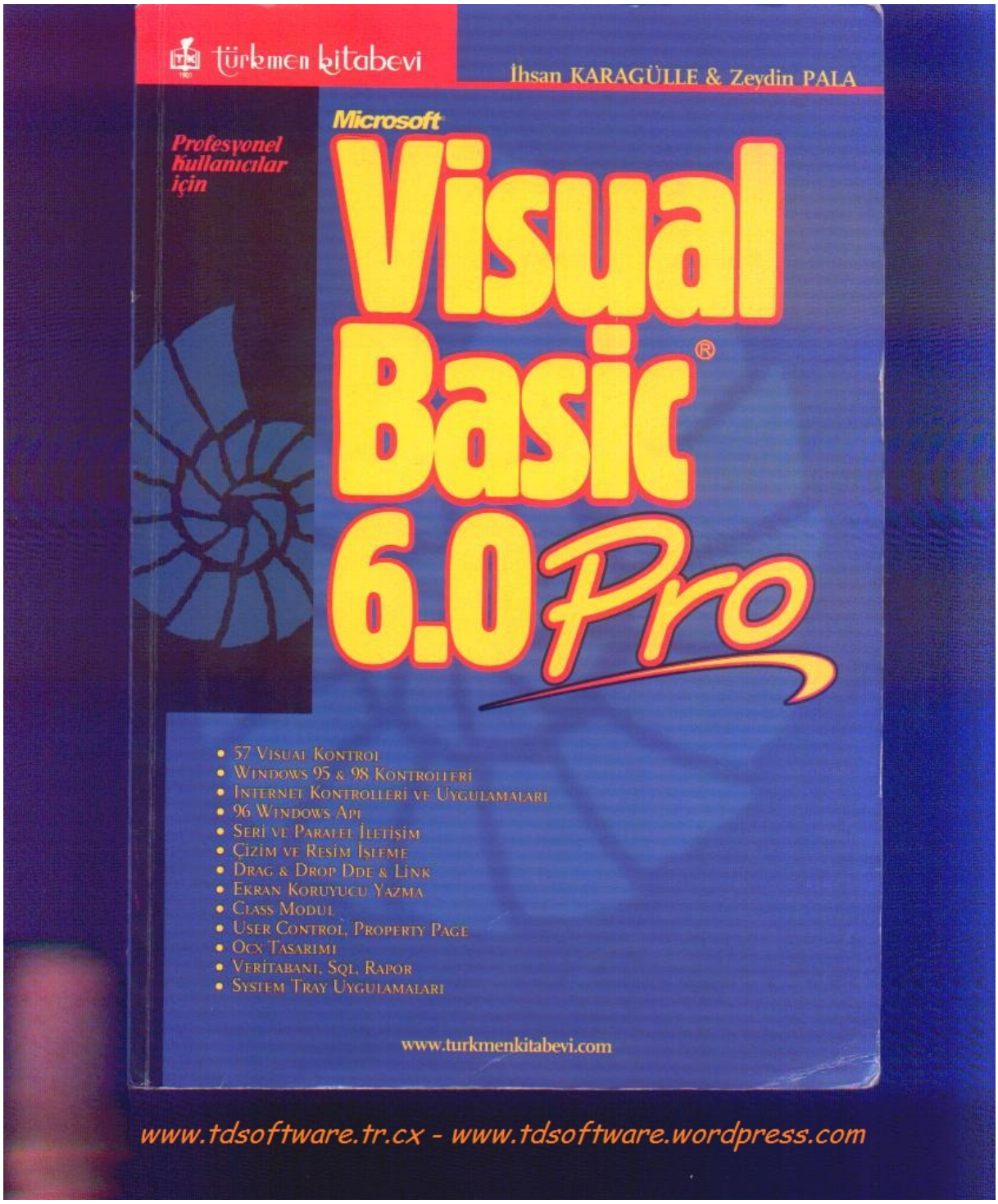


maktan kitapları iyice yıprandığı için aynı kitaptan bir tane daha aldıklarını ifade ediyorlardı. Okuyucularımızın bu memnuniyetini duymak bize fazlasıyla şevk verdi ve daha iyilerini hazırlama yönünde çabalarımızı artırdı. Ayrıca okuyucularımızdan gelen "ah keşke şu konu da olsa, yeni versiyonda şu konuyu da mutlaka anlatın, yeni baskıyı dört gözle bekliyoruz" şeklindeki isteklerini de dikkate alarak bu yönde örneklere ve konulara da birinci ve ikinci kitabımızda yer verdik.

İnşallah daha nice güzel eserlerle beraber olacağız.

***

Visual Basic Pro 6.0 kitabı toplam olarak on beş bölümden meydana gelmektedir:

Birinci bölümde, Visual Basicteki genel amaçlı kontroller anlatılmaktatır. **İkinci bölümde,** Windows 95 ve 98 kontrolleri anlatılmakta ve modern Windows uygulamalarının yazımı üzerinde durulmaktadır. **Üçüncü bölümde,** İnternet kontrolleri anlatılmakta ve internet uygulamaları verilmektedir. **Dördüncü bölümde,** okurlarımızdan gelen yoğun istek üzerine kitaba yerleştirdiğimiz Seri ve Paralel iletişimin VB ile nasıl yapılacağı anlatılmaktadır. **Beşinci bölümde,** Visual Basicte kullanılan diğer nesneler (ekran, yazıcı vb.) anlatılmaktadır. **Altıncı bölümde**, Menü tasarımı ve popup menülerin nasıl hazırlanacağı anlatılmaktadır. **Yedinci bölümde,** Bir çok programcının ilgisini çeken çizim konusu anlatılmaktadır. **Sekizinci bölümde,** Windowsun sunduğu önemli teknolojilerden biri olan Drag & Drop işlemlerinin VB ile nasıl programlanacağı anlatılmaktadır. **Dokuzuncu bölümde**, Yine Windows teknolojilerinden biri olan DDE & Link olayı anlatılarak Windows altında çalışan programların birbirleriyle nasıl haberleşebilecekleri programlarla anlatılmaktadır. **Onuncu bölümde**, Ekran koruyucu programların nasıl yazıldığı örneklerle anlatılmaktadır. **Onbirinci bölümde,** Class Module anlatılarak kullanıcının kendi sınıflarını nasıl tanımlayabileceği anlatılmaktadır. **Onikinci bölümde,** Visual Basic'in kalbini oluşturan kontrollerin bulunduğu OCX dosyalarının nasıl oluşturulacağı, kullanıcının kendi kontrollerini nasıl hazırlayacağı ve bu kontrollerle birlikte kullanabileceği Property Page'lerin nasıl yapacağı anlatılmaktadır. **Onüçüncü bölümde,** Veri tabanı konusu ele alınmaktadır. **Ondördüncü bölümde,** Windows API'leri anlatılmakta ve kullanıcının VB ile yapamadığı işlemleri API'lerle nasıl yapabileceği gösterilmektedir. Bu bölümde 96 tane Windows apisi anlatılmıştır. **Onbeşinci bölümde,** Bazı ekler yer almaktadır. Bu bölümde: VB'de ve Quick Basicte ortak olan komutlar ile kontrollerin bulundukları OCX dosyalarının listesi verilmektedir.

Kitabın bütün okuyucularımıza faydalı olması dileklerimizle.....

100. Yıl Üniversitesi Ahlat Meslek Yüksekokulu
Haziran 1999, Ahlat/Bitlis

Öğr. Gör. İhsan Karagülle
ikaragulle@hotmail.com

Öğr. Gör. Zeydin Pala
zpala@hotmail.com

İÇİNDEKİLER

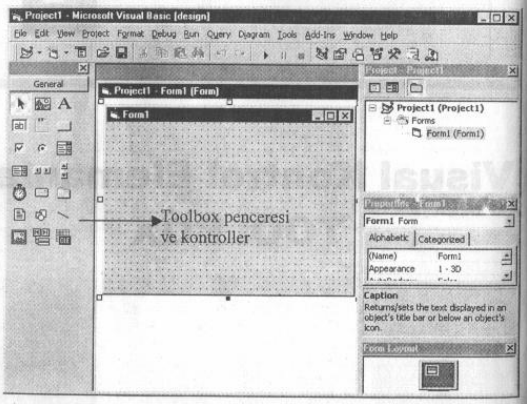
Bölüm 1

Visual Kontrol Elemanları
TOOLBOX

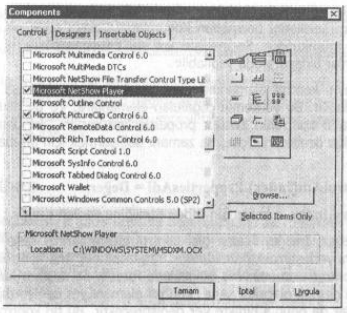
VB KONTROL ELEMANLARI (TOOLBOX)

Visual dillerin kullanıcıya sunduğu yeniliklerin başında şüphesiz ki kontrol elemanları gelmektedir. Kullanıcı artık küçük bir kontrol için uzun uzadıya program kodu yazmak zorunda kalmayacak, kontrol elemanları vasıtasıyla program modülleri daha kısa ve daha anlaşılır bir şekilde kontrol edilebilecektir.

Visual Basic programı çalıştırıldığı zaman Toolbox penceresinde, standart Visual Basic elemanları görüntülenir.



Tabii ki kontrol elemanları sadece bunlarla sınırlı değildir. Visual Basic'in OCX uzantılı olan bu dosyaları VB'nin ilgili dizinlerinde (Genellikle Windows'un System dizininde) bulabilirsiniz. Bunları Toolbox penceresine alıp kullanabilmek için **Project** menüsündeki **Components** seçeneğini ile açılan aşağıdaki pencereyi kullanacağız.



Pencerede işaretlediklerinizi projenize eklenmiş olacaktır. Eğer istediğiniz OCX dosyası listede yoksa **Browse** düğmesini kullanarak onu da bu listeye dahil edebilirsiniz.

Yine **Toolbox** penceresinde var olan bir kontrol elemanını kaldırmak için yukarıdaki pencereden o kontrolün işaretini kaldırmanız yeterlidir. Bir kontrol elemanını ToolBox'tan çıkarmak için o elemanın program içerisinde kullanılmıyor olması gerekir.

Her kontrol elemanı kendisine ait olan özellikleri (Properties) sayesinde kullanıcıya çok sayıda seçenek sunmaktadır. Bunlar kullanılarak programın gerek tasarımı, gerekse çalışma zamanı esnasında programa istenen şekil ve estetik verilebilmektedir. Properties penceresinde bulunan özelliklerin bir kısmı hem tasarım hem de çalışma zamanında değiştirilebilir. Tasarım anında değiştirilemeyecek olan özellikler properties penceresinde yer almamaktadır. Bunlar program modüllerinde kod yazılarak değiştirilebilir. Bazı özellikler ise hem çalışma zamanı hem de tasarım zamanı değiştirilemez, yalnız okunabilirler.

Bir kontrolün properties'ini çalışma zamanı değiştirmek için o properties'e aşağıdaki gibi atama yapılır.

KontrolAdı.PropertiesAdı = Değer

Buradaki KontrolAdı o kontrolün **Name** properties'ine verilen isimdir.

Kullanıcı her bir kontrol elemanını seçmek suretiyle formun istenilen yerine ve istenilen boyutta farenin sol tuşunu basılı tutarak çizebilir. Veya kontrol elemanı çift tıklatılarak da form üzerine alınabilir. Form üzerine alınan bir kontrol elemanı kesme, kopyalama, yapıştırma, taşıma ve birçoğu boyutlandırma işlemlerini

destekler. Form üzerindeki kontrolleri fare ile çerçeveleyip, seçmek suretiyle yerleşim şekillerini bozmadan konumlarını , veya ortak olan özelliklerini değiştirmek mümkündür. Seçme işlemini **shift** tuşuna basılı tutup seçilmek istenen elemanlar tıklanarak da yapılabilir.

Eğer aynı kontrol elemanından birden fazla kullanılacaksa kopyalama metodu kullanılabilir. Bu yöntemle çoğaltılan elemanlara, VB bir dizi numarası verir. Bu numara o kontrolün **Index** properties'ine verilir. Dizi olarak tanımlanmış bir kontrolün özelliklerini çalışma zamanı değiştirmek için atama aşağıdaki formatta yapılır.

KontrolAdı(Index).PropertiesAdi = Değer

Buradaki Index, o kontrolün Index özelliğine verilen sayıdır

Picturebox, Image, Frame gibi kontrol elemanlarının içine birden fazla kontrol elemanı yerleştirmek mümkündür. Yapılacak şey Ana kontrol elemanı seçmek ve Ana kontrol elemanının iç bölgesinde farenin sol tuşunu kullanmak suretiyle elemanı çizmek olacaktır. Ana kontrol elemanı yer değiştirdiği zaman içindeki elemanlar da onunla birlikte yer değiştirecektir. Bu tip kontroller diğer kontrolleri gruplamak için kullanılır ve birçok yararlı özellikleri vardır.

Bu bölümde her kontrol elemanı tek tek ele alınarak, bunların açıklanması, özellikleri (Properties), sahip olduğu olaylar (Events) ve o nesneye uygulanabilecek yöntemler (Methods) örneklerle açıklanmıştır. Özelliklerin bir kısmı her kontrol elemanı için aynı anlam ifade ettiğinden dolayı sadece bir eleman için verilmektedir. Ortak olmayan özellikler ise her eleman için ayrı ayrı ele alınmaktadır. Yine bazı özellikler ortak olmalarına rağmen birkaç eleman için değişik anlam ifade ettiğinden dolayı tekrar açıklanmaktadır.

Textbox(Metin Kutusu)

Bilgi girişi için kullanılan bir kontroldür. Programın çalışması veya tasarımı esnasında metin, kullanıcı tarafından değiştirilebilmesinin yanında kesme, kopyalama, yapıştırma, aşağı-yukarı, sağa-sola ilerleyebilme ve birden fazla satır girebilme gibi özelliklere de sahiptir.

Properties

Properties penceresi o anda seçilmiş olan elemana ait özellikleri görüntüler. Bu özellikler kullanıcı tarafından değiştirilebilir. Form üzerine alınan her bir nesne VB'nin default olarak verdiği bir isimle yerleşir. Kullanıcı bu isimle kontrol elemanını kullanabileceği gibi istediği ismi verebilme hakkına da sahiptir.

Text

Text kutusuna girilen metin nesnenin **Text** özelliğine atanacaktır. Bu özellik kullanılarak kullanıcının girdiği metin üzerinde işlem yapılır.

Text kutusunun **text** özelliği verilmeden sadece ismi kullanılırsa **text** özelliği kullanılmış gibi olur. Yani **isim.text** ile tek başına **isim** ile aynı anlamdadır.

Aşağıdaki her iki satır da aynı anlamdadır ve Text1 kutusunun içine "Visual Basic" yazılmasını sağlar.

```
Text1.Text = "Visual Basic"
Text1 = "Visual Basic"
```

Bütün kontrollerin bir default özelliği vardır. Text kutusunun default özeliği Text propertiesidir.

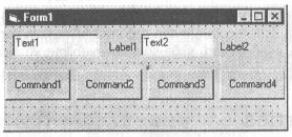
Kullanıcının Text kutusuna yazdığı bilgiyi öğrenmek için de bu özellik kullanılır.

```
ad = Text1.Text
no = Text2.Text
```

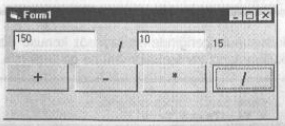
Eğer Text kutusunun içeriğinde bir sayı söz konusu ise bu sayıyı öğrenirken Val komutunu kullanmanız yanlışlıkların önüne geçmek için iyi olacaktır.

```
ad = Text1.Text
no = Text2.Text
sinav_notu = Val(Text3.Text)
```

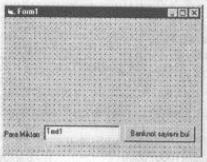
ÖRNEK: Kullanıcının iki text kutusuna girdiği sayılar üzerinde komut düğmeleri aracılığı ile dört işlemi yapabileceği basit bir hesap makinası yapalım. Bunun için aşağıdaki formu hazırlayın.



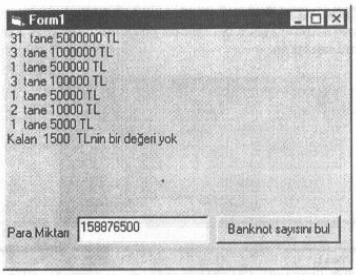

```
'ÖRNEK:Text
Private Sub Command1_Click()
 Label1 = "+"
 Label2 = Val(Text1) + Val(Text2)
End Sub

Private Sub Command2_Click()
 Label1 = "-"
 Label2 = Val(Text1) - Val(Text2)
End Sub

Private Sub Command3_Click()
 Label1 = "*"
 Label2 = Val(Text1) * Val(Text2)
End Sub

Private Sub Command4_Click()
 Label1 = "/"
 Label2 = Val(Text1) / Val(Text2)
End Sub

Private Sub Form_Load()
 Command1.Caption = "+"
 Command2.Caption = "-"
 Command3.Caption = "*"
 Command4.Caption = "/"
End Sub
```



ÖRNEK:Örnek olarak kullanıcının bir text kutusuna gireceği para miktarının içindeki banknotların sayısını bulacak bir program yazalım. Programımız için aşağıdaki formu hazırlayın.




```
'ÖRNEK : Text
Private Sub Command1_Click()
    Dim i, j
    Cls
    i = Val(Text1.Text) 'Text kutusuna yazılanı sayıya çevir

    j = Int(i / 5000000) 'kaç tane tam 5000000 var
    Print j; " tane 5000000 TL"
    i = i Mod 5000000 'kalanı bul

    j = Int(i / 1000000) 'kaç tane tam 1000000 var
    Print j; " tane 1000000 TL"
    i = i Mod 1000000 'kalanı bul

    j = Int(i / 500000)
    Print j; " tane 500000 TL"
    i = i Mod 500000

    j = Int(i / 100000)
    Print j; " tane 100000 TL"
    i = i Mod 100000

    j = Int(i / 50000)
    Print j; " tane 50000 TL"
    i = i Mod 50000

    j = Int(i / 10000)
    Print j; " tane 10000 TL"
    i = i Mod 10000

    j = Int(i / 5000)
    Print j; " tane 5000 TL"
    i = i Mod 5000
    Print "Kalan "; i; " TLnin bir değeri yok"
End Sub
```

Yukarıdaki programda kullandığımız iki değişkenimiz var. i değişkeni para miktarını, j değişkeni ise banknot miktarını göstermektedir.

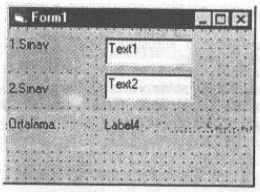
Bir banknottan kaç tane olduğunun bulunabilmesi için para miktarının o banknota bölünmesi ve sonucun tam kısmının alınması gerekir. j = Int(i / 5000000) satırı parayı 5 milyona bölerek tam kısmını almaktadır.

Para miktarındaki bir banknottan kaç tane olduğunu bulduktan sonra kalan miktarı bulmak için de **mod** operatörü kullanılabilir. i = i mod 5000000 işlemi para miktarındaki 5000000'luk banknotlar çıktıktan sonra kalan para miktarını verir.

ToolTipText

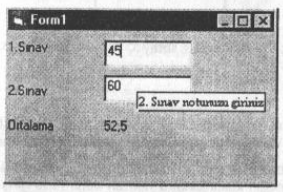
Windows altında çalışan bir çok modern programda, program öğelerinin birinin üzerinde fare ile kısa bir süre durduğunuzda açılan bir balonla onun ne işe yaradığını yazan mesajlar görmüşsünüzdür. VB'de de bu iş o kontrolün **ToolTipText** özelliğine atanacak metinle yapılır. Bu özelliğe verdiğiniz metin, kullanıcının o kontrol üzerinde kısa bir süre durmasıyla, küçük bir kutu içerisinde gösterilecektir.

Örnek olarak aşağıdaki formu hazırlayın. Text1 ve Text2'ye girilen notların ortalamasını Label4 içerisinde gösterelim. Kullanıcı Text1 üzerinde durunca 1.Sınav Notunuzu Giriniz, Text2 üzerinde durunca 2.Sınav Notunuzu Giriniz ve Label4 üzerinde durunca da 2 Sınavın ortalaması diye yazsın.




```
'ÖRNEK : ToolTipText
Private Sub Form_Load()
    Label4.ToolTipText = "2 Sınavın Ortalaması"
    Text1.ToolTipText = "1.Sınav Notunuzu Giriniz"
    Text2.ToolTipText = "1.Sınav Notunuzu Giriniz"
End Sub

Private Sub Text1_Change()
    'Text kutusunun içeriği değiştiğinde ortalamayı bul
    Label4 = (Val(Text1) + Val(Text2)) / 2
End Sub

Private Sub Text2_Change()
    'Text kutusunun içeriği değiştiğinde ortalamayı bul
    Label4 = (Val(Text1) + Val(Text2)) / 2
End Sub
```

Programı çalıştırdıktan sonra kontrollerin üzerinde fare ile kısa bir süre durursanız, aşağıdaki gibi o kontrolle ilgili açıklamayı gösteren bir kutu açılacaktır.



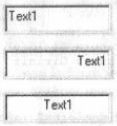
MultiLine

Bu özellik **True** ise text kutusuna birden fazla satır girilebileceğini gösterir. **False** ise tek satır girilebilir.

Normalde bu özellik **False**'dir ve enter tuşuna bastığınızda yazı bir alt satıra geçmez ve bip sesiyle uyarı verilir. Birden fazla satır girişi gerektiren durumlarda (örneğin adres girişi) bu özellik **True** yapılabilir.

Alignment

Nesne içerisindeki yazının sola, sağa veya ortaya yazılmasını sağlar.



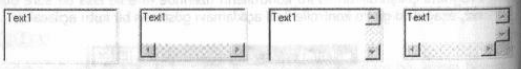
0 : Sola

1 : Sağa

2 : Ortaya

ScrollBars

MultiLine özelliğinin **true** olması durumunda etkili olan bu özellik text kutusu içindeki yazıyı aşağı-yukarı ve sola-sağa kaydırmak için kaydırma çubuklarının eklenmesini sağlar. Şu dört değeri alabilir:



0 : Yok 1 : Yatay 2 : Dikey 3 : Yatay ve dikey

MaxLength

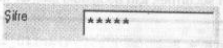
Text kutusuna girilebilecek maksimum karakter sayısını belirler. 0 verilirse bu sınır kaldırılır. Verilen sınır sayısı kadar karakterden fazlasını kabul etmez.

Kullanıcının belirli sayıdan daha fazla karakter girmesini önlemek için bu özellik kullanılır. Örneğin kullanıcının 3 harften daha fazla giriş yapmasını önlemek için bu özelliğe 3 değeri verilebilir.

MaxLength özelliği sadece kullanıcının girebileceği karakter sayısını sınırlar. Program kodu ile yapılan atamalarda bu sınır aşılabilir.

PasswordChar

Text kutusunu şifre girmek için kullanacaksınız; kullanıcının girdiği karakterlerin etraftakiler tarafından görülmesini istemezsiniz. **PasswordChar** özelliğine bir karakter girerek kullanıcının girdiği bütün karakterlerin bu karakterle görülmesini sağlayabilirsiniz. Kullanıcı ne yazarsa yazsın yazdığı karakter kadar ekranda **passwordchar** görecektir. Nesnenin içeriği ise kullanıcının yazdığı şey olacaktır. Örneğin buraya '*' girerseniz girilen bütün karakterler '*' ile gösterilecektir.



Locked

Text kutusunun **Locked** özelliği **True** yapıldıktan sonra kullanıcı Text kutusu üzerinde hiç bir değişiklik yapamaz.

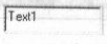
Örneğin kullanıcı şifreyi bilmediğinde veya belli sürelerle text kutusunda değişiklik yapılması istenmediğinde bu özellik True yapılabilir.

```
Private Sub Form_Load()
 If InputBox("Şifreyi giriniz") <> "abc" Then
  MsgBox ("Şifreyi bilemediniz. Bilgiler üzerinde değişiklik
          yapamazsınız.")
  Text1.Locked = True
 End If
End Sub
```

BorderStyle

Nesnenin ekran üzerindeki sınırlarının çerçeve şeklini belirler.

0 : Çerçeve yok



1 : Tek çizgili çerçeve

Appearance

Bu özellik kontrolün üç boyutlu görünümünü ayarlar. Bu özelliğin değerine 1 verilirse kontrol üç boyutluymuş gibi, 0 verilirse düz olarak gösterilir. Kontrolün üç boyutlu gösterilmesi daha güzel bir görüntü oluştururken, düz gösterilmesi daha hızlı olmasını sağlar.

SelStart,SelLength

Windows altında bir çok işlem için bloklama (seçme) yapmak gerektiğini biliyorsunuz. Kullanıcı Text kutularında seçme işlemi yapabilir. Kullanıcının seçtiği kısmı öğrenmek için bu iki özellik kullanılabilir. Ayrıca bu iki özelliği kullanarak siz de program kodu ile seçme işlemini yaptırabilirsiniz.

Bu özellikler seçmenin text içinde başlayacağı konumu ve uzunluğunu belirler.

Örneğin ilk 3 harfi seçtirmek için aşağıdaki satırlar kullanılabilir.

```
Text1.SelStart = 0
Text1.Sellength = 3
```

Tümünü seçtirmek için ise aşağıdaki satırlar kullanılabilir.

```
Text1.SelStart = 0
Text1.Sellength = Len(Text1)-1
```

SelText

Seçilmiş metin bu özellikle öğrenilip değiştirilebilir. **SelText** özelliğine yapılan atamalarda, eğer seçilmiş bir kısım varsa o kısım silinir, yoksa kursörün bulunduğu noktaya sıkıştırır.

```
Text1.SelText = "Merhaba"
```

Satırı ile text1 içinde seçilmiş olan kısmın yerine Merhaba yazılacaktır.

ÖRNEK:Örneğin seçili bir kısım varsa orayı yoksa da text kutusunun içeriğinin tamamını büyük/küçük harfe çevirecek bir program yazalım. (LCase fonksiyonu küçük harfe, UCase fonksiyonu ise büyük harfe çevirir)

Örneğimiz için aşağıdaki formu oluşturun.


```
'ÖRNEK : SelText,SelLength
Private Sub Command1_Click()
    If Text1.SelLength > 0 Then
    'Seçili bir kısım varsa
        'Seçili kısmı büyük harfe çevir
        Text1.SelText = UCase(Text1.SelText)
    Else
    'Seçili bir kısım yoksa
        'Tümünü büyük harfe çevir
        Text1.Text = UCase(Text1.Text)
    End If
End Sub

Private Sub Command2_Click()
    If Text1.SelLength > 0 Then
    'Seçili bir kısım varsa
        'Seçili kısmı küçük harfe çevir
        Text1.SelText = LCase(Text1.SelText)
    Else
    'Seçili bir kısım yoksa
        'Tümünü küçük harfe çevir
        Text1.Text = LCase(Text1.Text)
    End If
End Sub
```

HideSelection

True ise Text kutusunda seçilen metnin, kontrol başka bir nesneye geçtiğinde seçilen kısmın gizlenmesini sağlar.

False ise kontrol başka bir nesneye geçse de seçilen kısmın görülmesini sağlar.

Enabled

Nesnenin aktif, veya pasif olmasını sağlar. **False** yani pasif olması durumunda kullanıcı nesneyi görür ancak üzerinde işlem yapamaz.

Text kutusunun **Enabled** özelliği **False** ise kullanıcı giriş yapamaz.

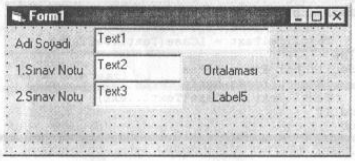
Visible

True ise nesne görülür, **False** yapılırsa görülmez. Belli şartlar gerçekleştiğinde gözükmesini istediğiniz kontrollerin visible özelliğini kullanabilirsiniz.

CausesValidation

Text kutusu kontrolü kaybettiğinde yani kullanıcı text kutusundan başka bir kontrole fare veya klavye yardımı ile geçtiğinde **Validate** olayının çalışıp çalışmayacağını belirler. Normalde bu özellik true'dir ve **Validate** olayı çalışır. Bu olaya yazacağınız kodla kullanıcının girdiği bilgiyi anında kontrol edebilir ve gerekiyorsa uyarabilirsiniz.

ÖRNEK: Örnek olarak not girişi yaptığımız bir program düşünelim. Kullanıcı notu olarak 0'dan küçük veya 100'den büyük bir rakam girdiğinde uyarılsın ve bunu düzeltmeden başka bir kontrole geçmesi önlensin. Örneğimiz için aşağıdaki formu hazırlayın.



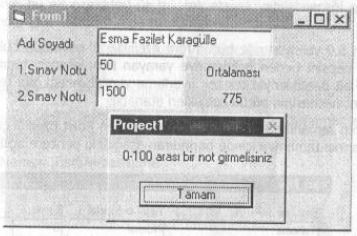
Text2 ve Text3 kontrollerinde 0-100 arasında olmayan bir rakam girilmesini önlemek istiyoruz.

```
'ÖRNEK:CausesValidation, Validate
Private Sub Form_Load()
 Text1.CausesValidation = True
 Text2.CausesValidation = True
End Sub

Private Sub Text2_Change()
 'bir değişiklik olduğunda ortalamayı bul
 Label5 = (Val(Text3) + Val(Text2)) / 2
End Sub

Private Sub Text3_Change()
 'bir değişiklik olduğunda ortalamayı bul
 Label5 = (Val(Text3) + Val(Text2)) / 2
End Sub

Private Sub Text2_Validate(Cancel As Boolean)
 If Val(Text2) < 0 Or Val(Text2) > 100 Then
  MsgBox ("0-100 arası bir not girmelisiniz")
  Cancel = True 'Çıkmasını önle
 End If
End Sub
```



```
Private Sub Text3_Validate(Cancel As Boolean)
 If Val(Text3) < 0 Or Val(Text3) > 100 Then
  MsgBox ("0-100 arası bir not girmelisiniz")
  Cancel = True 'Çıkmasını önle
 End If
End Sub
```

Text kutularından çıkarken **Validate** olayı meydana gelecek ve verdiğimiz şartı kontrol edecektir. Eğer girilen bilgi şartımıza uymuyorsa mesaj verecek ve oraya doğru bir değer girmeden başka bir yere geçmesine engel olacaktır.



FontBold

True ise nesne için kullanılan yazıyı koyu yapar.

FontItalic

True ise nesne için kullanılan yazıyı eğik yapar.

FontStrikethru

True ise nesne için kullanılan yazının ortasını çizer.

FontUnderline

True ise nesne için kullanılan yazının altını çizer.

FontName

Nesne için kullanılan yazının fontunu belirler. Bu font çalışma esnasında sistemde bulunan bir fontun numarası veya ismi verilerek değiştirilebilir.

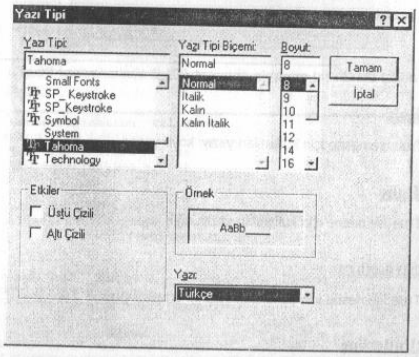
FontSize

Nesne için kullanılan yazının puntosunu belirler. maksimum 2048 olabilir.

Font

VB'nin 5.0 versiyonu ile bütün font özelliklerini bir arada bulunduran ve tek seferde hepsini birden belirlemeye yarayan Font nesnesi tanımlanmıştır. Bu nesnenin alt özellikleriyle tek tek ayarlama yapılabileceği gibi tek seferde de başka bir font nesnesinin bütün özellikleri atanabilir.

Tasarım zamanında properties penceresinden **Font** özelliği çift tıklanırsa yazı tipi ayarlarının tamamını içinde barındıran aşağıdaki pencere açılır.



Tasarım zamanında yukarıdaki pencere kullanılarak yazı tipi ayarları kolayca yapılabilir. Program çalışırken bu ayarları değiştirmek için ise aşağıdaki özellikler kullanılmalıdır.

Bu nesnenin şu alt özellikleri vardır.

Font.Name:	Fontun ismi
Font.Size:	Punto olarak büyüklüğü
Font.Strikethrough:	Üstü çizili durumu
Font.UnderLine:	Altı Çizili durumu
Font.Bold:	Kalın durumu
Font.Italic:	Eğiklik durumu
Font.Charset:	Ülke kodu

Font özelliğinin bu alt özellikleri ile tek tek atama yapılabileceği gibi başka bir font özelliğinden Set komutu ile de toplu atamalar yapılabilir.

Örneğin Text1 kutusunun yazı tipini kalın yapmak için

```
Text1.Font.Bold=True veya
Text1.FontBold=True şeklinde kullanabiliriz.
```

Text1 kutusunun yazı tipini Text2 kutusunun yazı tipi ile tamamen aynı özelliklere sahip olmasını sağlamak için ise

```
Text1.FontName= Text2.FontName
Text1.FontSize= Text2.FontSize
Text1.FontBold= Text2.FontBold
....
```

gibi bütün özellikleri tek tek eşitlemek yerine

```
Set Text1.Font=Text2.Font
```

ataması ile tek satırda yapabiliriz.

BackColor

Nesnenin zemin rengini değiştirir.

Bu özelliği oluşturan değer 7 digitlik bir hexadesimal sayıdır (0 ile &HFFFFFFF arasında) En düşük iki dijit kırmızı, sonraki iki dijit yeşil, ve diğer iki dijit mavi rengin yoğunluğunu gösterir. Bu ikili dijitlerin düşük değerli olmaları rengin koyuluğunu sağlar. En yüksek seviyeli dijitin değeri 0 değilse Windows'un system renklerinin kullanılacağını gösterir. Properties penceresinde olmayan renkler, bu dijitler ayarlanarak elde edilebilir. Kullanılabilecek renk sayısı $16^2 * 16^2 * 16^2 = 16.777.216$ renktir.

Durum D.	Mavi		Yeşil		Kırmızı	
x	x	x	x	x	x	x

Örneğin en açık tonlu yeşili elde etmek için &H100FF00 değeri kullanılır.

Renkleri ayarlamak için **QBColor( )** fonksiyonu da kullanılabilir. **QBColor( )** fonksiyonuyla 16 temel renkten biri alınabilir. Örneğin 2 nolu renk için aşağıdaki satır kullanılabilir.

```
Text1.BackColor = QBColor (2)
```

ForeColor

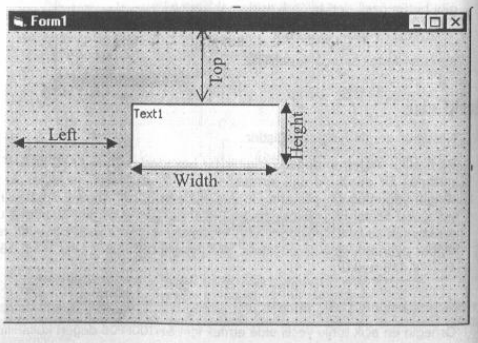
Yazının rengini belirler. Yukarıda anlatılan renk bileşimleri burada da geçerlidir.

Left,Top

Nesnenin sol ve üst noktalarının koordinatlarını belirler. Bu koordinatlar ekran koordinatları olmayıp içinde bulunduğu nesneye bağlı koordinatlardır. Bir nesne **Form, PictureBox** veya **Frame** içinde bulunabilir. İçinde bulunduğu nesnenin sol üst koordinatları 0,0 olmak üzere **left** ve **top** bu noktaya olan uzaklıktır.

Height,Width

Nesnenin boyu ve enini belirler. Bu iki özellik değiştirilerek kontrolün boyutları ayarlanabilir.

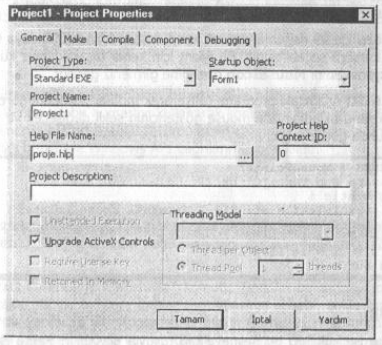


VB'de ölçü birimi olarak twip kullanılır. Ancak istenirse formun **ScaleMode** özelliği değiştirilerek ölçü birimi olarak piksel, veya metrik birimler verilebilir.

HelpContextId

Kontrolle ilgili yardım dosyasındaki konu numarası bu özellikle belirlenir.

Programınıza ait yardım dosyası varsa **Project-Properties** menüleri ile açılan aşağıdaki pencerenin **Help File Name** kutusuna bu yardım dosyasının ismi verilir.



Help dosyası belirlendikten sonra help dosyasındaki hangi konunun hangi kontrole karşılık geldiği **HelpContextID** özelliği ile belirlenir. Kullanıcı bu kontroldeyken F1 tuşuna bastığında belirlediğiniz yardım dosyası açılır ve bu özelliğe numarasını verdiğiniz konu gösterilir.

Index

Aynı isimli birden çok nesne oluşturulmuşsa VB bu nesneleri bir dizi olarak görür. Index parametresi bu nesnenin dizideki kaçıncı eleman olduğunu belirler, bu nesneye bu dizi indexiyle ulaşılır.

Index özelliğine bir sayı verilmişse o kontrole isminden sonra bu sayı verilerek ulaşılabilir. Örneğin Index özelliği 2 olan bir text kutusunun Text özelliğine

Text1(2).Text

satırı ile ulaşılabilir.

Bir kontrolü dizi olmaktan çıkarmak için ise Index özelliğini silmeniz gerekir.

MousePointer, MouseIcon

Mouse göstergesinin nesne üzerine geldiğinde alacağı şekli belirler. 0 ile 15 arası bir sayıdır. 0 normal hali 11 bekleme hali vb.

Bu özelliğe 99 değerini vererek kendi tasarladığınız (ICO veya CUR dosyasından) bir kursörü seçebilirsiniz. Bu işlem için kendi tasarladığınız kursörün bulunduğu dosyanın adını **MouseIcon** özelliğine girmeniz gerekir.

ÖRNEK: Aşağıdaki programı çalıştırarak mouse şekillerini görebilirsiniz. Her text kutusunu tıkladığınızda mouse göstericinin şekli değişecek ve numarası text kutusunda görülecektir.

```
'ÖRNEK : MousePointer
Private Sub Text1_Click ()
    Static i
    text1.Text = str(i)+ " Numaralı mouse pointer"
    text1.MousePointer = i
    i = i + 1
    If  i = 16 Then End
End Sub
```

TabStop

True ise kullanıcı bu nesneye tab tuşuyla ulaşabilir. **False** ise tab tuşuyla bu kontrol üzerine gelinmez, mouse ile, yazılım yoluyla veya varsa kısayol tuşu ile gelebilir.

Yoğun bilgi girişi gerektiren formlarda sık kullanılmayacak kontrollerin tabstop özelliklerini false yaparsanız kullanıcının bilgi girişi daha kolay olacaktır. Çünkü genelde yoğun bilgi girişi olan yerlerde tab tuşu sık kullanılır. Gereksiz yere ekrandaki bütün kontrolleri dolaşmamak için bazılarının bu özelliğini false yapmak gerekir.

TabIndex

Form üzerinde kullanıcının ulaşabileceği her nesnenin bir TabIndexi vardır. TabIndex kullanıcının **tab** tuşuyla kontroller arasında dolaşırken bu kontrollerin

sıralamasını belirler. Örneğin tab indexi 5 olan bir nesnede iken tab tuşuna basılırsa kontrol tab indexi 6 olan nesneye geçer.

Normalde her yerleştirilen kontrole otomatik olarak bir tab sırası verilir. Form tasarımınız bittikten sonra programı çalıştırıp tab tuşuyla bütün kontrolleri gezerek tab tuşunun sırayla gidip gitmediğini kontrol etmeniz gerekir. Çünkü araya yerleştirdiğiniz kontrollerin tab sıraları bozulacaktır.

Tag

Bu özelliğin Visual Basic için hiçbir anlamı yoktur. Kullanıcı bu parametreyi bir değişken gibi kullanabilir. Bu özellik bir string değişken olarak kullanılmalıdır.

hWnd

Windows altında çalışan kontrollerin **handle** diye adlandırılan tanıtıcı bir numarası vardır. Windows bu numarayı kullanarak kontrolleri tanır. Visual Basic'te bu numara nesnenin **hWnd** özelliği ile programın çalışması esnasında öğrenilebilir. **hWnd** özelliği genellikle Windows API çağrılarında gereklidir.

Parent

Nesnenin üzerinde bulunduğu form'a ulaşmayı sağlar. Nesnenin isminden sonra verilen **parent** özelliği o nesneyi değil nesnenin üzerinde bulunduğu formu temsil eder.

Örneğin Text1.Parent.Print "Merhaba" komutu Text1'in bulunduğu forma Merhaba yazacaktır. Böyle bir şeye ihtiyaç duyulmasının sebebi; çoklu formla çalışma esnasında yazılacak alt programların bütün formlar için kullanılabilmesinin sağlanmasıdır. Ayrıca bir defa yazılan bir alt programın değiştirilmeye gerek duyulmadan diğer programlarda da kullanılabilmesini sağlar.

Şöyle bir alt programımız olduğunu düşünelim: Alt programa girecek olan nesneyi bulunduğu form içinde ortalasın.

```
Private Sub Kaynagi_Ortala (Source As Control )
  Source.Left = (Source.Parent.Width - Source.Width) / 2
  Source.Top = (Source.Parent.Height - Source.Height) / 2
End Sub
```

Alt programı kullanmak için ortalanmak istenen nesnenin isminin form ismiyle birlikte alt programa giriş olarak verilmesi gerekir.

```
Kaynagi_Ortala  form2.Text1
```

Şeklinde alt program çağrıldığında Source=form2.text1 ,Source.Parent=form2 olacaktır. Dolayısıyla **Source.Left** satırı **Form2.Text1.Left** gibi, **Source.Parent.Width** satırı ise **Form2.Width** gibi işleme girecektir.

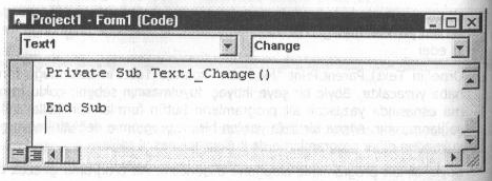
Görüldüğü gibi alt programımız formdan ve kontrolden bağımsız bir haldedir. Dolayısıyla herhangi bir formdan veya programdan çağrılabilir.

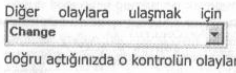
Events

Nedir bu olaylar?

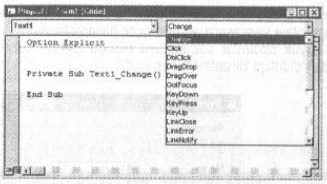
Windows'ta; bir nesneyi tıklamak (Click, DblClick), Mouse'un bir nesne üzerine gelmesi halinde açıklama ifadelerinin görüntülenmesi (MouseMove), Iconların taşınması olayı (Drag), Bir dosyanın fare ile tıklatılarak bir yerden başka bir yere alınması olayı (DragDrop), Mouse'un bazı bölgelerde gezinmesi halinde şeklinin değişmesi (DragOver), Mouse'un tuşları basılı tutularak çizim yapılması (MouseDown, MouseUp) vb. gibi yapılan her hareket bir olayı temsil etmektedir.

Form ve Form üzerinde bulunan her kontrol elemanı çift tıklatılarak o elemana ait kod penceresine ve **olay** prosedürlerine ulaşılır. Örneğin text kutusunu çift tıklarsanız Text kutusunun Change olayına ulaşırsınız.



Diğer olaylara ulaşmak için ise pencerenin sağ üst köşesindeki 

kutusunu kullanabilirsiniz. Bu listeyi aşağı doğru açtığınızda o kontrolün olayları listelenecektir.



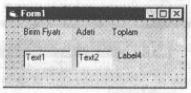
Yukarıdaki listeden istenen olaya ulaşılarak ona ait kod yazılabilir.

Aşağıda her kontrol elemanına ait **olaylar (Events)** tek tek ele alınarak incelenmiştir.

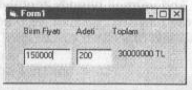
Change( )

Text kutusunun içerdiği metin değiştiği zaman bu olay meydana gelir. Bu olay kutu içerisinde tuş vuruşlarıyla kullanıcı tarafından meydana getirilebileceği gibi program vasıtasıyla text kutusuna atama yapılırken de meydana gelir.

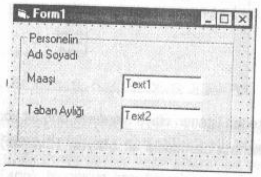
ÖRNEK:Örneğin aşağıdaki formda birim fiyatın yazıldığı Text1 veya Adetin yazıldığı Text2'de herhangi bir değişiklik olduğunda anında toplamın bulunup Label4'e yazılması istenir. Böyle durumlarda Change olayı kullanılabilir.



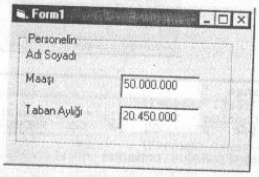

```
'ÖRNEK : Change
Private Sub Text1_Change()
    Label4.Caption = Val(Text1) * Val(Text2) & " TL"
End Sub
Private Sub Text2_Change()
    Label4.Caption = Val(Text1) * Val(Text2) & " TL"
End Sub
```



ÖRNEK: Aşağıdaki form dizaynı kullanılarak text kutularına girilen sayısal ifadeler Change olayı vasıtasıyla anında üçer basamaklı olarak ayrılmaktadır. Bu olay özellikle büyük rakamlar söz konusu olduğu zaman kullanıcıyı okuma kolaylığı bakımından oldukça rahatlatacaktır.



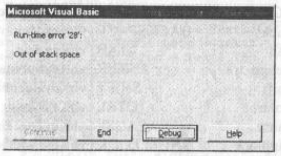

```
'ÖRNEK : Change
Private Sub Text1_Change()
 Dim X
 X = Text1.SelStart
 Text1 = Format(Text1, "###,###")
 Text1.SelStart = X + 1
End Sub
Private Sub Text2_Change()
 Dim X
 X = Text2.SelStart
 Text2 = Format(Text2, "###,###")
 Text2.SelStart = X + 1
End Sub
```



Bu olayı kullanarak kod yazmak başınızı ağrıtabilir. Çünkü **change** olayı birbirini tekrarlayan seferlerde meydana gelebilir ve programı sonsuz bir döngüye sokabilir. Bu durum aşağıdaki örnekte *"Out Of Stack Space"* mesajıyla kendini gösterir.


```
Private Sub Text1_Change( )
      static i
      i = i + 1
      text2 = " Text1'in içeriği" & i & ". defa değişti"
End Sub

Private Sub Text2_Change( )
      static j
      j = j + 1
      text1 = " Text2'nin içeriği"& j & ". defa değişti"
End Sub
```

Yukarıdaki örnekte Text1, Text2 nin içeriğinin değiştiriyor böylece de Text2'de Change olayı meydana geliyor. Text2 nin Change olayı da Text1'i değiştirdiği için birbirlerini sürekli olarak çağırıyorlar. Bu işlem sonsuz bir döngüye girdiği için sonuçta stack taşmasına sebep olur.



Click( )

Genellikle farenin tıklanmasıyla meydana gelen bir olaydır. Aynı olay bazı nesneler için klavye ile de gerçekleşebilir. Bu durum, farenin yaptığı iş klavyeden de yapılabiliyorsa meydana gelir. (ComboBox'larda aşağı, yukarı tuşları, komut düğmelerinde enter, boşluk gibi.)

DblClick( )

Nesnenin üzerine mouse ile çift tıklanması durumunda meydana gelir.

ÖRNEK: Click olaylarına örnek olarak bir text kutusu üzerine tıklandığında yazı renginin, çift tıklandığında da Zemin renginin değişmesini sağlayacak bir program yazalım.

```
'ÖRNEK : Click, DblClick
Private Sub Text1_Click ( )
 Text1.ForeColor = QBColor(Rnd * 15)
End Sub
```

```
Private  Sub Text1_DblClick ( )
 Text1.BackColor = QBColor(Rnd * 15)
End Sub
```

KeyDown (KeyCode As Integer, Shift As Integer)

Bu olay klavyeden bir tuşa basıldığında meydana gelir ve basılı tutulduğu müddetçe devam eder. **KeyPress** olayından önce meydana gelir ve **KeyPress** olayının tanımadığı tuşları da tanır. Bu tuşların genel bir listesi için aşağıdaki tablo-ya bakınız.

KeyCode

Basılan tuşun Scan kodudur. Klavyedeki tuşların Scan kodları aşağıdaki gibidir.

KeyCode	Tuş	KeyCode	Tuş
vbKeyBack	BACKSPACE	vbKeyMultiply	*
vbKeyTab	TAB	vbKeyAdd	+
vbKeyReturn	ENTER	vbKeySubtract	-
vbKeyShift	SHIFT	vbKeyDecimal	.
vbKeyControl	CTRL	vbKeyDivide	/
vbKeyPause	PAUSE	vbKeyPageUp	PAGE UP
vbKeyNumlock	NUM LOCK	vbKeyPageDown	PAGE DOWN
vbKeyCapital	CAPS LOCK	vbKeyEnd	END
vbKeyEscape	ESC	vbKeyHome	HOME
vbKeySpace	SPACEBAR	vbKeyLeft	Sol
vbKeyPrint	PRINT SCREEN	vbKeyUp	Yukarı
vbKeyInsert	INS	vbKeyRight	Sağ
vbKeyDelete	DEL	vbKeyDown	Aşağı
vbKeyA- vbKeyZ		A-Z harfleri	
vbKey0- vbKey9		0-9 rakamları	
vbKeyNumpad0- vbKeyNumpad9		sağdaki 0-9 rakamları	
vbKeyF1- vbKeyF16		F1-F16 tuşları	

Shift : Shift, Alt ve Control tuşlarının durumunu belirtir.

1: vbShiftMask : Shift tuşu maskesi,

2: VbCtrlMask : Control tuşu Maskesi,

4: VbAltMask : Alt tuşu Maskesi,

shift parametresi bu maskelerden biri ile *AND* işlemine tabi tutularak bu tuşlardan hangilerinin basıldığı anlaşılabilir.


```
text1 = ""
If shift AND vbShiftMask Then Text1 = "Shift Tuşu Basılı"
If shift AND VbCtrlMask Then Text1 = Text1+"Control Tuşu Basılı"
If shift AND VbAltMask Then Text1 = Text1+"Alt Tuşu Basılı"
```

Yukarıdaki gibi bir kod ile bu tuşlardan hangisinin veya hangilerinin basılı olduğu anlaşılabilir.

ÖRNEK: Örneğin Ctrl ve yön tuşları ile bir text kutusunu taşıyacak, Alt ve yön tuşları ile de boyutlarını değiştirecek bir kod yazalım. Bunun için formunuzun üzerine bir text kutusu yerleştirip aşağıdaki kodu yazın. Programı çalıştırdıktan sonra Ctrl ve yön tuşları ile bir text kutusunu taşıyabilecek, Alt ve yön tuşları ile de boyutlarını değiştirebileceksiniz.

```
'ÖRNEK : KeyDown
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
    'Ctrl tuşu basılı ise
    If Shift And vbCtrlMask Then
        Select Case KeyCode
            Case vbKeyLeft: Text1.Left = Text1.Left - 5
            Case vbKeyRight: Text1.Left = Text1.Left + 5
            Case vbKeyUp: Text1.Top = Text1.Top - 5
            Case vbKeyDown: Text1.Top = Text1.Top + 5
        End Select
    End If
    'Alt tuşu basılı ise
    If Shift And vbAltMask Then
        Select Case KeyCode
            Case vbKeyLeft: Text1.Width = Text1.Width - 5
            Case vbKeyRight: Text1.Width = Text1.Width + 5
            Case vbKeyUp: Text1.Height = Text1.Height - 5
            Case vbKeyDown: Text1.Height = Text1.Height + 5
        End Select
    End If
End Sub
```

KeyUp (KeyCode As Integer, Shift As Integer)

Bu olay basılan tuş bırakıldığı anda meydana gelir. Her klavye eventin'de oluşmaz. Şöyle ki A harfine uzun bir süre basarak birden çok A girişi yapılır. Bu giriş esnasında **KeyUp** olayı sadece son A harfinin yazılmasından sonra meydana gelir.

KeyPress(KeyAscii As Integer)

Bu olayı klavyedeki tuşların bir kısmı meydana getirir. Bu olay basılan tuşların ASCII kodları varsa meydana gelir. **KeyDown** olayından sonra, **KeyUp** olayından önce meydana gelir.

Klavye eventi sırasıyla şu olayları meydana getirir: **KeyDown, KeyPress, Change, KeyUp**

ÖRNEK: KeyPress olayını kontrol ederek küçük harfleri büyük harfe çeviren ve rakamları kabul etmeyen bir örnek aşağıda verilmiştir.

```
'ÖRNEK : KeyPress
Private  Sub Text1_KeyPress (keyascii As Integer)
 If  keyascii >Asc("0") AND keyascii <= Asc("9") Then
  keyascii = 0 'Rakam ise iptal et
 Else
  keyascii =  Asc (Ucase (Chr(keyascii)))'büyük harfe çevir
 End If
End Sub
```

ÖRNEK: Aşağıdaki örnekte ise kullanıcının bastığı tuşlar sona değil de başa ekleniyor. Tabi ki sonuçta kullanıcı ters yazıyor.

```
ÖRNEK : KeyPress
Private  Sub Text1_KeyPress (keyascii As Integer)
    text1 = Chr(keyascii) + text1'Basılan tuşu başa ekle
    keyascii = 0'ye tekrar eklenmemesi için iptal et
End Sub
```

ÖRNEK: Aşağıdaki örnek ise **KeyPress , KeyUp, KeyDown** ve **Change** olaylarının daha açık bir şekilde anlaşılmasını sağlayacaktır. Bir tuşa bastığınızda oluşan key olayları sırasıyla yazılacaktır.

```
'ÖRNEK : KeyPress, KeyUp, KeyDown, Change
Private  Sub Text1_KeyUp (keycode As Integer, shift As Integer)
 Static i
 text6 = text6 + "KeyUp"
 'text6 olayların meydana geliş sırasını gösterir
 i = i  + 1 'Olayın kaçıncı defa çalıştığını görmek için
 text3 = i+". KeyUp, KeyCode:"+Str(keycode)+" shift:"+Str(shift)
End Sub

Private  Sub Text1_Change ()
 text6 = text6 + "Change"
End Sub

Private  Sub Text1_KeyDown (keycode As Integer, shift As Integer)
 Static i
 text6 = text6 + "KeyDown"
 i = i + 1
 text2 =i+". KeyDown, KeyCode:"+Str(keycode)+" shift:"+Str(shift)
End Sub

Private  Sub Text1_KeyPress (keyascii As Integer)
 Static i
```



```
 text6 = text6 + "KeyPress"
 i = i + 1
 text4 = i + ". KeyPress, KeyAscii:" + Str(keyascii)
End Sub
```

KeyPress,KeyDown,KeyUp Olaylarının tanıdıkları tuş grupları

Tuş Grubu	KeyPress	KeyDown	KeyUp
Num,Caps,Scroll	Hayır	Evet	Evet
Control, Alt, Shift	Hayır	Evet	Evet
Alt + Bir harf	Hayır	Evet	Evet
Ctrl + Bir harf	Evet	Evet	Evet
Kısayol Tuşları	Hayır	Evet	Hayır
Fonksiyon Tuşları	Hayır	Evet	Evet
Yön Tuşları	Hayır	Evet	Evet
Normal Tuşlar	Evet	Evet	Evet

MouseDown, MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)

Bu olaylar; bir kontrolün üzerinde iken mouse düğmelerinden birinin basılıp bırakılması esnasında meydana gelir. Burada **MouseDown** farenin basılmasına, **MouseUp** ise farenin basılmış tuşunun bırakılmasına karşılık gelir.

Button

Button parametresi farenin basılı tuşlarını ifade eder.

1: vbLeftButton:Farenin sol tuşuna,

2: vbRightButton:Farenin sağ tuşuna,

4: vbMiddleButton:Farenin orta tuşuna basıldığını gösterir.

Shift:

Basılan **control, alt** ve **shift** tuşların durumlarını gösterir. Bu parametre **Keydown** olayında açıklanmıştır.

X,Y:

Fare göstergesinin bulunduğu konumun koordinatlarını verir.

ÖRNEK: Aşağıdaki örnekte kullanıcı text kutusunda farenin sol tuşuna bastığında text kutusunun zemin rengi ile yazı rengi değiştiriliyor , bıraktığında ise orijinal haline geliyor.

```
'ÖRNEK : MouseUp,MouseDown
Private  Sub text1_mousedown (Button As Integer, Shift As
Integer, X As Single, Y As Single)
    Dim c
    c = text1.BackColor
    text1.BackColor = text1.ForeColor
    text1.ForeColor = c
End Sub

Private  Sub Text1_MouseUp (Button As Integer, Shift As Integer,
X As Single, Y As Single)
    Dim c
    c = text1.BackColor
    text1.BackColor = text1.ForeColor
    text1.ForeColor = c
End Sub
```

MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

Mouse'un bir form yada bir kontrol elemanı üzerinde dolaşması sonucu **MouseMove** olayı meydana gelir. Dolaşma olayı devam ettiği müddetçe bu olay meydana gelir.

ÖRNEK: Aşağıda verilen örnekte fare **Text1** kutusu üzerine geldiği zaman farenin bulunduğu koordinatlar text1 kutusuna yazılmakta, ayrıldığı zaman ise farenin başka yerde olduğu belirtilmektedir.

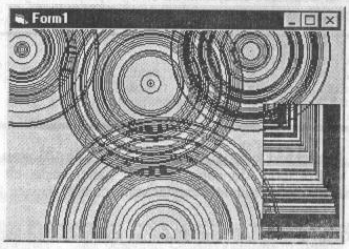
```
'ÖRNEK : MouseMove
Private  Sub Form_MouseMove (Button As Integer, Shift As Integer,
X As Single, Y As Single)
  Text1 = "Fare artık üzerimde değil..."
End Sub

Private  Sub Text1_MouseMove (Button As Integer, Shift As
Integer, X As Single, Y As Single)
  Text1 = "Fare Şu anda " & "   Dikey :" & Y & "   Yatay :" & X &
"koordinatlarında üzerimde gezmektedir."
End Sub
```

ÖRNEK: Farenin sol tuşu basılı tutularak taşındığında dikdörtgen, sağ tuşu basılı tutularak taşındığında ise daire çizecek bir programı mouse olaylarını kullanarak yazalım.

```
'ÖRNEK : MouseMove, MouseDown
Option Explicit
Dim mx, my
```



```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X
As Single, Y As Single)
    mx = X
    my = Y
    PSet (X, Y)'başlangıç koordinatını belirlemek için nota koy
End Sub

Private  Sub Form_MouseMove (Button As Integer, Shift As Integer,
X As Single, Y As Single)
    'Sağ tuş basılı ise çember çiz
    If Button = vbRightButton Then Circle (mx, my), Abs(mx - X)
    'Sol tuş basılı ise dikdörtgen çiz
    If Button = vbLeftButton Then Line (mx, my)-(X, Y), , B
End Sub
```



GotFocus()

Klavye kontrolünün bu nesneye geçmesi halinde meydana gelen bir olaydır. Bir nesneye kontrolün geçebilmesi için görülebilir (Visible) ve aktif (Enabled) olması gerekir.

ÖRNEK: Örneğin text kutusuna kontrol geçtiğinde içindeki metnin seçilmesini isteyelim.

```
'ÖRNEK : GotFocus
Private  Sub Text1_GotFocus ( )
 Text1.selstart = 0
 Text1.sellength = Len( Text1 )
End Sub
```


LostFocus()

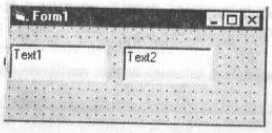
Bu olay herhangi bir kontrol elemanının kontrolü kaybetmesi esnasında meydana gelir. Bir eleman, kontrolü ya Tab tuşuyla yada fare tarafından tıklanmasıyla alır. Tab tuşu elemanlar arasında geçişi sağladığından dolayı elemanlara kontrolü verme veya geri alma olaylarında rol oynar. LostFocus olayının gerçekleşmesi için form üzerinde birden fazla kontrol elemanı yer almalıdır. Form üzerindeki menülerin seçilmesi *LostFocus* olayını meydana getirmez. Bu olay daha çok kontrolü kaybeden elemanın içeriğini değerlendirmek için kullanılabilir.

ÖRNEK: Örneğin kullanıcının girdiği sayı istediğimiz bir aralıkta değilse elmanın kontrolü kaybetmesini önleyelim. Bu örneğin çalışmasını görmek için form üzerine birden fazla eleman koymanız gerekir.

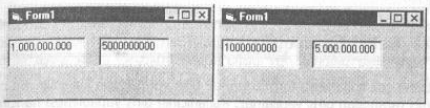
```
'ÖRNEK : LostFocus
Private  Sub Text1_LostFocus ()
 If  Val( text1 ) < 1 Or Val( Text1 ) > 100 Then
     Text1.SetFocus'Kontrolü tekrar Text1'e bırak
 End If
End sub
```

Burada, kontrol text kutusuna geçtikten sonra 1 ile 100 arası bir sayı girilmediği sürece kontrol form üzerindeki (menüler hariç) başka bir elemana geçmez. **SetFocus** bir metod olup Methods'lar kısmında açıklanmıştır.

ÖRNEK: Şimdi aşağıdaki form tasarımını kullanarak text kutularından herhangi birisinin focus kontrolünü kaybetmesi halinde içindeki sayısal verinin üçer basamak ayrılmasını, focus kontrolünü alması halinde ise tekrar eski haline döndürmesini sağlayalım:



Programı çalıştırıp tab tuşuna bastığımız zaman aşağıdaki görüntüyü elde ediyoruz:



Programın kodu aşağıdaki gibi olacaktır:

```
'ÖRNEK : LostFocus,Gotfocus
Private Sub Text1_GotFocus()
  Text1 = Val(Format(Text1, "#"))
End Sub

Private Sub Text1_LostFocus()
  Text1 = Format(Text1, "###,###")
End Sub

Private Sub Text2_GotFocus()
  Text2 = Val(Format(Text2, "#"))
End Sub

Private Sub Text2_LostFocus()
  Text2 = Format(Text2, "###,###")
End Sub
```

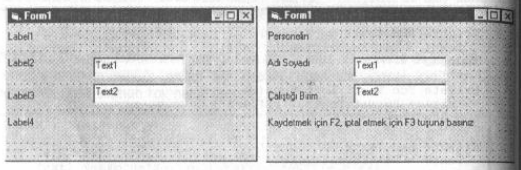
Validate(Cancel As Integer)

6.0 versiyonuyla ile eklenen bu özellik LostFocus olayında anlattığımız kontrol olayını yapmaya yarar. Eğer nesnenin **CausesValidation** özelliği **True** ise bu olaya yazdığınız kod çalışacak ve nesnenin içeriğini kontrol edebilmenizi sağlayacaktır. Eğer nesnenin içeriği uygun değilse **Cancel** parametresine **True** atanarak kontrolün tekrar aynı nesneye bırakılması sağlanabilir.

```
Private Sub Text1_Validate(Cancel As Boolean)
 If Val(Text1) < 0 Or Val(Text1) > 100 Then
  MsgBox ("0-100 arası bir not girmelisiniz")
  Cancel = True 'Çıkmasını önle
 End If
End Sub
```

Label Control(Etiket)

Daha çok kullanıcıya form üzerinde bilgi vermek için kullanılır. Bu nesneye kullanıcı tarafından giriş yapılamaz. Kullanıcının giriş yapabilmesi haricinde text kutusuna çok benzer.



Properties

Caption

Nesnenin üzerindeki yazı bu özellikle belirlenir. Tasarım anında properties penceresi aracılığı ile Label üzerindeki yazı belirlenebileceği gibi çalışma zamanında da aşağıdaki gibi bir kodla **Caption** özelliğine atanarak yapılabilir.

```
Label1.Caption = "Öğrencinin Adı Soyadı:"
```

Caption özelliğinde & karakterinin kullanılması bu karakterden sonraki karakteri kısa yol tuşu yapar.

```
Label1.Caption = "Öğrencinin &Adı Soyadı:"
```

Yukarıdaki satır A harfini kısayol tuşu yapar. Alt-A tuşlarına basılması durumunda klavye kontrolü TabIndex'i Label'den bir büyük olan elemana bırakır.

Bu kontrol çoğunlukla diğer kontrollere açıklama yazmak için kullanılır. Örneğin değişik bilgilerin girileceği text kutularının veya diğer kontrollerin önüne Labeller konarak bunlara neyin girileceği belirtilir. Labelin açıkladığı kontrolün tabIndex'i Labelin tabindex'inden bir büyük verilerek kısayol tuşuna basılması durumunda kontrolün açıkladığı nesneye bırakılması sağlanabilir.

Multiline özelliği **True** olan bir text kutusuna veya bir label kontrolüne kod yazarak bir kaç satırlık bilgi eklenmek istediğinde her satır dolduğunda yazı kendiliğinden bir aşağı satıra inecektir. Satırın bittiği yeri kendiniz belirlemek istiyorsanız, buraya **Chr(10) + Chr(13)** satırlarını eklemeniz gerekir. Chr (13) bir alt satıra, Chr (10) ise satır başına gelmeyi sağlar.

```
Label1.Caption="Birinci satır"+chr(13)+chr(10)+"ikinci satır"
```

Autosize

True ise nesnenin boyutları içeriğinin boyutlarına göre yeniden ayarlanacaktır. İçerik değiştiği anda kendiliğinden boyutlar da değişecektir.

BackStyle

Bu özellik nesnenin üzerinde bulunduğu konuma uymasını sağlar.

0 ise üzerinde bulunduğu nesnenin görülmesini sağlar, yani bir cam üzerine yazılmış yazı gibi davranır.

1 ise kendi zemin rengi üzerinde bulunur ve altındaki nesneyi göstermez.

Command Button (Komut Düğmesi)

Bir olayın kullanıcı tarafından başlatılması için programlarda çok kullanılan kontrollerden biridir.

Properties

Caption

Komut düğmesi üzerinde yazılacak olan mesajı içerir. Diğer nesnelerin **Caption** özelliklerinde olduğu gibi altı çizili harf o nesnenin kısa yol tuşudur ve alt tuşuyla birlikte altı çizili harfe basılması durumunda nesne aktif hale gelir.

Altı çizili harfi (kısayol tuşu) belirlemek için Caption'da o harfin önüne **&** konur.

Örneğin aşağıdaki gibi bir kodla komut düğmesinin kısayol tuşu Alt+d harfi olarak belirlenebilir.

```
Command1.Caption = "Yaz&dır"
```

Style

Bu özellik komut düğmesinin yazılı mı, resimli mi olacağını belirler.

0 ise komut düğmesinin üzerinde **Caption** özelliği ile belirlenen yazı bulunur.

1 ise komut düğmesinin üzerinde **Picture** özelliği ile belirlenen resim bulunur.

Picture, DisabledPicture, DownPicture

Eğer komut düğmesinin **Style** özelliğine 1 verilerek resimli bir komut düğmesi olacağı belirtilmişse bu üç özellikle komut düğmesinin üzerinde bulunacak resim belirlenebilir.

Picture özelliği ile belirlenen resim komut düğmesinin aktif (Enabled özelliği True) iken,

DisabledPicture özelliği ile belirlenen resim komut düğmesinin pasif (Enabled özelliği False) iken,

DownPicture özelliği ile de komut düğmesinin basılı iken gösterilecek resim belirlenir.

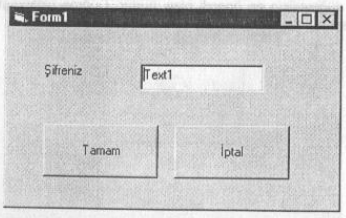
Default

Bir komut düğmesinin **default** özelliği **True** ise o düğmenin bulunduğu form üzerinde Entere basılması durumunda o düğme tıklanmış gibi olur.

Cancel

Bu özellik de **default** özelliği gibidir ancak **ESC** tuşu ile aktif hale gelir. **Cancel** özelliği **True** olan bir komut düğmesinin bulunduğu formda ESC tuşuna basılmasıyla bu komut düğmesi aktif yapılmış olur.

ÖRNEK: Örnek olarak aşağıdaki formu oluşturun.



```
'ÖRNEK: Default, Cancel
Private Sub Form_Load()
 'Enter tuşuna basılırsa command1 aktif olsun
 Command1.Default = True
 'ESC tuşuna basılırsa command2 aktif olsun
 Command2.Cancel = True
 'Text1 kutusunu şifre girişi için kullan
 Text1.PasswordChar = "*"
End Sub

Private Sub Command1_Click()
 If Text1.Text = "1234" Then
  MsgBox ("Doğru şifre")
  End
 Else
  MsgBox ("Yanlış şifre")
 End If
End Sub

Private Sub Command2_Click()
 End
End Sub
```

Programı çalıştırıp şifreyi yazdıktan sonra Enter tuşuna basarsanız Tamam düğmesinin, ESC tuşuna basarsanız İptal düğmesinin aktif olduğunu göreceksiniz.

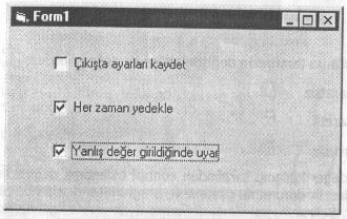
Events

Click( )

Komut düğmesinin en önemli olayı budur. Düğme tıklandığında yapılması istenen işlem bu olaya yazılır.

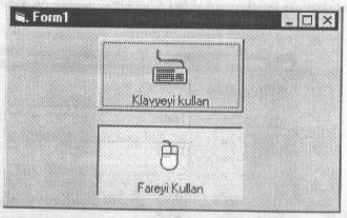
CheckBox (İşaret Kutusu)

Windows'ta çok kullanılan kontrollerden biri de kullanıcının belirli özellikleri aktif veya pasif hale getirmek için kullandığı CheckBox'lardır. Anlaşılırlığı ve kullanımı kolay olması sebebiyle kolay kullanılabilir arabirimler oluşturmak için oldukça faydalı bir kontroldür.



Komut düğmesinde olduğu gibi **Style** özelliği ile **CheckBox** kontrolünün resimli olması da sağlanabilir. **Picture, DisabledPicture** ve **DownPicture** özellikleri ile de kontrolün üç durumu için farklı resimler yüklenebilir.

Aşağıdaki formda **Style** özellikleri 1 yapılmış ve **Picture** özellikleri ile resim yüklenmiş iki **CheckBox** görülmektedir.



Properties

Alignment

0: ☐ Check1 İşaret solda

1: Check1 ☐ İşaret sağda

Value

İkisi kullanıcı tarafından değiştirilebilen üç değer alabilir:

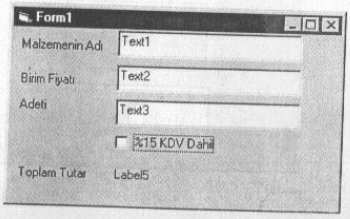
0 : İşaretsiz ☐

1 : İşaretli ☑

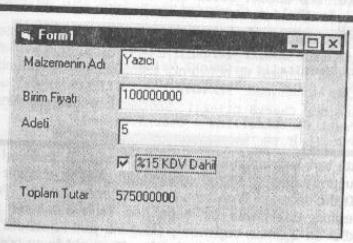
2 : Belirsiz ☑

İlk iki değer kullanıcı tarafından kontrol tıklanarak değiştirilebilir. Üçüncü değer ise belirsizlik durumunu gösterir ve program tarafından bu durum aktif hale getirilebilir. Belirsizlik, value'nin değerinin 2 olmasıyla ilgili değildir sadece o kontrolün ifade ettiği değerin belirsiz olduğunu gösterir. Örneğin CheckBoxu bir metnin Bold olup olmadığını temsil ettiğini düşünelim. Seçilen yazının tamamı Bold ise bu kutunun işaretli olması, değilse işaretsiz olması ama bir kısmı Bold bir kısmı değilse kutunun içeriğinin belirsiz olması gerekir.

ÖRNEK: Aşağıdaki formda yapılan satışa %15 KDV'nin dahil edilip edilmeyeceğini checkbox ile belirleyelim. Eğer CheckBox işaretli ise tutara %15 KDV eklensin.




```
'ÖRNEK: CheckBox, Value
Private Sub Check1_Click()
 hesapla
End Sub

Private Sub Text2_Change()
 hesapla
End Sub

Private Sub Text3_Change()
 hesapla
End Sub

Sub hesapla() 'hesaplamayı yapacak alt programımız
 Dim bf, adet, tutar
 bf = Val(Text2)
 adet = Val(Text3)
 If Check1.Value = 1 Then 'check1 işaretli ise
  tutar = bf * adet + bf * adet * 15 / 100
 Else
  tutar = bf * adet
 End If
 Label5 = tutar
End Sub
```

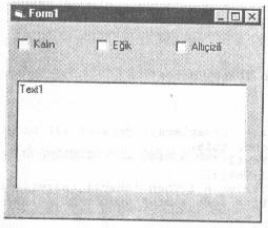


Events

Click( )

CheckBox'da click olayını boşluk tuşu, Kısayol tuşu ve mouse tıklaması meydana getirir. Check kutusunun işareti değiştirilirse bu olay meydana gelir. Aynı zamanda bu olayın meydana gelmesiyle birlikte **Value** özelliğinin değeri değişir.

ÖRNEK: Örneğin bir text kutusunun içeriğinin kalın, eğik ve altı çizili yapılması CheckBoxlar'la sağlayalım. Örneğimiz için formunuzun üzerine bir text kutusu ve üç tane CheckBox yerleştirin.



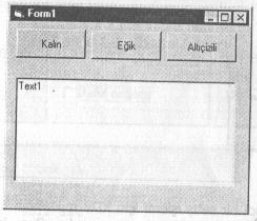

```
'ÖRNEK : CheckBox
Private  Sub Check1_Click ()
  Text1.FontBold = Check1.Value
End Sub

Private  Sub Check2_Click ()
  Text1.FontItalic = Check2.Value
End Sub

Private Sub Check3_Click()
    Text1.FontUnderline = Check3.Value
End Sub
```

Örnekte text kutusunun **FontItalic** ve **FontBold** özellikleri direkt olarak check kutusunun **Value** özelliğine eşitlenmiştir. Hatırlayacağınız gibi **FontBold** ve **FontItalic** değerleri **True** veya **False** değerlerini alabilir. **False** değeri 0 değerine eşittir, 0 olmayan bir değer ise **True** olduğunu gösterir. Dolayısıyla CheckBox'un 0 olması seçilmemiş olduğunu yani **FontBold**'un **False** olduğunu, 1 olması ise seçilmiş yani **FontBold**'un **True** olduğunu gösterir. Ancak bu işlemin tersi bir atama geçerli değildir. Yani ;

Check1.Value = Text1.FontItalic demek hataya sebep olur. Çünkü VB'de **False** 0 olarak gösterilirken **True** ise -1 olarak gösterilir. Mutlak değer fonksiyonu kullanılarak Check1.Value = Abs(Text1.FontItalic) problem halledilebilir.

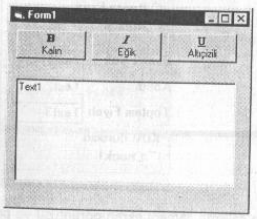
ÖRNEK: Yukarıdaki örneği CheckBox'ların resim özelliğini kullanarak yapalım. Kod olarak hiçbir fark olmayacak, sadece checkbox'ların üzerine birer resim yerleştirerek daha güzel bir görünüm oluşturacağız.

Bunu yapmak için her üç CheckBoxun'da **Style** özelliğini properties penceresinden 1 yapın. CheckBox'larımız aşağıdaki gibi birer komut düğmesi görünümüne girecektir. Ancak işlev olarak CheckBox görevi görecekler. Yani tıklandığında basılı kalacak, tekrar tıklandığında normale dönecekler.

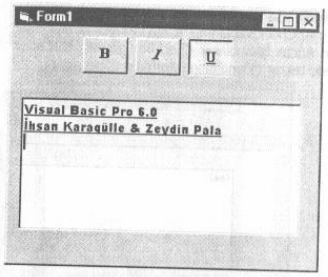


Şimdi bunların üzerine uygun birer resim yerleştirelim. Gerekli resimler GRAPHICS\BITMAPS\TLBR_W95 dizini altında bulunur.

Kalın düğmesini seçin ve properties penceresinden **Picture** özelliğini çift tıklayın. Açılan pencereden HICS\BITMAPS\TLBR_W95 dizinine gidin. Bu dizinde bulunan **BLD** isimli dosya kalın düğmesi için uygun bir resimdir. Diğer iki düğme için de aynı yöntemle **ITL** ve **UNDRLN** dosyalarını seçin. Böylece CheckBox'larımız aşağıdaki gibi olacaktır.



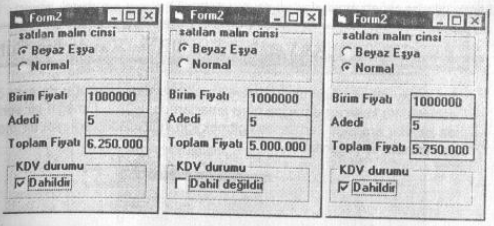
Son olarak da CheckBox'ların **Caption** özelliklerini silin ve içindeki resimlere uygun bir boyutta küçültün. Böylece CheckBox'larla araç çubuklarındaki düğmelerden elde etmiş olacağız.



ÖRNEK: Şimdi bir satış işlemi için KDV miktarının toplam satışa dahil edilip edilmeyeceğini belirleyen bir program yapalım. Bunu için aşağıdaki form tasarımını kullanacağız:

Aşağıdaki formu hazırlayalım. Burada option1 ve option2 kontrolleri KDV miktarının eşyaya göre değişimini belirler. Check1 kutusu işaretli ise KDV'li fiyat Text3 kutusuna, değilse KDVsiz fiyat text3 kutusuna aktarılıyor.

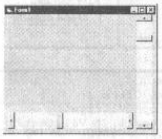
Programımızın çalışma zamanı ekran görüntüsü ve program kodu aşağıdaki gibi olacaktır:




```
'ÖRNEK : CheckBox
Private Sub Check1_Click ()
  Dim topfiyat, topKDVlifiyat
  topfiyat = Val(text1) * Val(text2)
  'Eğer check1 işaretli ise
  If check1.Value = 1 Then
    'Beyaz eşya seçili ise KDV %25
    If option1.Value = True Then
       check1.Caption = "Dahildir"
       topKDVlifiyat = topfiyat + topfiyat * 25 / 100
       text3 = topKDVlifiyat
    'Beyaz eşya seçili değil ise normal seçilidir KDV %15
    Else
       check1.Caption = "Dahildir"
       topKDVlifiyat = topfiyat + topfiyat * 15 / 100
       text3 = topKDVlifiyat
    End If
  'Check1 seçili değilse KDV yok
  Else
    check1.Caption = "Dahil değildir"
    text3 = topfiyat
  End If
  text3 = Format(text3, "###,###")
End Sub
```

Horizontal&Vertical ScrollBar (Kaydırma Çubuğu)

Biri yatay diğeri dikey olmak üzere her iki kaydırma çubuğu da aynı özellikle- sahiptir. Ve kullanım amacına göre amaçları da değişir. Örneğin bir veri taban da kayıtlar arasında ileri ve geri gitmek için kullanılabileceği gibi bir metinde ve ya resimde de kullanılabilir.

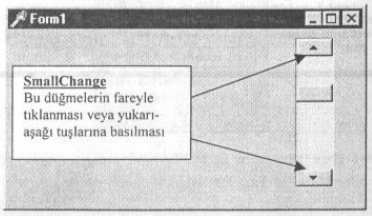


Properties

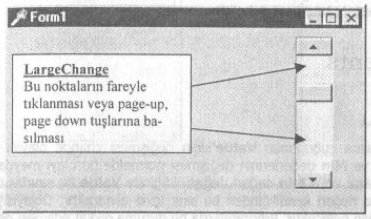
Value

Kaydırma çubuğunun temsil ettiği değeri gösterir, bu değer max ile min arasın bir sayıdır ve kaydırma çubuğu üzerinde değişik şekillerde değiştirilebilir.

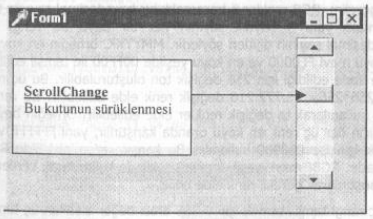
- Kaydırma çubuğunun uçlarındaki iki ok ile azaltılıp, çoğaltılabilir. Bu o **SmallChange** olarak adlandırılır.



- Direkt kaydırma çubuğu üzerine mouse ile tıklanarak değer artırılıp azaltıla- bilir. Bu olay **LargeChange** olarak adlandırılır.



- Kaydırma çubuğunun üzerindeki hareket eden kutucuk mouse ile istenen konuma getirilerek değer azaltılıp, çoğaltılabilir. Bu olayda **ScrollChange** olarak adlandırılabilir.



Max, Min

Kaydırma çubuğunun alabileceği (temsil edeceği) maksimum ve minimum de- ğerdir.

LargeChange

Kaydırma çubuğu üzerine tıklanması durumunda **Scroll.Value**'nin nekadarlık bir değişime tabi tutulacağını belirler.

SmallChange

Kaydırma çubuğunun iki kenarındaki oklarla **Scroll.Value**'nin ne kadarlık bir de- ğişime tabi tutulacağını belirler.

Events

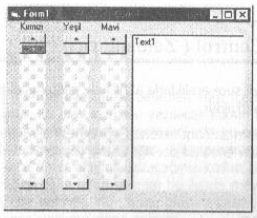
Change( )

Kaydırma çubuğunun **Value**'sinin değişmesi change olayını meydana getirir. **Max** ve **Min** değerlerinin değişmesi normalde bu olayı meydana getirmez. An- cak **Max** veya **Min** değeri değiştirildiğinde **Value** bu sınırlar dışında kalıyor ise **Value** değeri kendiliğinden bu sınır içine alınacaktır. Dolayısıyla **Change** olay meydan gelecektir. Programlarda bu duruma dikkat edilmesi gerekir.

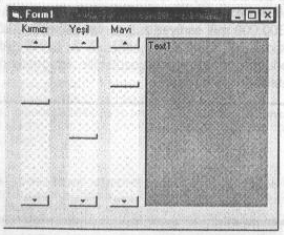
ÖRNEK: Örnek olarak kullanıcının kırmızı, yeşil ve mavi renklerin karışımından oluşan 16 milyon renkten birini seçme imkanı verecek bir program yazalım.

Önce RGB renkler (Red-Green-Blue,Kırmızı-Mavi-Yeşil) renkler nasıl temsil edili onu görelim. RGB renkler 6 basamaklı bir hexadesimal sayı ile temsil edilir. Bu 6 basamaklı yani 3 baytlık sayının her bir baytı bir rengi temsil eder. Bu hexadesimal sayının dijitleri şöyledir. MMYYKK, örneğin en koyu kırmızı 0000FF, en koyu mavi FF0000 ve en koyu yeşilde 00FF00 ile temsil edilir. Her bir renk iki dijitle ifade edildiği için 256 değişik ton oluşturulabilir. Bu üç rengin karışımı da 256*256*256= 16.777.216 değişik renk elde edilebilir. Bu renkler değişik ton- larda kullanılarak ta değişik renkler elde edilebilir. Örneğin beyaz rengi elde et- mek için her üç renk en koyu oranda karıştırılır, yani FFFFFF verilir. Siyahı elde etmek için ise 000000 kullanılır. Bu karışım işlemini VB'de RGB komutu yap- maktadır. RGB(mavi, yeşil, kırmızı) şeklinde kullanılarak verilen oranlarda renk- ler karıştırılarak gerekli renk elde edilir.

Bu üç rengi temsil etmek için 0-255 arası değer alabilen üç tane **ScrollBar** yer- leştirelim. Bunların her biri bir temel rengi temsil etsin (Red-Kırmızı, Green-Yeşil, Blue-Mavi), bunun için de **ScrollBar**'ların **Max** değerlerini 255 yapalım.




```
'ÖRNEK : Vscroll
Private Sub Form_Load()
 VScroll1.Max = 255
 VScroll2.Max = 255
 VScroll3.Max = 255
End Sub

Private Sub VScroll1_Change()
Text1.BackColor=RGB(VScroll1.Value,VScroll2.Value,VScroll3.Value)
End Sub

Private Sub VScroll2_Change()
Text1.BackColor=RGB(VScroll1.Value,VScroll2.Value,VScroll3.Value)
End Sub

Private Sub VScroll3_Change()
Text1.BackColor=RGB(VScroll1.Value,VScroll2.Value,VScroll3.Value)
End Sub
```



Timer Control (Zamanlayıcı)

Programda belirli süre aralıklarla aktif hale gelip belirli işleri yapabilmek amacıyla kullanılan bir kontroldür.

Bu kontrol tasarım zamanı ekranda görülmesine rağmen çalışma esnasında görülmez. Kontrolün enabled özelliği **False** yapılarak herhangi bir anda pasif hale getirilebilir.

Properties

Enabled

False verilerek Timer nesnesinin çalışması durdurulur. Tekrar **True** yapılıncaya kadar Timer olayı meydana gelmez.

Interval

Timer olayının gerçekleşeceği mili saniye cinsinden zaman periyodudur. Alabileceği değerler 1-65535 arasıdır. 0 değeri timer'i pasif hale getirir.(Timer'i pasif hale getirmek için Enabled özelliğini False yapmak daha mantıklıdır.)

Bu kontrolün tek dezavantajı aktif hale geleceği zaman periyodunu uzun verememenizdir. En büyük periyot 65535 mili saniye yani yaklaşık 65 saniyedir. Bu dezavantaj da **static** değişkenler kullanılarak belli bir ölçüde giderilebilir. Örneğin 5 dakikalık periyotlarla çalıştırmak istediğiniz bir kodunuz olduğunu düşünelim. Bu süre 300.000 mili saniyedir. Interval değerini 60.000'e ayarlayarak kontrolün her dakikada bir aktif hale gelmesini ve her 5 defa aktif hale geldiğinde istediğimiz kodu çalıştırarak sorun çözülebilir. Aşağıdaki örnek incelenerek geçici çözüm görülebilir.

```
'ÖRNEK : Timer Control
Private  Sub Timer1_Timer ()
 Static i
 i = i + 1
 If  i mod 5 = 0 Then
 '5 dakika doldu gerekli kod buraya yazılabilir.
 End If
End Sub
```


Events

Timer()

Timer kontrolünün interval özelliği ile belirtilen süre içerisinde periyodik olarak bu olay meydana gelir. Bu olay içerisine yazılacak kodun hızlı olması gerekir. Bu olay periyodik olarak sürekli meydana geleceği için bu olay içerisine yazılan kodun uzun olması Windows altında çalışan diğer programlarında yavaşlamasına sebep olacaktır. Ayrıca timer olayına yazacağınız kodun çalışması interval özelliğine verdiğiniz süreden daha uzun sürerse bu arada meydana gelmesi gereken timer olayları meydana gelmez. Yani timer olayına yazdığınız kodun tamamı çalıştırılmadan yeni timer olayı meydana gelmez.

ÖRNEK: Timer kontrolünü kullanarak formun başlığında saati gösterecek bir program yapalım. Bunun için **interval** özelliği 1000 (1 saniye) olan bir timer kontrolü form üzerine yerleştiriniz.

```
'ÖRNEK : Timer Control
Private  Sub Timer1_Timer ()
 form1.Caption = "Saat : " + Time
End Sub
```

ÖRNEK: Option buttondan yapılmış bir top programı. Program için forma bir option button ve bir timer yerleştirin.

```
'ÖRNEK : Timer Control
Private Sub Form_Load()
 Option1.Caption = ""
 Timer1.Interval = 50
End Sub

Private Sub Timer1_Timer()
 Static sx, sy, x, y

 If IsEmpty(sx) Then
  'İlk defa çalışıyorsa
  sx = 50 'x eksenindeki artım
  sy = -50 'y eksenindeki artım
  x = Option1.Left
  y = Option1.Top
 End If

 If y <= 0 Or y >= Form1.ScaleHeight - Option1.Height Then
 'Formun üstüne veya altına ulaşıldı ise artımı ters çevir
  sy = -sy 'artış yönünü ters çevir
  Beep
 End If
```

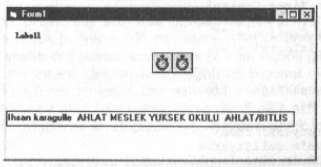


```
 If x <= 0 Or x >= Form1.ScaleWidth - Option1.Width Then
  'Formun soluna veya sağına ulaşıldı ise artımı ters çevir.
  sx = -sx 'artış yönünü ters çevir
  Beep
 End If
 x = x + sx
 y = y + sy
 Option1.Left = x
 Option1.Top = y
End Sub
```

ÖRNEK:Şimdi de Timer kullanarak bir text kutusuna yanıp sönme efekti verelim. Bunun için formun üzerine **Interval** özelliği 100 olan bir timer ve bir text kutusu yerleştirin. Timer olayında yazı rengi ile zemin rengini sürekli değiştirerek yanıp sönme efekti vereceğiz.

```
'ÖRNEK : Timer Control
Private  Sub Timer1_Timer ()
  Dim c
  c = Text1.BackColor
  Text1.BackColor = Text1.ForeColor
  Text1.ForeColor = c
End Sub
```

ÖRNEK: Ve bir animasyon programı. Programımız için aşağıdaki formu oluşturun. Bir Label, iki Timer ve bir Text kutusu yerleştirin. Label1'in **Index** özelliğine 0 verin ve Text kutusunun içeriğine animasyonunu istediğiniz yazıyı girin.

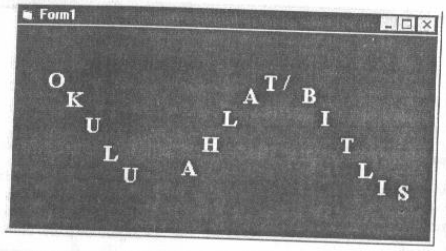


Text kutusundaki metni dairesel bir efekt vererek label kontrolünün kopyalarında göstereceğiz.

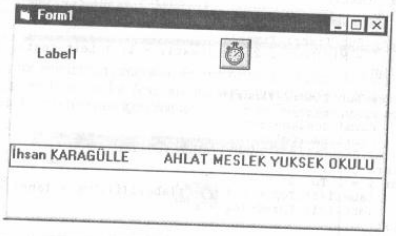
```
'ÖRNEK : Timer Control
Private Sub Form_Load()
  timer1.Interval = 150
  timer2.Interval = 5000
  Dim i, ds, y
```



```
  label1(0).AutoSize = True
  label1(0).FontSize = 15
  label1(0) = "M"
  label1(0).AutoSize = False
  label1(0).BackStyle = 0
  ds = 15
  Show
  label1(0) = ""
  label1(0).Top = 1200
  label1(0).Visible = True
  For i = 1 To ds + 4
    Load label1(i)'yeni labeller oluştur
    've dairesel olarak yerleştir
    y = Cos(i / 2) * Sqr(562500 - ((i) * 10) ^ 2) + 1200
    label1(i).Move label1(i - 1).Left + label1(i - 1).Width, y
    label1(i).Visible = True
  Next
  timer1.Enabled = True
End Sub

Private Sub Text1_Change()
  yaz (text1)
End Sub

Private Sub Timer1_Timer()
  text1 = Mid(text1, 2, Len(text1) - 1) + Left(text1, 1)
End Sub

Private Sub Timer2_Timer()
  Dim i, y, r
  y = Form1.ScaleHeight / 2
  r = Form1.BackColor
  Form1.BackColor = label1(0).ForeColor
  'ters çevir
  For i = 0 To 19
      label1(i).Top = 2 * y - (label1(i).Top + label1(i).Height)
      label1(i).ForeColor = r
  Next
End Sub

Private Sub yaz(text As String)
  Dim i, a, s
  If Len(text) > 19 Then
    s = 19
  Else
    s = Len(text) - 1
  End If
  For i = 0 To s
    a = Mid(text, i + 1, 1)
    label1(i) = a
  Next
End Sub
```



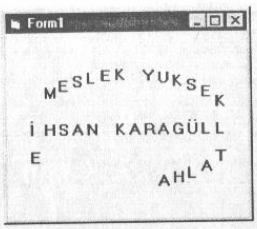
ÖRNEK: Yine bir yazı animasyonu. Bu animasyon ise dairesel bir harekete sahip. Yine örneğimiz için aşağıdaki formu oluşturup Label1'in **Index** özelliğine 0 verin.




```
'ÖRNEK : Timer Control
Private Sub Form_Load()
    timer1.Interval = 150
    Dim i, ds, y
    label1(0).AutoSize = True
    label1(0).FontSize = 10
    label1(0) = "M"
    label1(0).AutoSize = False
    label1(0).BackStyle = 0
  ds = 15
  Show
  label1(0) = ""
  label1(0).Top = 1200
  label1(0).Visible = True
```



```
  For i = 1 To 13
    Load label1(i)'yeni labeller oluştur
    label1(i).Move label1(i - 1).Left + label1(i - 1).Width,
label1(i - 1).Top
    label1(i).Visible = True
  Next
  i = 14
  Load label1(i)
  label1(i).Left = label1(0).Left
  y = Sqr(562500 - ((i - 13 - 7.5) * 100) ^ 2) + 1200
  label1(i).Move label1(0).Left, y
  label1(i).Visible = True
  For i = 15 To ds + 12
    Load label1(i)
    y = Sqr(562500 - ((i - 13 - 7.5) * 100) ^ 2) + 1200
    label1(i).Move label1(i - 1).Left + label1(i - 1).Width, y
    label1(i).Visible = True
  Next
  i = ds
  Load label1(i + 13)
  y = 1200 - Sqr(562500 - ((i - 7.5 - ds + 1) * 100) ^ 2)
  label1(i + 13).Move label1(0).Left, y
  label1(i + 13).Visible = True
  For i = ds + 1 + 13 To 2 * ds - 2 + 13
    Load label1(i)
    y = 1200 - Sqr(562500 - ((i - 13 - 7.5 - ds + 1) * 100) ^ 2)
    label1(i).Move label1(i - 1).Left + label1(i - 1).Width, y
    label1(i).Visible = True
  Next
End Sub

Private Sub Text1_Change()
  yaz (text1)
End Sub

Private Sub Timer1_Timer()
  text1 = Mid(text1, 2, Len(text1) - 1) + Left(text1, 1)
End Sub

Private Sub yaz(text As String)
  Dim i, a, s
  If Len(text) > 41 Then
    s = 41
  Else
    s = Len(text) - 1
  End If
  For i = 0 To s
    a = Mid(text, i + 1, 1)
    label1(i) = a
  Next
End Sub
```

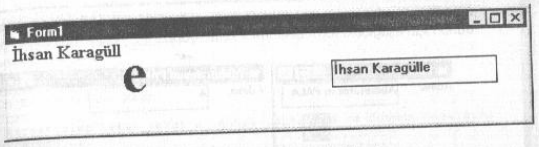



ÖRNEK: Şimdi de etkileyici editörler oluşturmak için kullanılabilecek bir programı yine Timer aracılığıyla yapalım.

Örneğimiz için form üzerine **Index** özelliği 0 olan bir Label ve **Interval** özelliği 10 olan bir Timer yerleştirin. Ayrıca kullanıcının metin girebilmesi için birde Text kutusu koyun.

```
'ÖRNEK : Timer Control
Private Sub Form_Load()
  Load label1(1)'yeni bir label daha oluştur
  label1(1).Visible = True
  'öncekinin yanına yerleştir
  label1(1).Left = label1(0).Left + label1(0).Width
  label1(0) = ""
  text1 = ""
End Sub
Private Sub Text1_Change()
  If Len(text1) = 0 Then Exit Sub
  label1(0) = Left(text1, Len(text1) - 1)
  label1(1) = Right(text1, 1)
  label1(0).FontSize = 12
  label1(1).Left = label1(0).Left + label1(0).Width
End Sub
Private Sub Timer1_Timer()
  Static f, i, j
  If IsEmpty(f) Then
    f = label1(0).FontSize
    i = 5
  End If
  If f > 50 Or f < 5 Then i = -i
  f = f + i
  label1(1).FontSize = f
End Sub
```

Programı çalıştırırsanız, siz yazarken en son yazdığınız harfin sürekli büyüyüp küçüldüğünü ve etkileyici bir efekt oluştuğunu göreceksiniz.



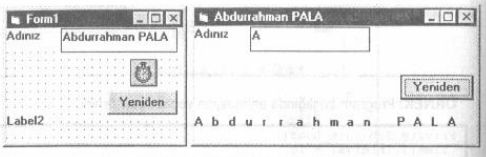
ÖRNEK: Program başlığında animasyon yapacak bir örnek:

```
Private Sub Form_Load()
 Timer1.Interval = 10
 Caption = "Bizim Program"
End Sub

Private Sub Timer1_Timer()
 Static t, x, artim
 If IsEmpty(t) Then
  t = "Version 1.0 İhsan Karagülle"
  artim = -1
 End If
 'yazının sonuna veya başına gelince artımı ters çevir
 If x = Len(t) Then artim = -artim
 If x <= 0 Then artim = -artim
 x = x + artim
 'Artım pozitifse ileri, negatifse geri doğru gel
 If artim > 0 Then Caption = "Bizim Program " & Left(t, x)
 If artim < 0 Then Caption = "Bizim Program " & Right(t, x)
End Sub
```

ÖRNEK: Yine bir yazı animasyonu örneği:

```
Private Sub Form_Load()
 Timer1.Interval = 10
 Label1.AutoSize = True
 Label1.FontName = "Courier New"
End Sub

Private Sub Timer1_Timer()
 Static t, x, artim
 If IsEmpty(t) Then
  t = "Bizim Program Version 1.0 İhsan Karagülle"
  artim = -1
 End If
 If (2 * x) >= Len(t) - 1 Then artim = -artim
 If x <= 0 Then artim = -artim
 x = x + artim
 Label1.Caption = Left(t, x)+Space(Len(t) - 2 * x) + Right(t, x)
End Sub
```

ÖRNEK: Şimdi karakterleri teker teker aşağıya düşüren bir program yapalım. Bunun için aşağıdaki form tasarımını kullanıyoruz:



Text1 kutusuna girilen isimin karakterleri kadar label elemanı oluşturulup, text1 deki her bir harf teker teker alınarak label başlıklarına ve formun başlığına aktarılıyor. Yeniden düğmesine basıldığı zaman aynı işlem tekrarlanıyor. Programın kodu aşağıdaki gibidir:

```
'ÖRNEK : Timer
Option Explicit
Dim adl, ad
Private Sub konrololustur ()
Dim i
caption = ""
ad = text1
'Girilen ismin uzunluğunun bir eksiği kadar label oluştur
For i = 1 To Len(ad) - 1
Load label2(i)
'Labelleri yanyana diz
label2(i).Left = label2(i - 1).Left + label2(i - 1).Width
label2(i).Visible = True
Next
End Sub

Private Sub Command1_Click ()
Dim i
On Error Resume Next
'Label kontrollerini hafızadan at
For i = 1 To Len(caption) - 1
Unload label2(i)
Next
text1 = caption
'Label kontrollerini yeniden oluştur
konrololustur
timer1.Enabled = True
'Timer olayını çağır
Timer1_Timer
End Sub
Private Sub Form_Load ()
'Kontrololustur alt programını çağır
```



```
konrololustur
End Sub
Private Sub Text1_Change ()
adl = text1
End Sub
Private Sub Timer1_Timer ()
On Error Resume Next
Static i
'karakterler teker teker alınarak labellere ve formun başlığına
aktarılıyor.
i = (i + 1) Mod (Len(ad) + 1)
adl = Mid(ad, i, Len(text1))
text1 = adl
label2(i - 1) = Mid(ad, i, 1)
caption = caption + label2(i - 1)
If i = Len(ad) Then timer1.Enabled = False
End Sub
```

ÖRNEK: Yılan gibi kıvrım kıvrım kıvranan karakter dizisine ne dersiniz? Evet bu program için form üzerine 2 adet timer, command1 ve index özelliği 0 olan bir label yerleştirin.

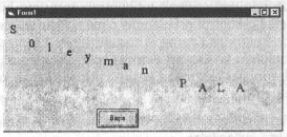
```
'ÖRNEK : Timer
Option Explicit
Dim k, i, j, aduz

Private Sub kontrol_et2()
If j = 0 Then
        i = i - 1
        Label1(i).Top = Label1(i + 1).Top
        j = i - 1
        k = Label1(j + 1).Top + 120
      Else
        j = j - 1
    End If
End Sub

Private Sub kontrol_et()
If j = 0 Then
        i = i - 1
        Label1(i).Top = Label1(i + 1).Top
        j = i - 1
        k = Label1(j + 1).Top - 120
      Else
        j = j - 1
    End If
End Sub

Private Sub Command1_Click()
Dim ad
```



```
ad = InputBox("Adınızı Giriniz: ", "Adi girişi", "Süleyman PALA",
0, 0)
aduz = Len(ad)
Label1(0).Caption = Mid(ad, 1, 1)
Dim n
    For n = 1 To aduz
        Load Label1(n)
        Label1(n).Top = 120
        Label1(n).Left = Label1(n - 1).Left + Label1(n - 1).Width
        Label1(n).Visible = True
        Label1(n).Caption = Mid(ad, n + 1, 1)
        Label1(n).FontSize = 16
        Label1(n).FontName = "Times New Roman"
        Label1(n).BorderStyle = 0
        Label1(n).BackStyle = 0
    Next
    i = aduz
    j = aduz - 1
   Timer2.Enabled = 0
   Timer1.Enabled = 1
   Label1(aduz).Top = 360
End Sub

Private Sub Timer1_Timer()
    Timer2.Enabled = 0
    If j = aduz Then
        k = Label1(aduz).Top + 120
     Else
       If j <> 0 Then
           k = Label1(j + 1).Top - 120
         Else
           kontrol_et
       End If
    End If
    If k = 120 Then
        j = aduz
        Exit Sub
     Else
        Label1(j).Top = k
    End If
    kontrol_et
    If i = 0 Then
        i = aduz
        j = aduz - 1
        Label1(aduz).Top = 1560
        Timer2.Enabled = 1
    End If
End Sub

Private Sub Timer2_Timer()
    Timer1.Enabled = 0
    If j = aduz Then
        k = Label1(aduz).Top - 120
```



```
      Else
        If j <> 0 Then
            k = Label1(j + 1).Top + 120
         Else
           kontrol_et2
        End If
    End If
    If k = 1800 Then
        j = aduz
        Exit Sub
     Else
        Label1(j).Top = k
    End If
    kontrol_et2
    If i = 0 Then
        i = aduz
        j = aduz - 1
        Label1(aduz).Top = 360
        Timer1.Enabled = 1
    End If
End Sub
```



Frame Control (Çerçeve)

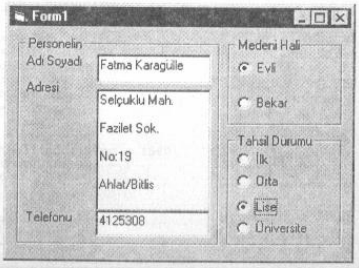
Bu kontrol tek başına değil, diğer kontrolleri gruplamak için kullanılır. Kontrolleri bu kontrol ile gruplamanın bir çok avantajı vardır.

Bu çerçeveler içine konan kontroller, çerçeveye bağımlıdırlar ve konumları bu çerçeve dışına taşamaz. Özellikle birkaç kontrolü birden görünür veya görünmez yapmak için hepsinin **Visible** özelliği tek tek değiştirmek yerine çerçevenin **Visible** özelliği değiştirilerek çerçeve içindeki tüm kontroller aynı anda görünmez yapılabilir. Aynı durum taşıma için de geçerlidir her birini tek tek taşımak yerine, çerçeve taşınır. Çerçevelerin buna benzer birçok faydaları vardır. Özellikle **OptionButton**'ların kullanılmasında çerçeve kullanmak zorunlu hale gelebilir.

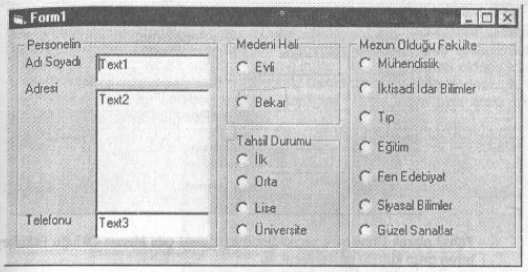
Frame'lerle gruplanan kontrollerin koordinatları artık forma göre değil grup kontrolünün sol üst köşesine göre belirlenir.

- Form üzerine yerleştirilmiş bir kontrolü taşıyarak bir **Frame** kontrolü üzerine getirmekle o kontrol gruplanmış olmaz. **Frame** kontrolü içerisine bir kontrol yerleştirirken önce **Frame** kutusunu seçin. Eğer zaten form üzerine yerleştirilmiş bulunan kontrolleri **Frame** kontrolü içine almak istiyorsanız o zaman form üzerindeki kontrolleri kesin (cut) ve **Frame** kontrolünü seçtikten sonra buraya yapıştırın (paste).

Frame'ler aşağıda olduğu gibi **OptionButton** (Seçenek Düğmesi) kontrollerini gruplamak için kullanılırlar.



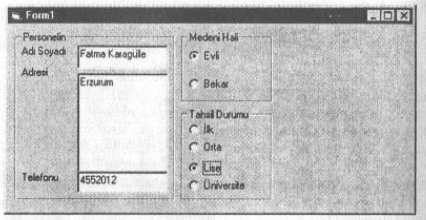
Ayrıca **Frame**, kontrolleri sadece bazı şartlarda gösterilmesi gereken kontrolleri de bir arada tutarak bunların kolayca gizlenip/gösterilmesini sağlarlar.



Örneğin yukarıdaki form dizaynında Mezun Olduğu Fakülte'yi temsil eden **OptionButton** düğmelerinin sadece Tahsil=Üniversite olması durumunda gösterilmesi gerekir. Mezun Olduğu Fakülteyi belirten yedi tane seçenek düğmesini gizleyip, göstermek için sadece onun içinde bulunduğu **Frame4** kontrolünü gizleyip göstermek yeterlidir. Aşağıdaki kod yazılırsa, sadece tahsili üniversite olanlar için mezun olduğu fakülteyi gösteren seçenek düğmeleri görülecektir.

```
'Örnek:Frame
Private Sub Form_Load()
 Frame4.Visible = False
End Sub

Private Sub Option3_Click() 'ilk
 Frame4.Visible = False
End Sub

Private Sub Option4_Click() 'orta
 Frame4.Visible = False
End Sub

Private Sub Option5_Click() 'lise
 Frame4.Visible = False
End Sub

Private Sub Option6_Click() 'üniversite
 'tahsil üniversite ise fakültesini göster
 Frame4.Visible = True
End Sub
```

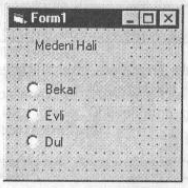

Yukarıdaki formda tahsil olarak Lise veya daha düşük bir seçenek seçildiğinde aşağıdaki gibi Mezun Olduğu Fakülteyi gösteren grup kutusu gizlenecektir.



Tahsil olarak Üniversite seçildiğinde aşağıdaki gibi Mezun Olduğu Fakülteyi gösteren grup kutusu gösterilecektir.

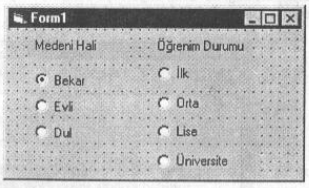
Option Button Control (Seçenek Düğmesi)

OptionButton kontrolü CheckBox'dan farklı olarak bir kaç seçenekten sadece birini seçme imkanı veren bir kontroldür. Bu kontrolün tek başına kullanılması anlamsızdır. Bir kaç seçenekten birini seçme imkanı veren bir kontrol olduğu için en az iki tane birlikte kullanılmalıdır. Gruptaki **OptionButton** düğmelerinden biri seçildiğinde diğeri kendiliğinde seçilmiş özelliğini kaldırır. Yani aynı anda bir grupta iki tane işaretli düğme bulunmaz.



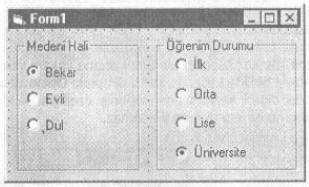
Yukarıdaki formda düğmelerden biri seçildiğinde diğerinin seçilmişliği kendiliğinden kalkacaktır.

Öğrenim durumunu da aynı formda düğmelerle göstermek istersek:



Bu durumda kullanıcı medeni hal ve öğrenim durumunu birlikte seçemeyecektir. Çünkü yedi tane düğmeden sadece biri seçilebilir. Medeni hali seçtiğinde öğrenim durumunun seçilmişliği kalkacak, öğrenim durumunu seçtiğinde de diğeri kalkacaktır. OptionButton'ların bu özelliğinden dolayı bunların bir gruplayıcı kont-

rolün içinde kullanılma mecburiyeti vardır. Bu iş için Frame veya PictureBox kontrolü kullanılabilir.



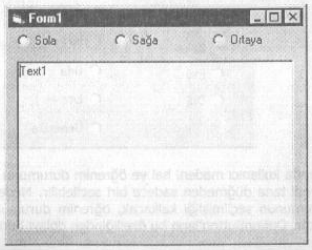
Bu şekilde düğmeler Frame'lere yerleştirildikten sonra her iki grup kendi arasında bağımsız olarak hareket edebilecektir.

Properties

Value

True olması option düğmenin seçilmiş olduğunu gösterir.

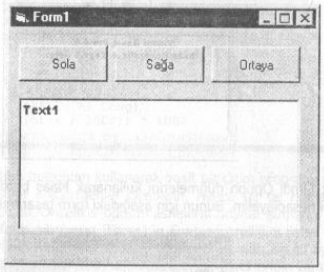
ÖRNEK: Örnek olarak bir Text kutusu içindeki yazının alignment'ini üç tane OptionButton aracılığı ile sola, sağa veya ortaya yerleştirecek bir program yapalım. Örneğimiz için aşağıdaki formu oluşturun.




```
'ÖRNEK: Option Button
Private Sub Option1_Click()
 Text1.Alignment = 0 'sola
End Sub

Private Sub Option2_Click()
 Text1.Alignment = 1 'sağa
End Sub

Private Sub Option3_Click()
 Text1.Alignment = 2 'ortaya
End Sub
```

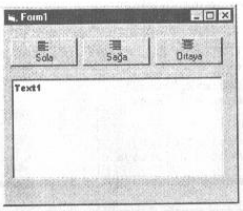
ÖRNEK: Yukarıdaki örneği OptionButton'ların resim özelliklerini kullanarak birer resimli düğme olarak tasarlayabiliriz.

Bunu yapmak için her üç OptionButton'unda **Style** özelliğini properties penceresinden 1 yapın. OptionButton'larımız aşağıdaki gibi birer komut düğmesi görünümüne girecektir. Ancak işlev olarak OptionButton görevi görecekler. Yani gruptakilerden sadece biri seçili olabilecektir.

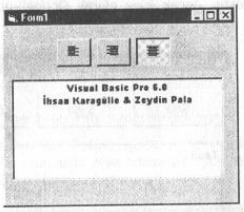


Şimdi bunların üzerine uygun birer resim yerleştirelim. Gerekli resimler GRAPHICS\BITMAPS\TLBR_W95 dizini altında bulunur.

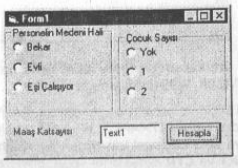
Sola düğmesini seçin ve properties penceresinden **Picture** özelliğini çift tıklayın. Açılan pencereden HICS\BITMAPS\TLBR_W95 dizinine gidin. Bu dizinde bulunan **LFT** isimli dosya Sola düğmesi için uygun bir resimdir. Diğer iki düğme için de aynı yöntemle **RT** ve **CTR** dosyalarını seçin. Böylece OptionButton'larımız aşağıdaki gibi olacaktır.



Son olarak da OptionButton'ların **Caption** özelliklerini silin ve içindeki resimlere uygun bir boyutta küçültün. Böylece OptionButton'larla araç çubuklarındaki düğmelerden elde etmiş olacağız.



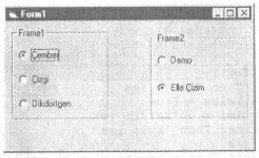
ÖRNEK: Şimdi Option düğmelerini kullanarak Maaş bordrosunda bulunan aile yardımını hesaplayalım. Bunun için aşağıdaki form tasarımını kullanacağız:



Burada personelin medeni halini option1, option2, option3 düğmeleri belirlerken, Çocuk sayısını da option4, option5, option5 düğmeleri belirler.


```
'ÖRNEK : Frame ve OptionButton
Private Sub Command1_Click()
 Dim aileyardimi, mks As Long
 Dim sonuc
 mks = Val(Text1)
 If Option1 Then 'Bekar ise
    Option4.Value = True
    aileyardimi = 0 'Yardım yok
 ElseIf Option2 Then 'Evli ise
    'çocuk yok
    If Option4 Then aileyardimi = mks * 300
    '1 çocuklu
    If Option5 Then aileyardimi = mks * 350
    '2 ve da fazla çocuklu
    If Option6 Then aileyardimi = mks * 450
 ElseIf Option3 Then 'Eşi çalışıyor ise
    'çocuk yok
    If Option4 Then aileyardimi = mks * 0
    '1 çocuklu
    If Option5 Then aileyardimi = mks * 50
    '2 ve da fazla çocuklu
    If Option6 Then aileyardimi = mks * 100
 End If
 sonuc = "Aile yardımı: "
 sonuc = sonuc & Format(BinlikAt(Val(aileyardimi)), "###,###")
 sonuc = sonuc & " TL"
 Form1.Caption = sonuc
End Sub


Function BinlikAt(x As Long)
 BinlikAt = (Int(x / 1000)) * 1000
End Function
```

ÖRNEK: Option buttonları kullanarak basit bir çizim programı yapalım. Program çember, çizgi ve dikdörtgenler çizebilsin. Ayrıca bir de rasgele çizim yapacak demo kısmı olsun. Öncelikle bütün nesnelerin orijinal isimlerini kullanarak aşağıdaki gibi bir form oluşturun. Timer1'in Enabled özelliğini False ve Interval değerini de 100 yapın. Frame'ler içindeki option buttonları ise iki ayrı dizi olarak düzenleyin. Yani Çember, Çizgi ve Dikdörtgen seçeneklerinin isimleri aynı Index özellikleri de sırasıyla 0,1 ve 2 olsun. Önce Index özelliklerine sırasıyla 0,1,2 verin ve sonra da isimlerini (name özelliği) Option1 yapın. Demo ve Elle çizim seçeneklerinin de isimleri aynı ve Index'leri 0 ve 1 olsun. Bunun için de önce Indexlerini sırasıyla 0,1 yapın ve isimlerini Option2 olarak değiştirin.



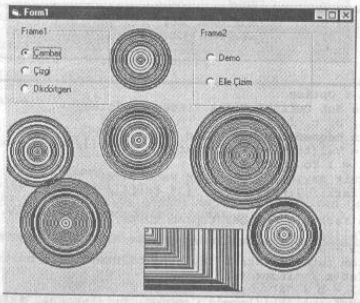
Elle çizim seçeneği aktifken farenin sol tuşuna basılı tutarak seçtiğimiz şekli çizeceğiz. Demo seçeneği aktif iken ise program timer'ı kullanarak kendi çizim yapacak.

```
'ÖRNEK : Frame ve OptionButton
option explicit
Dim mx, my, r, x1, y1

Private  Sub Option2_Click (Index As Integer)
 cls
 'demo modu seçili ise timer aktif hale getirilir.
 timer1.Enabled = option2(0).Value
End Sub

Private  Sub Timer1_Timer ()
 Dim md As Integer
 md = Rnd * 2
 option1(md).Value = True 'rasgele bir çizim modu seç
 Select Case md
  Case 0:Circle (Rnd * 5000, Rnd * 5000), Rnd * 5000
  Case 1:Line -(Rnd * 5000, Rnd * 5000)
  Case 2:Line(Rnd * 5000,Rnd * 5000)-(Rnd * 5000,Rnd * 5000), , B
 End Select
End Sub


Private  Sub Form_MouseDown (button As Integer, Shift As Integer,
x As Single, y As Single)
 If button = 1 Then'sol tuş basılı ise çizime başlandı
  If option1(0).Value = True Then 'çember seçili ise
   'yarıçapı sıfır, merkez koordinatları tıklanan nokta
   r = 0
   mx = x
   my = y
  End If
  If option1(1).Value = True Then 'Çizgi işaretli ise
   Line (x, y)-(x, y) 'Tıklanan noktaya bir çizgi koy
  End If
  If option1(2).Value = True Then 'Dikdörtgen işaretli ise
   'Köşe koordinatlarını tıklanan nokta olarak belirle
   x1 = x
```

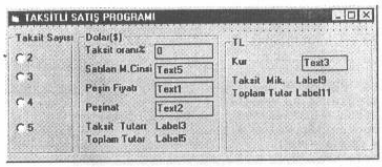


```
   y1 = y
  End If
 End If
End Sub

Private  Sub Form_MouseMove (button As Integer, Shift As Integer,
x As Single, y As Single)
 If button = 1 Then 'Sol tuş basılı ise
  If option1(0).Value = True Then
   r = Abs(x - mx)
   Circle (mx, my), r
  End If
  If option1(1).Value = True Then
   Line -(x, y)
  End If
  If option1(2).Value = True Then
   Line (x1, y1)-(x, y), , B
  End If
 End If
End Sub
```

Programda, farenin sol tuşuna basıldığı anda çizim yapılacak şeklin başlangıç koordinatları belirlenir. Çember ise bu koordinat merkezin, Çizim ise başlangıç noktasının, Dikdörtgen ise ilk köşenin koordinatlarıdır. Demo modu seçildi ise timer aktif hale getirilerek timer fonksiyonundaki rasgele çizim yapacak kod çalıştırılır.



ÖRNEK: Şimdi option kontrollerini kullanarak taksitli satış programını yapalım. Bunun için aşağıdaki form tasarımını kullanıyoruz:



Burada taksit sayısı option kontrolleri belirliyor. Bunlar dizi olarak (isimleri aynı, index özellikleri 0,1,2,3 olarak) tanımlanmıştır. Bunlardan herhangi birisi seçildiği zaman kutulara girilen veriler belirli işlemlere tabi tutularak taksitli satış sonuçları hem "TL" olarak hem de "$" olarak bulunuyor.

Programımızın çalışma zamanı görüntüsü ve program kodu aşağıdaki gibi olacaktır:


```
'ÖRNEK : Option
Private Sub Option1_Click (index As Integer)
 Dim a, b, c, fark
 'Peşin ödemeden sonra kalan fark
 fark = Val(text1) - Val(text2)
 For i = 1 To index + 2
  'taksit sayısıyla yansıyan vade farkı
  a = fark / (index + 2) * i * text4 / 100 + a
 Next
 'Vade farkıyla beraber toplam tutar
 a = Val(text1) + a
 label5 = a & "$"
 'toplam tutarın taksit sayısına bölünmesi ile bulunan miktar
 b = Int(a / (index + 2))
 label3 = b & "$"
 label9=Format(Val(label3.Caption)*Val(text3), "###,###") & "TL"
 label11=Format(Val(label5.Caption)*Val(text3), "###,###") & "TL"
End Sub
```


Shape Control(Şekil Kontrol elemanı)

Shape grafiksel bir kontrol elemanı olup dikdörtgen, kare, elips, çember, oval kare ve oval dikdörtgen şekillerini oluşturmaya yarar.

Properties

Shape

Shape kontrolünün çizeceği şekli belirler. 0-5 arası bir değer alabilir.

Shape	Şekil	Shape	Şekil
0	Dikdörtgen	3	Çember
1	Kare	4	Oval dikdörtgen
2	Elips	5	Oval kare

ÖRNEK: Aşağıda verilen örnekte Timer aracılığıyla her 1sn'de bir şekil çizdirilmektedir. Bu olay programa eklenen bir kodla sürekli tekrarlatılmaktadır. Programda dikey kaydırma çubuğu,shape ve timer kullanıldı.

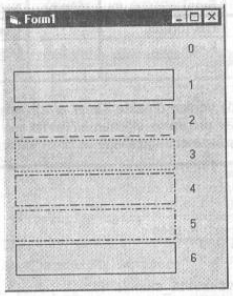
```
'ÖRNEK : Shape
Private Sub Form_Load ()
 shape1.BorderStyle = 1
 shape1.BorderWidth = 5'çizgi kalınlığı 5
 timer1.Interval = 1000
 VScroll1.Min = 0
 VScroll1.Max = 5
End Sub
Private  Sub VScroll1_Change ()
 shape1.Shape = VScroll1.Value
End Sub
Private  Sub Timer1_Timer ()
 VScroll1.Value = VScroll1.Value + 1
 'Tüm şekiller gösterildi ise gösterime baştan itibaren devam et
 If VScroll1.Value >= 5 Then VScroll1.Value = 0
End Sub
```

BorderStyle

Bu özellik nesnenin çerçeve biçimini belirler. Çerçeve biçimleri ve **BorderStyle** özelliğinin aldığı değerler aşağıdaki tabloda gösterilmiştir:

BorderStyle	Çerçeve Biçimi
0	Zemin rengiyle uyumlu, görülmez
1	Solid(Tam çerçeve)
2	Dash(Çizgi)
3	Dot(Nokta)
4	Dash dot (Çizgi,nokta)
5	Dash dot dot (Çizgi,nokta,nokta)
6	Inside solid(Şekil ile çerçeve kenarları çakışık şekilde)

Aşağıdaki şekilde farklı **BorderStyle** özellikleriyle çizilmiş dikdörtgenler görülmektedir.



BorderWidth

Bu özellik çerçeve kalınlığını ifade eder. 1 ile 8192 arasındaki değerleri alabilir. **BorderStyle** özelliğinin sadece **1-Solid** ve **6-İnside solid** biçimlerindeyken etkisi görülür.

FillColor

Şeklin iç boyama rengini belirler.

FillStyle

Şeklin içini boyamak için kullanılacak deseni belirler.

FillStyle	Örnek	Boyama şekli
0		Tam Dolu
1		Üzerinde bulunduğu nesnenin zemin rengi
2		Yatay Çizgili
3		Dikey Çizgili
4		Sola Eğik
5		Sağa Eğik
6		Kareli
7		Çapraz

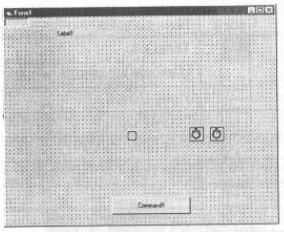
ÖRNEK: Bu stillerin tümünü göstermek için bir program yazalım. Bunun için form üzerine Index özelliği 0 olan bir Shape yerleştirin.

```
'ÖRNEK : FillStyle
Private Sub Form_Load ()
 Dim i
 shape1(0).Left = 100
 For i = 1 To 7
  Load shape1(i)'Yeni shape oluştur
  shape1(i).Left = 100
 'Bir önceki şeklin altına yerleştir.
  Shape1(i).Top = shape1(i - 1).Top + shape1(i - 1).Height + 20
  shape1(i).FillStyle = i
  shape1(i).Visible = True
 Next
End Sub
```

Örneği çalıştırdığınızda Form üzerine bir tane yerleştirmemize rağmen yedi tane FillStyle'leri farklı olan şekil görüntülenecektir. Bunun sebebi programdaki **Load** satırıdır. İleride daha detaylı göreceğimiz bu komut, ismi verilen nesneyle aynı özelliklerde yeni nesneler oluşturur. Bunu kullanabilmek için o nesnenin bir dizi olması gerekir. Form üzerindeki nesnenin index'ini 0 yapmamızın sebebi de budur. **Load Shape1(i)** satırıyla indexi i olan yeni shape nesneleri oluşturuluyor.

ÖRNEK: Shape kontrolünü bir top gibi kullanarak tek kişilik tenis oyunu yapalım. Örneğimiz için formunuzun üzerine şunları yerleştirin: Top olarak kullanmak için bir Shape, kutuları oluşturmak için bir PictureBox yerleştirin ve Index özelliğine 0 vererek dizi olmasını sağlayın. İki tane timer yerleştirin. Bunlardan birini topun hareketi için diğerini de kutuların rengini değiştirmek için kullanacağız.

Puanı göstermek için de bir Label yerleştirin. Ayrıca raket olarak kullanmak için bir Command düğmesi yerleştirin.

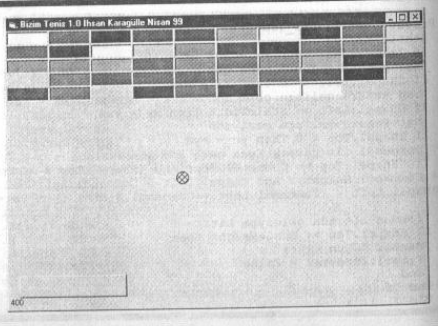



```
ÖRNEK: Shape
Option Explicit
Dim satirsayisi, sutunsayisi, kutusayisi, puan
Private Sub Command1_MouseMove(Button As Integer, Shift As
Integer, X As Single, Y As Single)
 Command1.Left = Command1.Left + X
End Sub

Private Sub Form_Load()
 Dim i, j, k
 Caption = "Bizim Tenis 1.0 İhsan Karagülle Nisan 99"
 Show
 ScaleMode = 3 'pixel
 KeyPreview = True
 satirsayisi = 5
 sutunsayisi = 10
 kutusayisi = satirsayisi * sutunsayisi
 Command1.Caption = ""
 Timer1.Interval = 10
 Timer2.Interval = 10
 Command1.Top = ScaleHeight - 50
 Label1.Top = ScaleHeight - Label1.Height
 Label1.Left = 0
 Label1 = "0"
 Shape1.Shape = 3 'çember
 Shape1.FillStyle = 7
 Shape1.Width = 18
 Shape1.Height = 18
 Picture1(0).Width = 60
 Picture1(0).Height = 20
 Width = Picture1(0).Width * sutunsayisi * Screen.TwipsPerPixelX
+ 30
```



```
 Picture1(0).Move 0, 0
 Picture1(0).BackColor = QBColor(1)
 For i = 1 To satirsayisi
  For j = 1 To sutunsayisi
   k = k + 1
   Load Picture1(k)
   Picture1(k).Left = Picture1(k - 1).Left + Picture1(k).Width
   Picture1(k).Top = (i - 1) * Picture1(k).Height
   Picture1(k).Visible = True
   Picture1(k).BackColor = QBColor(i)
  Next
  Picture1(k).Left = 0
 Next
End Sub

Sub Form_KeyDown(keycode As Integer, Shift As Integer)
 Select Case keycode
  Case vbKeyLeft: 'sol
   Command1.Left = Command1.Left - 15
   If Command1.Left <= 0 Then Command1.Left = 0
  Case vbKeyRight: 'sağ
   Command1.Left = Command1.Left + 15
   If Command1.Left >= Form1.ScaleWidth - Command1.Width Then
Command1.Left = Form1.ScaleWidth - Command1.Width
 End Select
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X
As Single, Y As Single)
 Command1.Left = X
End Sub

Private Sub Timer1_Timer()
 Static xa, ya
 If IsEmpty(xa) Then xa = 10: ya = 10
 'sol kenara gelince geri dön
 If Shape1.Left <= 0 Then xa = -xa
 'sağ kenara gelince geri dön
 If Shape1.Left >= ScaleWidth Then xa = -xa
 'üst kenara gelince geri dön
 If Shape1.Top < 0 Then ya = -ya
 'command1 ile çarpıştıysa geri dön
 If (Shape1.Top >= Command1.Top) And (Shape1.Top < (Command1.Top
+ Command1.Height)) And Shape1.Left > Command1.Left And
Shape1.Left < (Command1.Left + Command1.Width) Then ya = -ya

 'formun altında geldiyse bitir
 If Shape1.Top >= ScaleHeight Then
  MsgBox "Oyun bitti"
  Timer1.Interval = False
  End
 End If
 Dim i, j, k
```

```
'kutulardan birine çarpınca geri dön ve o kutuyu kaldır
k = -1
For i = 1 To sutunsayisi
 For j = 1 To satirsayisi
  k = k + 1
  If Picture1(k).Visible = True And Shape1.Left >=
Picture1(k).Left And Shape1.Left < (Picture1(k).Left +
Picture1(k).Width) And Shape1.Top >= Picture1(k).Top And
Shape1.Top <= (Picture1(k).Top + Picture1(k).Height) Then
    Picture1(k).Visible = False
    kutusayisi = kutusayisi - 1
    ya = -ya
    puan = puan + 100
    Label1 = puan
  End If
 Next
Next
Shape1.Left = Shape1.Left + xa
Shape1.Top = Shape1.Top + ya
If kutusayisi = 0 Then
 MsgBox ("Tebrikler")
 End
End If
End Sub

Private Sub Timer2_Timer()
 'renkleri değiş
 Picture1(Rnd * (satirsayisi * sutunsayisi - 1)).BackColor =
QBColor(Rnd * 15)
End Sub
```



Line(Çizgi Kontrol elemanı)

Bu kontrol elemanı form üzerine bir çizgi şeklinde alınır. Alınan bu çizgiyi yatay, dikey ve eğik bir biçimde göstermek, istenilen boyuta getirmek mümkündür. Bu kontrolün yerine Line (X1,Y1)-(X2,Y2) metodunu kullanmak bellek açısından daha faydalıdır.

Properties

X1,X2, Y1 ,Y2

X1 ve *Y1* noktaları çizgi kontrol elemanının başlangıç koordinatlarını, *X2* ve *Y2* ise bitiş koordinatlarını belirler. Yatay kontrolleri X1 ve X2, Dikey kontrolleri ise Y1 ve Y2 belirler.

PictureBox(Resim kutusu)

Bu kontrol elemanı Bitmap, Icon, Metafile, Jpeg ve Gif gibi resimleri görüntülemek için kullanılır. Ayrıca metodlar kullanılarak PictureBox içine çizimlerde yapılabilmektedir. Bu kontrolün bir özelliği de Frame kontrolü gibi diğer kontrolleri gruplayabilmesidir.

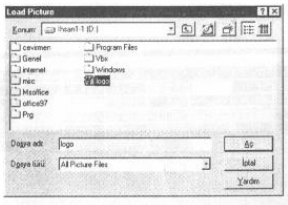
Properties

Picture

Bu özellik kullanılarak Picture kutusunun göstereceği resim belirlenir. Bu resim BMP, ICO, JPG, GIF veya WMF türünden olabilir.

Picture kutusuna resim yüklenmesi:

- Tasarım anında Picture özelliğinin properties penceresinden çift tıklanmasıyla açılan aşağıdaki pencereden yapılabilir.



- **Loadpicture** komutu ile dosyadan yüklenebilir.

```
picture1.Picture = LoadPicture ("\windows\yaprak.bmp")
```

- Başka bir nesnenin Picture özelliğinden aktarılabilir.

```
picture1.Picture = Picture2.Picture
```

- Clipboard'dan yapılabilir.

```
picture1.Picture = Clipboard.Getdata( )
```


ÖRNEK: Picture özelliğini kullanarak resim kutusuna dosyadan veya panodan resim yükleyecek ve kaydedecek bir program yapalım. Bunun için form üzerine bir Picturebox, bir DriveListBox, bir FileListBox , bir DirectoryListBox ve üç tanede komut düğmesi yerleştirerek aşağıdaki formu oluşturun.


```
'ÖRNEK : Picture
Private Sub Form_Load ()
 'Sadece resim dosyalarını göster
 File1.Pattern = "*.BMP;*.ICO;*.WMF;*.DIB;*.JPG;*.GIF"
 Picture1.AutoSize = True
End Sub

Private Sub Drive1_Change()
 'Sürücü değiştiğinde dizin/dosya gösteren kontrolleri değiştir
 ChDrive Drive1.Drive
 Dir1.Path = Drive1.Drive
End Sub

Private Sub Dir1_Change ()
 'Dizin değiştiğinde o dizindeki dosyaları göster
 File1.Path = Dir1.Path
 ChDir( File1.Path)
End Sub

Private  Sub File1_Click()
 On Local Error GoTo ResimHatası
 'Tıklanan dosyayı yükle
 Picture1.Picture = LoadPicture(File1.Filename)
Exit Sub

ResimHatası:
 MsgBox ("Resim Hatalı")
 Exit Sub
End Sub
```

```
Private Sub Command1_Click()
 Clipboard.Clear 'Panoyu sil
 Clipboard.SetData Picture1.Picture 'Resmi panoya kopyala
End Sub

Private Sub Command2_Click()
 Picture1.Picture = Clipboard.GetData 'Panodan resim al
End Sub

Private Sub Command3_Click()
 'Dosya adı sor ve o dosyaya kaydet
 SavePicture Picture1.Picture, InputBox("Dosya adı")
End Sub
```

Image

Bu özellik sadece okunabilir bir özelliktir. Ve Picture kontrolü içine methods'lar (print, line, circle gibi) kullanılarak yapılan yazım ve çizimleri temsil eder. Methods'lar kullanılarak yapılan yazım ve çizimlerin kaydedilebilmesi veya başka bir nesne içine alınabilmesi için bu özellik kullanılır. Bu özelliğin etkili olabilmesi için **AutoReDraw** özelliğinin **True** olması gerekir.

```
Picture1.Circle (0,0),150
Picture2.Picture=Picture1.Image
```

Yukarıdaki satırlar ile Picture1 içine bir daire çizilmekte ve bu çizim Picture2 nesnesi içerisinde de gösterilmesi sağlanmaktadır. (Picture1.AutoReDraw=True olmalıdır)

AutoRedraw

Bu özelliğin aldığı **True** veya **False** değerleri ile nesnenin kendini otomatik olarak yenilemesi yada yenilememesi sağlanır. Bu özellik sayesinde arka plana düşmüş bir form yada üzeri kapatılmış bir nesnenin üzerinin açılması ile nesnenin kendisini yeniden çizdirmesi sağlanır. Picture kutusunda methods'lar kullanılarak yapılmış yazım ve çizimler varsa PictureBox'un kendini yenileyebilmesi için bu özellik True olmalıdır. Aksi takdirde methods'larla yapılan işlemler doğru olarak görüntülenmeyecektir.

Ayrıca Line, Circle, Print gibi metodlarla oluşturulmuş çizimlerin yazdırılabilmesi, bir dosyaya kaydedilebilmesi gibi işlemler için de **AutoRedraw** özelliğinin **True** olması gerekir.

ScaleLeft, ScaleTop

ScaleLeft ve ScaleTop özellikleri Left ve Top özelliklerinden farklıdır. Left ve Top nesnenin form içindeki koordinatlarını belirlerken ScaleLeft ve ScaleTop çizimin yapılacağı bölgenin koordinatlarını belirler. Default olarak 0,0 dır, yani yapılacak çizimler nesnenin sol üst köşesi orijin olmak üzere yapılır. Pozitif değerler orijini sol yukarı taşırken, negatif değerler sağ aşağı taşır.

ScaleHeight, ScaleWidth

ScaleHeight ve ScaleWidth özellikleri çizimin yapılacağı bölgenin genişliğini belirler. Nesne içine yapılacak çizimler bu koordinatlar dışına taşamaz.

Bu özellikler nesne içinde picture özelliği ile yüklenen resmi etkilemez. Sadece nesnenin method'larıyla (Circle, Line, Print vb.) yapılmış yazım ve çizimleri etkiler.

CurrentX,CurrentY

Bu özellikler methods'lar kullanılarak yapılan çizim ve yazımlarda aktif pixelin koordinatları belirlenir.

Aşağıdaki kod 1500,2000 koordinatlarına Merhaba yazacaktır.

```
Private Sub Command1_Click()
    CurrentX = 1500
    CurrentY = 2000
    Print "Merhaba"
End Sub
```

Align

Bu özellik kullanılarak Picture kontrolünün sürekli olarak formun bir kenarında kalması sağlanabilir. Örneğin bir araç çubuğunun sürekli olarak formun en üstünde kalması istenir veya bir durum çubuğunun formun sürekli en altında kalması istenir. İşte bu özellik kullanılarak herhangi bir kod yazmaya gerek kalmadan kendisini sürekli aynı kenarda tutması sağlanabilir.

0-None	Yerleştirildiği koordinatlarda sabit kalır
1-Align Top	Sürekli üstte
2-Align Bottom	Sürekli altta
3-Align Left	Sürekli solda
4-Align Right	Sürekli sağda

Align özelliğine 0 olmayan bir değer verildiğinde, formun boyutlarının değişmesiyle kontrol kendini otomatik olarak boyutlandırır. Form boyutları büyüdüğünde kendi de ona uygun olarak büyür.

Events

Resize( )

Kontrolün boyutlarının değişmesi halinde bu olay meydana gelir. Picture boyutları iki şekilde değişir. Birincisi, programın herhangi bir yerinden kontrolün genişliğinin veya yüksekliğinin değiştirilmesiyle, ikincisi de PictureBox'un AutoSize özelliği True ise Picture kontrolünün içindeki resmin değişmesiyle PictureBox'un boyutları resmin boyutlarına ayarlanacağından *Resize* olayı meydana gelir.

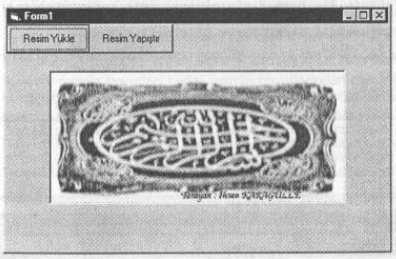
ÖRNEK: Örnek olarak BMP, ICO, DIB, WMF, GIF ve JPG resimlerini gösterecek bir program yapalım ve resim her zaman formun ortasında bulunsun. Bunun için form üzerine AutoSize özelliği True olan bir picture box , birinin Caption'u "Resim Yükle" ve diğerininki "Resim Yapıştır" olan iki Command Button yerleştirin. Ayrıca dosya yükleme işlemlerimiz için de bir CommonDialog kontrolü yerleştirin.

Resmin her zaman formun ortasında bulunmasını istediğimizden PictureBox'un Resize olayına resmi ortalayacak kodu yazmamız gerekir. PictureBox'un AutoSize özelliği True olduğundan resmin boyutları değiştiğinde Resize olayı meydana gelecek ve resim ortalanacaktır. Ancak Form'un boyutlarının değiştirilmesi durumunda resim ortada olmayacaktır. Formun boyutları değiştiğinde de resmin ortalanması için aynı kodu formun Resize olayına da yazmamız gerekir.

```
'ÖRNEK : Resize
Private Sub Command1_Click ()
  On Local Error GoTo iptal
  cmdialog1.Filter = "Bitmap| *.bmp| Icon| *.ico| Dib| *.dib|
Metafile| *.wmf| Jpeg| *.jpg| Gif| *.gif| Bütün Resim
Dosyaları|*.bmp;*.ico;*.dib;*.wmf;*.gif;*.jpg| Bütün
Dosyalar|*.*||"
  cmdialog1.Action = 1
  On Local Error GoTo hata
  picture1.Picture = LoadPicture(cmdialog1.Filename)
iptal:
   'Diyalog kutusunda iptal seçildi ise alt programdan çık
  Exit Sub
hata:
  'Resim yüklenirken hata oluştuysa hata mesajı ver ve çık
  MsgBox(cmdialog1.Filename+"/Bu dosya yüklenemiyor/" + Error)
  Exit Sub
End Sub
```

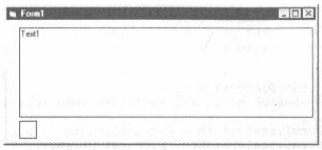


```
Private  Sub Command2_Click ()
 'Clipboard(pano) daki resmi resim kutusunda göster
  picture1.Picture = clipboard.GetData()
End Sub

Private  Sub Picture1_Resize ()
  'PictureBox'un boyutları değişirse resmi ortala
  Dim x, y
  x = (Form1.scalewidth - picture1.Width) / 2
  y = (Form1.scaleheight - picture1.Height) / 2
  picture1.Left = x
  picture1.Top = y
End Sub

Private  Sub Form_Resize ()
'Formun boyutları değişirse resmi ortala
  Dim x, y
  x = (Form1.scalewidth - picture1.Width) / 2
  y = (Form1.scaleheight - picture1.Height) / 2
  picture1.Left = x
  picture1.Top = y
End Sub
```



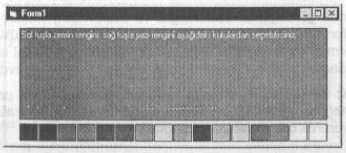
Programda gördüğünüz ScaleHeight ve ScaleWidth, Height ve Width'ten farklı olarak formun iç kısmının boyutlarıdır. Yani formun menüsü ve kenar çizgileri hariç formun büyüklüğüdür. Programda yapıştırma işlemini ise Clipboard nesnesini kullanarak yaptık. Clipboard nesnesini ilerleyen bölümlerde ayrıca göreceğiz.

ÖRNEK: Başka bir örnek olarak ta Picture kutularını renk paleti olarak kullanalım. Örneğimiz için aşağıdaki formu oluşturun ve Picture kutusunun index özelliğine 0 vererek bunun bir dizi olacağını belirtin.



Bu Picture'den çalışma zamanı 15 tane daha oluşturarak her birinde farklı bir renk göstereceğiz. Kullanıcı bunlardan birini sol tuşla tıklarsa text kutusunun zemin rengini, sağ tuşla tıklarsa da yazı rengini değiştirebilecek.

```
'ÖRNEK : Picture
Private Sub Form_Load()
 Dim i
 Picture1(0).BackColor = QBColor(0)
 For i = 1 To 15
  Load Picture1(i) 'Yeni bir picture oluştur
  'Öncekinin yanına al
  Picture1(i).Move Picture1(i-1).Left + Picture1(i-1).Width + 15
  'i numaralı rengi ver
  Picture1(i).BackColor = QBColor(i)
  've göster
  Picture1(i).Visible = True
 Next
End Sub

Private Sub Picture1_MouseDown(Index As Integer, Button As
Integer, Shift As Integer, X As Single, Y As Single)
 If Button = 1 Then 'sol tuşla tıkladıysa
  'Zemin rengini değiştir
   Text1.BackColor = QBColor(Index)
 Else
  'Yazı rengini değiştir
  Text1.ForeColor = QBColor(Index)
 End If
End Sub
```



Methods

PaintPicture (KaynakPicture, hedefx, hedefy, hedefgenişlik, hedefyükseklik, kaynakx, kaynaky, kaynakgenişlik, kaynakyükseklik, işlem)

Resim işleme için geliştirilmiş bir metoddur. Aslında bu metod BitBlt Api'sinin yaptığı işlemleri yapar. Bu metodla bir resmin ölçeklenmesi (büyültüp, küçültme), taşınması, ters çevrilmesi ve üzerinde işlem yapılması (negatifinin alınması gibi işlemler) mümkündür.

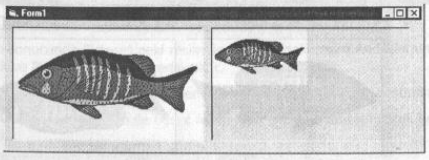
KaynakPicture parametresi ile işleme girecek resim belirlenir. PaintPicture metodunda hedef resim ise kontrolün kendisidir. Örneğin Picture1'in PaintPicture metodu kullanılıyorsa hedef resim Picture1 içindeki resimdir. Kaynak resim olarak ta aynı kontrolün Picture özelliği verilebilir.

Kaynakx, Kaynaky parametreleri ile işlenecek resmin sol üst köşe koordinatları ve **kaynakgenişlik, kaynakyükseklik** parametreleriyle de boyutları belirlenir.

Hedefx, Hedefy, Hedefgenişlik, HedefYükseklik parametreleri ile de işlenmiş resmin gösterileceği kontroldeki koordinatları ve boyutları belirlenir.

ÖRNEK: Kaynak resmin boyutları hedef resmin boyutlarından daha küçük verilerek resim küçültülebilir veya daha büyük verilerek resim büyültülebilir.

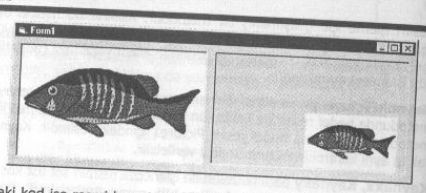
```
Private Sub Picture1_Click()
Picture2.PaintPicture Picture1.Picture, 0, 0, Picture1.ScaleWidth
/ 2, Picture1.ScaleHeight / 2, 0, 0, Picture1.ScaleWidth,
Picture1.ScaleHeight, &HCC0020
End Sub
```



Yukarıdaki kodla Picture1 içindeki resmi %50 küçülterek Picture2 içine almak mümkündür.

ÖRNEK: Kaynak ve hedef resmin koordinatları farklı verilerek de resmi taşımak mümkündür.

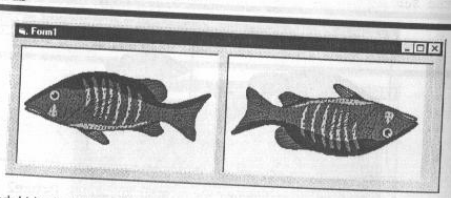
```
Private Sub Picture1_Click()
Picture2.PaintPicture Picture1.Picture, Picture1.ScaleWidth / 2,
Picture1.ScaleHeight/2,Picture1.ScaleWidth/2,
Picture1.ScaleHeight/2, 0, 0, Picture1.ScaleWidth,
Picture1.ScaleHeight, &HCC0020
End Sub
```



Yukarıdaki kod ise resmi hem %50 ölçeklendirmekte, hem de Picture2'nin ortasından başlamasını sağlamaktadır.

ÖRNEK: Hedef resmin boyutlarını negatif vererek resmi ters çevirmek mümkündür. Sadece hedefin genişliği negatif verilirse yatay olarak, yüksekliği negatif verilirse dikey olarak resim ters çevrilmiş olur. Her ikisi birden ters verilerek resim her iki yönde de ters çevrilebilir.

```
Private Sub Picture1_Click()
Picture2.PaintPicture Picture1.Picture, Picture1.ScaleWidth,
Picture1.ScaleHeight, -Picture1.ScaleWidth, -
Picture1.ScaleHeight, 0, 0, Picture1.ScaleWidth,
Picture1.ScaleHeight, &HCC0020
End Sub
```



Yukarıdaki kodla Picture1 içindeki resim her iki yönde de ters çevrilerek Picture2 içine alınmıştır.

Bu metodla resmin bir parçasını başka bir yere almak ta mümkündür. Ancak bu işlem için PicClip kontrolünü kullanmak daha avantajlıdır.

İşlem parametresi ile kaynak resime uygulanacak işlem belirlenir. Bu parametreye aşağıdaki değerlerden biri verilebilir.

&H550009 : Hedef resmin tersi alınır.

ÖRNEK: Örneğin picture1 içindeki resim tıklanınca tersi renge girmesi için

```
Private Sub Picture1_Click()
Picture1.PaintPicture Picture1.Picture, 0, 0,Picture1.ScaleWidth,
Picture1.ScaleHeight, 0, 0, Picture1.ScaleWidth,
Picture1.ScaleHeight, &H550009
End Sub
```

Picture1 içindeki resim tıklanınca tersi renge girecektir. Tekrar tıklandığında ise tekrar tersi alınacağı için orijinal haline döner.

&H330008: Kaynak resmin tersi alınarak hedefe kopyalanır.

&H1100A6: Kaynak resim ile hedef resim OR işlemine tabi tutulur ve sonucun tersi (NOT işlemi) alınarak hedefte gösterilir.

&H8800C6: Kaynak resim ile Hedef resim AND işlemine tabi tutularak hedefte gösterilir.

```
Private Sub Picture1_Click()
Picture2.PaintPicture Picture1.Picture, 0, 0,Picture1.ScaleWidth,
Picture1.ScaleHeight, 0, 0, Picture1.ScaleWidth,
Picture1.ScaleHeight, &H8800C6
End Sub
```

Bu işlem sonucunda Picture1'deki resim ile Picture2'deki resim And işlemine tabi tutulacak ve Picture2 içinde gösterilecektir.

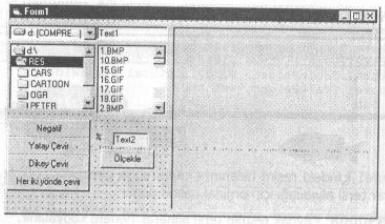
&HCC0020 : Kaynak resim hiçbir işleme tabi tutulmadan hedefte gösterilir.

&H660046 : Kaynak resim ile hedef resim XOR işlemine tabi tutularak hedefte gösterilir.

Kullanabileceğiniz diğer işlemlerin sembolik isimleri ise şöyledir: vbDstInvert, vbMergeCopy, vbMergePaint, vbNotSrcCopy, vbNotSrcErase, vbPatCopy,

vbPatInvert, vbPatPaint, vbSrcAnd, vbSrcCopy, vbSrcErase, vbSrcInvert, vbSrcPaint

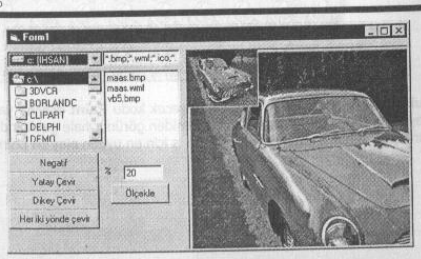
ÖRNEK: Örnek olarak kullanıcının seçeceği resim üzerinde ölçekleme, çevirme ve negatif alma işlemleri yapabilecek bir program yapalım. Örneğimiz için aşağıdaki formu hazırlayın.




```
ÖRNEK : PaintPicture
Private Sub Command1_Click()
    On Error Resume Next
    Picture1.PaintPicture Picture1.Picture, 0, 0,
Picture1.ScaleWidth, Picture1.ScaleHeight, 0, 0,
Picture1.ScaleWidth, Picture1.ScaleHeight, &H550009
End Sub

Private Sub Command2_Click()
    On Error Resume Next
    Picture1.PaintPicture Picture1.Picture, Picture1.ScaleWidth,
0, -Picture1.ScaleWidth, Picture1.ScaleHeight, 0, 0,
Picture1.ScaleWidth, Picture1.ScaleHeight, &HCC0020
End Sub

Private Sub Command3_Click()
    On Error Resume Next
    Picture1.PaintPicture Picture1.Picture, 0,
Picture1.ScaleHeight, Picture1.ScaleWidth, -Picture1.ScaleHeight,
0, 0, Picture1.ScaleWidth, Picture1.ScaleHeight, &HCC0020
End Sub

Private Sub Command4_Click()
    On Error Resume Next
    Picture1.PaintPicture Picture1.Picture, Picture1.ScaleWidth,
Picture1.ScaleHeight, -Picture1.ScaleWidth, -
Picture1.ScaleHeight, 0, 0, Picture1.ScaleWidth,
Picture1.ScaleHeight, &HCC0020
End Sub
```



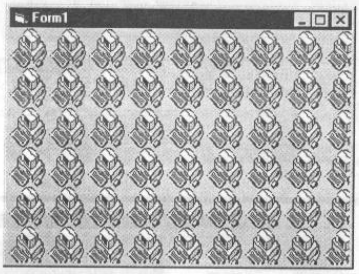
```
Private Sub Command5_Click()
    On Error Resume Next
    Text2 = Val(Text2)
    Picture1.PaintPicture Picture1.Picture, 0, 0,
Picture1.ScaleWidth * Text2 / 100, Picture1.ScaleHeight * Text2 /
100, 0, 0, Picture1.ScaleWidth, Picture1.ScaleHeight, &HCC0020
End Sub

Private Sub Dir1_Change()
    File1.Path = Dir1.Path
    ChDir File1.Path
End Sub

Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
    ChDrive Dir1.Path
End Sub

Private Sub File1_Click()
    On Error Resume Next
    Picture1.Picture = LoadPicture(File1.filename)
End Sub

Private Sub Form_Load()
    Text1 = "*.bmp;*.wmf;*.ico;*.dib;*.gif;*.jpg;*.emf;*.cur"
    Picture1.AutoSize = True
    Picture1.AutoRedraw = True
    Text2 = 50
End Sub

Private Sub Text1_Change()
    On Local Error Resume Next
    File1.Pattern = Text1
End Sub
```



ÖRNEK: PaintPicture metodunu kullanarak bir resmi form üzerine döşeyecek program yazalım.

Örneğimiz için formunuzun üzerine bir **PictureBox** yerleştirin ve içine döşenmesini istediğiniz resmi alın.

```
ÖRNEK: PaintPicture
Private Sub Form_Load()
 Picture1.Visible = False
End Sub

Private Sub Form_Paint()
 Dim i, j
 For i = 0 To ScaleWidth Step Picture1.Width
  For j = 0 To ScaleHeight Step Picture1.Height
   PaintPicture Picture1.Picture, i, j, Picture1.Width,
Picture1.Height, 0, 0
  Next
 Next
End Sub
```



Yukarıdaki örnekte, resmi forma döşeyecek kodu **Paint** olayına yazdık. Formun boyutları değiştiğinde veya ekranda yeniden görünür hale geldiğinde içindeki görüntünün yeniden çizilmesi gerekir. Bu iş için en uygun olay da paint olayıdır.

Image(Resim Gösterme Kontrolü)

Bu eleman grafiksel bir kontrol olup resimleri görüntülemek, boyutlandırmak ve taşımak için kullanılır. PictureBox'dan daha hızlı ve daha az system kaynağı kullanmasına rağmen PictureBox kadar gelişmiş özelliklere sahip değildir. PictureBoxtan farklı olarak gruplandırma yapamaz ve metodları kullanılarak yazım ve çizim yapamaz.

Properties

Picture

BMP, ICO, WMF, DIB, JPEG, GIF resim formatlarını destekler. Bu özellik, PictureBox kısmında detaylı olarak açıklanmıştır.

Stretch

Bu özellik resmin image kutusu kadar büyütülüp, büyütülemeyeceğini belirler.

True ise resim, kutunun içine sığdırılır. Yani resim büyük ise küçültülür, küçük ise büyültülür.

False ise Image kutusunun boyutları resmin boyutlarına ayarlanır. (**AutoSize** gibi davranır)

PicClip (Resim işleme kontrolü)*

Bu kontrolle BMP formatındaki resimleri parçalara ayırıp kullanmak mümkündür. Bu kontrol tasarım zamanı ekranda görülürken çalışma zamanı ekranda görülmez. Bu kontrolle parçalanan resimler Picture özelliği olan herhangi bir kontrolde kullanılır. Bu kontrol yalnız pixel scale modunu desteklediği için bu kontrolle parçaladığınız resimleri kullanacağınız kontrollerinde ScaleModunun pixel olması işlerinizi kolaylaştıracaktır.

Örneğin programınızda değişik durum ve zamanlarda göstereceğiniz çok sayıda resiminiz olduğunu düşünelim. Bu resimler herhangi bir olayı ifade eden resim, şekil olabileceği gibi komut düğmelerinin üzerinde gösterilecek resimler de olabilir. Programınızda değişik zamanlarda göstereceğiniz bu şekilleri programa almanın değişik yolları vardır. İlk yöntem; kullanılacak bütün resim dosyalarını da programla birlikte taşınarak ve programda LoadPicture ile yükleyerek kullanmak. Bu kadar çok resim dosyası programınızın taşınırlığını zorlaştıracağı gibi programınızın sürekli diskle ilişkisi olacağından orta çaplı programlarda bu yöntem pek uygun olmayabilir. İkinci bir yöntem ise form tasarlanırken bütün resimleri ayrı ayrı resim kutularına yüklemek ve programda gerekli olanları ekranda görüntülemek. Doğal olarak bu yöntemde bir çok resim kutusu kullanılacağı için programın boyutları büyüyecek ve system kayaklarının azalmasına sebep olacaktır. Bir diğer çözüm de bütün resimleri tek bir dosyada birleştirilerek PicClip kontrolüyle bunları seçip programın herhangi bir anında kullanmak. Bu son yöntemle hem tek kontrol kullanıyoruz ve hem de bütün resimleri tek bir dosya altında toplamış oluyoruz.

Ayrıca bu kontrol animasyonlar içinde uygundur. Bir animasyondaki resimleri toplu halde kaydedip bunları picclip içine alarak bir timer aracılığı ile arka arkaya gösterip animasyon yapmak mümkündür.

*Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft PictureClip Control" seçeneğini işaretlemeniz veya Browse düğmesi ile PICCLP32.OCX dosyasını bulup projeye eklemeniz gerekir.

Properties

Picture

PicClip kontrolü içine yüklenecek resmi belirleyen bu özellik vasıtasıyla diğer kontrollerdeki yöntemlerle resim yüklenir. PicClip kontrolünün Picture özelliğine yalnız BMP ve DIB formatındaki resimler yüklenebilir.

ClipX, ClipY, ClipWidth, ClipHeight, Clip

PicClip kontrolü içindeki resmin bir kısmını tanımlamaya yarar. ClipX, ClipY resimden alınacak parçanın sol üst köşe koordinatlarıyken, ClipWidth, ClipHeight bu parçanın boyutlarıdır. Clip ise bu özelliklerle sınırları belirlenmiş resimdir.

```
PicClip1.ClipX = PicClip1.Width / 2 -25
PicClip1.ClipY = PicClip1.Height / 2 -25
PicClip1.ClipHeight = 50
PicClip1.ClipHeight = 50
Picture1.Picture=PicClip1.Clip'Seçilen parçayı PictureBox'a yükle
```

Yukarıdaki kodlarla PicClip kontrolü içerisindeki resmin tam ortasında 50x50 piksellik bir parçanın PictureBox içerisinde gösterilmesi sağlanır.

PicClip kontrolündeki resimin tamamı başka bir nesne içine alınacaksa Picture özelliği kullanılabilir.

Cols, Rows, GraphicCell(Index)

PicClip kontrolü içindeki resmi yukarıdaki gibi bir parça kullanmak mümkünken bir matris gibi parçalara bölünerek de kullanılabilir. Cols özelliği resmin kaç sütuna, Rows özelliği de kaç satıra bölüneceğini belirler. Bu şekilde parçalara bölünen resmin istenilen parçası GraphicCell özelliğiyle gösterilir. Burada GraphicCell özelliğinin index parametresi matris şeklinde parçalanan resmin hangi parçasının kullanılacağını belirler. İlk parçanın indexi 0 iken son parçanın indexi **Cols x Rows – 1**'dir. PicClip içindeki resmi 3 x 4'lük bir matris olarak tanımlarsak GraphicCell özelliğinin alacağı index değerleri şöyle olacaktır.

0	1	2	3
4	5	6	7
8	9	10	11

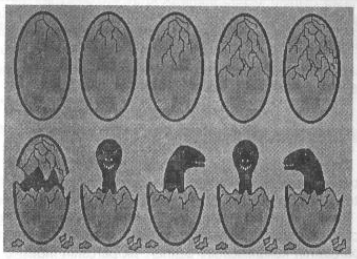
StretchX, StretchY

PicClip içinde parçalanan resmin bu özelliklerle verilen boyutlara sığdırılmasını sağlar. Bu işlemde seçilen parçanın boyutları bu özelliklerle verilen değerlerden daha küçükse parça üzerinde herhangi bir değişim olmaz, büyükse parça ölçeklendirilerek verilen boyutlara sığdırılır.

Örneğin PicClip içinde parçaladığınız resmi PictureBox içinde göstermek istediğimizi düşünelim. Seçilen parça PictureBox kontrolünün boyutlarını aşıyor olabilir. Seçilen parçayı PictureBox boyutlarına indirgemek için bu iki özelliğe PictureBox'un Scale boyutları verilerek seçilen parçanın boyutları PictureBox'un boyutlarına sığdırılır.

```
picclip1.StretchX = picture1.ScaleWidth
picclip1.StretchY = picture1.ScaleHeight
```

ÖRNEK: Picclip kontrolü ile animasyonlar yapılabilmektedir. Yapılacak animasyonu oluşturan bütün kareler çizilerek tek bir dosyada aşağıdaki gibi birleştirilir. Biz örnek olarak bir dinazorun yumurtadan çıkışını adım adım çizerek tek bir BMP dosyasında aşağıdaki gibi topladık.



Yukarıda gördüğünüz gibi animasyon 2 satır ve 5 sütunda toplanmış toplam 10 resimden oluşmaktadır. Yapılması gereken bu resim dosyasını **Picclip** kontrolünün **Picture** özelliğine girdikten sonra bu resimleri bir Timer aracılığı ile Image veya Picture kontrolü içinde göstermektedir.

- Örneğimiz için form üzerine bir Picclip yerleştirin ve Picture özelliğine hazırladığınız resim dosyasının adını girin. **Cols** özelliğini 5, **Rows** özelliğini 2 yapın.

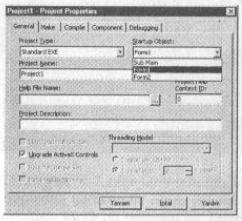
- Bit Timer yerleştirin ve Interval özelliğini 500 yapın. Bu ne kadar küçük olursa animasyon o kadar hızlı olacaktır.
- Resimleri göstermek için bir Image yerleştirin ve aşağıdaki kodu Timer olayına yazın.

```
Sub Timer1_Timer ()
 Static i
 'i numaralı resimi göster
 Image1.Picture = Picclip1.GraphicCell(i)
 i = i + 1
 i = i Mod 10 '10 olunca başa dön
End Sub
```

Form

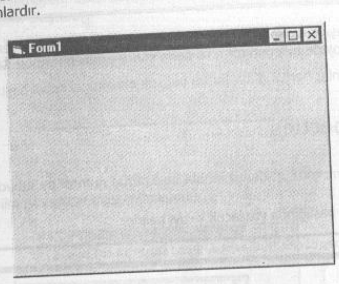
Windows arabiriminin en temel kontrolü formlardır. Windows'ta hemen hemen her program formlar üzerinde çalışır. Zaten Windows kelimesinin Türkçe anlamı olan Pencereler de bu formlardır. Boyutlandırılabilir özelliği sayesinde aynı anda ekranda tek bir program olmak zorunda değildir.

- Formun propertieslerini Formun alt programlarında yazarken Formun ismini kullanmak zorunda değilsiniz, direk properties ismini vermeniz yeterlidir. Yani **Form1.Caption** ile sadece **Caption,** form1'in alt programlarında aynı etkiye sahiptir.
- Programınızda birden fazla form bulunacaksa ilk olarak ana form çalışacaktır. Diğer formları programınızda kullanacağınız **Form2.Show** gibi bir yöntemle aktif hale getirmelisiniz.
- Bir formdan başka bir formun bir özelliğine ulaşabilmek için kontrol adından önce formun adı da verilmelidir. Örneğin **Form2** üzerindeki **Label1**'e ulaşmak için **Form2.Label2.Özellik** şeklinde kullanılır.
- Programınızda birden fazla formunuz varsa ilk oluşturduğunuz form ana formdur ve program çalışmaya o formla başlar. Eğer başlangıçta çalışacak formu değiştirmek istiyorsanız; **Project-Properties** menüleri ile açılan aşağıdaki pencerenin **Startup Object** listesinden istediğiniz formu seçebilirsiniz.



Yukarıdaki pencerede gördüğünüz gibi programlar çalışmaya formdan başlayabileceği gibi ismi Main olan bir alt programdan da başlayabilir. Bunun için bir Modül oluşturulur ve Sub Main ile başlayan bir alt program yazılır. Böylece program buradan başlar.

- Formlar ikiye ayrılır. Biri programlarda kullandığımız tek başına çalışan SDI (Single Document Interface) formlardır. Şu ana kadar gördüğümüz formlar bu tip formlardır.



- İkincisi de MDI (Multi Document Interface) formlardır. MDI formlar içinde kendisine bağlı formlar bulundururlar. Bu formlar (Child form) MDI forma bağlıdır. MDI formla birlikte hareket ederler. Aşağıda bir **MDI** form ve onun içinde bulunan **Child** formlar görülmektedir.

SDI Formlar

Tek başına çalışan formlara **SDI** form denir. Normalde yeni bir projeye başlarken projede bulunan form ve daha sonra eklenen formlar **SDI** formlardır.

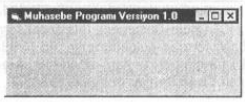
Bu formlar herhangi bir forma bağımlı olmaksızın kendi başlarına çalışırlar.

Properties

Caption

Formun başlığında yazılacak yazıyı belirler.

```
form1.caption="Muhasebe Programı Versiyon 1.0"
```



Icon

Formu tanıtacak iconu belirler. Seçilecek ikon form minimize edildiğinde, Alt+Tab geçişlerinde bu formu temsil eder. Ayrıca programa ait bir kısayol oluşturulduğunda bu ikon kullanılır.

Properties penceresinden bu özellik çift tıklanarak açılan pencereden GRAPHICS\ICONS dizinine geçilerek çok sayıda ikon bulunabilir.

BorderStyle

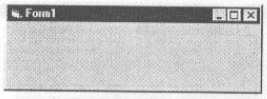
Formun çerçeve şeklini belirleyen bu özellikle formun boyutlarının değiştirilmesini veya kapatılabilmesini vereceğiniz değerlerler engelleyebilirsiniz.

0:vbBSNone

Çerçevesi, Başlığı, Kontrol kutusu, Ekranı Kapla (Maximize), Simge durumuna Küçült (Minimize) düğmeleri olmayan bir form oluşturur. Bu değer verildiğinde kullanıcı formu boyutlandıramaz, taşıyamaz ve kapatamaz.

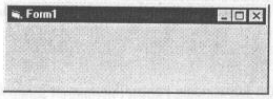
1:VbFixedSingle

Boyutları değiştirilemeyen, fakat konumu kullanıcı tarafından değiştirilebilen yani taşınabilen bir form oluşturur.



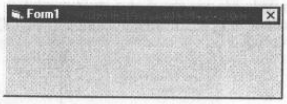
2:VbSizable

Default değer budur ve formun bütün özelliklerini sunabilir. Formun kapatılması, boyutlandırılabilmesi ve taşınması mümkündür.



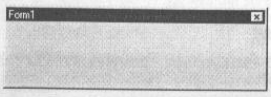
3:VbFixedDouble

Form kullanıcı tarafından boyutlandırılamaz. Yalnız formu kapatabilir veya taşıyabilir. 1 değerinden farkı; maximize ve minimize düğmelerinin olmamasıdır.



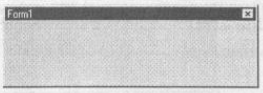
4:VbFixedToolWindow

Normal forma göre başlığı daha küçük olan, kontrol menüsü olmayan ve boyutlandırılamayan bir form oluşturur. Bu form daha çok ToolBox pencereleri gibi araçların bulunduğu formlar için uygundur.



5:VbSizableToolWindow

4 değerindeki gibi bir form oluşturur. Farklı olarak bu pencerenin boyutları kullanıcı tarafından değiştirilebilir.

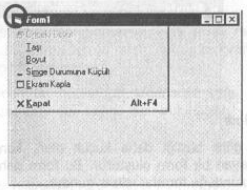


MaxButton, MinButton

Formun sağ üst köşesindeki maximize ve minimize düğmelerinin görüntülenmesini veya görüntülenmemesini sağlar. Bu değerler True ise formun minimize veya maximize edilmesi herhangi bir koda gerek kalmadan gerçekleşir.

ControlBox

Formun sol üst köşesindeki kontrol kutusunun görüntülenmesini veya görüntülenmemesini sağlar. Bu özellik False yapılmış ise form kullanıcı tarafından Alt+F4 tuş bileşimiyle kapatılamaz. True ise herhangi bir koda gerek kalmadan form kapanır, bu form programın ana formu ise program sona erer. Tabi bundan Unload ve QueryUnload olayıyla programın haberi olur. Bu olaylara yazacağınız kodla formun kapanmasını önlemeniz de mümkündür.

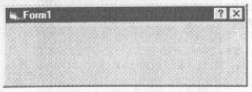


WhatsThisButton

Windows 95 versiyonunda pencerelerde [?] düğmesi de bulunabilir. **WhatThisButton** özelliğine **True** verilirse form üzerinde bu düğme bulunur.

Form üzerinde bu düğmenin bulunabilmesi için ayrıca

- ControlBox özelliği True
- BorderStyle özelliği Fixed Single, Sizable veya Fixed Dialog
- MinButton ve MaxButton özellikleri False olmalıdır.



Bildiğiniz gibi bu düğmenin özelliği, ekrandaki kontrollerin ne işe yaradığını kullanıcıya bildirmektir. Kullanıcı bu düğmeyi tıkladıktan sonra ekranda bir kontrolü tıklarsa, o kontrolün ne işe yaradığını kullanıcıya bildirmek gerekir. Bildirilecek yardım metni, her kontrolün **WhatThisHelpID** özelliği ile belirlenir.

Moveable

Bu özellik formun kullanıcı tarafından taşınıp taşınamayacağını belirler. **False** verilirse kullanıcı formu taşıyamaz.

ShowInTaskbar

Bu özellik formun, Windows görev çubuğunda yer alıp almayacağını belirler. Bu değere **False** verilirse form görev çubuğunda görülmez. Daha çok diyalog kutusu ve Toolbox olarak kullanılacak formların araç çubuğunda görülmesi gerekmez.

AutoRedraw

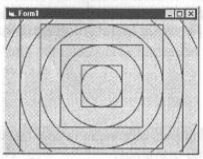
False ise form üzerine başka bir form geldiğinde veya form boyutlandırıldığında form üzerine yapılan yazım ve çizimler yenilenmeyecektir. **True** ise form boyutlandırılırken veya üzeri kapatılırken formun içeriği kaydedilecek ve formun içeriğinin kaybolmamasını sağlayacaktır. Eğer form üzerine yazım veya çizim yapıyorsanız bu özelliği True yaparak çizim ve yazılarınızın sürekli olarak form üzerinde kalmasını sağlayabilirsiniz. Ancak bu işlem hafızanın bir miktar azalmasına ve işlemlerin yavaşlamasına sebep olacaktır.

- Form üzerine **Print, Circle, Line** veya **Point** metodlarıyla yaptığınız yazım ve çizimleri kaydetmek veya yazdırmak istiyorsanız **AutoRedraw** özelliğini **True** yapmanız gerekir.

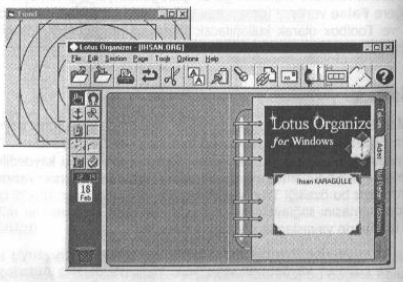
Bu özelliğin etkisini görmek için Formun **AutoRedraw** özelliği **False** iken aşağıdaki kodu yazıp mouse ile formu tıklayın.

```
'ÖRNEK:AutoRedraw
Private Sub Form_Click()
 Dim i, x, y
 x = ScaleWidth / 2
 y = ScaleHeight / 2
 For i = 0 To ScaleWidth Step 500
  Circle (x, y), i
  Line (x + i, y + i)-(x - i, y - i), , B
 Next
End Sub
```

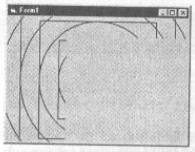
Form üzerine tıkladığınızda form üzerine daireler ve dikdörtgenler çizilecektir.



Formu boyutlandırdığınızda veya üzerine başka bir form getirdiğinizde formun ekranda görülmeyen kısmı silinecektir. Örnek olarak formun bir kısmının aşağıdaki gibi başka bir program tarafından kapatıldığını kabul edelim.



Tekrar programımıza ait forma döndüğünüzde çizimin altta kalan kısmının aşağıdaki gibi silindiğini göreceksiniz.



AutoRedraw özelliğini **True** yaparak aynı işlemleri tekrarlarsanız formun silinmediğini ancak işlemlerin biraz yavaşladığını göreceksiniz.

FillColor,FillStyle

Circle ve **Line** metodu ile form üzerine çizilen çember ve kutuların iç boyama rengini ve desenini belirler. **FillStyle** özelliğine verilebilecek değerler ve etkileri aşağıdaki gibidir.

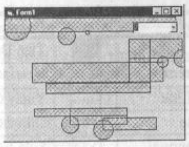
	FillStyle	Örnek	Boyama şekli
0	vbFSSolid		Tam Dolu
1	vbFSTransparent		Üzerinde bulunduğu yerin zemin rengi
2	vbHorizontalLine		Yatay Çizgili
3	vbVerticalLine		Dikey Çizgili
4	vbUpwardDiagonal		Sola Eğik
5	vbDownwardDiagonal		Sağa Eğik
6	vbCross		Kareli
7	vbDiagonalCross		Çapraz

ÖRNEK: Örnek olarak kullanıcının seçtiği stilde rasgele çizim yapacak bir program yazalım. Örneğimiz için formunuza bir **ComboBox** ve **Timer** yerleştirin.

```
'ÖRNEK : FillColor, FillStyle
Private Sub Form_Load()
    Timer1.Interval = 100
    Dim i
    For i = 0 To 7
        Combo1.AddItem i
```

```
    Next
    Combo1.ListIndex = 0
End Sub

Private Sub Combo1_Click()
    FillStyle = Combo1.ListIndex
    FillColor = QBColor(Rnd * 15)
    Cls
End Sub

Private Sub Timer1_Timer()
    Dim x, y, r
    x = Rnd * ScaleWidth
    y = Rnd * ScaleHeight
    r = Rnd * ScaleWidth / 10
    If Int(Rnd * 2) Then
        Line -(x, y), , B
    Else
        Circle (x, y), r
    End If
End Sub
```

Programı çalıştırdığınızda 0-7 arası rakamlar Combo kutusuna eklenecektir. Buradan seçeceğiniz değere göre o stilde rasgele çizim yapılacaktır.



FontTransparent

True ise form üzerine **Print** metodu ile yazılacak yazıların zemin rengi yoktur dolayısıyla yazı altında resim veya başka bir yazı varsa bunu gösterecektir.

False ise yazı kendi zemin rengi üzerine yazılır ve altındaki nesneyi göstermez.

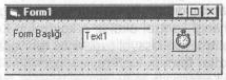
WindowState

Formun çalışması üç durumda olur ve bu durumlar **WindowState** özelliğiyle belirlenebilir veya formun durumu bu özellikle öğrenilebilir.

0:vbNormal:	Normal, ekranın herhangi bir kısmında pencere içinde
1:VbMinimized:	Minimized, simge durumunda
2:VbMaximized:	Maximized, Ekranın tamamını kaplayacak şekilde

Bu özelliğe yukarıdaki değerlerden biri atanarak formun durumu değiştirilebilirken, bu değer okunarak ta formun durumu öğrenilebilir.

ÖRNEK: Örnek olarak program simge durumunda çalışırken program başlığını kayan yazı şeklinde gösterecek bir program yapalım. Bunun için şu basit formu oluşturun.



Formun simge durumunda çalıştığını öğrenmek için WindowState özelliğini, yazıyı sürekli kaydırmak içinde Timer kontrolünün Interval olayını kullanacağız.

```
'ÖRNEK : WindowState
Private  Sub Form_Load ()
  WindowState = 1 'Formu simge durumuna getir
  Text1.Text = Form1.Caption
  Timer1.Interval = 100
End Sub
Private  Sub Text1_Change ()
  Form1.Caption = Text1
End Sub
Private  Sub Timer1_Timer ()
   If WindowState = 1 Then    'Form minimize ise
   'Baş harfi sona ekle
   Form1.Caption = Mid(Form1.Caption,2)+Left(Form1.Caption, 1)
  End If
End Sub
```


StartupPosition

Form yüklenirken ekrandaki koordinatlarının neye göre belirleneceğini bu özelliği etkiler.

0: Form tasarlanırken bulunduğu koordinatlarda açılır.

1: Form içinde bulunduğu formun ortasında açılır. (MDI Child formlar için)

2: Form ekranın ortasında açılır.

3: Formun koordinatları Windows tarafından belirlenir.

KeyPreview

Form aktifken basılan tuşlardan formun haberdar edilip edilmeyeceğini belirleyen önemli bir özelliktir.

False: Herhangi bir kontrol üzerinde iken basılan tuşlar yalnız o kontrolün KeyPress, KeyDown, KeyUp olaylarını meydana getirir.

True: Herhangi bir kontrol üzerinde basılan tuşlar önce Form'un daha sonra o kontrolün KeyPress, KeyDown, KeyUp olaylarını da meydana getirir.

Programınızda herhangi bir kontrole ait olmayan kısa yol tuşlarınız varsa bu tuşlara basılıp basılmadığını sizin kontrol etmeniz gerekecektir. Normalde bir tuşa basıldığında hangi kontrol aktifse o kontrolün Key olayları meydana gelecektir. Bu da her kontrol için kısayol tuşlarınızı tanıyacak kod yazacaksınız demektir. İşte bu olaya çözüm formun KeyPreview özelliğini True yaparak form üzerindeki her Key olayının Formda da meydana gelmesini sağlamaktır. Örneğin **ESC** tuşuna basıldığında programın kapanmasını istiyorsanız bu özelliği **True** yapıp Formun **Keypress** olayına aşağıdaki kodu yazabilirsiniz.

```
Private Sub Form_KeyPress(KeyAscii As Integer)
 If KeyAscii = 27 Then End '27 ESC tuşunun kodudur
End Sub
```

Count

Form içindeki menüler dahil kontrol sayısını verir.

Controls(Index)

Aslında bu forma ait bir properties değildir. Ancak konunun akışı içerisinde burada vermeyi uygun gördük. Formun **Count** özelliğiyle birlikte kullanıldığında o-

dukça önemli işler yapabilirsiniz. Bu özellik form üzerindeki menüler dahil Index numaralı nesnenin adını temsil eder.

ÖRNEK: Formdaki bütün kontrollerin zemin rengini değiştirmek istediğimizi düşünelim. Bu iş için bütün kontrollerin zemin rengini tek tek değiştirecek kod yazmak yerine Controls özelliğini kullanarak bütün kontrollere bir döngü içinde erişebiliriz. Formunuzun üzerine değişik kontrollerden yerleştirin ve aşağıdaki kodu yazın.

```
'ÖRNEK: Count, Controls
Private Sub Command1_Click()
 On Local Error Resume Next
 Dim i
 For i = 0 To Count - 1
  Controls(i).BackColor = QBColor(2)
 Next
End Sub
```

Yukarıdaki örneği herhangi bir programınıza eklerseniz ekrandaki bütün kontrollerin zemin renginin yeşil (**qbcolor(2)** satırından dolayı) olduğunu göreceksiniz. Ayrıca bu yöntemi kullanırken mutlaka **On Local Error Resume Next** (hata olursa sonraki satıra git) satırını kullanmanız gerekir. Çünkü her kontrolün **BackColor** özelliği olmayabilir. Bu durumda hatayı önlemesi için yukarıdaki satır kullanılmalıdır.

ÖRNEK: Bu iki özelliği daha iyi anlayabilmek için şöyle bir örnek verelim. Formun boyutları kullanıcı tarafından küçültüldüğünde form üzerine yerleştirdiğiniz kontrollerin bir kısmı ya görünmeyecek yada bir kısmı görünecektir, büyültüldüğünde ise formun bir kısmı boş olacaktır. Buda hiç hoş olmayan bir görüntü oluşturur. Halbuki form boyutları değiştiğinde üzerindeki kontrollerin boyutlarını da formla orantılı olarak değiştirirsek formun görüntüsü daha güzel olabilir. Bu işlemi Formun Resize olayına yazacağımız kodla yapabiliriz. Formun eski boyutuyla yeni boyutu arasındaki oranı form üzerinde görülen bütün nesnelerin Left,Width ve Top, Height özellikleriyle çarparsak form ile üzerindeki kontroller orantılı olarak değişecektir. Ancak bunu menüler hariç ekranda görülen bütün kontrollere uygulamak gerektiği için her kontrolün Left ve Width özelliklerini değiştirecek kod oldukça uzun olacaktır. Bunu yerine Controls(Index) özelliğiyle form üzerindeki bütün kontrollerinkini bir for-next döngüsü içinde gerçekleştirebiliriz.

Örneğimiz için bir form üzerine istediğiniz kadar kontrolü istediğiniz şekilde yerleştirip aşağıdaki kodu yazınız.

```
'ÖRNEK : Count, Controls(Index)
Private  Sub Form_Resize ()
  Static eskiboyut
  Dim i, k
  'Program ilk defa çalışıyorsa eski boyut formun genişliğidir.
  If IsEmpty(eskiboyut) Then eskiboyut = form1.Width
  If WindowState <> 1 Then 'Form minimize değilse
     k = form1.Width / eskiboyut'Değişim oranı
     eskiboyut = form1.Width
     On Local Error Resume Next
     For i = 0 To form1.Count - 1
       'Form üzerindeki i. kontrolün boyutlarını değiştir
       controls(i).Left = controls(i).Left * k
       controls(i).Width = controls(i).Width * k
     Next
  End If
End Sub
```

Bu kod form üzerindeki bütün kontrollerin Left ve Width özelliklerini değiştirmeye çalışıyor. Ancak bütün kontrollerin bu özellikleri olmayabilir veya bu özellikler değiştirilemeyebilir. Sadece bu şarta uyanları da ayırmak zor olacağı için bir hata yakalayıcı işimize yarayacaktır. Kodumuz böyle bir kontrolü değiştirmeye çalıştığında hata yakalayıcımız bir sonraki satıra geçilmesini sağlayacak böylece Left ve Width özellikleri değişebilenler değiştirilecek, değiştirilemeyenler ise atlanacaktır. Bu örneğimiz sadece yataydaki değişimlere karşılık olarak kontrolleri boyutlandırıyor, aynı kodu Top ve Width için de yazarsanız kontroller her iki yönde de boyutlanır.

Ancak bu yöntem ideal bir yöntem değildir, Left ve Width özellikleri Integer tipte olduğundan formu küçültüp, tekrar büyüttüğünüzde kontrollerin tam olarak orijinal haline gelmediğini göreceksiniz. Integer değişkenler yapı itibariyle ondalık sayıları tutmadığından

```
Dim i as integer
i = 12
i = i / 5
i = i *5 'i = 12 değil i = 10
```

işlemi i'ye 12 değerini vermeyecektir. İşte bu yuvarlatma sebebiyle kontrollerinizin yerleri ve genişlikleri biraz değişecektir. Özellikle formu bir kaç defa boyutlandırdığınızda kontroller birbirine karışabilir. Buna çözüm olarak ta işlemleri en son boyutlardan değil orijinal boyutlardan yapmaktır. Bunun için de form üzerindeki kontrollerin tümünün boyutlarını başlangıçta bir diziye alıp daha sonraki boyutlandırma işlemlerinde bu orijinal değerlerle boyutlandırmak bu problemi çözecektir.

Form üzerindeki kontrollerin orijinal boyutlarını tutacağımız bir diziye ihtiyacımız olacaktır. Left, Width ve Top, Height özellikleri için dört ayrı dizi yerine bir modül'ün declaration kısmında şu tipi tanımlayalım


```
Type Kontrol
        Left As Integer
        Width As Integer
        Top As Integer
        Height As Integer
End Type
```

Bu tipi tanımladıktan sonra formun Declaration kısmında bu tipten bir dinamik dizi tanımlayarak form üzerindeki kontrol sayısı kadar bir dizi oluşturabiliriz. Programda bir kaç form varsa her form için aynı işlemi yapmanız gerekecektir.

```
Dim Kont() As Kontrol 'Formun Declaration kısmında dinamik dizi tanımı
```

Şimdi yapılacak iş formun Load olayında form üzerindeki bütün kontrollerin orijinal boyutlarını bu diziye almak ve formun Resize olayında bu değerleri kullanarak boyutlandırmak

```
'ÖRNEK : Count, Controls(Index)
Private  Sub Form_Load ()
  Dim i
  ReDim kont(form1.Count) As kontrol
  On Local Error Resume Next
  For i = 0 To form1.Count - 1
    kont(i).Left = form1.Controls(i).Left
    kont(i).Width = form1.Controls(i).Width
    kont(i).Top = form1.Controls(i).Top
    kont(i).Height = form1.Controls(i).Height
  Next
End Sub

Private  Sub Form_Resize ()
  Static orjboyutx, orjboyuty
  Dim i, kx, ky
  If orjboyutx = 0 Then
    'İlk defa çalışıyor
    orjboyutx = form1.Width
    orjboyuty = form1.Height
    Exit Sub
  End If
  If windowstate <> 1 Then
     kx = form1.Width / orjboyutx
     ky = form1.Height / orjboyuty
     On Local Error Resume Next
     For i = 0 To form1.Count - 1
       Controls(i).Left = kont(i).Left * kx
       Controls(i).Width = kont(i).Width * kx
       Controls(i).Top = kont(i).Top * ky
       Controls(i).Height = kont(i).Height * ky
     Next
  End If
End Sub
```


Bu kod diğerine göre daha iyi olmasına rağmen yine problemler çıkacaktır. Örneğin text kutularının veya combo kutularının yükseklikleri belirli bir değerin altına inmeyecektir. Bu durumda kontrollerin fontsize özelliklerini de ölçeklendirmeniz gerekebilir.

ActiveControl

Form üzerindeki aktif kontrolün ismi gibi davranır. Örneğin

```
form1.ActiveControl.Left = 0
```

satırı form üzerindeki aktif kontrolü sola alır. Burada aktif kontrol TextBox ise bu satır

```
Form1.TextBox.Left = 0
```

gibi, aktif kontrol ListBox ise

```
Form1.ListBox.Left = 0
```

gibidir. (Screen nesnesinin ActiveControl özelliğine bakınız)

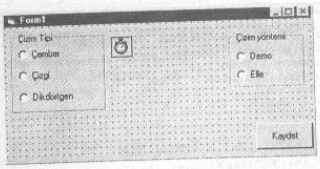
Picture

Form üzerinde gösterilecek resmi belirler. PictureBox'un Picture özelliği gibidir.

Image

Formun **AutoRedraw** özelliği **True** ise form üzerine yapılan yazım ve çizimlerin kaybolmadığından bahsetmiştik. Formun AutoRedraw özelliği True ise form üzerine yapılan çizim ve yazımlar Formun Image özelliğine kaydedilir. Bu özellik yalnız okunabilir bir özelliktir. Eğer form üzerine yaptığınız çizimleri kaydetmek, panoya kopyalamak veya başka bir kontrol içinde göstermek istiyorsanız Formun Image özelliğini kullanmalısınız.

Daha önce Frame ve OptionButton için yaptığımız örneği biraz daha geliştirelim. O örnekte sadece form üzerine çizim yapıyorduk. Şimdi ise form üzerine yapılan çizimi kaydetme imkanı sağlayalım, ayrıca yapılan çizimin minyatür halini de bir Image kutusunda gösterelim. Geliştireceğimiz bu örneğimiz için aşağıdaki formu şu adımlarla oluşturun.



- Formun üzerine yapılan çizimlerin kaydedilmesi için Formun AutoRedraw özelliğini True yapın
- Formun sol ve sağ köşelerine iki tane Frame kontrolü yerleştirin
- Birinci Frame içine bir OptionButton yerleştirin ve iki kopyasını da hemen altına yerleştirin.
- İkinci Frame içine de bir OptionButton yerleştirin ve bir kopyasını da hemen altına yerleştirin.
- İkinci Framenin altına, Stretch özelliği True olan bir Image kontrolü yerleştirin
- Image kontrolünün altına da bir Command Button yerleştirin
- Interval özelliği 1000 ve Enabled özelliği False olan bir Timer kontrolü yerleştirin.
- Kontrollerin Caption özelliklerine yukarıda gördüğünüz değerleri verin.

```
'ÖRNEK : Image, AutoRedraw
Option Explicit
Dim mx, my, r, x1, y1

Private Sub Form_Resize ()
 If windowstate <> 1 Then   'Form minimize değilse
  'Birinci çerçeveyi sol üst köşede tut
  frame1.Left = 50
  frame1.Top = 0
  'İkinci çerçeveyi sağ üst köşede tut
  frame2.Left = form1.Width - frame2.Width - 150
  frame2.Top = 0
  'ImageBox'u çerçevenin altına yerleştir
  image1.Left = frame2.Left
  image1.Top = frame2.Top + frame2.Height + 100
  'Komut düğmesini ImageBox'un altına yerleştir
  command1.Left = frame2.Left
  command1.Top = image1.Top + image1.Height + 100
 End If
End Sub
```

```
Private Sub Form_MouseDown (button As Integer, Shift As Integer,
x As Single, y As Single)
 If button = 1 Then
  If option1(0).Value = True Then
   r = 0
   mx = x
   my = y
  End If
  If option1(1).Value = True Then
   Line (x, y)-(x, y)
  End If
  If option1(2).Value = True Then
   x1 = x
   y1 = y
  End If
 End If
'Çizimleri picture kutusunda da göster
 image1.Picture = form1.Image
End Sub

Private Sub Form_MouseMove (button As Integer, Shift As Integer,
x As Single, y As Single)
 If button = 1 Then
  If option1(0).Value = True Then
   r = Abs(x - mx)
   Circle (mx, my), r
  End If
  If option1(1).Value = True Then
   Line -(x, y)
  End If
  If option1(2).Value = True Then
   Line (x1, y1)-(x, y), , B
  End If
 End If
 'Çizimleri picture kutusunda da göster
 image1.Picture = form1.Image
End Sub

Private Sub Option2_Click (Index As Integer, Value As Integer)
 Cls
 'demo modu seçili ise timeri aktif hale getir
 timer1.Enabled = option2(0).Value
End Sub

Private Sub Timer1_Timer ()
 Dim md As Integer
 'rasgele bir çizim modu seç
md = Rnd * 2
 'Hangi çizim modu seçili ise işaretle
option1(md).Value = True
Select Case md
 Case 0: Circle (Rnd * 5000, Rnd * 5000), Rnd * 5000
```



```
 Case 1: Line -(Rnd * 5000, Rnd * 5000)
 Case 2: Line(Rnd*5000, Rnd*5000)-(Rnd * 5000, Rnd * 5000), , B
End Select
'Çizimleri image kutusunda da göster
image1.Picture = form1.Image
End Sub

Private Sub Command1_Click ()
 Dim dosyaadi
 On Local Error GoTo hata
 dosyaadi = InputBox("Dosya Adı")
 If dosyaadi <> "" Then'İptal seçilmedi ise
  SavePicture form1.Image, dosyaadi
 End If
Exit Sub
hata:
 MsgBox (dosyaadi & " geçerli bir isim değil")
 Exit Sub
End Sub
```

Örnekte görüldüğü gibi formun Resize olayına koyduğumuz kodla formun - minimize durumu hariç- boyutları değiştiğinde kontrollerimizin formun sol ve sağ üst köşelerinde kalmasını sağladık. Ayrıca form üzerine yaptığımız bütün çizimlerin Image kutusu içerisinde de görülmesini sağladık. Image kontrolünün Stretch özelliğini True yaptığımız için form üzerindeki çizimler küçültülerek Image içerisinde gösteriliyor. Form üzerindeki çizimleri kaydetmek için de *SavePicture* komutunu kullandık. Bu komutun birinci parametresi kaydedilecek resim, ikincisi de dosya adıdır. Bu komut form üzerine yapılan çizimlerin BMP formatında kullanıcının girdiği dosyaya kaydeder.

MDIChild

Windows'un formlara verebildiği diğer bir özellik de form içinde form oluşturulabilmesidir. Bu formlara örnek olarak program yöneticisini verebiliriz. Program yöneticisinin bir tane ana formu varken bu forma bağımlı olan "Program Grubu" olarak adlandırılan Childleri de vardır. Bu Childler form içinde kalırlar. Programınızda bir MDI form varsa diğer formların MDIChild özelliğini True yaparak bu tür Child formlar oluşturulabilir. Bu konu MDI Formlar kısmında ayrıca anlatılmıştır.

hDC

Windows altında oluşturulan kontrollerin birer handle numarası olduğunu, Windows'un kontrollere bu numara ile ulaştığını ve VB'de bu numaranın hWnd özelliği ile öğrenilebileceğini daha önce görmüştük. hDC numarası da bir handle numarasıdır ancak özel bir numaradır. Bu numara Windows uygulamaları ile Device

Driver (birim sürücüleri, ekran, yazıcı gibi) arasında bir bağlantı kurar. Bu numara da hWnd özelliği gibi API'lerde kullanılır.

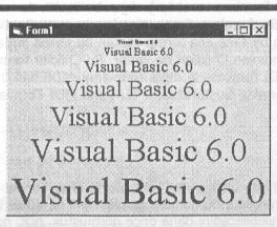
Handle numarası Windows tarafından verilen herhangi bir numaradır ve o kontrol yok edilene kadar aynıdır. Yani program çalıştığı sürece bu sayı sabittir ancak program ikinci kez çalıştığında önceki numara ile aynı numara değildir. Ancak hDC numarası hiç bir zaman sabit değildir. Yani program çalışırken bu numara değişebilir. Dolayısıyla bu numara bir değişkende saklanarak program boyunca kullanılamaz, her kullanıştan önce tekrar okunmalıdır.

Bu numaranın kullanıldığı API çağrılarında, ilgili nesnenin AutoReDraw özelliği True yapılmalıdır.

CurrentX, CurrentY

Form üzerine yapılan başlangıç noktası olmayan çizim ve yazımlar aktif pixelin bulunduğu bölgeden başlar. Bu aktif pixelin x ve y koordinatlarını CurrentX ve CurrentY özellikleriyle belirlenir. Özellikle form üzerine Print metodu ile yazacağınız yazılarda bu özellikler yazının koordinatlarını belirler.

```
ÖRNEK : Currentx
Private Sub Form_Load()
    Show
    Dim t, i
    t = "Visual Basic 6.0"
    FontName = "Times New Roman"
    For i = 1 To 10
        FontSize = i * 5
        'Yazı için formun ortasını bul
        CurrentX = (ScaleWidth - TextWidth(t)) / 2
        Print t
    Next
End Sub
```




```
ÖRNEK : Currentx,Currenty
Private Sub Form_Load()
    Show
    Dim t, i, x1, y1
    t = "İhsan Karagülle"
    FontName = "Times New Roman"
    FontSize = 15
    For i = 0 To 135
        x1 = ScaleWidth / 2 + 30 * i * Cos(i * 0.1)
        y1 = ScaleHeight / 3 + 30 * i * Sin(i * 0.1)
        ForeColor = QBColor(i Mod 16)
        CurrentX = x1
        CurrentY = y1
        Print t
    Next
End Sub
```

```
'ÖRNEK : Currentx,Currenty
Private Sub Form_Load()
    Show
    Dim t, i, x1, y1, x, y
    x = 100'yazının koordinatları
    y = 100
    t = "İhsan Karagülle  Zeydin Pala"
    FontName = "Times New Roman"
    FontSize = 15
    For i = 0 To 15
        x1 = x + 10 * i
        y1 = y + 10 * i
        ForeColor = QBColor(i)
        CurrentX = x1
        CurrentY = y1
        Print t
  Next
End Sub
```

Events

Load( )

Form ilk defa belleğe yüklenirken bu olay meydana gelir. Dolayısıyla program başlarken yapması gereken işler genellikle bu olayda yaptırılır.

- Formun **Visible** özelliği ile gizlenip tekrar gösterilmesi bu olayı meydana getirmez, çünkü form zaten bellektedir.
- Programınızda birkaç tane form varsa program çalışmaya başladığında sadece ana form görülecektir. Diğer formları **Show** metodu ile sizin göstermeniz gerekir.
- **Load** olayı, formla ilgili ilk çalışan ve bir daha çalışmayan olay olması sebebiyle form ekranda gösterilmeden önce form üzerinde yapılacak işlemler (kontrollerin yerlerini değiştirmek, kontrollere ilk değerler vermek gibi) burada yapılabilir.
- Bu olayın çalışmasını bitirdikten sonra form ekranda görülür. Bu olay bitmeden görüntülenmesi için **Show** metodunu kullanmanız gerekir.
- Formun **Load** olayına yazdığınız kodla bir nesneye set focus yapamazsınız. **Load** olayı çalışmasını bitirmeden form ekranda görülmeyeceği için klavye kontrolünü o nesneye bırakmaya çalışmak hataya sebep olur.
- Ayrıca formun **Load** olayında, ekran üzerine metodlar kullanılarak yapılan yazım ve çizimler (Print, Line gibi) görülmez. Bu problemleri aşmak için **Load** olayında **Show** metodu ile formun olay bitmeden gösterilmesini sağlayabilirsiniz veya formun **Autoredraw** özelliğini **True** yapabilirsiniz.
- Bir form belleğe yüklenirken sırasıyla şu olaylar meydana gelir: **Load, Initialize, Resize, Activate, Paint**.

Activate( ), Deactivate( )

Programınızda bir kaç form varsa bu formlardan aynı anda yalnız biri aktiftir. Aktivitenin programdaki formlardan diğerine geçmesi durumunda aktiviteyi kaybeden formun **Deactivate** olayı, aktif olan formun da **Activate** olayı meydana gelir. Bu olaylar yalnız sizin programınızdaki formlar arası geçişte meydana gelir. Windows altında çalışan başka bir programın aktiviteyi ele geçirmesi veya kaybetmesi bu olayları meydana getirmez.

Unload(Cancel As Integer)

Form herhangi bir şekilde kapatılırken bu olay meydana gelir. Bu olaya yazacağınız kodla program kapatılmadan önce yapılması gereken işleri yapabilirsiniz. Ayrıca **Cancel = True** yazarak ta formun kapatılmasını önleyebilirsiniz. Formun **Unload** olayı şu hallerde meydana gelir:.

- Formun sol üst köşesindeki kontrol kutusunda "kapat" seçildiğinde,
- Programda **Unload** komutu kullanıldığında,
- Windows Görev Yöneticisinden (Task Manager) "Göreve Son Ver" düğmesinin seçilmesiyle,
- Windows'tan çıkılmaya çalışıldığında,
- Form bir MDIChild ise MDI formun kapatılmasıyla.

Bu olaylardan herhangi birinin yapılmasıyla Windows formun **Unload** olayını çağırır. Bu olay içerisinde **Cancel = True** yapmazsanız formunuz kapanır.

Programın herhangi bir yerinden **End** komutu ile sonlandırılması durumunda **UnLoad** olayı gerçekleşmez. Bu yüzden **UnLoad** olayına kod yazdığınız programlarda **End** kodu ile programı sona erdirmemelisiniz. Bunun yerine **Unload Me** komutunu kullanabilirsiniz.

ÖRNEK: Örnek olarak çıkarken kaydedeyim mi diye sormasını istiyorsak bu kod için en uygun yer **Unload** olayıdır.

```
'ÖRNEK : Unload
Private  Form_Unload( Cancel As Integer )
   dim c
   'Evet,Hayır,Iptal düğmeleri olan ? iconlu mesaj kutusu
   c=msgbox("Çıkmadan önce çizim kaydedilsinmi", 3+32 , "Çıkış")
   select case c
         case 6 : 'Evet seçildi
                  'Kayıt için gerekli kodu buraya yazın
         case 7 : 'Hayır seçildi herhangi bir işleme gerek yok
         case 2 : 'İptal seçildi
                  Cancel = True 'formu kapatma
   End Select
End Sub
```

QueryUnload(Cancel As Integer, UnloadMode As Integer)

UnLoad olayıyla aynı işi yapar ancak formun kimin tarafından kapatılmaya çalışıldığını da öğrenebilirsiniz. Ayrıca bu olay **UnLoad** olayından önce meydana gelir ve burada **Cancel = True** ile kapatma olayı iptal edilirse **UnLoad** olayı meydana gelmez.

UnLoad olayında bir formun 5 değişik şekilde kapanacağını belirtmiştik. **QueryUnload** olayındaki **UnloadMode** parametresiyle formun hangi şekilde kapatılmaya çalışıldığını öğrenebilirsiniz. Bu parametrenin alacağı değerler şöyledir.

UnloadMode		Anlamı
0	vbFormControlMenu	Kontrol kutusunda "kapat" seçildi
1	vbFormCode	Programda Unload komutu kullanıldı
2	vbAppWindows	Windows'tan çıkılmaya çalışıldı
3	vbAppTaskManager	Task Managerden "Göreve Son Ver" seçildi
4	vbFormMDIForm	-Form bir MDIChild ise- MDI form kapatıldı

ÖRNEK: Örnek olarak doğru şifre girilmediği sürece Windows'tan çıkışı önleyecek bir program yapalım.

Windows'u kapatmak istediğinizde, Windows açık olan bütün uygulamalara kendini kapatmalarını bildirir. Yani programların **QueryUnload** ve **Unload** olayları aktif hale getirilir. O halde biz Windows'tan çıkılacağını anlayabilmek için yazacağımız programın **QueryUnload** olayına gerekli kodu yazmamız gerekir. Windows kapatılmak istendiğinde programımız o anda bellekte ise **QueryUnload** olayı aktif hale gelecek ve doğru şifre girilmezse Windows kapatılmayacaktır. Programı sürekli bellekte tutmak içinde "Başlangıçta" grubuna koymamız gerekir. Ayrıca programın ekranda görülmemesi içinde formu gizlemeliyiz.

```
'ÖRNEK : QueryUnload
Private Form_Load( )
 'Formu gizle
 Form1.Hide 'Form1.Visible = False ile aynı anlamda
End Sub

Private  Form_QueryUnload( Cancel As Integer, UnloadMode As
Integer )
 Select Case unloadmode
  Case 0: 'form görülmediği için kullanıcının formu bu yöntemle
kapatması mümkün değil
  Case 1: 'Program kendini kapatabilir
  Case 2: 'Windows kapatılmaya çalışılıyor
   If InputBox("Çıkış için şifre:") <> "Ahlat" Then cancel = True
  Case 3: 'Task Manager kapatmaya çalışıyorsa
   MsgBox ("Windows çalıştığı sürece ben de çalışmak zorundayım")
   cancel = True
 End Select
End Sub
```


Resize( )

Formun boyutlarının değişmesi halinde bu olay meydana gelir. Formun genişliğinin veya yüksekliğinin değişmesi, ayrıca formun **Minimize** edilmesi bu ola meydana getirir.

KeyPress, KeyDown, KeyUp

Formun Key olaylarının diğer key olaylarından önemli bir farkı, formu **KeyPreview** özelliği **True** ise form üzerinde basılan bütün tuşları, tuşun bas dığı kontrolün key olaylarından önce formun Key olaylarının algılamasıdır.

Eğer programda kullandığınız genel tuşlar varsa bunların kontrolüne bu olayla yazacağınız kodla sağlayabilirsiniz.

Methods

Move sol, üst, genişlik, yükseklik

Formun veya herhangi bir kontrolün konumunu ve boyutlarını tek seferde değ tirir.

Kontrolün Left, Top, Width, Height propertiesleriyle dört ayrı seferde yapaca nız işi bu metodla tek seferde yapabilirsiniz. Propertieslerle yapmanız durumu da kontrol dört defa şekil değiştirecek ve dolayısıyla daha ağır ve çirkin bir g rüntü oluşacaktır.

Aşağıdaki her iki komut grubu da aynı işi yapar.

```
'Propertieslerle
Form1.Left = 100
Form1.Top = 100
Form1.Width = Screen.Width - 200
Form1.Height = Screen.Height - 200
```

```
'Metodla
form1.Move 100, 100, Screen.Width - 200, Screen.Height - 200
```

Hide

Formun **Visible** özelliğinin False **yapılmasını** yani formun ekranda görülmem sini sağlar.

Show [stil]

Formun **Visible** özelliğinin **True** yapılması gibidir. Ancak parametre ile kullanılırsa formun gösterim şeklini de etkiler

Burada **stil** kullanılmazsa veya 0 kullanılırsa **Visible = True** ile aynı işi yapar.

Stil değeri 1 verilirse form **Modal** olarak gösterilir. Form modal ise, form gizlenmeden veya kapatılmadan program içindeki diğer formlara ulaşım engellenir, yani form ekranda olduğu sürece kontrolü elinde tutar, programınızdaki diğer formlara geçişi engeller. (Mesaj pencerelerinde olduğu gibi)

PrintForm

Form üzerinde görülen bütün kontrollerin yazdırılmasını sağlar. Form ve **PictureBox** üzerindeki methodlarla yapılan yazım ve çizimlerin de yazdırılabilmesi için **AutoRedraw** özelliklerinin **True** olması gerekir. Bu özellikler **True** ise Form üzerinde görülen menüler ve Form çerçevesi hariç her şey aynen yazıcıdan çıkar.

Refresh

Form üzerinde yapılan değişikliklerin anında gösterilmesini sağlar. Normalde form veya başka bir kontrol (Diskle ilgili kontroller hariç) üzerinde yapılan değişiklikler, kontrol Windows'a geçtiğinde Windows tarafından veya kontrol tarafından yapılır. Ancak yapılan değişikliğin anında ekranda görüntülenmesini istiyorsanız bu metodla kontrole kendini yenilemesini söyleyebilirsiniz.

ÖRNEK: Örnek olarak bir liste kutusuna 1000 tane eleman eklemek istediğimizi düşünelim.

```
For i = 1 to 1000
 List1.Additem i * 0.005 /100 + i * 10.548 /9.55
Next
```

Bu kodla belirli bir süre beklemeden sonra listeye elemanların tümünün aynı anda eklendiğini görürsünüz. Çünkü For-Next döngüsü bitmeden kontrol Windows'a geçmeyeceği için listeye eleman eklendiği anda listede görülmeyecektir. Ancak döngü bittikten sonra kontrol Windows'a geçecek ve elemanların tümü aynı anda listeye eklenecektir. Bu yöntemle döngü bitene kadar kullanıcı ekranda hiçbir değişiklik göremez ve hiçbir iş yapamaz.

Aynı kodu **Refresh** metoduyla yazıp farkı görebilirsiniz.


```
For i = 1 to 1000
 List1.Additem i * 0.005 /100 + i * 10.548 /9.55
 List1.Refresh 'Liste kutusunu güncelle
Next
```

Bu kodda, List kutusuna kendisini güncellemesini söylediğimiz için her eleman eklendiğinde List kutusu kendini yenileyecektir ve dolayısıyla bu kod biraz daha yavaş çalışacaktır. Ancak kullanıcı bekleme süresi içinde ekranda yapılan işlemleri de görebilecektir. Fakat yine kullanıcı bu süre içinde hiç bir iş yapamaz.

Aynı kodu bir başka yöntemle yazarsak

```
For i = 1 to 1000
 List1.Additem i * 0.005 /100 + i * 10.548 /9.55
 Doevents 'Kontrolü Windows'a ver ve işlemlerin yapılmasını sağla
Next
```

Bu kod aslında böyle uzun işlemler yapan kodlar için en ideal fakat en yavaş çözümdür. Çünkü burada **DoEvents** komutuyla kontrol Windows'a bırakılmaktadır. Böylece hem liste kutusu kendini güncellemekte, hem sistemde çalışan diğer programlarında çalışmaya devam etmesini sağlamakta ve hem de kullanıcının başka işler de yapabilmesini sağlamaktadır.

Bu örnekte görüldüğü gibi uzun işlemler yapan bir kodunuz varsa, kod aralarında **Doevents** komutunu kullanarak kontrolün Windows'a geçmesini mutlaka sağlayınız.

Refresh metodu bazı kontrollerde zorunlu olarak kullanılmalıdır. Bu kontroller: FileListBox , DirectoryListBox ve diskle ilgili diğer kontrollerdir. Örneğin formunuzda FileListBox varsa bu kontrolün gösterdiği dizinden bir dosya silinmesi veya eklenmesi halinde bu dosyanın listeden çıkarılması veya eklenmesini kontrole Refresh metoduyla bildirmeniz gerekir.

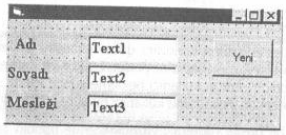
```
Kill "gereksiz.sil"
File1.Refresh
```

SetFocus

Klavye kontrolünün **SetFocus** yapılan nesneye geçmesini sağlar. Bir nesneye **SetFocus** metodunu uygulayabilmek için o nesnenin **Enabled** özelliğinin **True** olması gerekir aksi takdirde hata oluşacaktır.

Bu özellik daha kolay kullanılabilir arabirimler oluşturmak için faydalıdır.

ÖRNEK: Örneğin aşağıdaki gibi basit bir formumuz olsun.



Kullanıcı "Yeni" düğmesini tıklayınca text kutularının içeriğini boşaltalım. Kullanıcının bu düğmeyi tıklamasıyla kontrol komut düğmesine geçecek ve orada kalacaktır. Halbuki "Yeni" düğmesi tıklandıktan sonra tekrar aynı düğmenin tıklanması mantıklı olmayacağı için kontrolün komut düğmesinde değil Text1 kutusunda olması programın daha kullanılır olmasını sağlayacaktır.

```
'ÖRNEK : SetFocus
Private Sub Command1_Click ()
  text1 = ""
  text2 = ""
  text3 = ""
  text1.SetFocus'Kontrolü text1 kutusuna ver
End Sub
```

Bu örnek küçük ve basit olduğundan kontrolü bıraktığımız Text1 kutusunun enabled özelliği kontrol etme ihtiyacı duymadık. Ancak program daha karışık ise klavye kontrolünün devredilmek istendiği nesne programın herhangi bir yerinde pasif yapılmış olabilir. Böyle durumlarda nesnenin pasif olmadığını kontrol ettikten sonra klavye kontrolünün verilmesi gerekir.

```
If Text1.Enabled Then Text1.SetFocus
```

Zorder mod

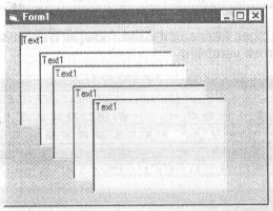
Bu metod kullanıldığı nesneyi en öne veya en arkaya getirmek için kullanılır. Burada modun 0 olması kontrolün diğerlerinin önüne, 1 olması arkasına getirilmesini sağlar.

Formlar tıklandığında kendiliğinden en öne gelecektir. Ancak tıklama yapılmadan en öne getirilmek veya en arkaya götürülmek istenirse bu metod kullanılır.

Bu metod daha çok kendiliğinden öne geçmeyen kontroller için kullanılması gerekir.

ÖRNEK: Örnek olarak 5 tane büyük text kutusuna ihtiyacımız olduğunu ancak formumuzda o kadar yer olmadığını düşünelim. Text kutularımız aşağıdaki gibi

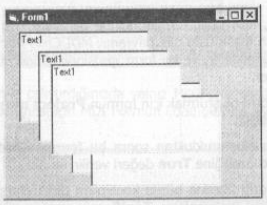
yerleştirdikten sonra **Click** olaylarına aşağıdaki kodu yazarak hangisi tıklanırsa onun en öne gelmesini sağlayabiliriz.




```
Private Sub Text1_Click(Index As Integer)
  Text1(Index).ZOrder 0
End Sub
```

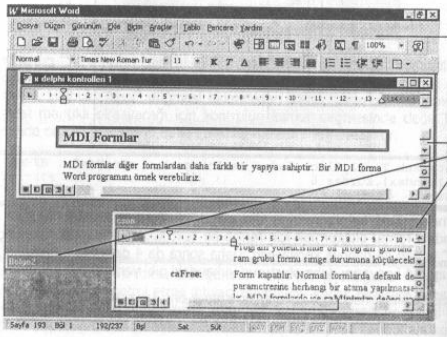
Bu örnekte Text kutularını birer dizi olarak oluşturarak koddan tasarruf ettik. Yukarıdaki formu oluşturmak için forma bir text kutusu yerleştirip onu seçin ve Ctrl+C tuşları ile panoya kopyalayın. Daha sonra da 4 defa Ctrl+V tuşuna basarak bunun 4 kopyasını daha oluşturup yukarıdaki gibi yerleştirin.

Böylece kullanıcı hangisini kullanmak isterse onu tıklayacak ve o en öne gelecektir.



MDI FORMLAR

MDI formlar diğer formlardan daha farklı bir yapıya sahiptir. MDI forma Word programını örnek verebiliriz.



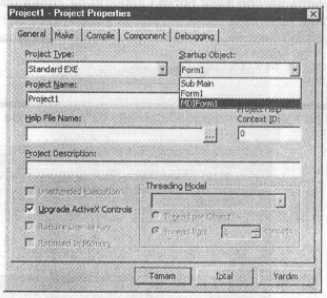
MDI Form, Wordün kendi ekranıdır **Child Form**, açılan dokümanların pencere-

Bir MDI form, MDI Child formları içinde barındırır. Yani bir nevi MDI form, MDI Child formların masa üstü gibi davranır. MDI Child formlar ancak bir MDI form içerisinde bulunabilirler, ve bu form içerisinde simge durumuna küçültülebilir veya taşınabilirler.

- Bir MDI form oluşturmak için formun **Project** menüsünden **Add MDI Form** komutu seçilir.
- MDI form oluşturulduktan sonra bu formun Child Formu olacak formların **MDIChild** özelliğine **True** değeri verilir.
- Bir MDI form üzerine bütün kontrolleri direk olarak yerleştiremezsiniz. MDI form üzerine **PictureBox, ToolBox** kontrolleri gibi **Align** özelliği olan kontroller yerleştirilebilir. Diğer kontroller ise ancak bu kontrollerin içine yerleştirilebilirler. Ve bu iki kontrol formun masa üstünde yer işgal eder yani masa üstünü Child formlarla paylaşırlar. Bir MDI form üzerine bütün formu kaplayacak şekilde bir PictureBox yerleştirirseniz MDI Child formları ekranda göremeyeceksiniz.

ÖRNEK: Bir örneği adım adım geliştirelim.

- Öncelikle yeni bir proje başlatın. Şu anda programa ait bir formun projede olması gerekiyor.
- Yeni bir MDI formu da **Project** menüsünden **Add MDI Form** seçeneği ile oluşturalım.
- Projede bulunan Form1'in **MDIChild** özelliğini de **True** yaparak bir MDIChild oluşmasını sağlayalım.
- Ayrıca programın MDIForm'dan başlaması içinde **Project** menüsünden **Project Properties** seçeneği ile açılan aşağıdaki pencerenin **General** kısmındaki **Startup Object** seçeneğini **MDIForm1** yapalım.



- Programı bu haliyle çalıştırdığınızda yalnız MDIForm görülecektir. MDIChild formun da görülebilmesi için MDI Formun Load olayına aşağıdaki kodu ekleyin:

```
Private Sub MDIForm_Load
  Form1.Show
End Sub
```

- Bu MDIChild formların yenisini oluşturmak için, yeni formlar oluşturulup **MDIChild** özellikleri **True** yapılabilir. Ancak eğer bu Child formlar aynı kontrollere sahip olacaksa bu yeni formlar elimizdeki MDIChild formdan türetilebilir. Bu iş için Dim değimini şu şekilde kullanacağız.

Dim formadi As New OrjinalForm

Formadi: Oluşturulacak yeni formun adı

OrjinalForm: Türetilecek formun adı

```
Dim form As New form1
```

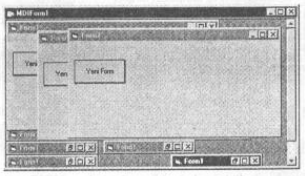
Yukarıdaki komutla Form1 formundan, Form isimli yeni bir form oluşturulabilir. Bu komut o formu oluşturur ancak görüntülemez. Görüntülemek için de aşağıdaki komutu kullanacağız.

```
Form.Show
```

- Şimdi bu komutları kullanarak programımıza yeni formlar oluşturalım. Bunun için Form1 üzerine **Yeni Form** Caption'lu bir komut düğmesi ve bu düğmenin **Click** koduna da aşağıdaki kodları yazalım.

```
Private Sub Command1_Click ( )
  Dim form As New form1
  form.Show
End Sub
```

Programı bu haliyle çalıştırıp aşağıdaki gibi bir görüntü elde edebilirsiniz.



Burada görüldüğü gibi Form1'den türetilen bütün formların **Caption**'ları Form1'dir. Bunu her form için artırarak değiştirelim. Bu iş için **Forms.Count** özelliğini kullanacağız. Bu özellik programda bulunan bütün formların sayısını verir, her yeni form oluşturulduğunda bu sayı bir artacağından oluşan formların başlıkları da farklı numaralarda olacaktır. Komut düğmesinin Click olayını şu şekle çevirelim.

```
Private Sub Command1_Click ( )
  Dim form As New form1
  form.Show
  form.Caption = "Form " & (forms.Count - 1)
End Sub
```

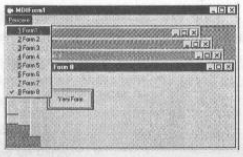

Forms.Count MDIForm dahil formların sayısını verdiği için bir eksiğini alarak Child formların sayısını bulmuş olduk. Böylece oluşan her forma Form1, Form2 gibi başlıklar verilecektir.

MDI Formda Menüler

- MDIChild formların menülerinin **WindowList** özelliği vardır. Bir menüye bu özellik verildiğinde MDI Child formların isimleri otomatik olarak bu menüye eklenir ve bu menüden seçilen form otomatik olarak aktif hale gelir. Bu işlemler için herhangi bir kod da gerekmez. Sadece ilgili menünün **WindowList** özelliğini **Menu Editor** penceresinde işaretlemek gerekir.

Bu özelliği örneğimizde kullanalım. Form1 formunu seçin ve **Tools-Menu Editor** menüleri ile aşağıdaki menu tasarım penceresini açıp; Pencere Caption'lu ve MnPencere isimli, WindowList özelliği True olan bir menü oluşturun.

Örneğimizi çalıştırın ve yeni formlar oluşturun. Oluşturduğunuz her formun Pencere menüsüne otomatik olarak eklendiğini, ve bunlardan birini seçtiğinizde de bunun aktif hale geldiğini göreceksiniz.

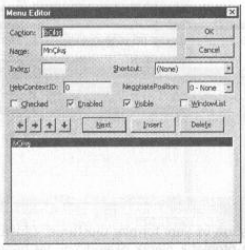


Programı çalıştırıp yeni formlar ekledikçe bu formların başlıklarının **Pencere** menüsüne eklendiğini göreceksiniz. Ve hiç bir koda ihtiyaç duymadan menüden bir formun seçildiğinde o formun aktif olduğunu da göreceksiniz.

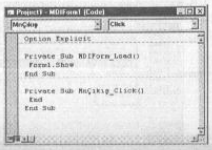
- MDI Formun bir çok özelliğinin diğer kontrollerin özelliklerinden bir farkı yoktur. Yalnız menülerde bir farklılık vardır. MDI formun menüleri olabilir ancak bu menüler bir MDIChild menü yokken menü çubuğunda görülür. Eğer bir MDIChild form varsa o Child formun menüleri menü çubuğu yerine geçer.

Şimdi de programı çalıştırıp komut düğmesi vasıtası ile birkaç form oluşturduktan sonra bütün Child formları kapatın. Ekranda sadece MDI formun kaldığını ve hiçbir menünün bulunmadığını göreceksiniz. Halbuki bu durumda dahi bazı işlemleri yapacak menülerin bulunması gerekir.

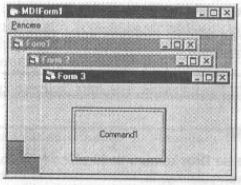
MDI Formumuza da Caption'u Çıkış ve ismi de MnÇıkış olan bir menü oluşturun. Bunun için MDI Forma geçin ve **Tools-Menu Editor** menüleri ile aşağıdaki menu tasarım penceresini kullanın.



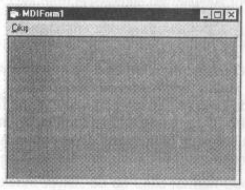
Çıkış işlemi için Menünün Click olayına **End** kodunu koyun.



Programı çalıştırın. Ekranda MDI Child formlar varken aşağıdaki gibi ona ait menüyü menü çubuğunda göreceksiniz.



Bütün MDI Child formları kapattıktan sonra ise aşağıdaki gibi MDI formumuza ait Çıkış menüsü görülecektir.



Properties

ScrollBars

Diğer formlardan farklı olarak MDI formlara kaydırma çubukları bu özellik vasıtasıyla verilebilir.

0: vbSBNone: Yok
1: vbVertical: Dikey
2: vbHorizontal: Yatay
3: vbBoth: Dikey ve yatay

ActiveForm

Screen nesnesindeki **ActiveForm** özelliği gibidir. O anda aktif olan MDI Child formu bu özellik temsil eder.

Örneğin aktif olan formun başlığına saati yazmak istersek aşağıdaki kodu bir Timer olayına yazabiliriz.

```
MDIForm1.ActiveForm.Caption = Time
```

AutoShowChildren

Bu özellik **False** ise **Dim** komutu ile türetilen MDI Child formlar **Load** komutu ile belleğe gizli olarak yüklenebilir.

Events

Unload

MDI veya MDI Child formların **Unload** olayının, normal bir Formun **Unload** olayından bir farkı yoktur. Ancak özel bir durum olarak bir MDI Child form kapatılmak istendiğinde formu bellekten çıkarmak yerine formu simge durumuna getirmek isteyebiliriz. O zaman MDI Child formun **Unload** olayına **Cancel = True** yazarak formun bellekten çıkarılmasını önleyebiliriz. Aynı zamanda burada formu minimize durumuna getirecek kodu da yazmamız gerekecektir. Olması gereken kod şöyledir:

```
FormAdi.WindowState = 1 'minimize
Cancel = True 'kapatmayı iptal et
```

Ancak eğer MDI Child formumuz örneğimizde olduğu gibi bir tek formdan türetilmişse kapatılmak istenen formun hangi form olduğunu bulmak yerine **Me** kelimesini kullanabiliriz. Bu kelime o anda aktif olan formun ismi yerine geçer.

Yani **Me.WindowState = 1** komutu ile o anki form hangisi olursa olsun bu formu minimize hale getirebiliriz.

Örneğimizdeki Form1'in **UnLoad** olayına aşağıdaki kodu yazarak form kapatılmak istendiğinde formun minimize hale gelmesini sağlayalım.

```
Private Sub Form_Unload (cancel As Integer)
  Me.WindowState = 1'minimize
  cancel = True'Kapatma işlemini iptal et
End Sub
```


Programımızı bu haliyle çalıştırırsanız artık MDI Formların kapanmadığını, minimize durumuna geldiğini göreceksiniz. Ancak artık ana formu da kapatamayacaksınız. Çünkü ana formu (MDI form) kapatmak istediğinizde bu form çocuklarına kendilerini kapatmasını söyleyecektir. Bu durumda da yukarıda yazdığımız kod devreye girecek ve Cancel = True ile kapatma işlemi iptal edilecektir. O halde MDI formun kapatılması için MDI Formun Unload olayına veya *QueryUnload* olayına *End* kodunu yazmamız gerekecektir.

Önce birinci yöntemi deneyelim. Yani MDI Formumuzun **UnLoad** olayına **End** kodunu yazalım.

```
Private Sub MDIForm_Unload (Cancel As Integer)
  End
End Sub
```

Programı çalıştırdığınızda işlemin başarılı olmadığını göreceksiniz. Yani MDI formun **Unload** olayındaki **End** koduna rağmen MDI form kapatılmak istendiğinde kapatılamamaktadır. Bunun sebebi **Unload** ve **QueryUnload** olaylarının meydana geliş sırasıdır. Bir MDI form kapatılmak istendiğinde meydana gelen olayların sırası şöyledir.

MDIForm_QueryUnload
Form_QueryUnLoad
Form_UnLoad
MDIForm_UnLoad

Bu sıralamadan görüldüğü gibi bir MDI form kapatılmak istendiğinde önce MDI formun QueryUnLoad olayı meydana gelir. Daha sonra MDI form Child formlarına kendilerini kapatmalarını söyler. Bu da Child formun QueryUnload olayını ve daha sonra da Unload olayını meydana getirir. Bu aşamalar geçildikten sonra yani Child formlar kapatılmayı engellemedikleri takdirde artık MDI formun kapanmaması için bir sebep yoktur ve MDI formun Unload olayı meydana gelir.

Bizim örneğimizde Child formların kapatılması yerine minimize edilmesi için gerekli kodu Child formun Unload olayına koyduğumuz için yani üçüncü aşamaya koyduğumuz için kapatılma iptal edilmekte ve kontrol MDI formun Unload olayına yani dördüncü aşamaya gelememektedir. O halde bizim MDI formu kapatmak için yazacağımız End kodunun dördüncü aşamadaki Unload olayına değil, birinci aşamada meydana gelen QueryUnload olayına yazmaktır.

```
Private Sub MDIForm_QueryUnload (Cancel As Integer, UnloadMode As
Integer)
    End
End Sub
```

Methods

Arrange

Bu yöntem ile MDI form içindeki Child formlar düzenlenebilir. Şu dört değerden birini alır:

	Arrange	Anlamı
0	vbCascade	Minimize durumunda olmayan bütün Child formlar basamaklanır.
1	vbTileHorizontal	Minimize durumunda olmayan bütün Child formlar yatay olarak döşenir.
2	vbTileVertical	Minimize durumunda olmayan bütün Child formlar dikey olarak döşenir.
3	vbArrangeIcons	Minimize durumunda olan bütün Child formların simgeleri yerleştirilir.

MDIForm içindeki bütün formları yatay olarak döşemek için:

```
mdiform1.Arrange 1
```

kullanılabileceği gibi

```
mdiform1.Arrange vbTileHorizontal
```

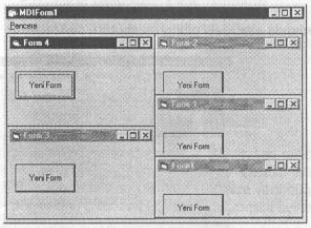
şeklinde sembolik ismi ile de kullanılabilir.

Örneğimizdeki Child form olan Form1 formunun Pencere menüsünü aşağıdaki gibi değiştirelim.

Burada dört alt menünün de ismini **MnArrange** olarak verin ve sırasıyla index'lerini 0,1,2,3 yapın. Yani menü bir dizi olarak tanımlanmaktadır. Bu da kodumuzu daha kısaltacaktır. Şimdi menünün Click olayına aşağıdaki kodu yazalım:

```
Private Sub MnArrange_Click (index As Integer)
 mdiform1.Arrange index
End Sub
```

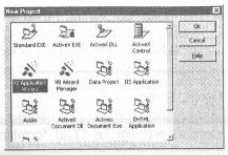
Artık pencere menüsü aracılığı ile Child formları döşeyebilir, basamaklayabilir veya simge durumunda olanların düzenlenmesini sağlayabilirsiniz. Aşağıdaki ekran görüntüsünde Child formların dikey olarak döşenmiş hali görülmektedir:



Wizard Kullanarak MDI Form Uygulaması Oluşturmak

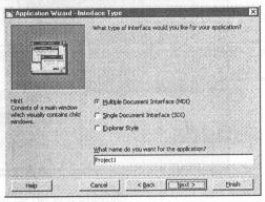
MDI formlar barındıran bir çok uygulama da genellikle standart bir çok menü ve işlem yer alır. Her seferinde bunlarla uğraşmak istemiyorsanız VB'nin wizardlarını kullanarak sizin için güzel bir MDI Form uygulamasını isteyebilirsiniz. VB gerekli bütün menüleri ve araç çubuklarını oluşturur ayrıca standart işlemler için gerekli kodu da yerleştirir. Size sadece kendi uygulamanızla ilgili kodları ve formları hazırlamak kalır.

Wizard kullanmak için **File-New Project** menüleri ile açılan aşağıdaki pencerede **VB Application Wizard** seçeneğini tıkladıktan sonra **Ok** düğmesine basın.



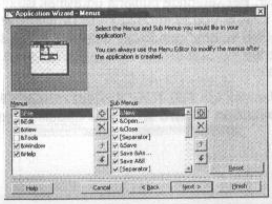
Uygulama sihirbazı devreye girecektir. Sihirbazın ilk adımını Next düğmesi ile geçin.

İkinci adımda aşağıdaki gibi projenin türü sorulacaktır:



Pencerede üç farklı seçene görüyorsunuz. İlk seçeneği kullanarak MDI form uygulaması oluşturmak istediğimizi belirtelim ve **Next** düğmesi ile sonraki adıma geçelim.

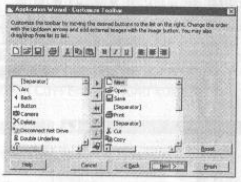
Üçüncü adımda ise projenizde bulunmasını istediğiniz menüler sorulacaktır.



Yukarıdaki pencerede bulunan menülerden kullanmak istemediğinizin başındaki işareti kaldırın. Veya yenilerini eklemek isterseniz penceredeki + düğmesine basın.

Menüleri belirledikten sonra **Next** düğmesi ile sonraki adıma geçebilirsiniz.

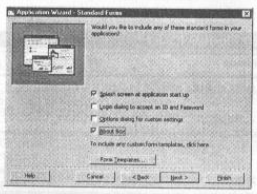
Dördüncü adımda ise projenizde bulunmasını istediğiniz araç çubukları sorulacaktır.



Projenizde kullanmak istediğiniz araç düğmelerini yukarıdaki pencereden belirleyebilirsiniz. Projenize yerleştirebileceğiniz düğmeler soldaki listede, projenize eklenmiş düğmeler ise sağdaki listede bulunur. Listenize yeni düğmeler eklemek için penceredeki > düğmesini, çıkarmak için de > düğmesini kullanabilirsiniz.

Düğmeleri belirledikten sonra **Next** düğmesi ile sonraki adıma geçebilirsiniz.

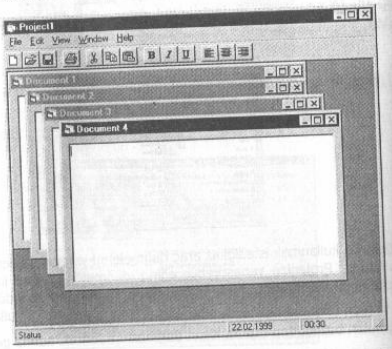
Beşinci ve **Altıncı** adımı **Next** düğmesi ile geçin. **Yedinci** adımda ise projenizde yer almasını istediğiniz hazır formlar sorulacaktır.



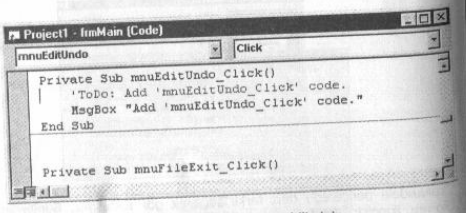
Yukarıdaki pencerede dört farklı seçenek görmektesiniz. Bunlardan ilki: Giriş formu, ikincisi: şifre formu, üçüncüsü: seçenekler formu ve dördüncüsü de hakkında formudur.

Bunlardan projenizde bulunması istediklerinizi işaretleyin ve **Finish** düğmesi
sihirbazı sona erdirin. VB sizin için güzel bir proje hazırladı.

Programı çalıştırırsanız bir çok menünün ve düğmenin üzerine düşen göre
yaptığını göreceksiniz.



Bazı menülerin yapacağı işe ait kodları doğal olarak sizin yazmanız gerekir. Z
ten gerekli yerlerde kod penceresinde "Add code" şeklinde açıklamalar göreceğ
siniz.



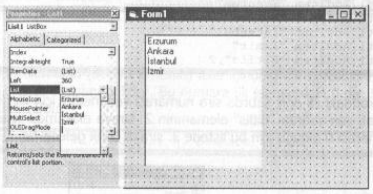
Bu noktalara programınız için gerekli kodları yazabilirsiniz.

ListBox (Listeleme Kutusu)

Visual Basic'in sağladığı dizilerinizi gösterebileceğiniz kontrollerdendir. Elemanları listelemek, sıralamak gibi özellikler sunan genel amaçlı bir kontroldür. DriveListBox, DirectoryListBox, FileListBox gibi türevleri vardır.

Listeye eleman eklemek için **AddItem** metodu kullanılabileceği gibi, tasarım anında **List** özelliği de kullanılabilir.

Eğer listeye tasarım anındayken elemanlar eklemek isterseniz Properties pencesinden List özelliğini aşağı doğru açın ve elemanları yazın. Ancak elemanlar arasında Enter tuşunu değil Ctrl+Enter tuşunu kullanın.



Methods

AddItem Eleman, SıraNo

Listeye eleman ilave etmek için kullanılan bir metoddur. Liste sıralı değilse eleman sona, sıralı ise alfabetik sıradaki yerine eklenir.

```
List1.AddItem "Erzurum"
List1.AddItem "Ankara"
List1.AddItem "İstanbul"
List1.AddItem "İzmir"
```

Bu kodlar aracılığı ile, aşağıdaki gibi bütün elemanlar eklendiği sıraya yerleştirilecektir.

Eğer elemanı sona değil de belli bir sıraya yerleştirmek isterseniz son parametre olarak yerleştirileceği sıra numarasını verebilirsiniz.

```
List1.AddItem "Erzurum"
List1.AddItem "Ankara"
List1.AddItem "İstanbul"
List1.AddItem "İzmir"
List1.AddItem "Bitlis",2
```

Bu koddaki ilk dört satırda sıra numarası verilmediği için ilk dört eleman eklenir. Ancak son satırda "Bitlis" elemanının 2. sıraya eklenmesi istenmektedir. İlk sıra numarası 0 olduğu için bu listede 3. sıraya denk gelecektir.

RemoveItem Index

Listedeki index nolu elemanı listeden çıkarmaya yarayan bir metoddur. İlk elemanın index numarası 0 olduğu için elemanların listedeki yerinin bir eksiği index olarak verilmelidir.

```
List1.RemoveItem 1'2. elemanı listeden çıkarır
List1.RemoveItem List1.Listindex'Seçili elemanı listeden çıkarır
```


Clear

Listedeki bütün elemanları siler, liste boşalır.

```
List1.Clear 'Liste boşaltıldı
```

Properties

ListCount

Liste içindeki eleman sayısını verir.

```
Label1.Caption="Listede " & List1.ListCount & " eleman var"
```

ListIndex

Listedeki aktif elemanın (seçili elemanın) liste içindeki numarasını verir veya listedeki aktif olacak elemanı belirler. Bu numara ilk eleman için 0 dır. Son eleman için ListCount - 1'dir.

```
List1.ListIndex = 5 'Listedeki 6. Elemanı seçer.
```

List(index)

Liste içindeki index numaralı elemanın değerini öğrenmek veya değiştirmek için kullanılır.

```
List1.List(1) = "Bursa" '2.elemanı Bursa yapar.
EskiDeger = List1.List(5) '5. Elemanı değişkene al
```

Bu özellik tasarım zamanında listeye eleman ilave etmek için de kullanılır. Properties penceresinden List özelliğini aşağı doğru açarak açılan listeye listede bulunmasını istediğiniz elemanları girebilirsiniz. Elemanları girerken bir alt satıra geçmek için Enter yerine Ctrl-Enter tuşlarını kullanmanız gerekir.

Text

Liste içinde seçili olan elemanın içeriğini verir. Sadece okunabilir bir özelliktir.

```
Print List1.Text
```

ile

```
Print List1.List(List1.ListIndex)
```

aynı sonucu verir.

MultiSelect

Bu özellik bir kutu içinde birden fazla elemanı seçme imkanı verir. İki değişik modu vardır:

0: Birden fazla eleman seçimi yapılamaz.
1: Mouse ile tıklanan her eleman seçilir veya seçilmişse seçilmişliği kaldırılır.
2: Bu modda Shift veya Ctrl tuşu basılı tutularak birden fazla seçim yapılır.

Bu özellik sadece tasarım zamanında properties penceresinden değiştirilebilir. Çalışma zamanında yapılacak atam ile listenin MultiSelect özelliği değiştirilemez.

Bu özellik aktif hale getirilmişse seçili olan elemanları öğrenmek için SelCount ve Selected özellikleri kullanılır.

SelCount, Selected (Index)

MultiSelect özelliği 0 olmayan listelerde birden fazla eleman seçilebileceği için **SelCount** özelliği ile seçili eleman sayısı, **Selected (Index)** özelliği ile de index numaralı elmanın seçili olup, olmadığı öğrenilebilir.

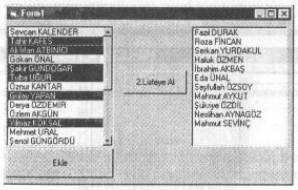
Seçili olan elemanları öğrenebilmek için listeyi baştan sona tarayacak bir döngü oluşturulur ve **Selected** özelliği ile her elemanın seçili olup olmadığı test edilir.

Aşağıdaki gibi bir kodla seçili elemanlar öğrenilebilir.


```
Dim i
For i=0 to List1.ListCount-1
 If List1.Selected(i) Then
       Print List1.List(i) ; "  elemanı seçili"
 End If
Next
```

ÖRNEK: Örnek olarak birinci listede seçili olan elemanları ikinci listeye aktaracak bir program yazalım. Örneğimiz için iki liste yerleştirin ve birinci listenin **MultiSelect** özelliğini 1 yapın. Ayrıca iki tane de komut düğmesi yerleştirin.

```
'ÖRNEK:Multiselect, Selected
Private Sub Command1_Click()
 Dim i
 For i = 0 To List1.ListCount - 1
  If List1.Selected(i) Then
   'seçili elemanı ikinci listeye ekle
   List2.AddItem List1.List(i)
  End If
 Next
End Sub

Private Sub Command2_Click()
 List1.AddItem InputBox("Eklenecek Eleman")
End Sub
```



Sorted

Listenin alfabetik sıralı olmasını sağlar. Listeye eklenen elemanları sona değil alfabetik sıraya yerleştir.

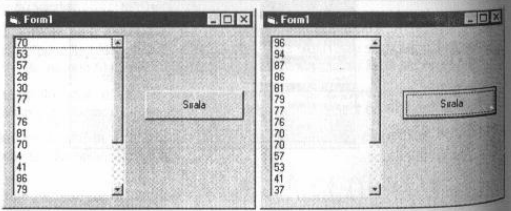
Ancak bu sıralama işlemi sayılar üzerinde doğru etkiyi göstermez. Çünkü alfabetik olarak 10 sayısı 2 sayısından önce gelir. Eğer sayıları sıralamak istiyorsanız sıralayacak kodu kendiniz yazmanız gerekecektir.

ÖRNEK: Örnek olarak bir listedeki sayıları büyükten küçüğe doğru sıralayacak program yapalım. Örneğimiz için forma bir liste kutusu ve bir komut düğmesi

yerleştirin. Ancak liste kutusunun **Sorted** özelliğini **True** yapmayın. Çünkü sıralama yapacak kodu kendimiz yazacağız.

Program çalıştığında listeye rasgele 20 tane rakam ekleyecek ve komut düğmesinin tıklanmasıyla bunları sıralayacaktır.

```
'ÖRNEK: Sayıları Sıralama
Private Sub Form_Load()
 Dim i
 'Listeye 20 tane rasgele sayı ekle
 For i = 1 To 20
  List1.AddItem Int(Rnd * 100)
 Next
End Sub

Private Sub Command1_Click()
 Dim i, j, c
 For i = 0 To List1.ListCount - 1
  For j = i To List1.ListCount - 1
   If Val(List1.List(j)) > Val(List1.List(i)) Then
    'elemanların yerini değiştir
    c = List1.List(i)
    List1.List(i) = List1.List(j)
    List1.List(j) = c
   End If
  Next
 Next
End Sub
```



Bu algoritmanın mantığı şöyledir: Listedeki her elemanı kendinden sonraki bütün elemanlarla karşılaştırır. Eğer kendinden daha büyük bir elemana rastlarsa onunla yer değiştirir. Bu işlem listedeki bütün elemanlar için tekrarlandığında liste büyükten küçüğe doğru sıralanmış olur.

- Eğer küçükten büyüğe doğru değiştirmek isterseniz if şartındaki > işaretini < olarak değiştirmeniz yeterlidir.

Bu örnekte basit bir sıralama algoritması kullandık. Eğer listedeki eleman sa fazla ise daha iyi bir algoritma kullanmanızı öneririz.

Ayrıca hangi algoritmayı kullanırsanız kullanın, sıralamayı liste üzerinde yapı yerine bir dizide yaptırırsanız işlem daha hızlı olacaktır. Yukarıdaki örneği bir içinde sıralattığınızda programın en az 20 kat daha hızlı çalıştığını göreceksini

Yukarıdaki örneği dizi ile yaparsak kod aşağıdaki gibi olacaktır.

```
Private Sub Command1_Click()
 Dim i, j, c
 ReDim x(List1.ListCount - 1)
 'listeyi diziye al
 For i = 0 To List1.ListCount - 1
  x(i) = List1.List(i)
 Next
 'diziyi sırala
 For i = 0 To List1.ListCount - 1
  For j = i To List1.ListCount - 1
   If Val(x(j)) > Val(x(i)) Then
    'elemanların yerini değiştir
    c = x(i)
    x(i) = x(j)
    x(j) = c
   End If
  Next
 Next
 'Diziyi listeye al
 For i = 0 To List1.ListCount - 1
  List1.List(i) = x(i)
 Next
End Sub
```

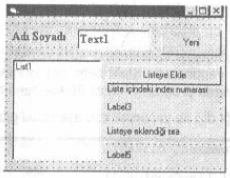
NewIndex

Listeye en son eklenen elemanın listindexini verir. Listenin **Sorted** özelliği **Fa** ise bu değer **ListCount - 1**'dir, **True** ise elemanın, sıralanmış listedeki yer indexidir.

ItemData(Index)

Index nolu elemana, listede görülmeyecek bir numara verilmesini sağlar. S lanmış listelerde herhangi bir elemanın listeye konulduğu sıra belli olamaya için bu özellik kullanılarak listeye konulan elemana sıra numarası verilebilir.

Örnek olarak sıralanmış bir isim listesi içindeki isimlerin listeye konuluş sıra verecek bir program yapalım. Bunun için aşağıdaki formu oluşturun. ListBo Sorted özelliğini True yapın.




```
'ÖRNEK : NewIndex, ItemData
Option Explicit
Private  Sub Command1_Click ()
 List1.AddItem  text1 'Texti listeye ekle
 List1.ItemData(list1.NewIndex) = List1.ListCount
End Sub

Private  Sub List1_Click ()
 label3 = list1.ListIndex + 1 'Seçilen elemanın listedeki sırası
 label5 = List1.ItemData(List1.ListIndex)'Eklendiği sıra nosu
End Sub
```

Listedeki elemanlardan biri seçildiğinde o elemanın listedeki numarası ve listeye eklendiği sıra numarası Label kutularına yazılacaktır.

List1.ItemData(list1.NewIndex) = List1.ListCount satırını biraz açalım:

List1.NewIndex listeye en son eklenen elmanın indexini,

List1.ItemData(list1.NewIndex) ise bu indexli elemana (yani listeye son eklenen elemana) verilecek numarayı belirliyordu. Bu numara elemanın listeye eklendiği sıra numarası olacağı için bunu liste içindeki eleman sayısını atayarak yapabiliriz.

Columns

Bu özellik ile liste kutusu bir kaç kolon yapılabilir.

0 ise: Liste kutusu tek sütundur ve listenin ekranda görülen kısmının dolmasıyla listeye dikey scrollbar eklenir.

0 değilse: Listenin genişliği verilen sayıda sütuna bölünür ve bir sütunun dolmasıyla ikinci sütuna geçilir. Listenin görülen kısmındaki sütunların dolmasıyla listeye yatay scrollbar eklenir. Burada her kolonun genişliği listenin genişliğinin verilen kolon sayısına oranıdır.

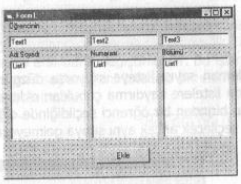
Örneğin Columns=5 ve Width = 3000 ise her kolonun genişliği 3000/5 = 600 olacaktır.

Columns özelliği 3 olan bir liste aşağıdaki gibi olacaktır:

TopIndex

O anda ekranda görülen en üstteki elemanın indexidir. Bu properties ile listenin o anda ekranda görülen kısmını öğrenebiliriz.

Bu özelliğin etkisini görebilmek için şöyle bir örnek yapalım. Bir öğrenciye ait bilgileri listboxlarda gösterelim. Örneğimiz için form üzerine ListBox yerleştirin ve bunun iki kopyasını da hemen yanına yerleştirin. Ayrıca aşağıdaki gibi text kutularını ve komut düğmelerini de yerleştirin.



Amacımız öğrenciyle ilgili bilgileri her üç liste kutusuna ayrı ayrı eklemek olduğundan komut düğmesinin Click olayına şu kodu yazalım.

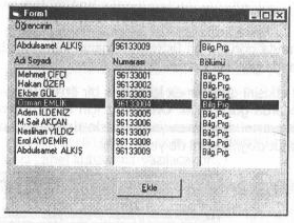
```
Private Sub Command1_Click ()
  list1(0).AddItem Text1
  list1(1).AddItem Text2
  list1(2).AddItem Text3
End Sub
```

Her üç listede öğrenciyle ilgili bilgileri tuttuğu için listeden bir eleman seçildiğinde diğer listelerden de aynı elemanın seçilmesi gerekir. Örneğin listeden öğrencinin adı seçildiğinde, öğrencinin numarası ve bölümünün de seçilmesini isteriz.

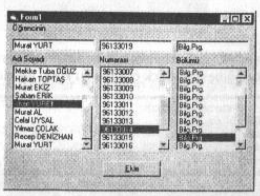
Bu işlem için listelerden biri tıklandığında diğer listelerin ListIndex özelliklerini eşitlersek listeler uyumlu hareket edecektir.

```
Private  Sub List1_Click (index As Integer)
  Dim i, ind
  'seçili elemanın listindexini al
  ind = List1(index).ListIndex
  For i = 0 To 2
    'Diğer listelerin Listindexlerini seçili indexe eşitle
    List1(i).ListIndex = ind
  Next
End Sub
```

Böylece listelerden birinden bir eleman seçildiğinde diğer listelerde de aynı öğrenciye ait özellikler seçilecektir.



Bu; listedeki eleman sayısı listeye sığıyorsa düzgün çalışacaktır. Listedeki elemanlar arttığında listelere kaydırma çubukları eklenecektir. Bu durumda listenin alt kısımlarından birinden bir öğrenci seçildiğinde diğer listelerde de bu öğrenciye ait özellikler seçilecek ancak aynı sıraya gelmeyecektir.



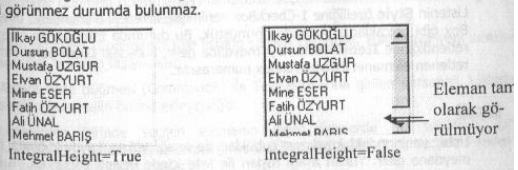
Görüldüğü gibi elemanlar listeye sığmadığında aynı öğrenciye sahip özellikler aynı sıraya gelmemektedir. İşte bu durumu düzeltmek için listelerin TopIndex özelliklerini de eşitlemek gerekir.

Sonuç olarak kodumuz aşağıdaki gibi olacaktır.

```
'ÖRNEK : TopIndex
Private Sub Command1_Click ()
  List1(0).AddItem Text1
  List1(1).AddItem Text2
  List1(2).AddItem Text3
End Sub

Private Sub List1_Click (index As Integer)
  Dim i, ind, tind
  'seçili elemanın listindexini al
  ind = List1(index).ListIndex
  tind = List1(index).TopIndex
  For i = 0 To 2
    'Diğer listelerin Listindexlerini seçili indexe eşitle
    List1(i).ListIndex = ind
    List1(i).TopIndex = tind
  Next
End Sub
```

IntegralHeight

Bu özelliğin değeri **True** ise Liste kutusu içindeki elemanları tam gösterebilmek için kendisini yeniden boyutlandırır. Normalde bu özellik True'dir ve liste kutusunun boyutu sadece belirli aralıklarla değiştirilebilir. Bu işlem listedeki elemanların tam olarak görülebilmesini sağlar. Yani listedeki bir elemanın yarısı görünür yarısı görünmez durumda bulunmaz.



Bu özelliğin **True** olması elemanların listede tam olarak görülmesini sağlarken, listenin form içindeki yerleşimini zorlaştırır.

Style

Liste kutusu bir check box dizisi gibide kullanılabilir. **Style** özelliğine 1 değeri verildiğinde listedeki elemanlar birer CheckBox gibi davranır. Yani kullanıcı listedeki elemanları işaretleyerek seçebilir.

Listedeki bir eleman seçildiğinde ItemCheck olayı meydana gelecektir.

Yukarıda **Style** özelliği 1 olan bir liste kutusu görülmektedir.

Events

Click

Listeden bir eleman seçildiğinde Click olayı meydana gelir.

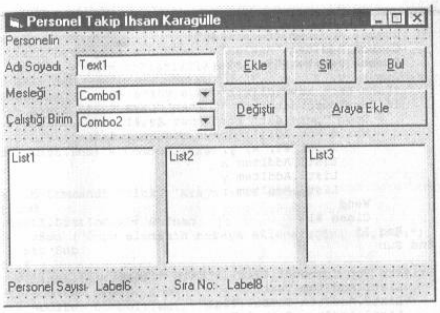
ItemCheck(Item As Integer)

Listenin **Style** özelliğine 1-CheckBox verilmişse listedeki elemanların birer Check Box gibi olacaklarını yukarıda görmüştük. Bu durumda listeden bir eleman işaretlendiğinde **ItemCheck** olayı meydana gelir. Buradaki **Item** parametresi işaretlenen elemanın listedeki Index numarasıdır.

Scroll

Liste, yanlarındaki kaydırma çubukları ile aşağı yukarı kaydırıldığında bu olay meydana gelir. Yukarı aşağı tuşları ile liste içinde gezinti yaparken de bu olay meydana gelebilir. Genel olarak listenin ekranda görülen kısmı değiştiğinde bu olay meydana gelecektir.

ÖRNEK: Genel bir örnek olarak personel bilgilerinin girilebileceği bir program yazalım. Örneğimiz için aşağıdaki formu oluşturun.



Örneğimizde:

Ekle düğmesi (Command1) ile Text kutularına girilenleri ayrı ayrı listelerin sonuna ekleyeceğiz.

Sil düğmesi (Command2) ile Listeden seçilen personeli sileceğiz.

Bul düğmesi (Command3) ile Text1'e ismi girilen elemanı listede bulup seçeceğiz.

Değiştir düğmesi (Command4) ile listede seçilen elmanı Text kutularına girilen yeni değerlerle değiştireceğiz.

Taşı düğmesi (Command5) ile listede seçilen personeli, kullanıcıya yeni sırasını sorarak, o sıraya taşıyacağız.

Araya Ekle düğmesi (Command6) ile Text kutularına girilen personeli, Listede seçilen personelin önüne ekleyeceğiz.

Listelerden birinde seçilen elemanın diğer listelerde de seçilmesi için **ListBox**'ların **Click** olayları aracılığı ile **ListIndex** özelliklerinin, seçilen listedeki ile aynı olmasını sağlayacağız.

Ve programdan çıkarken listeleri kaydedip girerken de yükleyeceğiz.

```
'ÖRNEK : ListBox
Private Sub Form_Load()
    Combo1.AddItem "Mühendis"
    Combo1.AddItem "İşçi"
    Combo1.AddItem "Programcı"
    Combo1.AddItem "Muhasebeci"
```

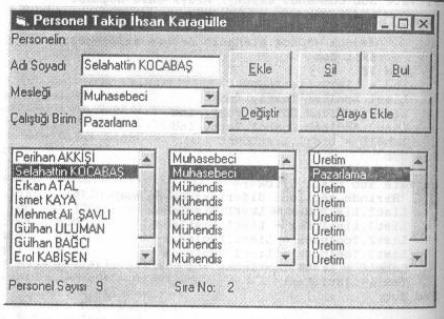


```
    Combo2.AddItem "Üretim"
    Combo2.AddItem "Pazarlama"
    Combo2.AddItem "Reklam"
    Combo2.AddItem "Satış"
    Dim x, y, z
    'Daha önce kaydedilmiş dosya varsa aç
    If Dir("pers.dat") <> "" Then'dosya varsa
        Open "pers.dat" For Input As #1
        While Not EOF(1)'dosya sonuna kadar
            Input #1, x, y, z
            List1.AddItem x
            List2.AddItem y
            List3.AddItem z
        Wend
        Close #1
    End If
End Sub

Private Sub Command1_Click()'Ekle düğmesi
    List1.AddItem Text1
    List2.AddItem Combo1.Text
    List3.AddItem Combo2.Text
    Label6 = List1.ListCount
End Sub

Private Sub Command2_Click()'Sil düğmesi
    If List1.ListIndex < 0 Then
        MsgBox ("Önce silinecek elemanı seçiniz")
        Exit Sub
    End If
    Dim ind, c
    c = MsgBox(List1.List(ind) & " silinsin mi", vbYesNo +
vbExclamation + vbDefaultButton2, "Sil")
    ind = List1.ListIndex
    If c = vbNo Then Exit Sub
    List1.RemoveItem ind
    List2.RemoveItem ind
    List3.RemoveItem ind
    Label6 = List1.ListCount
End Sub

Private Sub Command3_Click()'Bul düğmesi
    Dim i
    For i = 0 To List1.ListCount - 1
        If UCase(List1.List(i)) = UCase(Text1) Then
            'bulundu ise seç
            List1.ListIndex = i
            Exit Sub
        End If
    Next
    MsgBox (Text1 & " bulunamadı")
End Sub
```



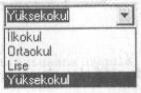
```
Private Sub Command4_Click()'Değiştir düğmesi
    Dim ind
    If List1.ListIndex < 0 Then
        MsgBox ("Önce değiştirilecek elemanı seçiniz")
        Exit Sub
    End If
    ind = List1.ListIndex
    List1.List(ind) = Text1
    List2.List(ind) = Combo1.Text
    List3.List(ind) = Combo2.Text
End Sub

Private Sub Command5_Click()'Araya ekle düğmesi
    Dim ind
    If List1.ListIndex < 0 Then
        MsgBox ("Önce elemanın nereye ekleneceğini seçiniz")
        Exit Sub
    End If
    ind = List1.ListIndex
    List1.AddItem Text1, ind
    List2.AddItem Combo1.Text, ind
    List3.AddItem Combo1.Text, ind
    Label6 = List1.ListCount
End Sub

Private Sub List1_Click()
    'Birinde seçileni diğerlerinde de seç
    List2.ListIndex = List1.ListIndex
    List3.ListIndex = List1.ListIndex
    List2.TopIndex = List1.TopIndex
    List3.TopIndex = List1.TopIndex
    Label8 = List1.ListIndex + 1
    Text1=List1.Text
End Sub

Private Sub List2_Click()
    List1.ListIndex = List2.ListIndex
    List3.ListIndex = List2.ListIndex
    List1.TopIndex = List2.TopIndex
    List3.TopIndex = List2.TopIndex
    Label8 = List2.ListIndex + 1
    Combo1.Text=List2.Text
End Sub

Private Sub List3_Click()
    List2.ListIndex = List3.ListIndex
    List1.ListIndex = List3.ListIndex
    List2.TopIndex = List3.TopIndex
    List1.TopIndex = List3.TopIndex
    Label8 = List3.ListIndex + 1
    Combo2.Text=List3.Text
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
    'Çıkışta listeleri pers.dat dosyasına kaydet
    Dim x, y, z, i
    Open "pers.dat" For Output As #1
    For i = 0 To List1.ListCount - 1
        x = List1.List(i)
        y = List2.List(i)
        z = List3.List(i)
        Write #1, x, y, z
    Next
    Close #1
End Sub
```



ComboBox (Açılan Liste)

Aşağı doğru açılabilen bir liste kontrolüdür. Genellikle, değerleri daha önceden belli olan elemanların seçimi için kullanılırlar. Liste kutusuna benzer ancak listedeki elemanlardan sadece seçileni ekranda görüntüler.



ComboBox kontrolüne eleman ekleme ve silme işlemi ListBox'ta olduğu gibidir.

Properties

Style

0,1,2 değerlerinden birini alabilen bu özellik ile ComboBox'un tipi belirlenir.

2: Aşağı doğru açılabilen fakat kullanıcı tarafından giriş yapılamayan combobox oluşturur. Kullanıcı listedeki değerlerden birini seçebilir fakat değiştiremez.

1: Aşağı doğru açılmayan fakat içeriği kullanıcı tarafından değiştirilebilen ComboBox oluşturur. Listedeki bir elemanı seçmek aşağı ve yukarı tuşlarıyla yapılır, listedeki bütün elemanlar aynı anda görüntülenmez.

0: Aşağı doğru açılan ve içeriği kullanıcı tarafından değiştirilebilen combobox oluşturur.

Text

Combobox'un kutusunda aktif olan elemanın içeriği bu özellikle öğrenilebilir.

Eğer liste stili 0 veya 2 ise bu değer, listedeki elemanlardan biri olmayabilir. Bu özellik kullanıcının kutuya girdiği metni öğrenmek için de kullanılır.

ListIndex

Kullanıcının kutudan seçtiği değeri öğrenmek için Text özelliği kullanılabilir. Seçtiği elemanın numarasını öğrenmek veya listedeki bir elemanı program koduyla aktif hale getirmek için de ListIndex özelliği kullanılabilir.

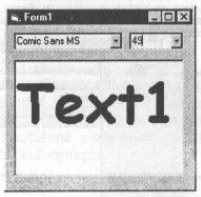
ÖRNEK: Örnek olarak bir Text kutusu içindeki yazının fontunu ve font boyutunu iki ComboBox aracılığı ile seçebilmeyi sağlayacak bir program yapalım. Programın Form_Load olayına yazacağımız kodla ekran fontlarını Combo1 içine, font büyüklüklerini de Combo2 içine ekleyeceğiz. Kullanıcının seçtiği font özelliklerini Text kutusuna uygulamak içinde Combo kutularının Click olayını kullanacağız.

Örneğimiz için formunuza bir Text kutusu ve iki tane ComboBox yerleştirin. Birinci kutuyu font seçme işlemi için, ikinci kutuyu ise font boyutunu seçtirmek için kullanacağız. Kullanıcı ancak var olan fontlardan birini seçebileceği için bu ComboBox'un Style özelliği 2 olmalıdır. Böylece kullanıcı bu listeye yeni bir şey yazamaz, sadece var olanlardan seçebilir. Font boyutunu seçeceği kutuya ise kullanıcı isterse kutuda olmayan bir rakam da girebilir. Bu yüzden de ikinci ComboBox'un Style özelliğini 0 yapın. Böylece kullanıcı isterse bu kutudan bir değer seçer, isterse kendi değerini kutuya yazar.

```
'ÖRNEK: ListIndex, Text
Private Sub Form_Load()
 Dim i
 'Ekran fontlarını Combo1 içine ekle
 For i = 0 To Screen.FontCount - 1
  Combo1.AddItem Screen.Fonts(i)
 Next
 Combo1.ListIndex = 0 'İlk elemanı seç
```



```
 'B-100 arası rakamları Combo2 içine ekle
 For i = 8 To 100
  Combo2.AddItem i
 Next
 Combo2.ListIndex = 0 'İlk elemanı seç
End Sub

Private Sub Combo1_Click()
 'seçilen font adını Text kutusuna uygula
 Text1.FontName = Combo1.Text
End Sub

Private Sub Combo2_Click()
 'seçilen font boyunu Text kutusuna uygula
 Text1.FontSize = Combo2.Text
End Sub
```

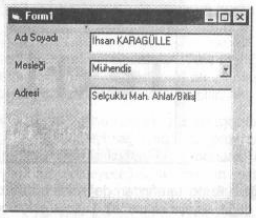


ÖRNEK: İçeriği kullanıcı tarafından değiştirilebilen ComboBox'larda kullanıcının kutuya yazdığı değer listeye eklenmez. Listeye eklenmek isteniyorsa kod yazmak gereklidir.

Örnek olarak bir personelin adını, mesleğini ve adresini öğrenmek isteyelim. Mümkün olabilecek bazı meslekleri ise bir ComboBox içinde listeleyelim, eğer personelin mesleği bunlardan biri değilse mesleğini ComboBox'a eklemesine izin verelim. Personelin adı ve adresi için birer text kutusu, mesleği için ise Style'i 0 olan bir ComboBox oluşturalım.

```
'ÖRNEK: ComboBox
Private  Sub Form_Load ()
    combo1.AddItem "Vasıfsız İşçi"
    combo1.AddItem "Mühendis"
    combo1.AddItem "Muhasebeci"
    combo1.AddItem "Eğitimci"
    combo1.AddItem "Programcı"
    combo1.AddItem "Teknisyen"
    combo1.ListIndex = 0'İlk elemanı kutuda göster
End Sub
```

```
Private  Sub Combol_LostFocus ()
    Dim i
    For i = 0 To combol.ListCount - 1
        If combol.Text = combol.List(i) Then
        'Kutuda yazılı olan listedeki elemanlardan biri ise
        'Değişim yok fonksiyondan çık
            Exit Sub
        End If
    Next
    'ComboBox'a giriş yapılmış
    combol.AddItem combol.Text'Girileni listeye ekle
End Sub
Private  Sub Combol_KeyPress (keyascii As Integer)
    If keyascii = 27 Then
        'ESC'ye basıldı ise
        combol.Text = combol.List(0)
    End If
End Sub
```



Kullanıcının Combobox'a herhangi bir değer girip girmediğini ComboBox'un **LostFocus** olayındaki yazdığımız kodla anlıyoruz. **Lostfocus** olayını kullanmamızın sebebi kontrol ComboBox'dan gittiğine göre yazılacak olan metnin de bitmiş olacağındandır. Burada ComboBox kutusunda yazılı olanla (Combo1.Text) listedeki bütün elemanları karşılaştırıyoruz, kutudaki yazı listede var ise bu yeni eklenmiş bir şey değildir ve fonksiyondan çıkıyoruz. Listede yok ise yeni girilmiş bir şeydir ve bunu listeye ekliyoruz. Ayrıca kullanıcının ComboBox'a yazdığı bir şeyi ESC ile iptal edebilmesi için de KeyPress olayındaki kod ile kutunun içindeki yazı listedeki bir elemana eşitlenmiştir.

ComboBox'lara kullanıcının yazdıklarını eklemek değişik yollarla da yapılabilir. Burada kullandığımız yöntem her zaman işe yaramayabilir. Örneğin ComboBox'ta bir şey yazdıktan sonra menüden bir seçeneğin seçilmesi ComboBox'da lostfocus olayını gerçekleştirmeyecektir. Eğer seçilen menüde ComboBox'un değerine ihtiyaç duyuyorsa doğru değeri alamayacaktır.

Alternatif bir yöntem olarak yalnız keypress olayını aşağıdaki gibi kullanabiliriz. Bu durumda ise kullanıcının listeye eklemek istediği yeni değeri yazdıktan sonra Entere basması gerekir. Entere basmaması durumunda listeye eklenmeyecektir.

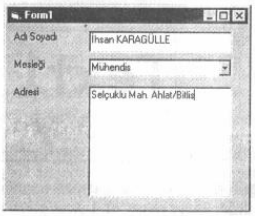
```
Private Sub Combol_KeyPress (keyascii As Integer)
    Dim i
    If keyascii = 27 Then 'ESC'ye basıldı ise
        combol.Text = combol.List(0)
    End If
    if keyascii = 13 Then
        For i = 0 To combol.ListCount - 1
         If combol.Text = combol.List(i) Then
         'Kutuda yazılı olan listedeki elemanlardan biri ise
         'Değişim yok fonksiyondan çık
             Exit Sub
         End If
     Next
     'ComboBox'a giriş yapılmış
     combol.AddItem combol.Text'Girileni listeye ekle
    End If
End Sub
```

ÖRNEK: Aşağıda verilen örnekte Bir kişiye ait derece/kademe verileri ve bunların karşılığı olan göstergeler verilmekte, bir combo kutusuyla seçilen derece/kademe ye karşı gelen değer "Fag" değişkenine aktarılıp 1000 sabiti ile çarpılarak sonuç bir label kutusuna yazdırılmaktadır. Bunun için Programda bir combo kutusu ile bir label kullanılmıştır.

```
'ÖRNEK1 : ComboBox
 Dim derece(0 To 2)
 Dim gosterge(0 To 2)
 Const sabit = 1000
Private  Sub Form_Load ()
 Dim i
 derece(0) = "1/1"
 derece(1) = "1/2"
 derece(2) = "1/3"
 gosterge(0) = "900 "
 gosterge(1) = " 910  "
 gosterge(2) = " 920 "
 For i = 0 To 2
  combol.AddItem derece(i)  'derece dizisini combo kutusuna al
 Next i
 combol.ListIndex=0'combo kutusunda bulunan ilk elemanı görüntüle
End Sub
Private  Sub Combol_Click ()
 Dim Fag
 'dereceye karşı düşen göstergeyi Fag değişkenine aktar
 Fag = gosterge(combol.ListIndex)
 Label1.Caption = Fag * sabit
End Sub
```


Visual Basic Pro 6.0 İ.Karagülle & Z.Pala


```
Private  Sub Combol_LostFocus ()
    Dim i
    For i = 0 To combol.ListCount - 1
        If combol.Text = combol.List(i) Then
        'Kutuda yazılı olan listedeki elemanlardan biri ise
        'Değişim yok fonksiyondan çık
            Exit Sub
        End If
    Next
    'ComboBox'a giriş yapılmış
    combol.AddItem combol.Text'Girileni listeye ekle
End Sub
Private  Sub Combol_KeyPress (keyascii As Integer)
    If keyascii = 27 Then
        'ESC'ye basıldı ise
        combol.Text = combol.List(0)
    End If
End Sub
```

Kullanıcının Combobox'a herhangi bir değer girip girmediğini ComboBox'un **LostFocus** olayındaki yazdığımız kodla anlıyoruz. **Lostfocus** olayını kullanmamızın sebebi kontrol ComboBox'dan gittiğine göre yazılacak olan metnin de bitmiş olacağındandır. Burada ComboBox kutusunda yazılı olanla (Combo1.Text) listedeki bütün elemanları karşılaştırıyoruz, kutudaki yazı listede var ise bu yeni eklenmiş bir şey değildir ve fonksiyondan çıkıyoruz. Listede yok ise yeni girilmiş bir şeydir ve bunu listeye ekliyoruz. Ayrıca kullanıcının ComboBox'a yazdığı bir şeyi ESC ile iptal edebilmesi için de KeyPress olayındaki kod ile kutunun içindeki yazı listedeki bir elemana eşitlenmiştir.

ComboBox'lara kullanıcının yazdıklarını eklemek değişik yollarla da yapılabilir. Burada kullandığımız yöntem her zaman işe yaramayabilir. Örneğin ComboBox'ta bir şey yazdıktan sonra menüden bir seçeneğin seçilmesi ComboBox'da lostfocus olayını gerçekleştirmeyecektir. Eğer seçilen menüde ComboBox'un değerine ihtiyaç duyuyorsa doğru değeri alamayacaktır.

Alternatif bir yöntem olarak yalnız keypress olayını aşağıdaki gibi kullanabiliriz. Bu durumda ise kullanıcının listeye eklemek istediği yeni değeri yazdıktan sonra Entere basması gerekir. Entere basmaması durumunda listeye eklenmeyecektir.

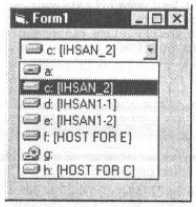
```
Private Sub Combol_KeyPress (keyascii As Integer)
    Dim i
    If keyascii = 27 Then 'ESC'ye basıldı ise
        combol.Text = combol.List(0)
    End If
    if keyascii = 13 Then
        For i = 0 To combol.ListCount - 1
         If combol.Text = combol.List(i) Then
         'Kutuda yazılı olan listedeki elemanlardan biri ise
         'Değişim yok fonksiyondan çık
             Exit Sub
         End If
     Next
     'ComboBox'a giriş yapılmış
     combol.AddItem combol.Text'Girileni listeye ekle
    End If
End Sub
```

ÖRNEK: Aşağıda verilen örnekte Bir kişiye ait derece/kademe verileri ve bunların karşılığı olan göstergeler verilmekte, bir combo kutusuyla seçilen derece/kademe ye karşı gelen değer "Fag" değişkenine aktarılıp 1000 sabiti ile çarpılarak sonuç bir label kutusuna yazdırılmaktadır. Bunun için Programda bir combo kutusu ile bir label kullanılmıştır.

```
'ÖRNEK1 : ComboBox
 Dim derece(0 To 2)
 Dim gosterge(0 To 2)
 Const sabit = 1000
Private  Sub Form_Load ()
 Dim i
 derece(0) = "1/1"
 derece(1) = "1/2"
 derece(2) = "1/3"
 gosterge(0) = "900 "
 gosterge(1) = " 910  "
 gosterge(2) = " 920 "
 For i = 0 To 2
  combol.AddItem derece(i)  'derece dizisini combo kutusuna al
 Next i
 combol.ListIndex=0'combo kutusunda bulunan ilk elemanı görüntüle
End Sub
Private  Sub Combol_Click ()
 Dim Fag
 'dereceye karşı düşen göstergeyi Fag değişkenine aktar
 Fag = gosterge(combol.ListIndex)
 Label1.Caption = Fag * sabit
End Sub
```

Drive List Box (Sürücü Listeleme Kutusu)

Sistemde bulunan sürücüleri listelemeye yarayan, ComboBox kontrolünden türemiş bir kontrol elemanıdır. Bunun vasıtasıyla programın çalışması esnasında istenilen sürücü seçimi yapılır. Bu seçim sürücüye geçişi sağlamaz. Sürücüye geçiş işlemi **Chdrive** gibi bir komutla yapılması gerekir.



Properties

Drive

Bu özellik kullanılarak kontrolün gösterdiği aktif sürücü öğrenilebilir veya değiştirilebilir.

Drive özelliğinin gösterdiği değer disket sürücüler için sürücü harfi ve : işareti iken ,harddisk sürücülerde buna sürücü etiketi de dahildir.

Yani A sürücüsü A: ile gösterilirken C: sürücüsü C:[Etiket] şeklinde gösterilir. Bu durum genellikle problem çıkarmaz, çünkü sürücü atamalarında sadece iki karakter dikkate alınır.

Yani **Chdrive "c:abcd"** gibi bir komut C sürücüsüne geçmeyi sağlar. Aynı durum **Drive** değerini değiştirirken de geçerlidir. **Drive1.Drive = "C:"** ataması C sürücüsüne geçmeyi sağlayacaktır.

Events

Change()

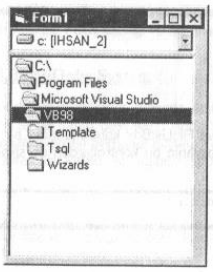
Kontrolden bir sürücü seçilmesi ile bu olay meydana gelir. Sürücü seçimi yapıldığında yapılması gereken kod buraya yazılmalıdır.

Bir **DirectoryListBox** ile bağlantı kurulacaksa bu işlem **Change** olayında aşağıdaki gibi yapılmalıdır.

```
Sub Drive1_Change()
  Dir1.Path=Drive1.Drive
  ChDrive Drive1.Drive
End Sub
```

Directory List Box (Dizin Listeleme Kutusu)

Sistemde bulunan sürücülerdeki dizinleri listelemeye yarayan, ListBox kontrolünden türemiş bir kontrol elemanıdır. Bu kontrol elemanı vasıtasıyla **Path** özelliği ile belirlenmiş sürücüde bulunan dizinler seçilebilir. Bu seçme işlemi aktif dizinin değişmesine sebep olmaz, bu işin **ChDir** gibi bir komutla yapılması gerekir.



Properties

Path

Bu özellik kullanılarak seçilmiş olan dizin öğrenilebilir ve değiştirilebilir. Bu özellikten dönen değer sürücü ismi dahil tam yoldur. Kontrolün göstermesi istenen sürücüde, yine sürücü isminin bu özelliğe atanması ile gerçekleştirilir.

Events

Change()

Listeden bir dizin seçilmesi ile bu olay meydana gelir. Dizin seçimi yapıldığında yazılması gereken kod buraya yazılmalıdır.

Bir **FileListBox** ile bağlantı kurulacaksa bu işlem **Change** olayında aşağıdaki gibi yapılmalıdır.

```
Sub Dir1_Change()
  File1.Path= Dir1.Path
  ChDir Dir1.Path
End Sub
```

Methods

Update

DirectoryListBox ve FileListBox kontrollerinde, bir dizin veya dosya silinirse silinen dizin veya dosyanın bu kontrollerden de silinebilmesi için ReFresh metodu kullanılmalıdır.

```
rmdir "geçici dizin"
dir1.update
```


FileList Box (Dosya Listeleme Kutusu)

Herhangi bir dizindeki dosyaları listelemeye yarayan, ListBox kontrolünden türemiş bir kontrol elemanıdır. Hangi dizindeki dosyaları göstereceği **Path** özelliği ile belirlenir.

Properties

FileName

Bu özellik ile seçili dosyanın ismi öğrenilebilir. Geri dönecek değer dosyanın adı ve uzantısıdır. Dosyanın yolu ise **Path** özelliği ile öğrenilebilir.

Archive,Hidden,Normal, ReadOnly, System

Bu özellikler kullanılarak dosya listeleme kutusunda Arşiv, Saklı, Sadece okunur, Normal ve System özeliklerine sahip dosyaların görüntülenmesi veya görüntülenmemesi sağlanır. Bu özelliklerden herhangi birisine **True** değeri verildiğinde o özelliğe sahip dosyalar listelenir, **False** verildiğinde listelenmez.

Path

Bu özellik ile listelenmek istenen sürücü ve dizin ismi belirlenir. Örneğin Windows dizinindeki dosyaları göstermesini istiyorsak File1.Path = "C:\windows" atamasını yapabiliriz.

Pattern

Bu özellik ile listelenmek istenen dosyalar filtrelenir. Bu işlem bildiğimiz joker karakterle (?,*) yapılır.

Örneğin yalnız EXE uzantılı dosyaları göstermek için :

File1.Pattern = "*.EXE" ataması yapılır.

Birden fazla tip belirlemek için ";" karakteri kullanılır. EXE ve COM dosyalarını listelemek için :

File1.Pattern = "*.EXE;*.COM" şeklinde kullanılır.

Events

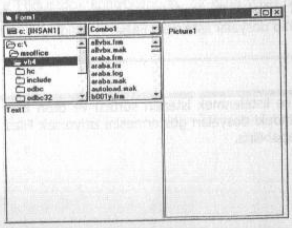
PathChange()

Bu olay **Path** özelliğinin değiştirilmesi sonucu meydana gelir. Yani **FileListBox** farklı bir yeri göstermeye başladığında meydana gelir.

PatternChange()

Bu olay **Pattern** özelliğinin değiştirilmesi sonucu meydana gelir. Yani **FileListBox** farklı türlerdeki dosyaları listelemeye başladığında bu olay meydana gelir.

ÖRNEK: Örnek olarak kullanıcının seçtiği dosyaya göre işlem yapacak bir program yapalım. Örneğimiz için aşağıdaki formu oluşturun ve Text kutusunda birden fazla satır gösterilebilmesi için MultiLine özelliğini True yapın.

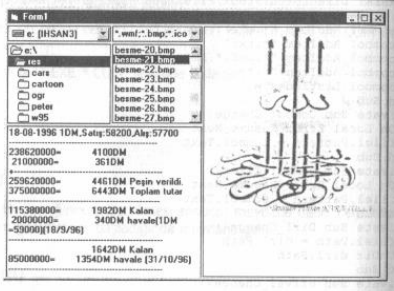


Örneğimizde kullanıcı bir programı tıklarsa o program çalıştırılsın, bir resim dosyasını tıklarsa picture kutusunda gösterilsin, başka bir dosyayı tıklarsa bu dosyanın içeriği text kutusunda gösterilsin. Eğer dosya text kutusuna sığmayacak kadar büyükse de Write (WordPad) programı çalıştırılarak dosya bu programla açılsın.

```
'ÖRNEK: DirectoryListBox,FileListBox,DriveListBox,ComboBox
Private Sub Form_Load()
  combo1.AddItem "*.exe;*.com;*.bat"
  combo1.AddItem "*.txt"
  combo1.AddItem "*.wmf;*.bmp;*.ico"
  combo1.AddItem "*.*"
  combo1.ListIndex = 0
End Sub
Private Sub Combo1_Change()
  On Local Error Resume Next
  file1.Pattern = combo1.Text
End Sub
Private Sub Combo1_Click()
  On Local Error Resume Next
  file1.Pattern = combo1.Text
End Sub
Private Sub Dir1_Change()
  file1.Path = dir1.Path
  ChDir dir1.Path
End Sub
Private Sub Drive1_Change()
  dir1.Path = drive1.Drive
  ChDrive drive1.Drive
End Sub

Private Sub File1_Click()
  On Local Error Resume Next
  Dim n, uz, x, t
  n = InStr(file1.FileName, ".") 'Noktanın yerini bul
  uz = Mid(file1.FileName, n)'noktadan sonrasını al,yani uzantıyı
  Select Case uz
    Case ".exe", ".com", ".bat", ".pif" 'program ise çalıştır
      x = Shell(file1.FileName, 1)
    Case ".wmf", ".ico", ".bmp" 'Resim ise göster
      picture1.Picture = LoadPicture(file1.FileName)
    Case Else 'Bunlardan biri değilse içeriğini göster
      If FileLen(file1.FileName) > 32000 Then
      'dosya text kutusunun alacağından büyük ise Write ile
göster
        x = Shell("write " & file1.FileName, 1)
      Else
        'değilse dosyayı text kutusunda göster
        t = ""
        'Dosyayı sıralı erişimli modda okumak için aç
        Open file1.FileName For Input As #1
          While Not (EOF(1)) 'Dosya sonuna gelene kadar oku
            Line Input #1, x'Bir satır oku
```



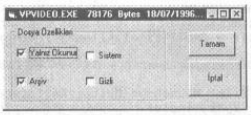
```
            t = t + x + Chr(13) + Chr(10)'Öncekinin altına ekle
          Wend
          text1 = t'Okunanları text kutusunda göster
        Close #1'Dosyayı kapat
      End If
    End Select
End Sub
```



ÖRNEK: Şimdi DriveListBox, DirListBox ve FileListBox elemanlarını kullanarak bir dosyayı kopyalayacak, adını değiştirecek, silecek, çalıştıracak, özelliklerini gösterecek yada yeni özellikler verecek bir program yapalım. Bunun için aşağıdaki form tasarımını kullanacağız.

Formdaki kontrolleri oluşturduktan sonra dosya menüsünde de yanda görülen menüleri oluşturun.

Programda seçili olan dosyaya ait özellikler aşağıdaki form ile gösteriliyor:



Programın kodu aşağıdaki gibi olacaktır:

```
'ÖRNEK:Drivelistbox, Dirlistbox, Filelistbox
Private Sub Dir1_Change()
 'Dizin yolunu ve dizin yol değişimini File1 e aktar
 File1.Path = Dir1.Path
 ChDir (File1.Path)
End Sub

Private Sub Dir1_Click()
 File1.Path = Dir1.Path
End Sub

Private Sub Drive1_Change()
 'Sürücü değişimini Dir1e aktar
 Dir1.Path = Drive1.Drive
End Sub

Private Sub File1_Click()
 Dim dosya
 Dim secili As Boolean
 With File1
  'Eğer seçili dosya varsa dosyanın adını,uzunluğunu ve tarihini
Labelin başlığına aktar
  If .ListIndex >= 0 Then
   secili = True
   dosya = .List(.ListIndex)
   Label1.Caption = dosya & "   " & FileLen(dosya) & "  Bytes  "
& FileDateTime(dosya)
  Else
   secili = False
   Label1.Caption = ""
  End If
 End With
 'Herhangi bir dosya seçili ise menüleri aktif değilse pasif yap
 mnuac.Enabled = secili
 mnusil.Enabled = secili
 mnuaddegistir.Enabled = secili
 mnuozellikler.Enabled = secili
 mnukopyala.Enabled = secili
End Sub
```

```
Private Sub File1_KeyDown(KeyCode As Integer, Shift As Integer)
  'Eğer entere basılmışsa dosyayı çalıştır.
   If KeyCode = 13 Then
     Dim calistir
      calistir = Shell(File1.filename, vbNormalFocus)
   End If
End Sub

Private Sub File1_MouseDown(Button As Integer, Shift As Integer,
X As Single, Y As Single)
   'eğer farenin sağ tuşu basılı ise popup menüyü aç
   If Button = 2 Then PopupMenu mnudosya
End Sub

Private Sub Form_Load()
   Dim dosya
   Dim secili As Boolean
   With File1
       If .ListIndex >= 0 Then
         secili = True
         dosya = .List(.ListIndex)
         Label1.Caption = dosya & "    " & FileLen(dosya) & "
Bytes  " & FileDateTime(dosya)
     Else
        secili = False
        Label1.Caption = ""
      End If
  End With
  mnuac.Enabled = secili
  mnusil.Enabled = secili
  mnuaddegistir.Enabled = secili
  mnuozellikler.Enabled = secili
  mnukopyala.Enabled = secili
 End Sub

Private Sub mnuac_Click()
  Dim calistir
  'Seçili dosyayı çalıştır.
  Calistir = Shell(File1.filename, vbNormalFocus)
End Sub

Private Sub mnuaddegistir_Click()
    Dim ad, eskiad, YeniAD, mesaj
    eskiad = UCase(File1.filename)
    ad = InputBox("Yeni Dosya İsmini Giriniz", " Dosya İsmi
Değiştir", eskiad)
    If ad = "" Then Exit Sub
    YeniAD = Ucase(LTrim(RTrim(ad)))
    On Local Error GoTo degismehatasi
    Name eskiad As YeniAD: File1.Refresh
    File1.SetFocus
Exit Sub
degismehatasi:
```



```
    mesaj = eskiad + " dosya ismi " + YeniAD + " ye çevrilemiyor.
"
    If Dir(YeniAD) = YeniAD Then
       mesaj = mesaj + YeniAD + " isimli dosya var. Yeni bir
isim verin"
    Else
       mesaj = mesaj + YeniAD + " ismi geçersiz bir isim veya " &
Error
    End If
    MsgBox (mesaj)
Exit Sub
End Sub

Private Sub mnucikis_Click()
  Dim cevap
   cevap = MsgBox("Programı terk etmek istediğinizden emin
misini?", vbYesNoCancel + vbQuestion + vbDefaultButton2 +
vbSystemModal, "Çıkış")
   If cevap = vbYes Then End
End Sub

Private Sub mnukopyala_Click()
    On Error GoTo kopyaHatasi
    Dim kaynak, hedef
    kaynak = File1.filename
    hedef = InputBox("Dosyanın kopyalanacağı yolun adını
giriniz", " Dosya kopyala", File1.Path + "\" + kaynak)
    FileCopy kaynak, hedef
    MsgBox ("Dosya Kopyalandı")
Exit Sub
kopyaHatasi:
   MsgBox ("Dosya kopyalanamıyor")
Exit Sub
End Sub

Private Sub mnuozellikler_Click()
   Form2.Show
   Dim ozellik
   'Dosyanın içerdiği özellikleri form2 deki checkboxlara aktar
    ozellik = GetAttr(Form1.File1.filename)
    If ozellik And 1 Then Form2.Check1.Value = 1 Else
Form2.Check1.Value = 0
    If ozellik And 32 Then Form2.Check2.Value = 1 Else
Form2.Check2.Value = 0
    If ozellik And 4 Then Form2.Check3.Value = 1 Else
Form2.Check3.Value = 0
    If ozellik And 2 Then Form2.Check4.Value = 1 Else
Form2.Check4.Value = 0
    Form2.Caption = Form1.File1.filename & "    " &
FileLen(Form1.File1.filename) & "  Bytes  " &
FileDateTime(Form1.File1.filename)
End Sub
```



```
Private Sub mnusil_Click()
   Dim cevap
   cevap = MsgBox("Bu dosyayı silmek istediğinizden emin
misini?", vbYesNoCancel + vbQuestion + vbDefaultButton2 +
vbSystemModal, "Sil")
  If cevap = vbYes Then
     If File1.ListIndex >= 0 Then
        Kill File1.List(File1.ListIndex)
        File1.Refresh
     End If
  End If
End Sub

'Bu bölüm form2 nin kod editörüne yazılacak
Option Explicit

Private Sub Command1_Click()
   Form1.Show
   Dim ozellik
   On Error Resume Next
   ozellik = GetAttr(Form1.File1.filename)
   If Form2.Check1.Value = 1 Then
      SetAttr Form1.File1.filename, ozellik + vbReadOnly
   End If
   If Form2.Check2.Value = 1 Then
      SetAttr Form1.File1.filename, vbArchive + ozellik
      ozellik = ozellik + vbArchive
   End If
   If Form2.Check3.Value = 1 Then
       SetAttr Form1.File1.filename, vbSystem + ozellik
       ozellik = ozellik + vbSystem
   End If
    If Form2.Check4.Value = 1 Then SetAttr Form1.File1.filename,
ozellik +   vbHidden
    Unload Me
End Sub

Private Sub Command2_Click()
    Unload Me
End Sub
```


Common Dialog(Diyalog Pencereleri)*

Windows tarafından sağlanan standart diyalog kutularının kullanımını sağlayan kontroldür. Bu kutular: "Aç", "Yeni Adla Kaydet", "Renk", "Help", "Yazdır" ve "Font" diyalog kutularıdır.

Diyalog kutularının modal olması sebebiyle bir seçim yapılmadan form üzerinde işlem yapılamaz.

Aşağıda anlatılacak olan bu kontroller form üzerinde tasarım zamanı görülmesine rağmen çalışma esnasında görülmez. Bu kontroller, **Action** özelliklerine değer atanmasıyla veya ilgili metod kullanılarak aktif hale gelirler.

CMDialog kontrolünde "İptal" düğmesinin seçilmesi halinde hata oluşturulup oluşturulmayacağını kullanıcının seçimine bırakmıştır. Oluşacak hata kontrollerin programlamasından kaynaklanan bir hata (BUG) değildir. Sadece "İptal" düğmesinin seçildiğini programa belirtmek için düşünülmüş bir yöntemdir.

Properties

Action

1-6 arası bir değer verilerek ilgili diyalog kutusu açılır.

1: "Aç" diyalog kutusu

CommonDialog1.Action = 1 veya CommonDialog1.ShowOpen

2: "Yeni Adla Kaydet" diyalog kutusu

CommonDialog1.Action = 2 veya CommonDialog1.ShowSave

3: "Renk" diyalog kutusu

CommonDialog1.Action = 3 veya CommonDialog1.ShowColor

4: "Font" diyalog kutusu

CommonDialog1.Action = 4 veya CommonDialog1.ShowFont

* Bu kontrolleri kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Common Dialog Control" seçeneğini işaretlemeniz veya Browse düğmesi ile COMDLG32.OCX dosyasını bulup projeye eklemeniz gerekir.

5: "Yazdır" diyalog kutusu

CommonDialog1.Action = 5 veya CommonDialog1.ShowPrint

6: Windows Help programını çalıştırır.

CommonDialog1.Action = 6 veya CommonDialog1.ShowHelp

Action özelliğine bu değerlerden birinin atanmasıyla ilgili diyalog penceresi açılır ve kullanıcının seçimini bekler. Bu pencerenin açılmasıyla yapılan işlemler diyalog penceresi kapanıncaya kadar askıya alınır.

Bu pencereler de temsil ettikleri işi yapmaz sadece o iş için düşünülmüş standart bir arabirim sağlarlar. Yani **Yeni Adla Kaydet** diyalog kutusu bir dosyayı yeni adla kaydetmez, sadece o iş için yapılacak standart işlemler için bir kutu açar ve kaydedilecek dosyanın ismini ve yolunu kullanıcının girmesini sağlarlar. Kaydetme işlemini sizin yapmanız gerekir.

CancelError

Diyalog pencerelerinde "İptal" düğmesinin seçilmesi halinde yakalanabilir bir hata oluşturulup oluşturulmayacağı bu özellikle belirlenir.

True : "İptal" düğmesi seçilirse hata oluştur

False : Hata oluşturma.

```
Private Sub Form_Click()
    On Local Error GoTo iptal
    CommonDialog1.Action = 1
    CommonDialog1.CancelError = True
    MsgBox (CommonDialog1.filename & " dosyası seçildi")
Exit Sub
iptal:
    MsgBox ("Dosya seçme işlemi iptal edildi")
    Exit Sub
End Sub
```

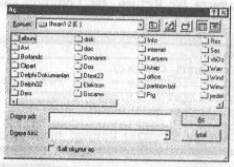
DialogTitle

Açılacak pencerenin başlığında yazılması istenen metni belirler. Bir değer atanmazsa standart başlık yazısı görüntülenir.

Aç ve Yeni Adla Kaydet Diyalog Pencereleri

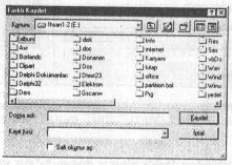
CommonDialog kontrolünün **Action** özelliğine 1 vererek veya **ShowOpen** metodunu kullanarak Aç diyalog penceresi aktif hale getirilir.

CommonDialog1.Action=1 veya **CommonDialog1.ShowOpen** satırı ile açılacak diyalog kutusu:



CommonDialog kontrolünün **Action** özelliğine 2 vererek veya **ShowSave** metodunu kullanarak Aç diyalog penceresi aktif hale getirilir.

CommonDialog1.Action=2 veya **CommonDialog1.ShowSave** satırı ile açılacak diyalog kutusu:



Properties

FileName

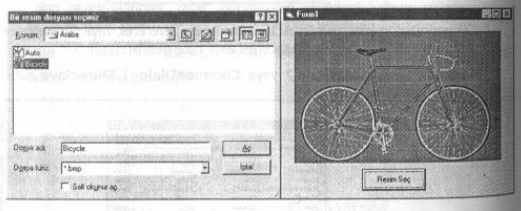
Yukarıdaki diyalog pencerelerinde **Dosya Adı** kutusunda yazılması istenen veya kullanıcı tarafından yazılan değerdir. Bu özelliğe, diyalog kutusu aktif hale getirilmeden bir değer atanması durumunda bu değer **Dosya Adı** kutusuna yazıla-

cak default değer olur. Diyalog kutusu kapandıktan sonra ise, bu özelliğin aldığı değer kullanıcının seçtiği veya yazdığı dosya adıdır. Bu özelliğin ifade ettiği dosya adına sürücü ve yol dahildir.

CommonDialog1.Action=1 veya CommonDialog1.Action=2 satırları ile diyalog penceresini açtıktan sonra kullanıcının seçtiği dosyanın adını öğrenmek için bu özellik kullanılabilir.

ÖRNEK: Örnek olarak formumuzda bulunan bir PictureBox içindeki resmi, CommonDialog penceresi ile açılacak bir dosyadan seçme imkanı verelim. Örneğimiz için formunuza bir Picture, bir CommonDialog ve bir Command Button yerleştirin.

```
'ÖRNEK: FileName
Private Sub Command1_Click()
 CommonDialog1.DialogTitle = "Bir resim dosyası seçiniz"
 CommonDialog1.Filter = "*.bmp|*.bmp"
 CommonDialog1.Action = 1
 Picture1.Picture = LoadPicture(CommonDialog1.FileName)
End Sub
```



FileTitle

Filename özelliğinden farklı olarak dosya adına yol dahil değildir. Sadece seçilen dosyanın ismi ve uzantısını gösterir.

DefaultExt

Kullanıcı dosya isminde dosya uzantısı yazmazsa bu özellikle belirlenen uzantı dosya adına eklenir.

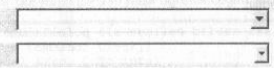
```
CommonDialog1.DefaultExt = "TXT" 'Default uzantı TXT
```


Diyalog penceresinde **Dosya Adı** kısmına kullanıcı DENEME yazarsa CommonDialog1.FileTitle = "DENEME.TXT" olacak, DENEME.X yazarsa CommonDialog1.FileTitle = "DENEME.X" olacaktır. Yani bir uzantı belirtmezse **DefaultExt** uzantısı ile belirlenen uzantı dosyaya eklenecektir.

Yaptığınız programın kullandığı standart bir dosya uzantısı varsa, bu özelliğe o uzantıyı vererek, kullanıcıyı her seferinde uzantı yazma zahmetinden kurtarırsınız.

Filter

Yukarıdaki her iki diyalog penceresinde de gördüğümüz



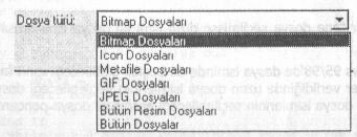
kutularına listelenecek dosyaların tiplerini seçmeye yarayacak açıklamalar koymaya yarar. Kullanım Formatı şöyledir.

CommonDialog1.Filter="Açıklama1|Filtre1|Açıklama2|Filtre2|.....|.....||"

Buradaki | karakteri, Ascii kodu 124 olan karakterdir (Alt Gr tuşu ile birlikte < tuşu) ve filtre parametresinin sonunda iki tane kullanılır. AçıklamaN parametreleri combo kutusunda görülmesi istenen yazıyı, FiltreN parametreleri de listedeki elemanın seçilmesiyle dosya kutusunda gösterilecek dosyaların filtresidir.

```
CommonDialog1.Filter = "Bitmap Dosyaları|*.BMP| Icon Dosyaları|
*.ICO| Metafile Dosyaları|*.WMF|GIF Dosyaları|*.GIF|JPEG
Dosyaları|*.JPG|Bütün Resim
Dosyaları|*.BMP;*.ICO;*.WMF;*.GIF;*.JPG|Bütün Dosyalar|*.*||"
CommonDialog1.ShowOpen
```

Yukarıdaki komut satırlarıyla "Dosya Türü:" combo kutusu aşağıdaki gibi olacaktır.



Listeden "Bitmap dosyaları" seçeneğinin seçilmesiyle diyalog penceresi içerisindeki dosya kutusunda yalnızca BMP uzantılı dosyalar görüntülenecektir.

FilterIndex

Filter özelliğiyle listeye eklenen seçimlerden hangisinin default olarak etkin olacağını belirleyen bir numaradır. Listedeki ilk elemanın numarası 1 dir.

InitDir

Diyalog kutularının ilk açıldığında listeleyeceği dizini belirlemeyi sağlar. Verilmezse aktif dizin kabul edilir.

Flags

Açılacak diyalog penceresinin bazı özelliklerini belirleyen bir özelliktir. Bu özelliğin alacağı değerler ve etkileri aşağıda verilmiştir. Bu değerler OR işlemine tabi tutularak birden fazla özellik verilebileceği gibi hangi özelliğin seçili olduğu da aşağıdaki değerlerle AND işlemine tabi tutularak öğrenilir.

&H200, cdlOFNAllowMultiselect:

Bu değerin **Yeni Adla Kaydet** diyalog kutusunda bir etkisi yoktur. **Aç** diyalog kutusunda ise birden çok dosyanın seçilebilmesini sağlar. Bu seçme işlemi Shift veya Ctrl tuş bileşimleriyle yapılır. Bu durumda FileTitle parametresi boş döner, **FileName** parametresinde ise, dosya isimleri arasında da bir boşluk olarak döner.

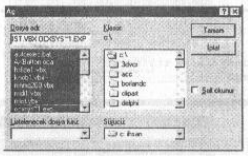
CommonDialog1.Filename = "SürücüİsmiDizinİsmi\ BirinciDosya İkinciDosya ÜçüncüDosya ..."

Örneğin Windows dizininde WIN.COM ve MSD.EXE dosyaları seçilmiş ise

CommonDialog1.Filename = "C:\WINDOWS\ win.com msd.exe" şeklinde olacaktır.

Eğer bir tane dosya seçilmişse dizin ismi ile dosya ismi arasında boşluk bulunmaz.

Windows 95/98'de dosya isminde de boşluk olabileceği için **Flags** parametresine bu değer verildiğinde uzun dosya isimlerinin seçilebileceği dosya penceresi yerine 8+3 dosya isimlerinin seçilebileceği aşağıdaki dosya penceresi açılacaktır.

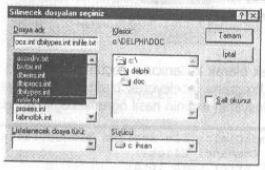


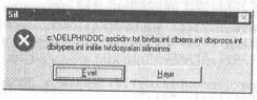
ÖRNEK: Örnek olarak kullanıcının seçtiği birden fazla dosyayı silecek bir program yazalım. Programı inceleyerek birden fazla dosyanın seçilebildiği pencerelerde her bir dosyanın isminin nasıl öğrenileceğini görebilirsiniz.

```
'ÖRNEK: CommonDialog ile seçilen birden fazla dosyayı silme
Private Sub Command1_Click()
    On Local Error GoTo iptal
    CommonDialog1.CancelError = True
    CommonDialog1.filename = "*.*"
    CommonDialog1.Flags = cdlOFNAllowMultiselect
    CommonDialog1.DialogTitle = "Silinecek dosyaları seçiniz"
    CommonDialog1.ShowOpen
    Dim c
    c = MsgBox(CommonDialog1.filename & "dosyaları silinsin mi",
vbYesNo + vbCritical, "Sil")
    If c = vbNo Then Exit Sub
    On Local Error GoTo hata
    Dim y, x, dizin, dosya
    x = CommonDialog1.filename
    y = InStr(x, " ")'ilk boşluğu bul
    If y <> 0 Then
        'Boşluk bulundu. Birden fazla dosya seçilmiş.
        dizin = Left(x, y)'Dizin ismini al
    Else
        'boşluk bulunamadı tek dosya seçilmiş
        Kill x
        Exit Sub
    End If
    ChDrive dizin
    ChDir dizin
    x = Mid(x, y + 2) 'dizin ismini çıkar
    Do
        y = InStr(x, " ") 'boşluk bul
        If y <> 0 Then
            dosya = Left(x, y - 1)
            Kill dosya
            x = Mid(x, y + 1)
        End If
    Loop While y > 0'Boşluk kalmayıncaya kadar devam et
Exit Sub
iptal:
```



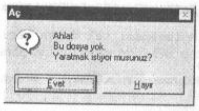
```
    MsgBox ("İşlem iptal edildi")
    Exit Sub
hata:
    MsgBox (dosya & " silinemedi " & Error)
    Resume Next
End Sub
```





&H2000, cdlOFNCreatePrompt:

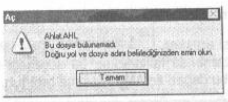
Bu değerin de **Yeni Adla Kaydet** diyalog kutusunda bir etkisi yoktur. **Aç** diyalog kutusunda yazılan dosya isminin diskte olmaması durumunda Windows'un şöyle bir mesaj kutusu çıkarmasına sebep olur.



Burada "Hayır" seçilmesi halinde diyalog kutusu kapanmayacak ve yeni bir dosya seçmenizi sağlayacaktır. "Evet" seçilmesi halinde ise gerçekte diskte olmayan dosya ismi geri gönderilir.

&H1000, cdlOFNExplorer:

Bu değerin de **Yeni Adla Kaydet** diyalog kutusunda bir etkisi yoktur. **Aç** diyalog kutusunda yazılan dosya isminin diskte olmaması durumunda Windows'un şöyle bir mesaj kutusu çıkarmasına sebep olur.

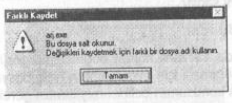


&H4, cdlOFNHideReadOnly:

Diyalog kutusundaki ☐ Salt okunur aç CheckBox'unun görüntülenmemesini sağlar.

&H8000, cdlOFNNoReadOnlyReturn:

Diyalog kutusunda Read Only (Sadece Okunabilir) dosya seçilmesini önler. Böyle bir dosya seçilmesi durumunda Windows'un aşağıdaki mesajı görüntülemesini sağlar.

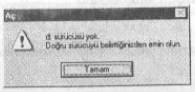


&H2, cdlOFNOverwritePrompt:

Bu değerin **Aç** diyalog kutusunda bir etkisi yoktur. **Yeni Adla Kaydet** diyalog kutusunda yazılan dosya ismiyle aynı isimli bir dosyanın diskte olması durumunda Windows'un aşağıdaki mesajla uyarmasını sağlar

&H800, cdlOFNPathMustExist:

Girilen dosya isminin yolu doğru değilse Windows'un aşağıdaki gibi bir mesaj vermesini sağlar.



&H1, cdlOFNReadOnly:

Diyalog penceresi açılırken ☐ Salt okunur aç CheckBox'unun işaretlenmiş olmasını sağlar. Aynı zamanda kullanıcının bu seçeneği işaretleyip işaretlemediği de Flags özelliğinin bu değeri ile AND işlemine tabi tutularak anlaşılır.

```
CommonDialog1.ShowOpen
If CommonDialog1.Flags And cdlOFNReadOnly Then
  'Salt okunur aç seçeneği işaretli
End If
```

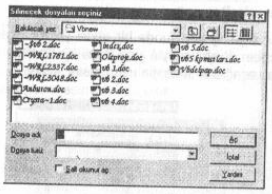
Yukarıdaki gibi bir kodla seçeneğin işaretlenip işaretlenmediği öğrenilebilir.

&H4000, cdlOFNShareAware:

Normalde diyalog kutuları, bir başka uygulama tarafından açık tutulan bir dosyanın seçilmesi durumunda bir hata mesajı ile kullanıcıyı uyarır. Eğer flags özelliğine bu değer verilirse diğer programlar tarafından açık tutulan dosyaların da seçilebilmesini sağlar.

&H10, cdlOFNHelpButton:

Diyalog kutusunda **Yardım** komut düğmesinin görüntülenmesini sağlar. Yardım düğmesinin etkili olması için uygulamanın Help dosyası ve HelpConTextID özelliği ile ilgili yardım konusunun belirlenmiş olması gerekir.



&H8, cdlOFNNoChangeDir:

Normalde diyalog kutuları sonraki seferlerde açıldığında en son kalınan dizindeki dosyaları gösterir. Eğer her seferinde aynı dizini göstermek istiyorsanız **Flags** parametresine bu değeri verebilirsiniz.

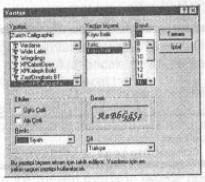
Font Diyalog Penceresi

Windows tarafından sağlanan standart **Yazıtipi** diyalog kutusunun kullanılmasını sağlayan bir kontroldür. Bu kontrolle sistemde bulunan fontlardan biri seçilebilir, rengi, büyüklüğü, şekli değiştirilebilir.

CommonDialog kontrolünün **Action** özelliğine 4 vererek veya **ShowFont** metodunu kullanarak Font diyalog penceresi aktif hale getirilir.

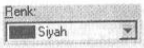
Font dialog penceresi açılmadan önce **Flags** özelliğine 1,2 veya 3 değerlerinden birinin verilmesi gerekir. Aksi takdirde Windows yüklü yazı tipi olmadığını bildirir bir mesaj verecektir.

CommonDialog.Action=4 veya **CommonDialog1.ShowFont** satırları ile açılacak diyalog penceresi:



Properties

Color

Diyalog penceresi içinde bulunan 

kutusunda seçilen rengin numarasıdır.

```
CommonDialog1.Flags = cdlCFEffects Or cdlCFBoth
CommonDialog1.ShowFont
text1.ForeColor = CommonDialog1.Color
```


FontBold , FontItalic, FontStrikeThru, FontUnderline

True veya **False** değeri alabilen bu özellikler diyalog penceresinde Koyu, İtalik, Üstü Çizili ve Altçizgi özelliklerinin durumunu belirler.

```
CommonDialog1.Flags = cdlCFEffects Or cdlCFBoth
CommonDialog1.ShowFont
text1.FontBold = CommonDialog1.FontBold
text1.FontItalic = CommonDialog1.FontItalic
text1.FontUnderline = CommonDialog1.FontUnderline
```

Fontname

Diyalog kutusunda seçilen/seçilecek fontun sistemdeki ismidir. Bu değer herhangi bir kontrolün fontname özelliğine atanabilir.

```
CommonDialog1.Flags = cdlCFEffects Or cdlCFBoth
CommonDialog1.ShowFont
text1.FontName = CommonDialog1.FontName
```

FontSize

Seçilen fontun büyüklüğünü verir.

```
CommonDialog1.Flags = cdlCFEffects Or cdlCFBoth
CommonDialog1.ShowFont
text1.FontSize = CommonDialog1.FontSize
```

Min, Max

Diyalog penceresindeki **Boyut** listesinde gösterilecek font boyutlarının minimum ve maximum değerlerini belirler.

Flags

Açılacak diyalog penceresinin bazı özelliklerini belirleyen bir değer alır. Alacağı değerler ve anlamları aşağıda verilmiştir. Bu değerler **OR** işlemine tabi tutularak bir kaç özellik birden aktif hale getirilebilir.

&H100, cdlCFEffects :

Diyalog penceresinde **Etkiler** ve **Renk** kutularının da görüntülenmesini sağlar.

&H1, cdlCFScreenFonts:

Ekran fontlarının listede görülmesini sağlar.

&H2, cdlCFPrinterFonts:

Yazıcı fontlarının da listede görülmesini sağlar.

&H3, cdlCFBoth:

Hem ekran hemde yazıcı fontlarının listede görülmesini sağlar.

&H20000, cdlCFScalableOnly:

Yalnız ölçeklenebilir fontların listede görülmesini sağlar. (True Type, Vektör fontlar vb.). Bu fontların özelliği büyültülüp/küçültüldüğünde şekillerinin bozulmamasıdır.

&H40000, cdlCFTTOnly:

Yalnız True Type fontların listede görülmesini sağlar.

&H8000, cdlCFWYSIWYG:

Yalnız, hem ekran hem de yazıcıda kullanılabilen fontların gösterilmesini sağlar. Bu flags değeri özellikle yazıcıdan alınacak çıktıyla ekranda görülen yazının aynı kalitede olması istendiği durumlarda kullanılmalıdır. Diğer durumda da ekranda görülen yazı yazıcıdan aynen çıkar ancak ekran ile yazıcının çözünürlükleri aynı olmadığı için yazım kalitesi düşük olacaktır.

Bu değerin etkili olması için &H1, &H2 ve &H20000 değerlerinin de verilmiş olması gerekir. Yani flags özelliğinin **&H28003** olması gerekir.

&H400, cdlCFANSIOnly:

Symbol fontları gibi yazı içermeyen fontların listede görülmemesini sağlar.

&H800, cdlCFNoVectorFonts:

Vektör fontların seçilmemesini sağlar.

&H4000, cdlCFFixedPitchOnly:

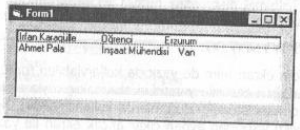
Yalnız "Fixed Pitch" fontların görüntülenmesini sağlar.

Windowsta kullanılan fontlardan çoğunluğunun (Variable Length Font-Times New Roman gibi) harf genişlikleri sabit değildir. "M" harfi "I" harfinden daha geniş bir alan kaplayacaktır. Fontların diğer bir kısmında ise (Fixed Pitch Fontlar-Courier New gibi) bu genişlik her harf için sabittir (Dos'ta olduğu gibi). Font genişliğinin sabit olması çirkin bir görüntü oluştururken özellikle tablolarda kullanılması verilerin aynı hizaya gelmelerini sağlayacaktır.

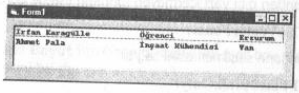
Bu durumu bir örnekle göstermeye çalışalım. Örnek için form üzerine Fontname'si "Times New Roman" olan bir ListBox koyun ve Formun Load olayına aşağıdaki kodu girin.

```
Private  Sub Form_Load( )
 Dim Ad As String *25, Meslek As String *20
 Dim  Memleket As String *15
 Ad = "İrfan Karagülle"
 Meslek = "Öğrenci"
 Memleket = "Erzurum"
 List1.AddItem Ad + Meslek + Memleket
 Ad = "Ahmet Pala"
 Meslek = "İnşaat Mühendisi"
 Memleket = "Van"
 List1.AddItem Ad + Meslek + Memleket
End Sub
```

Programı çalıştırdığınızda aşağıdaki görüntüyü elde edeceksiniz:

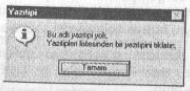


Gördüğünüz gibi string uzunlukları sabit olmasına rağmen listedeki elemanlar alt alta gelmemektedir. Bunun sebebi "Times New Roman" fontunun genişliğinin harfe göre değişmesindendir. Şimdi aynı örneği ListBox'un FontName'sini "Courier New" yaparak çalıştırdığınızda listedeki elemanlar aşağıdaki gibi alt alta gelecektir.



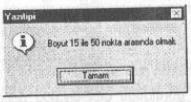
&H10000, cdlCFForceFontExist:

Kullanıcının listede olmayan bir font ismi yazması durumunda Windows'un aşağıdaki mesajı görüntülemesini sağlar ve geçerli bir font ismi seçmesi için kontrolü tekrar diyalog kutusuna bırakır.



&H2000, cdlCFLimitSize:

Dialog kontrolünün Max ve Min propertiesleriyle belirlenen sınırlar haricinde bir font büyüklüğü seçilmesine izin vermez ve aşağıdaki gibi bir mesaj görüntüler.



Flags'ın bu değerini kullanmadan önce CommonDialog kontrolünün Max ve Min propertieslerine geçerli aralığı atamış olmalısınız. Aksi taktirde Max ve Min 0 olarak kabul edilecek ve geçerli bir font büyüklüğü seçimi engellenecektir.

ÖRNEK: Örnek olarak bir text kutusunun fontunu, FontDialog kontrolünü kullanarak değiştirecek bir program yapalım. Bunun için form üzerine bir Text kutusu, bir CommonDialog kontrolü ve Caption'u "Font" olan bir de CommandButton yerleştiriniz. Ayrıca "Etkiler" ve "Renk" kutularını diyalog penceresinde gösterilmesi için CommonDialog kontrolünün **Flags** özelliğine 257 yazın.

```
'ÖRNEK : FontDialog
Private  Sub Command1_Click ()
  'Text kutusunun özelliklerini diyalog kutusunda seç
  CommonDialog1.FontName = text1.FontName
  CommonDialog1.FontSize = text1.FontSize
  CommonDialog1.FontBold = text1.FontBold
  CommonDialog1.FontItalic = text1.FontItalic
  CommonDialog1.FontUnderline = text1.FontUnderline
  CommonDialog1.FontStrikethru = text1.FontStrikethru
  CommonDialog1.Color = text1.ForeColor
   'İptal seçilmesi durumunda hatayı yakala
  On Local Error Goto Iptal
  'Etkiler ve renk kısımları ile ekran-yazıcı fontları
  CommonDialog1.Flags=cdlCFEffects Or cdlCFBoth
  CommonDialog1.ShowFont'Diyalog penceresini görüntüle
  'Diyalog penceresinde seçilenleri Text kutusunda aktif hale
getir.
  text1.FontName = CommonDialog1.FontName
  text1.FontSize = CommonDialog1.FontSize
  text1.FontBold = CommonDialog1.FontBold
  text1.FontItalic = CommonDialog1.FontItalic
  text1.FontUnderline = CommonDialog1.FontUnderline
  text1.FontStrikethru = CommonDialog1.FontStrikethru
  text1.ForeColor = CommonDialog1.Color
Exit Sub
Iptal:
      Exit Sub'İptal seçildi. Altprogramdan çık
End Sub
```

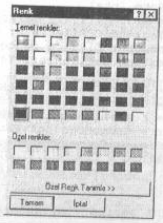

Yukarıdaki önekte görüldüğü gibi diyalog penceresini açmadan önce fontunu değiştirmek istediğimiz Text kutusunun özelliklerinin FontDialog kontrolünün özelliklerine atadık. Böylece kullanıcının yeni bir font seçmeden önce aktif değerleri de görebilmesini sağlamış olduk. Hata ayıklayıcı satırımız ise "İptal" seçilmesi halinde bir değişiklik yapmadan altprogramdan çıkmamızı sağlıyor.

Renk Diyalog Penceresi

Windows tarafından sağlanan standart **Renk** diyalog kutusunun kullanılmasını sağlayan bir kontroldür. Bu kontrolle sistem renklerinin veya kullanıcının tanımlayacağı özel renklerin kullanılması mümkündür.

CommonDialog kontrolünün **Action** özelliğine 3 vererek veya **ShowColor** metodunu kullanarak Renk diyalog penceresi aktif hale getirilir.

CommonDialog.Action=3 veya **CommonDialog1.ShowColor** komutu ile açılacak diyalog penceresi:



Properties

Color

Kullanıcının seçtiği renk numarasıdır. Herhangi bir kontrolün BackColor, ForeColor gibi renk özelliklerinden birine atanarak seçili renk aktif hale getirilebilir.

ÖRNEK: Bir Text kutusu içindeki zemin ve yazı renginin bu kutu aracılığı ile belirlenebileceği bir program yazalım. Kullanıcı Text kutusunu sol fare tuşu ile tıklarsa yazı rengini, sağ tuşla tıklarsa zemin rengini belirlesin.

```
ÖRNEK: Color
Private Sub Text1_MouseDown(Button As Integer, Shift As Integer,
X As Single, Y As Single)
 CommonDialog1.Action = 3
 If Button = 1 Then' sol tuş basılı ise
  Text1.ForeColor = CommonDialog1.Color
 Else
  Text1.BackColor = CommonDialog1.Color
 End If
End Sub
```

Flags

Açılacak diyalog penceresinin bazı özelliklerini belirleyen bir değer alır. Alacağı değerler ve anlamları aşağıda verilmiştir. Bu değerler **OR** işlemine tabi tutularak bir kaç özellik birden aktif hale getirilebilir.

&H2, cdCCIFullOpen :

Diyalog penceresinin normalde sadece renk kısmı açılır. Kullanıcının **Özel Renk Tanımla>>** komut düğmesini tıklamasıyla da pencerenin diğer yarısı açılır. Eğer pencerenin tamamının program tarafından açılmasını istiyorsanız **Flags** özelliğinin bu değerini kullanmalısınız.

&H4, cdlCCPreventFullOpen:

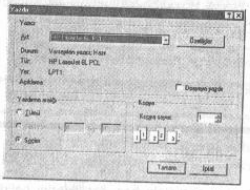
Diyalog penceresinde **Özel Renk Tanımla>>** komut düğmesinin pasif yapılmasını ve kullanıcı tarafından seçilememesini sağlar.

Yazdırma Diyalog Penceresi

Windows tarafından sağlanan standart **Yazdır** diyalog kutusunun kullanılmasını sağlayan bir kontroldür. Bu kontrolle; baskı kalitesini, basılacak sayfa aralığını, kopya sayısını ayarlamak ve **Yazıcı Ayarları** diyalog kutusunu kullanmak mümkündür.

CommonDialog kontrolünün **Action** özelliğine 5 vererek veya **ShowPrint** metodunu kullanarak Yazdır diyalog penceresi aktif hale getirilir.

CommonDialog.Action=5 veya **CommonDialog1.ShowPrinter** komutu ile açılacak diyalog penceresi:



Properties

PrinterDefault

Bu özelliğe **True** değeri verilirse kullanıcının seçtiği yazıcı ayarları varsayılan ayar olarak kaydedilir.

Copies

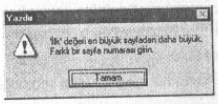
Diyalog penceresinde **Kopya Sayısı** kutusuna yazılan değeri gösterir.

FromPage, ToPage

Diyalog kutusundaki **İlk** ve **Son** sayfa numaralarını belirten sayıları gösterir. Bu değerlerin -1 olması bütün sayfalar anlamına gelir.

Min, Max

Diyalog kutusundaki **İlk** ve **Son** text kutularına girilebilecek minimum ve maximum sınırı belirler. Kullanıcının bu sınır dışında bir değer girmesi halinde Windows aşağıdaki gibi bir mesaj görüntüler.



MaxPage değerini, yazdırılacak dokümanınızın sayfa sayısına eşitleyerek daha büyük bir değerin verilmesini engelleyebilirsiniz.

Flags

Açılacak diyalog penceresinin bazı özelliklerini belirleyen bir değer alır. Alacağı değerler ve anlamları aşağıda verilmiştir. Bu değerler **OR** işlemine tabi tutularak bir kaç özellik birden aktif hale getirilebilir.

&H0, cdlPDAllPages:

Yazdır diyalog penceresinde **Tümü** seçeneğinin seçili olmasını sağlar. Veya Tümü seçeneğinin seçili olup olmadığı bu değerle öğrenilir.

&H1, cdlPDSelection:

Yazdır diyalog penceresinde **Seçili Kısım** seçeneğinin seçili olduğunu gösterir.

&H2, cdlPDPageNums:

Yazdır diyalog penceresinde **Sayfalar** seçeneğinin seçili olduğunu gösterir. Hangi sayfaların girildiği ise Max ve Min özellikleri ile öğrenilir.

&H20, cdlPDPrintToFile:

Yazdır diyalog penceresinde **Dosyaya Yazdır** seçeneğinin seçili olduğunu gösterir.

&H80000, cdlPDDisablePrintToFile :

Yazdır diyalog penceresinde **Dosyaya Yazdır** seçeneğinin pasif (seçilemez) olmasını sağlar.

&H100000, cdlPDHidePrintToFile:

Yazdır diyalog penceresinde **Dosyaya Yazdır** seçeneğinin görüntülenmemesini sağlar.

&H8, cdlPDNoPageNums:

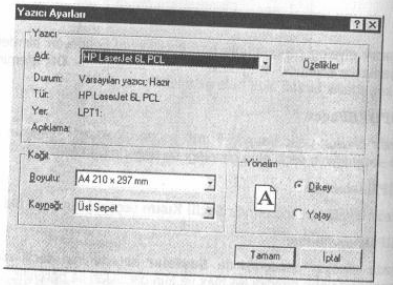
Yazdır diyalog penceresinde **Sayfalar** seçeneğinin ve **İlk, Son** text kutularının pasif olmasını sağlar.

&H4, cdlPDNoSelection:

Yazdır diyalog penceresinde **Seçili Kısım** seçeneğinin pasif olmasını sağlar.

&H40, cdlPDPrintSetup:

Yazdır diyalog penceresi yerine **Yazıcı Ayaları** diyalog penceresinin görüntülenmesini sağlar.

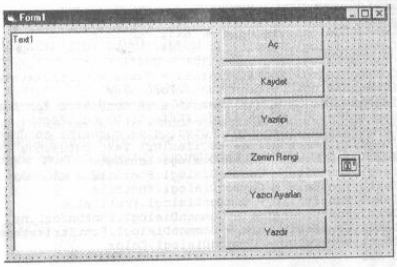


Bu pencere aracılığı ile bir ayar yapıldıktan sonra bu ayarların etkili olabilmesi için **Printer.EndDoc** komutu kullanılmalıdır.

```
CommonDialog1.Flags = &H40
CommonDialog1.Action = 5
Printer.EndDoc
```

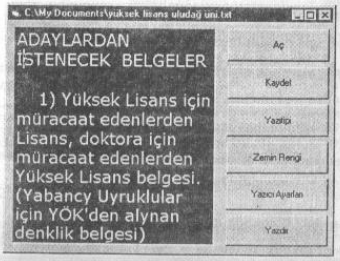
ÖRNEK:

Diyalog kutularının tümünü kullanabileceğimiz bir örnek yapalım. Örneğimiz için aşağıdaki formu hazırlayın.




```
'ÖRNEK: CommonDialog
Private Sub Command1_Click()
 Dim s
 CommonDialog1.DialogTitle = "Açmak için bir dosya seçiniz"
 CommonDialog1.Filter = "Metin dosyaları|*.txt|Bütün
dosyalar|*.*||"
 CommonDialog1.ShowOpen
 Open CommonDialog1.FileName For Input As #1
 Text1 = ""
 While Not EOF(1) 'dosya sonuna gelinceye kadar
  Line Input #1, s 'dosyadan oku
  've text kutusuna ekle
  Text1 = Text1 + Chr(13) + Chr(10) + s
 Wend
 Close #1
 Caption = CommonDialog1.FileName
End Sub

Private Sub Command2_Click()
 Dim s
 CommonDialog1.DialogTitle = "Kaydetmek için bir dosya adı
giriniz"
 CommonDialog1.DefaultExt = "txt"
 CommonDialog1.ShowSave
 Open CommonDialog1.FileName For Output As #1
 s = Text1
 Print #1, s 'dosyaya yaz
 Close #1
 Caption = CommonDialog1.FileName
End Sub

Private Sub Command3_Click()
  'Text kutusunun özelliklerini diyalog kutusunda seç
  CommonDialog1.FontName = Text1.FontName
```

```
    CommonDialog1.FontSize = Text1.FontSize
    CommonDialog1.FontBold = Text1.FontBold
    CommonDialog1.FontItalic = Text1.FontItalic
    CommonDialog1.FontUnderline = Text1.FontUnderline
    CommonDialog1.FontStrikethru = Text1.FontStrikethru
    CommonDialog1.Color = Text1.ForeColor
    'Etkiler ve renk kısımları ile ekran-yazıcı fontları
    CommonDialog1.Flags = cdlCFEffects Or cdlCFBoth
    CommonDialog1.ShowFont 'Diyalog penceresini görüntüle
    'Diyalog penceresinde seçilenleri Text kutusunda aktif yap
    Text1.FontName = CommonDialog1.FontName
    Text1.FontSize = CommonDialog1.FontSize
    Text1.FontBold = CommonDialog1.FontBold
    Text1.FontItalic = CommonDialog1.FontItalic
    Text1.FontUnderline = CommonDialog1.FontUnderline
    Text1.FontStrikethru = CommonDialog1.FontStrikethru
    Text1.ForeColor = CommonDialog1.Color
End Sub

Private Sub Command4_Click()
 CommonDialog1.ShowColor
 Text1.BackColor = CommonDialog1.Color
End Sub

Private Sub Command5_Click()
 CommonDialog1.Filter = &H40
 CommonDialog1.ShowPrinter
 Printer.EndDoc
End Sub

Private Sub Command6_Click()
 Printer.Print Text1
 Printer.EndDoc
End Sub
```



MSMasked (Formatlı Giriş)*

Bu kontrol standart Text kutusu kontrolüne benzer. Ancak Text kutusu formatlı girişlerin gerektiği durumlarda fazla kullanışlı değildir. Örneğin kullanıcının doğum tarihini girmesi gereken bir kutuda kullanıcının doğru girip girmediğini text kutusu ile anında kontrol etmek daha zordur. MsMasked kontrolü ile bilgilerin özel bir düzende girilmesi gerektiği durumlarda hem kullanıcıya hemde programcıya büyük kolaylıklar sağlar.

Properties

Mask

Kullanıcının ne tip giriş yapacağı bu özellikle belirlenir. Bu özelliğe atanacak değerler ve anlamları şöyledir.

Mask	Anlamı	Örnek Görüntü
Boş ("")	Giriş sınırlaması yok Standart Text kutusu gibi davranır.	Sınır yok
(###) ###-####	Kuzey Amerika tipi telefon numarası	(232) 145-5489
##-???-##	Tarih Bilgisi	12-May-96
##-##-##	Tarih Bilgisi	12-06-96
##:## ??	Saat Bilgisi	08:15 pm

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Masked Edit Control" seçeneğini işaretlemeniz veya Browse düğmesi ile MSMASK32.OCX dosyasını bulup projeye eklemeniz gerekir.

Mask	Anlamı	Örnek Görüntü
##:##	Saat Bilgisi	20:15

Mask özelliğine bu hazır değerlerden biri girilebileceği gibi aşağıdaki kurallar çerçevesinde kendi özel maskenizi de tanımlayabilirsiniz.

Karakter	Anlamı	Örnek
#	Yalnız rakam Ör."####"	1234
.	Ondalık basamak yeri Ör."####.###"	1234.045
:	Saat ayırıcı Ör. "##:##:##"	08:15:10
/	Tarih ayırıcı Ör. "##/##/####"	12/06/1996
&	Sadece Ascii kodu 32-126 ve 128-255 arasında olan karakterlerin girilmesine izin verir.Ör. "&&###&&"	a$12#2
A	Sadece harf ve rakam girişi imkanı verir. Ör. "AAA"	2XT
?	Sadece harf girişine izin verir. Ör. "???"	abc
\	#, &, A, ? karakterlerinin maske olarak kullanılmaması için bu karakterlerin önüne konur. Ör. "\###"	#12

Bunların haricinde herhangi bir karakter Mask özelliğine verilirse bu karakterler aynen konur ve kullanıcı bu karakterleri değiştiremez. Örneğin $f(x) = ax^n + bx + c$ şeklindeki bir fonksiyonun katsayılarını kullanıcının girmesini isteyelim. a,b,c ve n tek basamaklı sayılar olsun. Bu durumda Mask özelliğine şu ifadeyi vermemiz gerekir: "f(x)=#x^#+#x+#"

f(x)=_x^_+_x+_ f(x)=5x^2+4x+3

Kullanıcının 0.434.411 31 42 formatında telefon numarası girmesi için ise Mask özelliği: "#,###,### ## ##" olması gerekir.

_.___.___ __ __ 0.434.411 31 42

Kullanıcı belirlenen maske haricinde bir giriş yapamaz. Tanımlanan sınırlar haricinde bir karakter girmeye çalışırsa bu karakter göz ardı edilir ve ValidationError olayı meydana gelir.

Text

Kontrolün içeriği bu özellikle öğrenilir ve değiştirilir. Kontrol kutusundaki kullanıcının girmediği karakterlerde bu özelliğe dahildir.

ClipText

Sadece kullanıcının girdiği karakterler bu özellikle öğrenilir. Maske içinde verilen karakterler bu özelliğe dahil değildir. Bu özellikle kutu içerisine atama yapılmaz. Atama için Text özelliği kullanılır.

Örneğin Mask özelliği "#,###,### ## ##" olan bir Masked kutusunda kullanıcı 0.434.411 31 42 girmiş olsun.

Print MaskEdit1.Text ifadesi **0.434.411 31 42** çıktısını verirken

Print MaskEdit1.ClipText ifadesi **04344113142** çıktısını verir.

ClipMode

0 verilirse Cut ve Copy işlemlerinde Mask özelliğinde verilen karakterlerin alınmasını, 1 verilirse bu karakterlerin alınmamasını sağlar. Yani 0 değeri Text özelliğinin, 1 değeri ClipText özelliğinin panoya kopyalanmasını sağlar.

Format

Maske kutusu içerisindeki ifadenin istenen bir formatta görüntülenmesini sağlar. Örneğin kullanıcının girdiği 10/05/1996 tarihinin 10 May 1996 şeklinde yazdırılabilir. Bu özelliğe atanacak değerler ve anlamları şöyledir.

Format	Anlamı	Örnek Görüntü
Boş ("")	Giriş aynen gösterilir.	 Sınır yok
#,##0.00TL	Girilen sayıları para formatında gösterir. TL,Kuruş	6646464 6.646.464,00TL
0	Sayının tam kısmını görüntüler	1234,312 1234
#,##0	Sayıyı basamaklara ayırır.	4654544465 4.654.544.465
%0	Sayıyı yüzde olarak gösterir.	0,45 %45
0.00E+00	Sayıyı üstel formda gösterir.	4500000 4,50E+06
c	Tarih, saat veya her iki formatta gösterir.	10/1/96 10/1/1996
ddddd	Tarihi gg/aa/yyyy formatında gösterir.	10/may/96 10/5/1996
dddddd	Tarihi "gün gg ay yyyy" formatında gösterir.	6/5/96 Monday 6 May 1996
dd-mmm-yy	Tarihi gg-ay-yy formatında gösterir	13/5/1996 13-May-96

Format	Anlamı	Örnek Görüntü
hh:mm	Saati 24 saat formatında gösterir	1:5 PM 13:05
hh:mm AM/PM	Saati 12 saat formatında gösterir	13:5 01:05 PM
tttt	Saati sa:dk:sn formatında gösterir	5:40 5:40:00

Format özelliği kontrolün Text özelliğini etkilemez. Text özelliği kullanıcının girdiği hali gibidir. Format özelliği yalnız ekranda istenen formatta görüntülenmesini sağlar. Girişin formatlı halini yani kontrol nesnede iken ekranda göründüğü hali öğrenmek için FormattedText özelliği kullanılır.

FormattedText

Maskedit kutusunun format özelliği ile belirlenen formattaki bilgi bu özellikle öğrenilir.

MaxLength

Maskedit kutusuna maksimum 64 karakter girilebilir. İstenirse bu özelliğe 0 haricinde bir sayı atanarak girilebilecek maksimum karakter sayısı sınırlanabilir.

PromptChar

Maskedit kutusunda, mask özelliği ile belirlenen formattaki giriş sınırlarını göstermek için, girilmeyen her bir karakter yerine _ karakteri konur. Bu karakter istenirse PromptChar özelliği ile değiştirilebilir.

46_-__-____

PromptInclude

Text özelliği içinde prompt karakterinin gösterilip gösterilmeyeceği bu özellikle belirlenir.

AutoTab

True yapılırsa Mask özelliği ile belirlenen formatta bilgi girildikten sonra kontrolün bir sonraki nesneye geçmesini sağlar. Bir çok bilginin girilmesi gereken formlarda kullanıcıya hız kazandırmak için düşünülebilir.

Events

ValidationError (invalidtext As String, startposition As Integer)

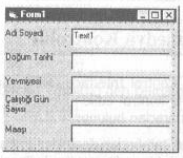
Kullanıcının Mask özelliği ile belirlenen sınırlar haricinde bir karakter girmesi durumunda bu karakter göz ardı edilir ve bu olay meydana gelir.

InvalidText parametresi ile geçersiz karakterle birlikte kutunun içeriği, **StartPosition** parametresi ile geçersiz karakterin girilmek istendiği konum öğrenilebilir.

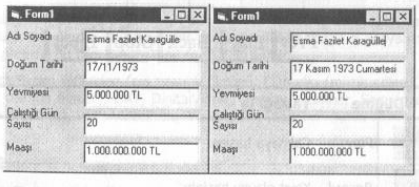
ÖRNEK: Örneğin kullanıcının "gg/ay/yyyy" formatında tarih girmesini istediğimiz bir masked kutusu olsun. Bu kutuda kullanıcının yanlış giriş yapmaya çalışması durumunda yol gösterecek mesajlar verelim.

```
'ÖRNEK : MaskEdit, ValidationError
Private  Sub Form_Load ()
  MaskedEdit1.Mask = "##/???/####"
End Sub

Private  Sub MaskedEdit1_ValidationError (invalidtext As String,
startposition As Integer)
Select Case startposition
  Case Is < 3: MsgBox ("Buraya günü giriniz. 05 gibi")
  Case 3 To 6: MsgBox ("Buraya ayı giriniz. May gibi")
  Case Is > 3: MsgBox ("Buraya yılı giriniz. 1996 gibi")
End Select
End Sub
```

ÖRNEK: Örnek olarak aşağıdaki formda bulunan doğum tarihi, yevmiye, çalıştığı gün ve maaş kutularında MaskEdit kontrolü kullanarak bunlara girilen bilginin formatlanmasını sağlayalım.




```
Private Sub Form_Load()
 MaskEdBox1.Format = "dd mmmm yyyy dddd"
 MaskEdBox1.Mask = "##/##/####"

 MaskEdBox2.Format = "#,##0 TL"
 MaskEdBox2.Mask = "#######"

 MaskEdBox3.Mask = "##"

 MaskEdBox4.Format = "#,##0 TL"
 MaskEdBox4.Mask = "##########"
End Sub
```

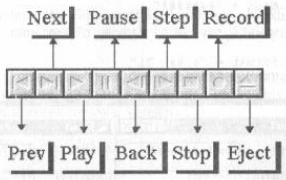


MCI (Multimedya Kontrolü)*

MCI kontrolü, Media Control Interface (MCI) sürücülerini kullanmamızı sağlar. Bu eleman aracılığıyla kontrol edilen elemanlar arasında CD-ROM sürücü, ses kartı vb. gibi donanım araçları bulunur.

MCI kontrolü Windows tarafından desteklenen Multi medya işlemlerini kullanıma sokar. Herhangi bir tipteki media aygıtını kullanabilmek için o aygıtın sürücüsünün Windows'a tanıtılmış olması gerekir.

MCI elemanı bir çok düğmeden oluşmaktadır. **MCI** kontrolü üzerindeki düğmeler ve bu düğmelerin görevlerini şöyle sıralayabiliriz:



Düğme		Yaptığı İş
▶	Play	Çalmaya başlar.
●	Record	Kayıt olayını başlatır.
❚❚	Pause	Çalma veya kaydetme olayını durdurur. Tıklandıktan sonra tekrar tıklanırsa çalma yada kaydetme olayını devam ettirir.
■	Stop	Çalma yada kaydetme olayını bitirir.

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Multimedia Control" seçeneğini işaretlemeniz veya Browse düğmesi ile MCI32.OCX dosyasını bulup projeye eklemeniz gerekir.

Düğme		Yaptığı İş
▶|	Next	Bir sonraki Track(iz) den çalma olayını devam ettirir. (CD'de sonraki parçaya)
|◀	Prev	Bir önceki Track(iz) den çalma olayını devam ettirir. (CD'de önceki parçaya)
|▶	Step	Okuyucu kafasını bir adım ileri alır.
◀|	Back	Okuyucu kafasını bir adım geri alır.
⏏	Eject	Sürücüden çıkarılır.(CD'de: CDROM sürücüden çıkarılır)

Properties

DeviceType

MCI aygıtının kullanılacağı birimin adı. Bu isim Windows tarafından belirlenen aşağıdaki stringlerdir. MCI aygıtı bir dosyayı çalabilmek için o dosyanın tipini bilmesi gerekir. Bu bilgi DeviceType özelliği ile verilir.

DeviceType: AVIVideo (avi uzantılı filmler), CDAudio (müzik cdleri), MMMovie (mov uzantılı filmler), DAT, DigitalVideo, Overlay, Scanner, Sequencer (mid uzantılı müzikler), VCR, VideoDisk, WaveAudio, Other

Filename

DeviceType belirlendikten sonra **Filename** özelliği ile çalınacak (veya kayıt yapılacak) dosyanın ismi belirlenir. Bazı birimler dosya kullanmayabilir. Örneğin **DeviceType** olarak **CDAudio** belirlendiğinde **FileName** özelliği etkisizdir.

Command

Dosya adını belirledikten sonra bu dosyayı açmak ve diğer işlemleri yapmak için Command özelliği ile yapılacak işlem belirlenir. Verilecek komutlar şunlardır.

Command: Open, Close, Play, Pause, Stop, Back, Step, Prev, Next, Seek, Record, Eject, Sound, Save.

Bir dosya çalınmadan önce açılması (open) ve bir başka dosyanın açılabilmesi için de açık olan dosyanın kapatılması (close) gerekir.

ÖRNEK: Örnek olarak kullanıcının seçtiği Wav dosyasını çalacak bir program yapalım. Örneğimiz için formunuzun üzerine bir **CommonDialog** kontrolü, bir **MCI** kontrolü ve bir komut düğmesi yerleştirin.

```
'ÖRNEK: DeviceType, FileName, Command
Private Sub Command1_Click()
 MMControl1.Notify = False
 MMControl1.Wait = True
 MMControl1.Shareable = False
 MMControl1.DeviceType = "WaveAudio"
 CommonDialog1.Filter = "Wav dosyaları|*.wav"
 CommonDialog1.DialogTitle = "Wav dosyası seç"
 CommonDialog1.ShowOpen
 MMControl1.FileName = CommonDialog1.FileName
 MMControl1.Command = "Open" 'Dosyayı aç
End Sub
```

Bu örnekte sadece **Open** komutunu verdiğimiz için dosya açılacak ancak çalınmaya başlamayacaktır. Dosya açıldıktan sonra MCI üzerindeki düğmeler aktif olacağı için kullanıcı istediği andan Play düğmesine basabilir. Otomatik olarak çalmasını istiyorsanız MMControl1.Command="Play1" komutunu yazabilirsiniz.

Wait

True verilirse MCI verilen komutu yerine getirinceye kadar kontrolü programa bırakmaz. Bu da üst üste iki MCI komutunun aynı anda gelmemesini sağlar.

Shareable

Cihaz açılmadan önce birçok uygulama tarafından kullanılıp kullanılmayacağı belirlenmelidir. Normalde **False**'dir ve diğer uygulamalar MCI aygıtını siz kullandığınız sürece kullanamazlar. **True** yaparsanız kullanabilirler. Bazı media tipleri shareable olarak çalışamazlar.

Orientation

0 için MCI kontrolü yatay, 1 için dikey görüntülenir.

PrevVisible, NextVisible, PlayVisible, PauseVisible, BackVisible, StepVisible, StopVisible, RecordVisible, EjectVisible

Belirtilen düğmelerin görüntülenmesini veya görüntülenmemesini sağlar.

PrevEnabled, NextEnabled, PlayEnabled, PauseEnabled, BackEnabled, StepEnabled, StopEnabled, RecordEnabled, EjectEnabled

Belirtilen düğmelerin aktif veya pasif olmasını sağlar.

AutoEnable

Bu özellik True ise düğmelerin durumu MCI tarafından otomatik olarak Enabled veya Disabled yapılır. Bu değer False yapılırsa düğmelerin program tarafından ayarlanması gerekir. Örneğin MCI aygıtı bir dosyayı çalarken Play düğmesinin pasif olması ve çalma bittiğinde ise aktif olması gerekir. Eğer bu özelliğe True verilmişse bu tür işlemler otomatik olarak yapılır.

CanEject, CanPlay, CanRecord, CanStep

Tipi belirlenen MCI aygıtının Eject, Play, Record ve Step olaylarını destekleyip desteklemediği bu özelliklerle öğrenilebilir. Bu özelliklerin değeri True ise destekleniyordur.

```
Dim msg
If mmcontroll.CanEject Then msg = "eject "
If mmcontroll.CanPlay Then msg = msg + "play "
If mmcontroll.CanRecord Then msg = msg + "record "
If mmcontroll.CanStep Then msg = msg + "step "
Msg = Msg + "Olayları destekleniyor."
MsgBox (Msg)
```

RecordMode

MCI aygıtı ile kayıt yapılacaksa **Command** özelliğine **Record** komutu verilmeden önce bu özellikle kayıt modu belirlenir. 0 için kayıt varolan konumdan itibaren 1 için varolan kayıt üzerine yapılır.

hWndDisplay

Çalınacak dosya görüntüler içeriyorsa, bu özellikle görüntülerin gösterileceği kontrol belirlenir. Örneğin bir AVI dosyasındaki video bir picture kutusunda gösterilecekse bu özelliğe PictureBox'un hwnd özelliği verilir.

```
MMControl1.hWndDisplay = Picture1.hWnd
```

Bu özelliğe 0 verilirse Windows varsayılan bir pencerede videoları gösterecektir.

Error

Verilen komut yerine getirilmişse bu özellik 0 değerini alır. Yani bu değer 0 değilse komut yerine getirilmemiştir. Hatanın mesajı ise ErrorMessage özelliği ile öğrenilebilir.

ErrorMessage

MCI aygıtına verilen komutta bir hata oluşursa bu hata VB tarafından algılanmaz ve herhangi bir hata mesajına sebep olmaz. Oluşan bu hatanın mesajını bu özellikle öğrenebilirsiniz.

```
If MMControl1.Error <> 0 Then
  MsgBox ("Hata:" & MMControl1.ErrorMessage)
End If
```

TimeFormat

MCI tarafından açılan dosyanın zaman formatı bu özellikle belirlenir. Bu özelliğin alacağı değerler ve anlamları şöyledir.

0. Mili saniye, 4 Byte'lık bir integer alanda Mili saniye olarak tutulur.
1. Saat:Dakika:Saniye, 4 Byte'lık bir integer alanda süre bu formatta tutulur. En düşük seviyeli byte saati, sonraki dakikayı ve üçüncü byte saniyeyi tutar. En yüksek seviyeli byte kullanılmaz.
2. Dakika:Saniye:Çerçeve, 4 Byte'lık bir integer alanda süre bu formatta tutulur. En düşük seviyeli byte dakikayı, sonraki saniyeyi ve üçüncü byte çerçeve sayısını tutar. En yüksek seviyeli byte kullanılmaz.
3. Çerçeve sayısı, 4 Byte'lık bir alanda tutulur.
4. Saat:Dakika:Saniye:Çerçeve, 24 çerçevelik paketler halinde, 4 Byte'lık bir alanın en düşük byte'ında saat ve diğerleride sırasıyla diğer byte'larda tutulur.
5. 4 değeri gibidir. 25 çerçevelik paketler tutulur.
6. 4 değeri gibidir. 30 çerçevelik paketler tutulur.
7. 4 değeri gibidir. 30 drop-frame paketler tutulur.
8. Byte, 4 Byte'lık bir alanda byte olarak uzunluğu tutulur.
9. Sample, 4 Byte'lık bir alanda örnek sayısı tutulur.
10. İz:Dakika:Saniye:Çerçeve, 4 Byte'lık bir alanda ilk byte dakikayı ve diğerleri sırasıyla diğer bytelarda tutulur.

Length

Açık olan dosyanın **TimeFormat** özelliği ile belirlenen birimde uzunluğu bu özellik tarafından öğrenilir.

Mode

MCI aygıtının o anki durumu bu özellik ile öğrenilebilir. Bu özelliğin alabileceği değerler ve anlamları şöyledir.

524: Birim açık değil
525: Birim durmuş
526: Birim çalıyor
527: Birim kaydediyor
529: Birim beklemede (Pause)
530: Birim hazır

UpdateInterval

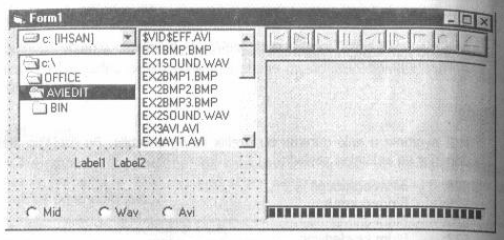
MCI kontrolü açık ve çalıyorsa çaldığı dosya hakkındaki bilgiyi belirli sürelerle öğrenmek gerekebilir. Bu süre UpdateInterval özelliği ile Mili saniye olarak belirlenebilir. Bu özellikle belirlenen süre periyotlarınca StatusUpdate olayı meydana gelir. Bu olay yalnız MCI aygıtı çalarken meydana gelir.

Events

StatusUpdate ( )

UpdateInterval özelliği ile belirlenen periyotlarla bu olay meydana gelir. Daha çok çalan dosyanın pozisyonu hakkında bilgi göstermek için kullanılır.

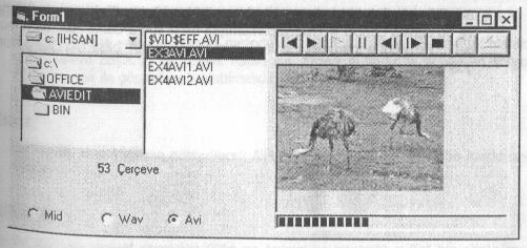
ÖRNEK: Örnek olarak Wav, Mid ve AVI dosyalarını çalacak bir program yapalım. Örneğimiz için aşağıdaki formu bir PictureBox ile birlikte oluşturun. Ayrıca COMMCTL32.OCX dosyasında bulunan **ProgressBar** kontrolünü de çalan dosyanın pozisyonunu göstermek için kullanacağız.




```
'ÖRNEK :MMControl

Private Sub Form_Load()
    MMControl1.Notify = False
    MMControl1.Wait = True
    MMControl1.Shareable = False
End Sub

Private Sub Dir1_Change()
    ChDir Dir1.Path
    File1.Path = Dir1.Path
End Sub

Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
    File1.Path = Drive1.Drive
    ChDrive Drive1.Drive
End Sub

Private Sub File1_Click()
    MMControl1.Command = "Close"'Açık olan varsa kapat
    MMControl1.filename = File1.filename
    MMControl1.Command = "Open"
    If MMControl1.Error <> 0 Then
        MsgBox ("Hata: " + MMControl1.ErrorMessage)
    End If
    MMControl1.Command = "Play"
        If MMControl1.Error <> 0 Then
        MsgBox ("Hata: " + MMControl1.ErrorMessage)
    End If
    Label1 = MMControl1.Length
    ProgressBar1.Max = MMControl1.Length
End Sub
```



```
Private Sub MMControl1_StatusUpdate()
    If MMControl1.Mode = 526 Then 'Çalıyor ise
        ProgressBar1.Value = MMControl1.Position
    End If
End Sub

Private Sub Option1_Click()
    File1.Pattern = "*.mid"
    MMControl1.DeviceType = "Sequencer"
    MMControl1.TimeFormat = 0'Mili saniye
    Label2 = "Mili Saniye"
End Sub

Private Sub Option2_Click()
    File1.Pattern = "*.wav"
    MMControl1.DeviceType = "WaveAudio"
    MMControl1.TimeFormat = 0'Mili saniye
    Label2 = "Mili Saniye"
End Sub

Private Sub Option3_Click()
    File1.Pattern = "*.avi"
    MMControl1.DeviceType = "AVIVideo"
    'Video görüntüleri Picture1 içinde gösterilecek
    MMControl1.hWndDisplay = Picture1.hWnd
    MMControl1.TimeFormat = 3'Çerçeve sayısı
    Label2 = "Çerçeve"
End Sub
```



Done (NotifyCode As Long)

MCI kontrolüne verilen komut yerine getirildikten (veya getirilemedikten) sonra bu olay meydana gelir. NotifyCode parametresinin aldığı değere göre yapılan işlemin sonucu hakkında bilgi alınabilir.

1: Komut başarıyla tamamlandı,

2: Komut başka bir komut işini bitirmeden verildi,

4: Komut kullanıcı tarafından iptal edildi,

8: Komut hatalı

BackClick, EjectClick, NextClick, PauseClick, PlayClick, PrevClick, RecordClick, StepClick, StopClick(Cancel As Integer)

Bu düğmelerden biri kullanıcı tarafından tıklandığında bu olaylardan biri aktif hale gelir. Eğer kullanıcı tarafından verilen bu komut iptal edilmek isteniyorsa Cancel = True yapılır.

BackCompleted, EjectCompleted, NextCompleted, PauseCompleted, PlayCompleted, PrevCompleted, RecordCompleted, StepCompleted, StopClick (ErrorCode As Long)

Bu düğmelerden biri kullanıcı tarafından tıklandığında Click olayı meydana gelir ve bu olayda Cancel = True yapılmamışsa daha sonra Completed olayı meydana gelir. ErrorCode parametresinin 0 olması komutun başarıyla tamamlandığını gösterir.

Animation* (Avi Gösterici)

Windows-95/98 ile birlikte gelen kontrollerden biri de Animation kontrolüdür. Bu kontrolü kullanarak AVI dosyalarını kolayca oynatabilirsiniz. Ne yazık ki bu kontrol bütün AVI dosyalarını desteklememektedir. Gösterilecek AVI dosyasının sıkıştırılmamış olması veya RLE algoritması ile sıkıştırılmış olması gerekir. Ayrıca içinde ses bulunan AVI dosyaları da desteklenmemektedir. Aslında bu kontrolün temel amacı klasik anlamdaki, film içeren avi dosyalarını göstermek değildir. Bu tür AVI dosyaları için MCI kontrolünü kullanmalısınız. Bu kontrolün görevi ise Windows 95 ve 98'den alışık olduğunuz, bir iş yaparken küçük bir animasyonu göstermektir. Örneğin Windows 95 veya 98'de bir dosya kopyalarken veya silerken karşınıza gelen aşağıdaki türde animasyonları göstermektir.

Bu tür animasyonları COMMON\GRAPHICS dizini altındaki VIDEOS dizini altında bulabilirsiniz.

Properties

AutoPlay

Bu özellik **True** yapılırsa, **Open** metodu ile açılan AVI dosyası otomatik olarak gösterilmeye başlanır. Bu özellik **False** ise, **Open** metodu ile açıldıktan sonra **Play** metodu ile gösterime başlatılmalıdır.

Center

Bu özellik True yapılırsa gösterilecek AVI filmi Animation kontrolünün içinde ortalanır.

BackStyle

Bu özelliğin değeri normalde 0'dır ve Animation kontrolü şeffaf bir görünüme sahiptir. Yani filmin altında kalan nesne görünür durumdadır. Bu özellik 1 yapılır-

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Windows Common Controls-2 6.0" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCOMCT2.OCX dosyasını bulup projeye eklemeniz gerekir.

sa Animation kontrolü kendi zemin rengine sahip olur ve altında kalan nesne görülmez.

Methods

Open Dosya

Animation kontrolü ile gösterilmek istenen AVI dosyası bu metodla açılır. Eğer AutoPlay özelliği True ise film otomatik olarak gösterilmeye başlanır. Diğer durumda Play metodu kullanılarak çalma başlatılabilir.

```
Animation1.Autoplay=true
Animation1.Open "c:\klip.avi"
```

Animation kontrolü her türde AVI dosyasını desteklediği için, Open metodundan önce aşağıdaki gibi bir hata yakalama kodu kullanılması yerinde olacaktır.

```
Sub Command1_Click()
 On Local Error Goto hata
 Animation1.Autoplay=true
 Animation1.Open "c:\klip.avi"
exit sub
hata:
 Msgbox("Bu dosya türü desteklenmiyor")
 Exit Sub
End Sub
```

Play

AutoPlay özelliği True iken bir dosya açıldığında Animation kontrolü otomatik olarak o dosyayı çalmaya başlar. Eğer AutoPlay özelliği False ise Play metodu ile çalma başlatılabilir.

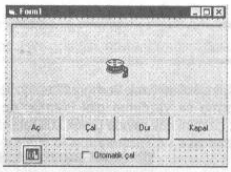
Stop

Bu metod kullanılarak AVI dosyasının çalması durdurulabilir.

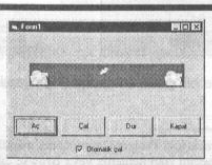
Close

Open metodu ile açılan AVI dosyası Close metodu ile kapatılabilir.

ÖRNEK: Örnek olarak kullanıcının seçeceği bir AVI dosyasını çalacak program yapalım. Örneğimiz için formunuza bir Animation kontrolü, bir CommonDialog kontrolü ve aşağıdaki diğer kontrolleri yerleştirin.




```
'ÖRNEK: Animation
Private Sub Check1_Click()
 Animation1.AutoPlay = Check1.Value
End Sub

Private Sub Command1_Click()
 On Local Error GoTo hata
 CommonDialog1.Filter = "avi (*.avi)|*.avi"
 CommonDialog1.ShowOpen
 Animation1.Open CommonDialog1.FileName
 Exit Sub
hata:
 MsgBox ("Bu dosya türü desteklenmiyor")
 Exit Sub
End Sub
Private Sub Command2_Click()
 On Local Error Resume Next
 Animation1.Play
End Sub
Private Sub Command3_Click()
 On Local Error Resume Next
 Animation1.Stop
End Sub

Private Sub Command4_Click()
 On Local Error Resume Next
 Animation1.Close
End Sub
```



Chart*

Bu kontrol elemanı kullanılarak veri grafikleri çizilebilir. Keza kopyalama, ve yapıştırma, kaydetme ve yazdırma gibi işlemler de gerçekleştirilebilire. Verilerin grafiksel olarak çizimini çok sayıda grafik stili tarafından desteklenir.

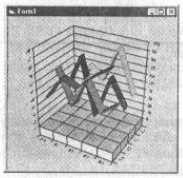


Chart kontrolünün bir çok özelliğini tasarım anında görerek değiştirmek istersenız; form üzerine aldığınız Chart kontrolünü sağ fare tuşu ile tıklayın ve açılan pencereden **Properties** komutunu seçin. Böylece; açılacak olan aşağıdaki pencere ile bir çok özelliği görerek değiştirebilirsiniz.

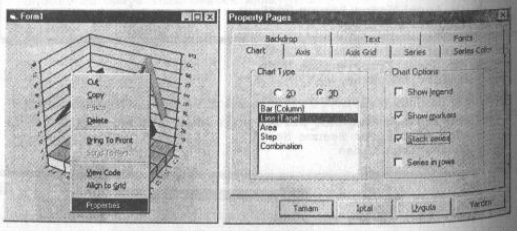


Chart kontrolü ile bir grafik oluşturmak için yapılması gereken işlemler şöyle özetleyebiliriz.

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Chart Control" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCHRT20.OCX dosyasını bulup projeye eklemeniz gerekir.

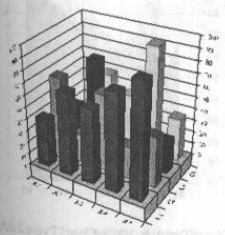
- **ChartType** özelliği ile grafiğin şeklini belirlemek,
- Kontrol üzerinde kaç ayrı grafiğin gösterileceğini **ColumnCount** ile belirlemek.
- Her bir grafiğe ait kaç veri olduğunu **RowCount** özelliği ile belirlemek.
- **Data** özelliği ile her bir noktanın değerlerini belirlemek.

Chart kontrolü ile bir grafiğin oluşturulmasını bu adımlarla özetleyebiliriz. Bu özelliklerin her birinin nasıl atanacağını sırasıyla anlatacağız . Burada örnek olarak adım-adım bir grafiğin nasıl oluşturulacağını da vereceğiz.

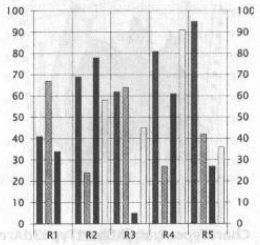
Properties

ChartType

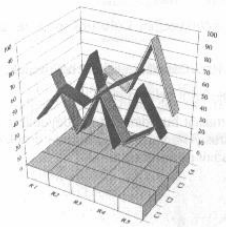
Grafik tipini belirlemek için bu özellik kullanılır. Alacağı değerler ve grafikler aşağıda verilmiştir.



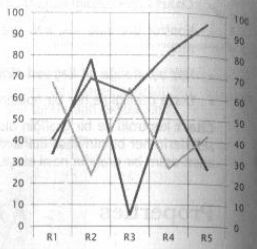
ChartType:0, VtChChartType3dBar



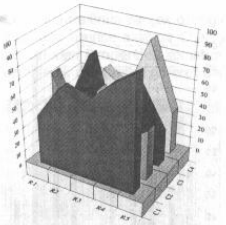
ChartType:1, VtChChartType2dBar



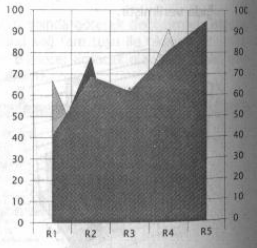
ChartType:2, VtChChartType3dLine



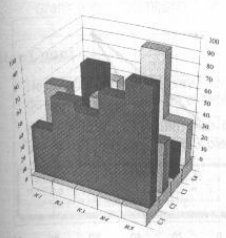
ChartType:3, VtChChartType2dLine



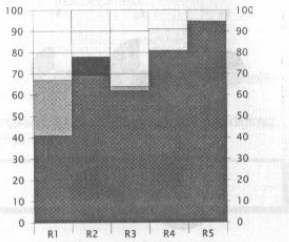
ChartType:4, VtChChartType3dArea



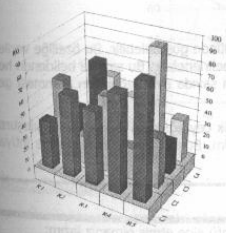
ChartType:5, VtChChartType2dArea



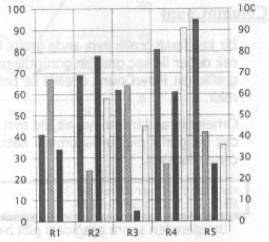
ChartType:6, VtChChartType3dStep



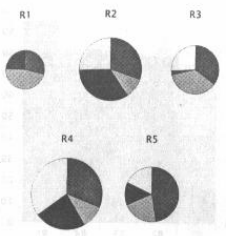
ChartType:7, VtChChartType2dStep

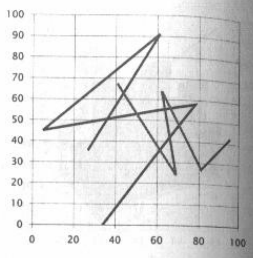


ChartType:8, VtChChartType3dCombination



ChartType:9, VtChChartType2dCombination





ChartType:14, VtChChartType2dPie **ChartType:16, VtChChartType2dXY**

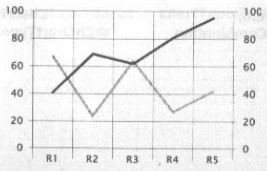
ColumnCount

Bir kaç farklı grafik aynı anda grafik kontrolünde gösterilebilir. Bu özelliğe verilecek değer ile kaç grafiğin görüntüleneceği belirlenebilir. Bu sayı ile belirlenen her grafik için **RowCount** özelliği ile belirlenen sayıda ayrı değerlerin atanması gerekir.

Örnek olarak bir firmaya ait, iki ayrı yılın, ilk çeyreğindeki gelirlerini karşılaştıracak bir grafik oluşturduğumuzu kabul edelim. Bunun için iki ayrı grafiğe ihtiyacımız olacaktır.

```
MsChart1.ChartType = 3
MsChart1.ColumnCount= 2
```

Bu işlemden sonra aşağıdaki gibi bir görüntü elde etmiş olmanız lazım:



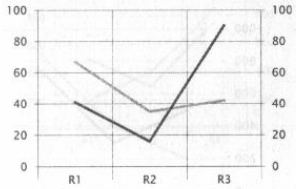
Grafik üzerindeki değerler şimdilik rasgele olarak seçilmiştir.

RowCount

Her bir grafiğin nokta (veya veri) sayısı bu özellik aracılığı ile belirlenir. Bu özelliğe yapılan atama ile veri ekseni o kadar parçaya bölünür.

Örneğimizde ki firmanın, yılın ilk çeyreğine ait gelirini karşılaştırmak istediğimiz için **RowCount** değerinin 3 olarak değiştirelim.

```
MsChart1.ChartType = 3
MsChart1.ColumnCount = 2
MsChart1.RowCount = 3
```



Row, Column

Üzerinde işlem yapılacak nokta bu özelliklerle belirlenir.

Data

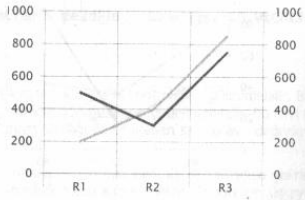
Row ve **Column** özellikleri ile belirlenen noktanın değeri bu özellikle verilebilir.

Örneğimizde firmanın iki yılın ilk çeyreğindeki gelirleri şöyle olsun.

1994		1995	
Ocak	500	Ocak	200
Şubat	300	Şubat	400
Mart	750	Mart	850

Grafiğimize, **Data** özelliği ile bu değerleri atayalım.


```
MsChart1.Column=1
MsChart1.Row=1
MsChart1.Data=500'1994 Ocak
MsChart1.Row=2
MsChart1.Data=300'1994 Şubat
MsChart1.Row=3
MsChart1.Data=750'1994 Mart
MsChart1.Column=2
MsChart1.Row=1
MsChart1.Data=200'1995 Ocak
MsChart1.Row=2
MsChart1.Data=400'1995 Şubat
MsChart1.Row=3
MsChart1.Data=850'1995 Mart
```



ChartData

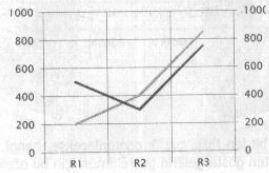
Grafik üzerindeki bir noktanın değerini belirlemek için **Data** özelliğini kullanıyorduk. Ancak yukarıdaki örnekte de gördüğünüz gibi toplu değer atamada **Data** özelliği pek uygun bir yöntem değildir. **ChartData** özelliği ile bütün değerler bir diziye atandıktan sonra, dizi ismi bu özelliğe verilebilir.

Tanımlanacak olan dizi 2 boyutlu olmalıdır. Dizinin ikinci boyutu hangi grafik setinin kullanılacağını, birinci boyutu grafik üzerindeki veriyi gösterir.

Örneğimizde iki farklı yıla ait 3 ayın grafiğini çizdiriyorduk. Bu durumda tanımlamamız gereken dizi x(3,2) şeklinde olmalıdır. Yani üç noktadan oluşan iki farklı grafik.

Ayrıca VB'de diziler ilk elemandan başlamaktadır. Chart kontrolünde ise Column ve Row numarası birden başlar. Bu yüzden diziyi (1 to n) şeklinde tanımlamak gerekir. Örneğimizdeki dizi X(1 To 3, 1 To 2) şeklinde olmalıdır.

```
Private Sub Form_Load()
 ReDim X(1 To 3, 1 To 2)
 X(1, 1) = 500 '1994 Ocak
 X(2, 1) = 300 '1994 Şubat
 X(3, 1) = 750 '1994 Mart
 X(1, 2) = 200 '1995 Ocak
 X(2, 2) = 400 '1995 Şubat
 X(3, 2) = 850 '1995 Mart
 MSChart1.ChartData = X
End Sub
```

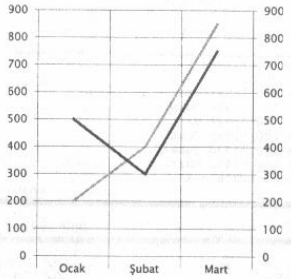


ColumnLabel, RowLabel

Grafikteki her verinin neyi temsil ettiğini belirten metin bu özelliklerle belirlenir. **Column** veya **Row** özelliği ile değiştirilmek istenen nokta seçildikten sonra bu iki özellikle o noktanın etiketi verilebilir.

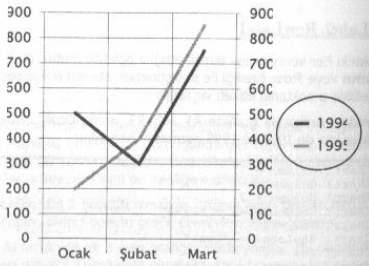
Yukarıdaki örnekteki grafikte R1, R2, R3 yerine Ocak, Şubat, Mart yazmasını, grafikler için de 1994 ve 1995 yazmasını isteyelim:

```
MSChart1.Column = 1
MSChart1.ColumnLabel = "1994"
MSChart1.Column = 2
MSChart1.ColumnLabel = "1995"
MSChart1.Row = 1
MSChart1.RowLabel = "Ocak"
MSChart1.Row = 2
MSChart1.RowLabel = "Şubat"
MSChart1.Row = 3
MSChart1.RowLabel = "Mart"
```



ShowLegend

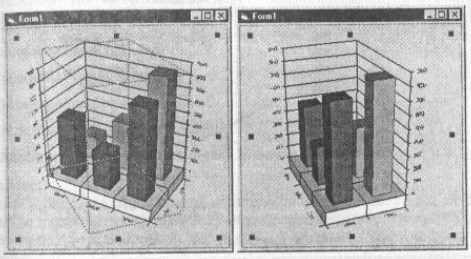
Kart üzerinde birden fazla grafik gösterilecekse, hangi rengin hangi grafiğe ait olduğunu belirten göstergelerin gösterilmesi için bu özellik **true** yapılmalıdır.



AllowDynamicRotation

Bu özellik True'dir. Ve hiç bir koda gerek kalmadan kullanıcı, 3 boyutlu grafikleri döndürebilir. Bu özellik False yapılırsa döndürme işlemini kullanıcı yapamaz. Ancak program kodu ile yaptırılabilir.

Bu özellik True iken kullanıcı Ctrl tuşunu basılı tutarak grafiği, fare yardımıyla döndürebilir.



AutoIncrement

Data özelliği ile grafik üzerindeki noktaların değerlerini belirlerken önce **Column** ve **Row** özelliği ile hangi noktanın değerini vereceğimizi belirtiyorduk. İstenirse bu özelliği **true** değeri verilerek her **Data** atamasında sonra yeni noktayı belirtmeye gerek kalmadan bir sonraki noktaya geçmesini sağlayabiliriz. (Önceki Vb versiyonlarında bulunan Graph nesnesi gibi)

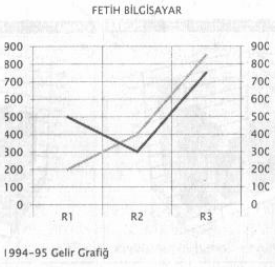
```
MsChart1.AutoIncrement=true
MsChart1.Row=1
MsChart1.Data=100
MsChart1.Data=300
MsChart1.Data=200
```

Yukarıdaki örnekte **AutoIncrement** özelliği True yapıldığı için **Data** özelliğiyle yapılan her atamadan sonra bir sonraki noktayı belirtmeye gerek kalmadan sonraki noktanın değeri atanabilmiştir.

TitleText, FootnoteText

Grafiğin üstüne ve altına yazılması istenen metinler bu iki özellikle belirlenir.

```
MSChart1.TitleText = "FETİH BİLGİSAYAR"
MSChart1.FootnoteText = "1994-95 Gelir Grafiği"
```



Title, Footnote

Grafiğin üstüne ve altına yazılacak yazıları **TitleText** ve **FootnoteText** özellikleri ile belirleyebiliyorduk. Bunlara ait diğer özellikleri (yazı tipi gibi) ise bu iki özelliğin alt özellikleri ile belirleyebiliriz. Her iki özellik de **Title** nesnesi olarak tanımlanmıştır ve aşağıdaki alt özellikleri vardır.

VtFont

Metnin yazı tipi özellikleri bu özellikle belirlenir.

```
MsChart1.Title.VtFont.Name="Times New Roman Tur"
MsChart1.Title.VtFont.Size=12
```

Location

Metnin yeri bu özeliğin aşağıdaki alt özellikleri ile belirlenir.

Location.Visible

Bu özelliğe False verilerek metin gizlenebilir.

Location.LocationType

Bu özelliğe aşağıdaki değerlerden biri verilerek metnin yazılacağı konum belirlenir.

VtChLocationTypeTop: Üst

VtChLocationTypeTopLeft: Sol üst

VtChLocationTypeTopRight: Sağ üst

VtChLocationTypeLeft: Sol

VtChLocationTypeRight: Sağ

VtChLocationTypeBottom: Alt

VtChLocationTypeBottomLeft: Sol alt

VtChLocationTypeBottomRight: Sağ alt

VtChLocationTypeCustom: **Rect** özelliği ile belirlenen koordinat.

Location.Rect

Metnin yerleşim yer olarak Custom verilmişse bu özelliğim Max ve Min alt özellikleri ile metnin yazılacağı alanın köşe koordinatları belirlenir. Max ve Min özelliklerinin de X ve Y alt özellikleri vardır.

```
MSChart1.Title.Location.LocationType = VtChLocationTypeCustom
MSChart1.Title.Location.Rect.Max.X = 800
MSChart1.Title.Location.Rect.Max.Y = 400
MSChart1.Title.Location.Rect.Min.X = 0
MSChart1.Title.Location.Rect.Min.Y = 0
```

TextLayout

Metnin yerleşimi bu özelliğin aşağıdaki alt özellikleri ile belirlenir:

TextLayout.Orientation

Metnin yatay mı veya dikey mi olacağı bu özellikle belirlenir. Alabileceği değerler ve anlamları şunlardır.

VtOrientationHorizontal: Metin yatay olarak yazılır.

VtOrientationVertical: Metin dikey olarak yazılır.

VtOrientationUp: Metin aşağıdan yukarıya doğru yazılır.

VtOrientationDown: Metin yukarıdan aşağıya doğru yazılır.

TextLayout.HorzAlignment

Metnin yatay yerleşimi bu özelliğe verilecek aşağıdaki değerlerle belirlenir.

VtHorizontalAlignmentLeft: Sola

VtHorizontalAlignmentRight: Sağa

VtHorizontalAlignmentCenter: Ortaya

TextLayout.VtVerticalAlignment

Metnin dikey yerleşimi bu özeliğe verilecek aşağıdaki değerlerle belirlenir.

VtVerticalAlignmentTop: Üste

VtVerticalAlignmentBottom: Alta

VtVerticalAlignmentCenter: Ortaya

Methods

EditCopy

Bu metod kullanılarak grafik panoya kopyalanabilir.

```
Private Sub Command1_Click()
 MSChart1.EditCopy
End Sub
```

EditPaste

Bu metodla panodaki bilgi grafiğe alınabilir.

```
Private Sub Command2_Click()
 MSChart1.EditPaste
End Sub
```

Events

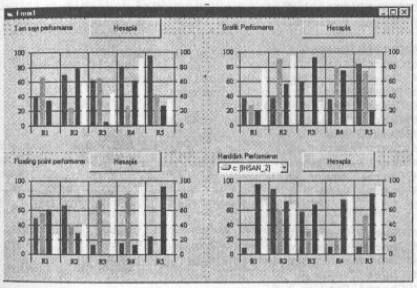
PointActivated(Series As Integer, DataPoint As Integer, MouseFlags As Integer, Cancel As Integer)

Grafikteki bir nokta çift tıklandığında bu olay meydana gelir. **Series** parametresi ile hangi grafiğin (birden fazla grafik bulunabiliyordu), **DataPoint** parametresi ile de hangi noktanın çift tıklandığı öğrenilebilir.

ÖRNEK: Örneğin kullanıcının bir noktayı çift tıkladığında o noktayı değiştirmesine imkan verelim:


```
Private Sub MSChart1_PointActivated(Series As Integer, DataPoint
As Integer, MouseFlags As Integer, Cancel As Integer)
 With MSChart1
  .Column = Series
  .Row = DataPoint
  .Data = InputBox("Yeni değer:", , .Data)
 End With
End Sub
```

ÖRNEK: Bilgisayarın performansını ölçerek sonuçları grafik halinde gösterecek bir program yapalım. Örneğimiz için aşağıdaki formu hazırlayın.



Yapacağımız programla bazı işlemler yaptırıp bilgisayarın bu işlemleri ne kadar sürede yapabildiğini bulacağız ve bu sonuçları grafikte göstereceğiz. Ayrıca karşılaştırmalı olabilsin diye bu P-II 233 ve 686-120 işlemcilerinin bu programla ölçülmüş performanslarını da grafikte göstereceğiz.

```
'ÖRNEK: Chart
Private Sub Form_Load()
 Caption = "Benchmark 1.0 İhsan Karagülle"
 MSChart1.RowCount = 3
 MSChart1.ColumnCount = 1
 MSChart1.chartType = VtChChartType2dBar
 MSChart1.Row = 1
 MSChart1.Data = 228
 MSChart1.RowLabel = "P II 233"
 MSChart1.Row = 2
 MSChart1.Data = 118
 MSChart1.RowLabel = "686-120"
 MSChart1.Row = 3
```

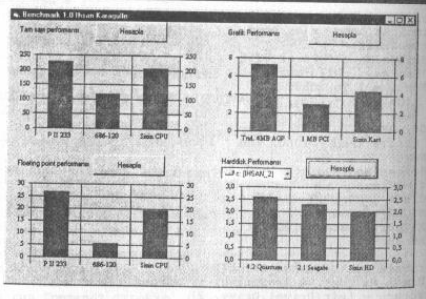


```
 MSChart1.Data = 0
 MSChart1.RowLabel = "Sizin CPU"

 MSChart2.RowCount = 3
 MSChart2.ColumnCount = 1
 MSChart2.chartType = VtChChartType2dBar
 MSChart2.Row = 1
 MSChart2.Data = 27
 MSChart2.RowLabel = "P II 233"
 MSChart2.Row = 2
 MSChart2.Data = 5.85
 MSChart2.RowLabel = "686-120"
 MSChart2.Row = 3
 MSChart2.Data = 0
 MSChart2.RowLabel = "Sizin CPU"

 MSChart3.RowCount = 3
 MSChart3.ColumnCount = 1
 MSChart3.chartType = VtChChartType2dBar
 MSChart3.Row = 1
 MSChart3.Data = 7.33
 MSChart3.RowLabel = "Trid. 4MB AGP"
 MSChart3.Row = 2
 MSChart3.Data = 3
 MSChart3.RowLabel = "1 MB PCI"
 MSChart3.Row = 3
 MSChart3.Data = 0
 MSChart3.RowLabel = "Sizin Kart"

 MSChart4.RowCount = 3
 MSChart4.ColumnCount = 1
 MSChart4.chartType = VtChChartType2dBar
 MSChart4.Row = 1
 MSChart4.Data = 2.6
 MSChart4.RowLabel = "4.2 Quantum"
 MSChart4.Row = 2
 MSChart4.Data = 2.3
 MSChart4.RowLabel = "2.1 Seagate"
 MSChart4.Row = 3
 MSChart4.Data = 0
 MSChart4.RowLabel = "Sizin HD"
End Sub

Private Sub Command1_Click()
 Dim i As Long, x As Long, t
 t = Timer 'işlemin başlangıcındaki zamanı öğren
 Screen.MousePointer = 12
 'Tam sayı performansı için bazı işlemler yaptır
 For i = 0 To 10000000
  x = x + i + 10
  x = 2 * i - 100 - x
 Next
 Screen.MousePointer = 0
```



```
 t = Timer - t 'Geçen süreyi bul
 MSChart1.Row = 3 '3. kolona yaz
 MSChart1.Data = 100 / t 'sürenin tersini al (Süre ne kadar azsa
performans o kadar çoktur)
End Sub

Private Sub Command2_Click()
 Dim i As Long, x As Long, t
 t = Timer
 Screen.MousePointer = 12
 'Ondalık sayı performansı için bazı işlemler yaptır
 For i = 1 To 1000000
  x = Log(i) * Sin(i) * Exp(Cos(i))
  x = x / i
  x = x * i
 Next
 Screen.MousePointer = 0
 t = Timer - t
 MSChart2.Row = 3
 MSChart2.Data = 100 / t
End Sub

Private Sub Command3_Click()
Dim i As Long, x As Long, t
 t = Timer
 Screen.MousePointer = 12
 'Tam sayı performansı için bazı çizimler yaptır
 For i = 1 To 20000
  Line (Rnd * 1000, Rnd * 1000)-(Rnd * 1000, Rnd * 1000)
  Line (Rnd * 1000, Rnd * 1000)-(Rnd * 1000, Rnd * 1000), Rnd *
100, BF
  Circle (Rnd * 1000, Rnd * 1000), Rnd * 1000, Rnd * 100
 Next
 Screen.MousePointer = 0
 t = Timer - t
 Cls
 MSChart3.Row = 3
 MSChart3.Data = 100 / t
End Sub

Private Sub Command4_Click()
 Dim i As Long, x As String, t, sürücü
 sürücü = Left(Drivel.Drive, 2)
 t = Timer
 Screen.MousePointer = 12
 x = "tesssssssssssst"
 'Harddisk performansı için dosyalara yazdır ve oku
 Open sürücü + "\test.$$$" For Random As #1
  For i = 1 To 150000
   Put #1, i, x
  Next
 Close #1
 Open sürücü + "\test.$$$" For Random As #1
```

```
  For i = 1 To 150000
   Get #1, i, x
  Next
 Close #1
 Open sürücü + "\test.$$$" For Input As #1
  While Not EOF(1)
   Input #1, x
  Wend
 Close #1
 x = "tesssssssssssst"
 Open sürücü + "\test.$$$" For Output As #1
  For i = 1 To 150000
   Write #1, x
  Next
 Close #1
 Kill sürücü + "\test.$$$"
 Screen.MousePointer = 0
 t = Timer - t
 MSChart4.Row = 3
 MSChart4.Data = 100 / t
End Sub
```



Bu programı EXE dosya haline getirip çalıştırırsanız sonuçlar daha net olacaktır. Çünkü programı VB içinden çalıştırdığınızda optimize edilmeden çalıştırılacak ve sonuçlar daha düşük çıkacaktır. EXE haline getirdiğinizde ise VB, bunun en iyi şekilde çalışması için optimize edecek ve gerçek sonuçlar ortaya çıkacaktır.

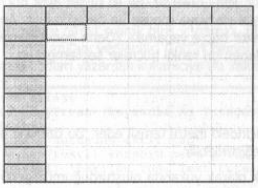
Grid (Izgara Kontrolü)*

Grid kontrolü, hücrelerden meydana gelen Excel'deki sayfalara benzer bir kontroldür.

Properties

Cols,Rows

Bu özellikler; bir hücrede bulunması gereken satır ve sütun sayılarını belirtirler. Bunun minimum değeri 1 dir. Bir ızgarada en fazla 2000 satır ve 400 sütun oluşturulabilir.

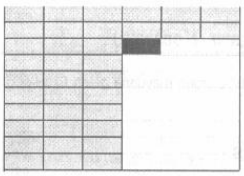


Grid1.Cols=6 ve Grid1.Rows=10 olan bir grid yukarıda görülmektedir.

FixedCols,FixedRows

Tablo üzerinde kullanıcının erişemediği ve sütun ve satırların başlığı olarak kullanan sütun ve satır sayısını belirler. Normalde her ikisini değeri de 1 dir. Bu tablo üzerinde gri renkli gördüğümüz kısımdır. Bu sayıyı artırabilir veya 0 yaparak kaldırabilirsiniz.

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Grid Control" seçeneğini işaretlemeniz veya Browse düğmesi ile GRID32.OCX dosyasını bulup projeye eklemeniz gerekir.



Grid1.FixedRows = 2 ve Grid1.FixedCols= 3 olan grid yukarıda görülmektedir.

Col,Row

Bir ızgarada, üzerinde işlem yapılacak hücre adreslerini belirtir. İlk satır için Row=0, ilk sütün için Col=0'dır. Bu iki değer aktif hücreyi belirler. Kullanıcı aktif hücreyi yön tuşlarıyla veya mouse ile değiştirebilir. Ancak ilk satır ve ilk sütundaki hücreleri aktif hücre yapamaz. (Daha genel olarak FixedRows ve FixedCols özelliği ile belirlenen gri renkli hücreler seçilemez)

Text

Aktif hücrenin içindeki metni temsil eder. Bu özellikle aktif hücrenin içeriği öğrenilebilir veya değiştirilebilir.

ÖRNEK: Örnek olarak yukarıda oluşturduğumuz grid'i haftalık ders programı için uygun hale getirelim. Bunun için Formun Load olayına aşağıdaki kodu yazalım.

```
'ÖRNEK :Row,Col,Text
Private Sub Form_Load ()
 Dim i
 grid1.cols=6 '6 sütun
 grid1.rows=10 '10 satırdan oluşan grid

 grid1.Row = 0 'ilk satırın
 grid1.Col = 1 '1. kolonuna
 grid1.Text = "Pt" 'Pt yaz
 grid1.Col = 2
 grid1.Text = "Sl"
 grid1.Col = 3
 grid1.Text = "Çr"
 grid1.Col = 4
 grid1.Text = "Pr"
 grid1.Col = 5
 grid1.Text = "Cu"
```



```
 grid1.Col = 0 'İlk sütunun
 For i = 1 To 9
  grid1.Row = i 'satırlarına
  grid1.Text = i & ".saat" 'saati yaz
 Next
End Sub
```

Bu kod aracılığı ile satır başlıklarına saatleri, sütun başlıklarına ise günleri yazdırmış olduk

	Pt	Sl	Çr	Pr	Cu
1.saat					
2.saat					
3.saat					
4.saat					
5.saat					
6.saat					
7.saat					
8.saat					
9.saat					

ÖRNEK: Kullanıcı hücrelere direkt olarak giriş yapamaz. Bir Text kutusu yardımı ile kullanıcıya giriş yapma imkanı verilebilir. Grid'in hemen üzerine bir text kutusu yerleştirip aşağıdaki kodları yazarsak kullanıcıya bu imkanı vermiş oluruz.

```
'ÖRNEK :Gride text girişi
Private  Sub Grid1_KeyPress (keyascii As Integer)
 text1 = grid1.Text
 text1.SetFocus 'Text kutusunu aktif yap
 text1.SelStart = Len(text1)'kursörü sona götür
 SendKeys Chr(keyascii) 'basılan tuşu Text kutusuna gönder
End Sub

Private Sub Grid1_Click ()
 text1 = grid1.Text
End Sub

Private  Sub Text1_Change ()
 grid1.Text = text1
End Sub
```

Kullanıcının grid üzerinde bir tuşa basması durumunda kontrolü text kutusuna bırakıyoruz ve text kutusunun change olayı ile de text içerisine yazılanların gride de yazılmasını sağlıyoruz.

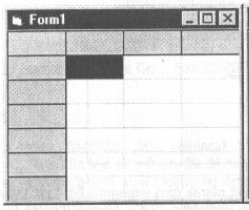
Grid kontrolüne tıklanınca bilgi girişi yapılmasını basitçe:

```
Grid1.text=InputBox("Değer")
```

satırı ile de yapabilirsiniz.

Programı çalıştırırsanız ilk sütuna ve ilk satıra giriş yapamadığınızı göreceksiniz. Bu ilk satır ve sütunu tıklamanız o hücrenin değil o sıradaki bütün hücrelerin seçilmesini sağlar. İlk satırı ve ilk sütunu ancak programda yazacağınız Grid1.Col=0 veya Grid1.Row=0 satırları ile seçebilirsiniz.

ÖRNEK: Örnek olarak bir işlem tablosu yapalım. Kullanıcının girdiği fiyata kar oranın ekledikten sonra satış fiyatını hesaplasın.



Form1

	Alış F	Kar O	Satış F
TV	5000	20	6000
M.Seti	3000	25	3750
Ütü	500	20	600
Masa	1500	17	1755
Takvim	100	35	135
Yazıcı	2000	7	2140

```
'ÖRNEK :Grid
Private Sub Form_Load ()
  grid1.Col = 0
  grid1.Row = 1
  grid1.Text = "TV"
  grid1.Row = 2
  grid1.Text = "M.Seti"
  grid1.Row = 3
  grid1.Text = "Ütü"
  grid1.Row = 4
  grid1.Text = "Masa"
  grid1.Row = 5
  grid1.Text = "Takvim"
  grid1.Row = 6
  grid1.Text = "Yazıcı"
  grid1.Row = 0

  grid1.Col = 1
  grid1.Text = "Alış F"
  grid1.Col = 2
  grid1.Text = "Kar O"
  grid1.Col = 3
  grid1.Text = "Satış F"
End Sub

Private Sub Grid1_Click ()
  Dim x, y
  grid1.Text = Val(InputBox("Değer"))
```



```
  grid1.Col = 1
  x = grid1.Text
  grid1.Col = 2
  y = Val(grid1.Text)
  grid1.Col = 3
  grid1.Text = x + x * y / 100
End Sub
```

CellSelected

Bu özellik; Col ve Row özellikleri ile belirlenen hücrenin seçili olup olmadığını belirtir.

SelStartCol, SelEndCol, SelStartRow, SelEndRow

Seçili bölgenin başlangıç ve bitiş adresleri bu özelliklerle belirlenir. Hücreleri seçmek veya seçili hücreleri öğrenmek için bu özellikler kullanılır. Bu özelliklerin değerleri FixedCols ve FixedRows hücrelerini içeremez.

Clip

Bir ızgara içindeki seçilmiş olan bölgenin içeriğini verir. Ayrıca bir çok hücreye birden atama imkanı verir. Aynı satırdaki hücreler arasında Chr(9) karakteri, bir sonraki sütuna geçiş içinde Chr(13) karakteri bulunur.

ÖRNEK: Örneğin yukarıda ders programı için yaptığımız örneğin ilk iki satırına şu dersleri koymak isteyelim.

"Visual Basic", "Matematik", "Nümerik", "Donanım", "DBase"
"Visual Basic", "Türk Dili ", "Nümerik", "Donanım", "DBase"

Bunu text özelliği ile yapmak istersek her hücre için ayrı ayrı atamamız gerekir.

```
Grid1.Row=1
Grid1.Col = 1
Grid1.Text ="Visual Basic"
Grid1.Col = 2
Grid1.Text = "Matematik"
Grid1.Col = 3
Grid1.Text = "Nümerik"
Grid1.Col = 4
Grid1.Text = "Donanım"
Grid1.Col = 5
Grid1.Text ="DBase"
Grid1.Row=2
Grid1.Col = 1
Grid1.Text ="Visual Basic"
```



```
Grid1.Col = 2
Grid1.Text ="Türk Dili "
Grid1.Col = 3
Grid1.Text = "Nümerik"
Grid1.Col = 4
Grid1.Text = "Donanım"
Grid1.Col = 5
Grid1.Text ="DBase"
```

Aynı işlemi Clip özelliğiyle yapmak ise daha az bir kodla gerçekleşecektir.

```
grid1.SelStartCol = 1
grid1.SelStartRow = 1
grid1.SelEndCol = 5
grid1.SelEndRow = 2
grid1.Clip = "Visual Basic" + Chr(9) + "Matematik" + Chr(9) +
"Nümerik" + Chr(9) + "Donanım" + Chr(9) + "DBase" + Chr(13) +
"Visual Basic" + Chr(9) + "Türk Dili " + Chr(9) + "Nümerik" +
Chr(9) + "Donanım" + Chr(9) + "DBase"
```

	Pt	Sl	Çr	Pr	Cu
1.saat	Visual	Matan	Nüme	Donan	DBase
2.saat	Visual	Türk	Nüme	Donan	DBase
3.saat					
4.saat					
5.saat					
6.saat					
7.saat					
8.saat					
9.saat					

ÖRNEK: Clip özelliğinin asıl önemi grid içindeki bilgilerin bir dosyaya yazılması ve okunması sırasında görülür. Dosyaya yazarken her bir hücreyi tek tek yazmak yerine Gridi seçtikten sonra Grid özelliği gridin bütün içeriğini alabilir aynı şekilde dosyadan okurken de tek bir işlemle bu işlemi gerçekleştirebiliriz.

Dosyaya yazmak için aşağıdaki kodu Form Load olayına yazalım

```
Dim s
Open "ders.dat" For Random As #1
 grid1.SelStartCol = 1
 grid1.SelStartRow = 1
 grid1.SelEndCol = grid1.cols-1
 grid1.SelEndRow = grid1.rows-1
 s=grid1.Clip
 Put #1,1,s
Close #1
```

Dosyadan okumak içinde aşağıdaki kodu Formun Unload olayına yazalım.


```
Dim s
Open "ders.dat" For Random As #1
 grid1.SelStartCol =1
 grid1.SelStartRow = 1
 grid1.SelEndCol = grid1.cols-1
 grid1.SelEndRow = grid1.rows-1
 Get #1,1,s
 grid1.Clip=s
Close #1
```

Graph kontrolüne QuickData özelliği ile araya Tab koyarak toplu değer atanabiliyordu. Bir Graph ile Gridin bağlantısını kurarken de

Graph1.QuickData = Grid1.Clip

satırını kullanarak değer aktarmasını hızlıca yapabilirsiniz.

ColAlignment(Kolon)

Numarası verilen kolondaki hücrelerin içeriğini sola, sağa ve ortaya yerleştirir. 0-2 arası bir değer alabilir.

0: Sola yanaşık

1: Sağa yanaşık

2: Ortada

Yukarıdaki ders programı örneğimize bu özelliği ekleyelim. Örneğimizdeki formun üzerine bir Option yerleştirin ve iki kopyasını da hemen yanına yerleştirin (yani bir dizi olmalarını sağlayın). Sırasıyla Caption özelliklerine de "Sola", "Sağa" ve "Ortaya" yazılarını yerleştirip aşağıdaki kodu Click olayına yazalım.

```
Sub Option1_Click (index As Integer)
  grid1.ColAlignment(grid1.Col) = index
End Sub
```

FixedAlignment (Kolon)

ColAlignment özelliği gibidir ancak yalnız Fixed olarak adlandırılan yani gri renkli kolonların alignmentini ayarlar.

ColWidth(Kolon)

Numarası verilen kolonun genişliğini ayarlamaya yarar. Atanan değerin birimi Twip cinsinden olmalıdır.

RowHeight(Satır)

Numarası verilen satırın yüksekliğini ayarlamaya yarar. Atanan değerin birimi Twip cinsinden olmalıdır.

Genişliği ve yüksekliği kullanıcı mouse ile de değiştirebilir.

FillStyle

0: Yapılan değişiklik sadece aktif hücreyi etkiler.

1: Yapılan değişiklik seçili olan bütün hücreleri etkiler.

GridLines

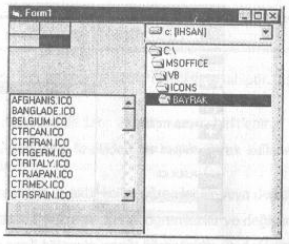
False verilerek hücreler arasındaki ayırma çizgilerinin görüntülenmemesi sağlanabilir.

Picture

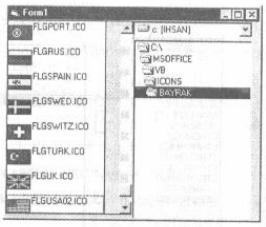
Bu özellik diğer nesnelerin Picture özelliği gibidir. Grid nesnesinin her bir hücresine bu özellikle resim eklenebilir.

Yukarında görüldüğü gibi her bir hücreye hem resim hem de yazı eklenebilmektedir.

ÖRNEK: Örnek olarak icon gösteren bir program yapalım. Bunun için aşağıdaki formu oluşturalım.




```
'ÖRNEK :Grid ile Icon gösterici
Private  Sub Form_Load ()
 grid1.FixedCols = 0
 grid1.FixedRows = 0
 grid1.Rows = 1
 grid1.Cols = 1
 file1.Pattern = "*.ico"
 file1.Visible = False
 grid1.ColWidth(0)=32*screen.TwipsPerPixelX + TextWidth("X") * 12
 grid1.RowHeight(0) = 32 * screen.TwipsPerPixelY
 dir1_change
End Sub

Private Sub Drive1_Change ()
 ChDrive drive1.Drive
 dir1.Path = drive1.Drive
End Sub

'Dizin değiştiğinde o dizindeki ikonları göster
Private Sub dir1_change ()
 On Local Error Resume Next
 ChDir dir1.Path
 file1.Path = dir1.Path

 Dim i
 grid1.Rows = file1.ListCount - 1 'satır sayısı=resim sayısı
 screen.MousePointer = 11
 For i = 1 To file1.ListCount
  grid1.Row = i - 1
  grid1.RowHeight(i - 1) = 32 * screen.TwipsPerPixelY
  grid1.Picture = LoadPicture(file1.List(i - 1))
  grid1.Text = file1.List(i - 1)
 Next
 screen.MousePointer = 0
End Sub
```

ScrollBars

0: Yok
1: Yatay
2: Dikey
3: Yatay ve Dikey

ScrollBars özelliği ile grid hücrelerinin nesne içerisine sığmadığı durumlarda kaydırma çubuklarının eklenmesi sağlanabilir.

LeftCol,TopRow

O anda ekranda görülen satır ve sütun numarası bu özelliklerle öğrenilebilir. Grid hücreleri kontrolün sınırlarını aşıyorsa ekranda görülen hücreler bu özelliklerle belirlenir.

Events

RowColChange( )

Aktif olan hücre kontrolünün başka bir hücreye geçmesi halinde yani o hücrenin aktif olması halinde **RowColChange** olayı meydana gelmektedir.

SelChange( )

Seçili alanın değişmesi halinde bu olay meydana gelir.

MsFlexGrid (Izgara Kontrolü)*

Grid kontrolü gibidir ancak bazı gelişmiş özelliklere sahiptir. Aşağıdaki özellikleri Grid'de olduğu gibi kullanılır.

1. Cols ve Rows özellikleri ile satır ve sütun sayısı belirlenir.
2. FixedCols ve FixedRows özellikleri ile başlık olarak kullanılacak satır ve sütunlar belirlenir.
3. Col ve Row özellikleri ile aktif kolon öğrenilebilir veya değiştirilebilir.
4. Text özelliği ile aktif hücrenin içeriği öğrenilebilir ve değiştirilebilir.
5. Clip özelliği ile seçili bölgenin içeriği öğrenilebilir ve değiştirilebilir.
6. ColWidth ve RowHeight özellikleri ile sütun genişliği ve satır yüksekliği belirlenebilir.
7. ColAlignment özelliği ile kolonların alignmentleri değiştirilebilir.

Bu kontrolün Grid kontrolünden farklı olan özelliklerini aşağıda göreceğiz.

Properties

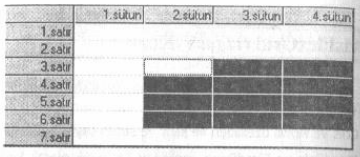
RowSel, ColSel

Grid içerisinde kullanıcı birden fazla hücre seçebilir. Seçili alanı öğrenmek veya değiştirmek için bu özellikler kullanılır.

Seçili alandaki sütunlar **Col** ve **ColSel** özellikleri ile, seçili alandaki satırlar ise **Row** ve **RowSel** özellikleri ile belirlenir.

Bir gridde aşağıdaki gibi bir alan seçili olsun:

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft FlexGrid Control" seçeneğini işaretlemeniz veya Browse düğmesi ile MSFLXGRD.OCX dosyasını bulup projeye eklemeniz gerekir.



Burada Col, Colsel, Row ve RowSel özelliklerinin alacağı değerler şöyle olacaktır:

Col: 2
ColSel: 4
Row: 3
RowSel: 6

Bu özelliklere bakara 2 ile 4. sütunlar arası ve 3 ile 6. satırlar arasında bulunan hücrelerin seçildiği anlaşılmaktadır.

Clip

Bir ızgara içindeki seçilmiş olan bölgenin içeriğini verir. Ayrıca bir çok hücreye birden atama imkanı verir. Aynı satırdaki hücreler arasında Chr(9) karakteri, bir sonraki sütuna geçiş içinde Chr(13) karakteri bulunur.

Bu özellik Grid kontrolündeki gibidir. Tek fark seçim için kullanılacak komutlardır. Gridde seçili alanın SelStartCol, SelEndCol, SelStartRow, SelEndRow özellikleri belirliyordu. MSFlexGrid kontrolünde ise seçili alanı Col, ColSel, Row, RowSel özellikleri belirler.

ÖRNEK: Örneğin aşağıdaki ders programı için kullanılan grid'in ilk iki satırına şu dersleri koymak isteyelim.

"Visual Basic", "Matematik", "Nümerik", "Donanım", "DBase"
"Visual Basic", "Türk Dili ", "Nümerik", "Donanım", "DBase"

	Pazartesi	Salı	Çarşamba	Perşembe	Cuma
1.ders					
2.ders					
3.ders					
4.ders					
5.ders					
6.ders					
7.ders					
8.ders					
9.ders					

Bunu text özelliği ile yapmak istersek her hücre için ayrı ayrı atamamız gerekir.

```
MsFlexGrid1.Row=1
MsFlexGrid1.Col = 1
MsFlexGrid1.Text ="Visual Basic"
MsFlexGrid1.Col = 2
MsFlexGrid1.Text = "Matematik"
MsFlexGrid1.Col = 3
MsFlexGrid1.Text = "Nümerik"
MsFlexGrid1.Col = 4
MsFlexGrid1.Text = "Donanım"
MsFlexGrid1.Col = 5
MsFlexGrid1.Text ="DBase"
MsFlexGrid1.Row=2
MsFlexGrid1.Col = 1
MsFlexGrid1.Text ="Visual Basic"
MsFlexGrid1.Col = 2
MsFlexGrid1.Text ="Türk Dili "
MsFlexGrid1.Col = 3
MsFlexGrid1.Text = "Nümerik"
MsFlexGrid1.Col = 4
MsFlexGrid1.Text = "Donanım"
MsFlexGrid1.Col = 5
MsFlexGrid1.Text ="DBase"
```

Aynı işlemi Clip özelliğiyle yapmak ise daha az bir kodla gerçekleşecektir.

```
MSFlexGrid1.Col = 1
MSFlexGrid1.Row = 1
MSFlexGrid1.ColSel = 5
MSFlexGrid1.RowSel = 2
MSFlexGrid1.Clip = "Visual Basic" + Chr(9) + "Matematik" + Chr(9)
+ "Nümerik" + Chr(9) + "Donanım" + Chr(9) + "DBase" + Chr(13) +
"Visual Basic" + Chr(9) + "Türk Dili " + Chr(9) + "Nümerik" +
Chr(9) + "Donanım" + Chr(9) + "DBase"
```

	Pazartesi	Salı	Çarşamba	Perşembe	Cuma
1.ders	Visual Basic	Matematik	Nümerik	Donanım	DBase
2.ders	Visual Basic	Türk Dili	Nümerik	Donanım	DBase
3.ders					
4.ders					
5.ders					
6.ders					
7.ders					
8.ders					
9.ders					

ÖRNEK: Clip özelliğinin asıl önemi grid içindeki bilgilerin bir dosyaya yazılması ve okunması sırasında görülür. Dosyaya yazarken her bir hücreyi tek tek yazmak yerine Gridi seçtikten sonra Clip özelliği ile gridin bütün içeriğini alabilir, aynı şekilde dosyadan okurken de tek bir işlemle bu işlemi gerçekleştirebiliriz.

Dosyadan okumak için aşağıdaki kodu Form Load olayına yazalım

```
Private Sub Form_Load()
 Dim i,a
 If Dir("ders.dat") <> "" Then 'dosya varsa
  Open "ders.dat" For Input As #1
  MSFlexGrid1.Col = 1
  MSFlexGrid1.Row = 1
  MSFlexGrid1.ColSel = MSFlexGrid1.Cols - 1
  MSFlexGrid1.RowSel = MSFlexGrid1.Rows - 1
  Input #1, s
  MSFlexGrid1.Clip = s
  Close #1
 End If
End Sub
```

Dosyaya yazmak için de aşağıdaki kodu Formun Unload olayına yazalım.

```
Private Sub Form_Unload(Cancel As Integer)
 Dim s
 Open "ders.dat" For Output As #1
 MSFlexGrid1.Col = 1
 MSFlexGrid1.Row = 1
 MSFlexGrid1.ColSel = MSFlexGrid1.Cols - 1
 MSFlexGrid1.RowSel = MSFlexGrid1.Rows - 1
 s = MSFlexGrid1.Clip
 Write #1, s
 Close #1
End Sub
```

Böylece Grid içindeki bilgileri kolayca dosyaya kaydedebilir ve programa girerken de o dosyadaki bilgileri gride yükleyebiliriz.

TextMatrix(Satır, Sütun)

Grid içindeki bilgileri öğrenmenin veya değiştirmenin bir yolu da bu özelliği kullanmaktır. Özellikle döngü türü işlemlerde bu özellik çok kullanışlıdır. **Satır** ve **Sütun** parametreleri ile belirlenen hücredeki bilgi bu özellikle öğrenilebilir veya değiştirilebilir.

```
MsFlexGrid1.TextMatrix(1,1) ="Visual Basic"
MsFlexGrid1. TextMatrix(1,2) = "Matematik"
MsFlexGrid1. TextMatrix(1,3)= "Nümerik"
MsFlexGrid1. TextMatrix(1,4)= "Donanım"
MsFlexGrid1. TextMatrix(1,5)="DBase"
MsFlexGrid1. TextMatrix(2,1) ="Visual Basic"
MsFlexGrid1. TextMatrix(2,2)="Türk Dili "
MsFlexGrid1. TextMatrix(2,3)= "Nümerik"
MsFlexGrid1. TextMatrix(2,4)= "Donanım"
MsFlexGrid1. TextMatrix(2,5)="DBase"
```


	Pazartesi	Salı	Çarşamba	Perşembe	Cuma
1.ders	Visual Basic	Matematik	Nümerik	Donanım	DBase
2.ders	Visual Basic	Türk Dili	Nümerik	Donanım	DBase
3.ders					
4.ders					
5.ders					
6.ders					
7.ders					
8.ders					
9.ders					

Sort

Bu özelliğe aşağıdaki değerlerden biri verilerek Grid içindeki sütunlardan birine göre sıralanabilir. Eğer seçili bir alan varsa sıralama işlemi sadece o alanda uygulanır. Seçili alan yoksa aktif sütuna göre sıralama yapılır.

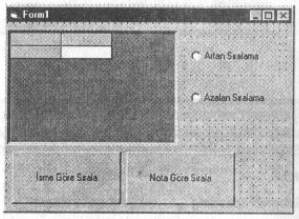
flexSortNone	0	Sıralama yok
flexSortGenericAscending	1	Artan sıralama
flexSortGenericDescending	2	Azalan Sıralama
flexSortNumericAscending	3	Artan Sayısal Sıralama.
flexSortNumericDescending	4	Azalan Sayısal Sıralama
flexSortStringNoCaseAsending	5	Büyük-Küçük harf ayrımı yapmayan, Artan Sıralama
flexSortNoCaseDescending	6	Büyük-Küçük harf ayrımı yapmayan, Azalan Sıralama
flexSortStringAscending	7	Büyük-Küçük harf ayrımı yapan, Artan Sıralama
flexSortStringDescending	8	Büyük-Küçük harf ayrımı yapan, Azalan Sıralama
flexSortCustom	9	Özel Sıralama. Sıralamanın nasıl olacağı **Compare** olayına yazılacak kodla yapılır.

Görüldüğü gibi MSFlexGrid kontrolünün oldukça kullanışlı bir sıralama mantığı vardır. Sayı sıralama ile alfabetik sıralama mantığı farklıdır. Alfabetik olarak sıraladığınızda 10 sayısı 2 sayısından küçük görülecektir. Bu yüzden sıralama işlemi hem alfabetik hem de nümerik olarak desteklenmektedir. Ayrıca alfabetik veya nümerik olmayan sıralamalar da desteklenir. Kendi özel veri tipleriniz veya özel şartlarınız varsa buna göre de sıralama yaptırabilirsiniz. Sort özelliğine 9 de-

ğeri verilirse, sıralama işlemi sırasında **Compare** olayı meydana gelecek ve büyük mü, küçük mü olduğu bu olaya yazacağınız koda bakılarak karar verilecektir.

ÖRNEK: Örnek olarak öğrenci isimlerinin ve notlarının bulunduğu bir Grid oluşturalım. Kullanıcı isterse isme göre, isterse nota göre sıralama yapabilsin.

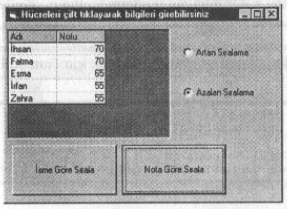
Örneğimiz için aşağıdaki formu oluşturun:




```
'ÖRNEK: Sort
Private Sub Form_Load()
 MSFlexGrid1.Cols = 2
 MSFlexGrid1.Rows = InputBox("Öğrenci Sayısı") + 1
 MSFlexGrid1.FixedCols = 0
 MSFlexGrid1.Row = 0
 MSFlexGrid1.Col = 0
 MSFlexGrid1.Text = "Adı"
 MSFlexGrid1.Col = 1
 MSFlexGrid1.Text = "Notu"
 Caption = "Hücreleri çift tıklayarak bilgileri girebilirsiniz"
End Sub
Private Sub Command1_Click()
 MSFlexGrid1.Col = 0 'isim sütunu
 If Option1 Then
  MSFlexGrid1.Sort = 5
 Else
  MSFlexGrid1.Sort = 6
 End If
End Sub
Private Sub Command2_Click()
 MSFlexGrid1.Col = 1 'not sütunu
 If Option1 Then
  MSFlexGrid1.Sort = 3
 Else
  MSFlexGrid1.Sort = 4
 End If
End Sub
```



```
Private Sub MSFlexGrid1_DblClick()
 'Çift tıklayarak değer girebilsin
 If MSFlexGrid1.Col = 0 Then
  MSFlexGrid1.Text = InputBox("Adı:" & MSFlexGrid1.Text)
 End If
 If MSFlexGrid1.Col = 1 Then
  MSFlexGrid1.Text = Val(InputBox("Notu:" & MSFlexGrid1.Text))
 End If
End Sub
```



MergeCells

İstenirse gridde, yan yana veya üst üste aynı bilgileri içeren hücrelerin birleştirilerek tek bir hücre halinde gösterilmesi sağlanabilir. Bu özelliğin değeri normalde 0'dır ve herhangi bir birleştirme işlemi uygulanmaz. Birleştirme isteniyorsa bu özelliğe 1 değeri verilebilir.

Bu özelliğe 1 değeri verildikten sonra hangi satır veya sütunlarda birleştirme uygulanacağı **MergeRow** veya **MergeCol** özellikleri ile belirlenir.

MergeRow(Satır), MergeCol(Sütun)

MergeCells özelliğine 0 haricinde bir değer verilerek birleştirme istenmişse hangi satır veya sütunlarda birleştirme işlemi yapılacağı bu özellikle belirlenir. Birleştirilmek istenen satır veya sütunun bu özelliklerine **true** değeri verilir.

örneği sadece 1. ve 2. satırdaki bilgiler aynı iken birleştirilmek isteniyorsa:

```
MSFlexGrid1.MergeRow(1)=True
MSFlexGrid1.MergeRow(2)=True
```

ÖRNEK: Örnek olarak ders programını girmek için kullandığımız gridimizi düşü-

nelim. Bir çok ders üst üste birden fazla saatte yer alır. Bunları ayrı ayrı hücrelerde göstermek yerine birleştirmek isteyelim.

	Pazartesi	Salı	Çarşamba	Perşembe	Cuma
1.ders	Visual Basic	Matematik	Nümerik	Donanım	Access
2.ders	Visual Basic	Matematik	Nümerik	Donanım	Access
3.ders	Delphi	Matematik	C++ Builder	İnternet	İnternet
4.ders	Delphi	C++ Builder	C++ Builder	İnternet	İnternet
5.ders	Delphi	C++ Builder	Access	C++ Builder	Excel
6.ders	Visual Basic	Delphi	Access	C++ Builder	Excel
7.ders	Visual Basic	Delphi	Access	C++ Builder	Word
8.ders					
9.ders					

Sütunlarda birleştirme işlemi yaptırmak için Formun Load olayına şu kodu ekleyebiliriz:

```
'ÖRNEK: MergeCells, MergeCol
Private Sub Form_Load()
 Dim i
 MSFlexGrid1.MergeCells = 1
 For i = 1 To MSFlexGrid1.Cols - 1
  MSFlexGrid1.MergeCol(i) = True
 Next
End Sub
```

Aşağıda gördüğünüz gibi aynı dersler alt alta geldiğinde bu hücreler birleştirilmekte ve tek bir hücre içinde gösterilmektedir.

	Pazartesi	Salı	Çarşamba	Perşembe	Cuma
1.ders	Visual Basic	Matematik	Nümerik	Donanım	Access
2.ders					
3.ders	Delphi	C++ Builder	C++ Builder	İnternet	İnternet
4.ders					
5.ders			Access	C++ Builder	Excel
6.ders	Visual Basic	Delphi			
7.ders					Word
8.ders					
9.ders					

ColPosition(Sütun), RowPosition (Satır)

Bu özellik kullanılarak numarası verilen sütunun veya satırın yeri başka bir sütuna veya satıra taşınabilir.

MSFlexGrid1.ColPosition(1) = 4 satırı ile 1 nolu sütun 4 nolu sütuna alınacaktır.

Aşağıdaki gibi bir gridimiz olsun:

	1.sütun	2.sütun	3.sütun	4.sütun
1.satır				
2.satır				
3.satır				
4.satır				
5.satır				
6.satır				
7.satır				

Bu grid için aşağıdaki kodu çalıştırdığımızda 1. sütun 4. sütun yerine gelecektir.

```
MSFlexGrid1.ColPosition(1) = 4
```

	2.sütun	3.sütun	4.sütun	1.sütun
1.satır				
2.satır				
3.satır				
4.satır				
5.satır				
6.satır				
7.satır				

Bu işlemin kullanıcı tarafından fare ile yapılabilmesini isterseniz Drag Drop işlemlerini kullanabilirsiniz. Kullanıcının seçtiği kolonu öğrenmek için **MouseCol,** satırı öğrenmek için de **MouseRow** özelliği kullanılabilir.

Aşağıdaki kodu MouseDown olayına yazarsanız tıklanan kolon 4. kolona alınır.

```
MSFlexGrid1.ColPosition(MSFlexGrid1.MouseCol) = 4
```

Picture

FlexGrid kontrolünün Picture özelliği ile FlexGrid'in görüntüsü resim olarak alınabilir. Grid'i aynen yazıcıdan çıkarmak için veya başka bir picture içinde göstermek için bu özellik kullanılabilir. Bu özellik Grid'in sadece ekranda görünen kısmının görüntüsünü değil tamamının görüntüsünü verir.

GridLines, GridLinesFixed

Bu özellik aracılığı ile hücreleri ayıran çizgilerin şekli belirlenebilir. Aşağıdaki değerlerden birini alabilir.

flexGridNone	0	Çizgi yok
flexGridFlat	1	Düz çizgi
flexGridInset	2	Üç boyutlu görünüme sahip, basık çizgi
flexGridRaised	3	Üç boyutlu görünüme sahip, kabarık çizgi

GridLinesFixed özelliği fixed kısmın çizgi stilini belirlerken, **GridLines** özelliği diğer kısımların çizgi stilini belirler.

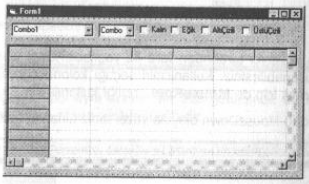
GridLineWidth

Bu özellikle gridde bulunan çizgilerin kalınlığı belirlenebilir. Bu özellik, sadece **GridLines** özelliğine 1 değeri verildiğinde etkili olur.

CellFont...

Aktif hücrenin font özellikleri **CellFontName, CellFontSize, CellFontBold, CellFontItalic, CellFontUnderline, CellFontStrikeThrough** özellikleri ile belirlenebilir.

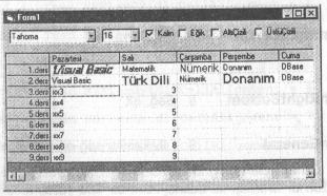
ÖRNEK: Örnek olarak aktif hücrenin font özelliklerini kullanıcının belirleyebileceği bir program yapalım. Örneğimiz için aşağıdaki formu hazırlayın.




```
Private Sub Form_Load()
 Dim i
 'Combo1'e font isimlerini ekle
 For i = 0 To Screen.FontCount - 1
  Combo1.AddItem Screen.Fonts(i)
 Next
 'Combo2'ye font boyutlarını ekle
 For i = 8 To 100 Step 4
  Combo2.AddItem i
 Next
End Sub

Private Sub Combo1_Click()
 'Seçilen fontu aktif hücreye uygula
 MSFlexGrid1.CellFontName = Combo1.Text
End Sub
```



```
Private Sub Combo2_Click()
 'Seçilen font boyutunu aktif hücreye uygula
 MSFlexGrid1.CellFontSize = Val(Combo2.Text)
End Sub

Private Sub Check1_Click()
 MSFlexGrid1.CellFontBold = Check1
End Sub

Private Sub Check2_Click()
 MSFlexGrid1.CellFontItalic = Check2
End Sub

Private Sub Check3_Click()
 MSFlexGrid1.CellFontUnderline = Check3
End Sub

Private Sub Check4_Click()
 MSFlexGrid1.CellFontStrikeThrough = Check4
End Sub

Private Sub MSFlexGrid1_Click()
 'Aktif hücrenin font özelliklerini kutulara yansıt
 Combo1.Text = MSFlexGrid1.CellFontName
 Combo2.Text = MSFlexGrid1.CellFontSize
 Check1.Value = Abs(MSFlexGrid1.CellFontBold)
 Check2.Value = Abs(MSFlexGrid1.CellFontItalic)
 Check3.Value = Abs(MSFlexGrid1.CellFontUnderline)
 Check4.Value = Abs(MSFlexGrid1.CellFontStrikeThrough)
End Sub
```

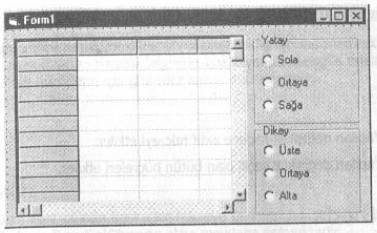


CellAlignment(Hücre)

Aktif hücrenin yatay ve dikey yerleşimi (sola, sağa, üste vb) bu özellikle değiştirilebilir. Alabileceği değerler ve anlamları şöyledir:

flexAlignLeftTop	0	Sol, üst	Visual Basic
flexAlignLeftCenter	1	Sol, orta	Visual Basic
flexAlignLeftBottom	2	Sol, alt	Visual Basic
flexAlignCenterTop	3	Orta, üst	Visual Basic
flexAlignCenterCenter	4	Orta, orta	Visual Basic
flexAlignCenterBottom	5	Orta, alt	Visual Basic
flexAlignRightTop	6	Sağ, üst	Visual Basic
flexAlignRightCenter	7	Sağ, orta	Visual Basic
flexAlignRightBottom	8	Sağ, alt	Visual Basic
flexAlignGeneral	9	Rakamlar sağ-ortaya ve metinler sol-ortaya.	

ÖRNEK: Örnek olarak grid içindeki yazının yatay ve dikey yerleşimini OptionButton'lar aracılığı ile belirleyecek bir kod yazalım. Örneğimiz için formunuza iki tane frame yerleştirin ve birinci Frame içinde bir Option Button yerleştirin. Daha sonra Ctrl+C tuşu ile bu Option Button'u panoya kopyalayın. Tekrar frameyi seçin ve Ctrl+C tuşları ile panodaki Option Button'u yapıştırın ve VB'nin soracağı soruyu Evet diyerek cevaplayın. Aynı yöntemle Frame1 içinde bir tane daha, Frame2 içine de üç tane daha OptionButton yerleştirin. Böylece bütün OptionButton'ların isimleri Option1 ve Index'leri 0,1,2,3,4,5 olacaktır. Bu yöntem bize kod yazmamızda kolaylık sağlayacaktır.




```
'ÖRNEK: CellAlignment
Private Sub Option1_Click(Index As Integer)
  If Option1(0) Then 'sola
    'üst
    If Option1(3) Then MSFlexGrid1.CellAlignment = 0
    'orta
    If Option1(4) Then MSFlexGrid1.CellAlignment = 1
    'alt
    If Option1(5) Then MSFlexGrid1.CellAlignment = 2
  End If
  If Option1(1) Then 'orta
    'üst
    If Option1(3) Then MSFlexGrid1.CellAlignment = 3
    'orta
    If Option1(4) Then MSFlexGrid1.CellAlignment = 4
    'alt
    If Option1(5) Then MSFlexGrid1.CellAlignment = 5
  End If
  If Option1(2) Then  'sağ
    'üst
    If Option1(3) Then MSFlexGrid1.CellAlignment = 6
    'orta
    If Option1(4) Then MSFlexGrid1.CellAlignment = 7
    'alt
    If Option1(5) Then MSFlexGrid1.CellAlignment = 8
  End If
End Sub
```

ColAlignment(Sütun)

CellAlignment özelliği gibidir. Ancak sadece bir hücrenin değil, numarası verilen sütundaki bütün hücrelerin alignmentini tek seferde değiştirmeye yarar.

```
MsFlexGrid1.ColAlignment(1)=0 '1 nolu sütundaki bütün hücreler
içindeki yazılar sol-üste yerleşsin.
```

FixedAlignment (Sütun)

ColAlignment özelliği gibidir ancak yalnız Fixed olarak adlandırılan yani gri renkli kolonların alignmentini ayarlar.

FillStyle

0: Yapılan değişiklik sadece aktif hücreyi etkiler.

1: Yapılan değişiklik seçili olan bütün hücreleri etkiler.

Methods

AddItem

Liste kutularında gördüğümüz bu metod Grid'e yeni bir satır eklemek için kullanılır. Normalde eklenen satır gridin sonunda yer alır. Eğer sona değil de araya eklenmek isterse parametre olarak ekleneceği yerin satır numarası verilir.

```
MSFlexGrid1.AddItem ""
```

Yukarıdaki gibi kullanıldığında gridin sonuna boş bir satır eklenir.

```
MSFlexGrid1.AddItem "",1
```

Yukarıdaki gibi kullanıldığında gridin başına bir satır eklenir ve diğer satırlar aşağı doğru kaydırılır. Fixed satırların üstüne yeni satır eklenememektedir. Bu yüzden eğer Fixed satır sayısı 1'den fazla ise index numarasını ona göre ayarlamanız gerekir. Örneğin fixed satır sayısı 2 ise yukarıdaki komuttaki değer de en az 2 olabilir.

Yukarıdaki her iki örnek satırda da satır eklenmekte fakat satırların içeriği boş bırakılmaktadır. Satırların içeriği de belirlenmek istenirse, satır bilgileri arasına chr(9) konularak ilk parametre olarak verilebilir.

```
MSFlexGrid1.AddItem "1.sütun"+chr(9)+"2.sütun"+chr(9)+"3.sütun",1
```

RemoveItem

Aradan bir satır silinmek istendiğinde bu metod kullanılabilir. Index numarası verilen satırı siler ve diğer satırları yukarıya doğru kaydırır.

```
MSFlexGrid1.RemoveItem 1
```


Clear

Gridin içindeki bütün bilgileri siler ancak satır ve sütunlar yerinde kalır. RemoveItem metodunda satırdaki bilgilerle birlikte satırın kendisi de silinmektedir. Bu metod ise hücrelerin içeriğini siler ancak hücreler kalır.

Events

RowColChange()

Aktif olan hücre kontrolünün başka bir hücreye geçmesi halinde yani o hücrenin aktif olması halinde **RowColChange** olayı meydana gelmektedir.

SelChange()

Seçili alanın değişmesi halinde bu olay meydana gelir.

EnterCell(), LeaveCell()

Bir hücre aktif hale geldiğinde **EnterCell** olayı, aktiviteyi kaybettiğinde ise LeaveCell olayı meydana gelir.

Compare(ByVal Row1 As Long, ByVal Row2 As Long, Cmp As Integer)

Sort özelliği ile sütundaki veriler alfabetik veya nümerik sıraya sokulabiliyordu. Ancak bazı bilgiler vardır ki bu iki sıralama yöntemine de uymazlar. Bu tip bilgileri sıralamak istediğimizde **Sort** özelliğine 9 değerini verdikten sonra karşılaştırma için gerekli kodu bu olaya yazmamız gerekir.

MSFlexGrid'in **Sort** özelliğine 9 değerini verdiğinizde her satır karşılaştırırken bu olay meydana gelir ve hangi satırların karşılaştırıldığı **Row1** ve **Row2** parametreleri ile programınıza bildirilir. Yazacağınız kodla bu iki satırın hangisinin büyük, eşit veya küçük olduğunu belirledikten sonra **Cmp** parametresi ile sonucu bildirmeniz gerekir. Böylece Grid özel durumlara göre de sıralanabilecektir.

Cmp parametresine verebileceğiniz değerler şunlardır:

1	Row1 büyük
0	Eşit
-1	Row2 büyük

Karşılaştırmada kullanılacak bilgilerden birinin satır numarası **Row1** diğeri de **Row2** parametresi ile bildirilir. Karşılaştırma aktif kolon üzerinde yapıldığı için bu bilgilerden faydalanılarak karşılaştırılacak bilgiler aşağıdaki gibi öğrenilebilir:

```
x1 = MSFlexGrid1.TextMatrix(Row1, MSFlexGrid1.Col)
x2 = MSFlexGrid1.TextMatrix(Row2, MSFlexGrid1.Col)
```

Karşılaştırma işlemi yapıldıktan sonra sonuç **Cmp** değeri ile bildirilmesi gerekir. En basitinden "Yüzbaşı" ile "Onbaşı" karşılaştırılırken aşağıdaki gibi bir yapı kullanılabilir.

```
x1 = MSFlexGrid1.TextMatrix(Row1, MSFlexGrid1.Col)
x2 = MSFlexGrid1.TextMatrix(Row2, MSFlexGrid1.Col)
if x1="Yüzbaşı" and x2="Yüzbaşı" then cmp=0
if x1="Yüzbaşı" and x2="Onbaşı" then cmp=1
if x1="Onbaşı" and x2="Yüzbaşı" then cmp=-1
if x1="Onbaşı" and x2="Onbaşı" then cmp=0
```

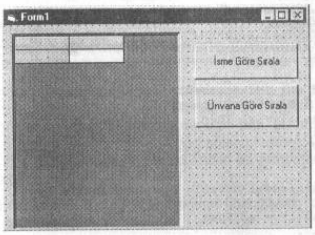
Tabii ki karşılaştırılacak bilgi sayısı artınca yukarıdaki gibi bir mantık çok uzun kod gerektirecektir. Bunun yerine karşılaştırılacak elemanlar sırayla bir diziye atanabilir ve bu dizideki index numaraları karşılaştırılabilir. Aşağıdaki örnekte bu yapı kullanılmıştır.

ÖRNEK: Örnek olarak ünvan ve isim bilgilerinin bulunduğu aşağıdaki gridi düşünelim.

Ünvanı	Adı Soyadı
Öğr.Gör	İhsan Karagülle
Öğr.Gör	Orhan Deligöz
Öğr.Gör	Zeydin Pala
Prof.	Ali Ak
Uzman	Veli Pak
Prof.	Ahmet Öz
Doç.	Murat Sakan
Uzman	Mustafa Bir
Prof.	Muhammet Ak
Yrd.Doç.	Ayşe Koce
Okutman	Fatma Gül
Okutman	Esma Gür

Bu listeyi isme göre sıralamak istediğimizde ikinci kolonu aktif yapıp **Sort** özelliğine **flexSortStringAscending** değerini vermemiz yeterlidir. Ancak ünvana göre sıralamak istediğimizde hangi ünvanın hangisinden önce geldiğini bilgisayarın bilmesini bekleyemeyiz. Bu durumda birinci kolonu aktif yaptıktan sonra **Sort** özelliğine 9 değerini vererek sıralamayı **Compare** olayı aracılığı ile belirleyebiliriz.

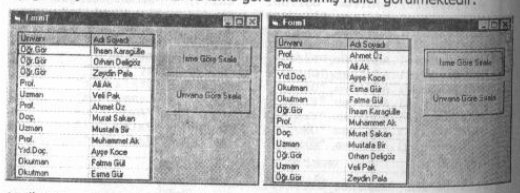
Örneğimiz için aşağıdaki formu hazırlayın:



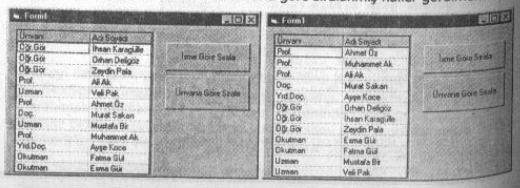

```
'ÖRNEK: Compare
Private Sub Form_Load()
 MSFlexGrid1.Cols = 2
 MSFlexGrid1.FixedCols = 0
 MSFlexGrid1.Rows = 1
 MSFlexGrid1.TextMatrix(0, 0) = "Ünvanı"
 MSFlexGrid1.TextMatrix(0, 1) = "Adı Soyadı"
 MSFlexGrid1.AddItem "Öğr.Gör" + Chr(9) + "İhsan Karagülle"
 MSFlexGrid1.AddItem "Öğr.Gör" + Chr(9) + "Orhan Deligöz"
 MSFlexGrid1.AddItem "Öğr.Gör" + Chr(9) + "Zeydin Pala"
 MSFlexGrid1.AddItem "Prof." + Chr(9) + "Ali Ak"
 MSFlexGrid1.AddItem "Uzman" + Chr(9) + "Veli Pak"
 MSFlexGrid1.AddItem "Prof." + Chr(9) + "Ahmet Öz"
 MSFlexGrid1.AddItem "Doç." + Chr(9) + "Murat Sakan"
 MSFlexGrid1.AddItem "Uzman" + Chr(9) + "Mustafa Bir"
 MSFlexGrid1.AddItem "Prof." + Chr(9) + "Muhammet Ak"
 MSFlexGrid1.AddItem "Yrd.Doç." + Chr(9) + "Ayşe Koce"
 MSFlexGrid1.AddItem "Okutman" + Chr(9) + "Fatma Gül"
 MSFlexGrid1.AddItem "Okutman" + Chr(9) + "Esma Gür"
End Sub

Private Sub Command1_Click()
 MSFlexGrid1.Col = 1 'Adı sütununu aktif sütun yap
 'Bu sütuna göre sırala
 MSFlexGrid1.Sort = flexSortStringAscending
End Sub

Private Sub Command2_Click()
 MSFlexGrid1.Col = 0 'Ünvanı sütununu aktif sütun yap
 'Bu sütuna göre sırala
 MSFlexGrid1.Sort = 9 'flexSortCustom
End Sub

Private Sub MSFlexGrid1_Compare(ByVal Row1 As Long, ByVal Row2 As
Long, Cmp As Integer)
 Dim x1, x2, unvan
```

```
  x1 = MSFlexGrid1.TextMatrix(Row1, MSFlexGrid1.Col)
  x2 = MSFlexGrid1.TextMatrix(Row2, MSFlexGrid1.Col)
  unvan = Array("Prof.", "Doç.", "Yrd.Doç.", "Dr.", "Öğr.Gör",
"Okutman", "Uzman")
  Dim u1, u2, i
  'Birinci satırdaki Unvanın sıra numarasını bul
  For i = 0 To UBound(unvan)
   If unvan(i) = x1 Then
    u1 = i
    Exit For
   End If
  Next
'İkinci satırdaki Unvanın sıra numarasını bul
  For i = 0 To UBound(unvan)
   If unvan(i) = x2 Then
    u2 = i
    Exit For
   End If
  Next
  'Hangisinin büyük olduğunu Cmp parametresi ile bildir.
  If u1 = u2 Then Cmp = 0
  If u1 > u2 Then Cmp = 1
  If u1 < u2 Then Cmp = -1
End Sub
```

Aşağıdaki şekillerde normal ve isme göre sıralanmış haller görülmektedir.



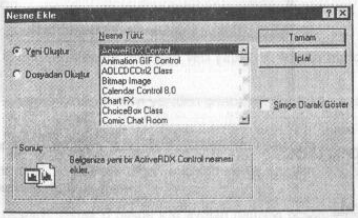
Aşağıdaki şekillerde ise normal ve ünvana göre sıralanmış haller görülmektedir.



OLE (Nesne Bağlama ve Gömme Elemanı)

Bu kontrol elemanı aracılığıyla OLE' yi destekleyen tüm programlar arasında bağlantı kurmak, bu bağlantı sonucunda istenilen veri transferi yapmak mümkündür. Dolayısıyla bir programa eklenmesi gereken bazı bölümleri eklemek yerine bu işi tam anlamıyla yapan programlarla bağlantı kurmak suretiyle kolayca halledilebilir. Bu olay programcıyı bir çok sıkıntıdan ve programın hacmini fazla büyütmekten kurtarır.

OLE nesnesini form üzerine alındığınızda standart OLE diyalog kutusu karşımıza otomatik olarak çıkacaktır:

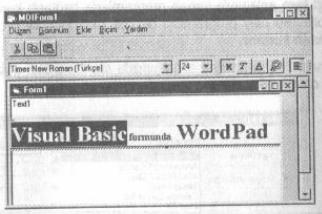


Bu pencerede OLE'yi destekleyen programların listesi yer almaktadır. Bunlardan herhangi birisi seçildiği zaman o programla direkt bağlantı kurularak programın kaynağına ulaşılır. Bu listedeki OLE'yi destekleyen programlardan bir seçilerek o uygulama ile bağlantı kurulabileceği gibi **Dosyadan Yarat** düğmesi kullanılarak ta daha önce kaydedilmiş bir dosya ile bağlantı kurulabilir.

OLE işlemlerinde bir nesne iki türlü kullanılabilir. Bunlar Linked (Bağlantılı) ve Embedded (Gömülü) olarak adlandırılır. Linked olarak kurulan bağlantılarda uygulama ile Nesne arasında paylaşılacak veri alınır. Embedded olarak kurulan ilişkilerde ise Nesnenin paylaşılacak verisi ile birlikte onu açıp düzenleyecek kodda alınır. Linked olarak kurulan bağlantılarda sadece veri alındığı için daha az yer kaplayacaktır ancak bu tip bağlantılarla oluşturulmuş dosyaların başka bilgisayara götürülmesi durumunda orada da çalışacağı garanti edilemez. Çünkü bağlantının kurulduğu program o bilgisayarda bulunmayabilir. Embedded olarak ilişki kurulduğunda ise veri ile birlikte programı düzenleyecek kodda nesneye dahil olduğu için bu tip dosyalar rahatlıkla diğer bilgisayarlarda götürülüp çalıştırılabilir.

Embedded olarak kaydedilen Ole dosyaları Linked olanlara göre daha çok yer kaplayacaktır.

Ayrıca modern Ole uygulamaları yerinde düzenlemeyi desteklerler. Aşağıda bir WordPad nesnesinin VB formuna alınmış hali görülmektedir. Görüldüğü gibi Formun menülerinin ve Araç çubuklarının yerini WordPad menüleri ve araç çubuğu almaktadır.



Properties

AutoActivate

Bu özellik, kullanıcının OLE kontrolünü aktif hale getirmek için kullanacağı metodları belirler. Bunlar arasında OLE kontrolünün çift tıklanması ve focus kontrolünü alması gibi metodlar yer alır.

Bu özelliğin aldığı değerler aşağıdaki gibidir:

0, vbOLEActivateManual

OLE elemanı otomatik olarak aktif hale gelmez. DoVerb metodu kullanılarak program kodlarıyla aktif hale getirilmesi gerekir.

1, vbOLEActivateGetFocus

OLE elemanı klavye kontrolünü aldığında aktif hale gelir.

2, vbOLEActivateDoubleclick

Varsayılan değer olup OLE elemanının çift tıklanması yada klavye kontrolü OLE elemanında iken enter tuşuna basılmak suretiyle OLE elemanı aktif hale gelir. Fakat buradaki çift tıklama olayı, *Double_Click* olayını meydana getirmez.

3, vbOLEActivateAuto

Eğer OLE elemanı bir nesne içeriyorsa, OLE elemanının çift tıklanması yada klavye kontrolünü alması durumunda bu değer onu otomatik olarak aktif hale getirir.

Action

Sadece çalışma zamanında kullanılabilen bu özellik aşağıdaki değerlerden birini alarak ilgili olayı meydana getirir.

0: Gömülü nesneyi oluşturur. **CreateEmbed** metodu ile aynıdır.

1: Dosya içindeki bir nesne ile bağlantı kurar. **CreateLink** metodu ile aynıdır.

4: Bir nesneyi panoya kopyalar. **Copy** metodu ile aynıdır.

5: OLE kontrol elemanına panodan veri yapıştırır. **Paste** metodu ile aynıdır.

6: OLE kontrolü içindeki veriyi yeniden günceller. **Update** metodu ile aynıdır.

7: Bir nesneyi işlem için açar. **DoVerb** metodu ile aynıdır.

9: Bir nesneyi kapatarak diğer uygulamalarla varolan bağlantıyı keser. **Close** metodu ile aynıdır.

10: İlgili nesneyi silerek o nesne için ayrılan hafızayı boşaltır. **Delete** metodu ile aynıdır.

11: Bir nesneyi veri dosyasına kaydeder. **SaveToFile** metodu ile aynıdır.

12: Bir dosyaya kaydedilen dosyayı yükler. **ReadFromFile** metodu ile aynıdır.

14: Nesne yerleştirme diyalog kutusunu görüntüler. **InsertObjDlg** metodu ile aynıdır.

15: Özel yapıştır diyalog kutusunu görüntüler. **PasteSpecialDlg** metodu ile aynıdır.

17: Nesnenin desteklediği işlemleri günceller. **FetchVerbs** metodu ile aynıdır.

18: Bir nesneyi OLE1 formatında kaydeder. **SaveToOle1File** metodu ile aynıdır.

AppIsRunning

Çalışma zamanında kullanılan bu özellik, OLE kontrolü içindeki uygulamanın o anda çalışıp çalışmadığını belirler. Ayrıca bu özelliğe True atanarak ilgili uygulama başlatılabilir ve False atanarak ta kapatılabilir.

UpdateOptions

Bağlantı kurulan verinin değişmesi sonucu OLE elemanı içindeki nesnenin nasıl güncelleneceğini belirler. Bu özelliğin aldığı değerler aşağıdaki gibidir:

0, vbOLEAutomatic

Varsayılan bir değer olup bağlantı kurulan verinin değiştiği her an OLE elemanı içindeki nesne de güncellenir.

1,vbOLEFrozen

Bağlantı kurulan veri kaydedilirken güncellenir.

2, vbOLEManual

Nesne sadece Update metodunun kullanılması sonucu güncellenir.

DisplayType

Ole kontrolünün form üzerinde nasıl gösterileceği bu özellikle belirlenir.

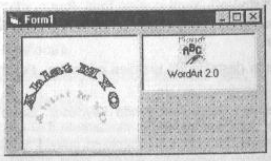
0, vbOLEDisplayContent

Nesnenin içeriği gösterilir.

1,vbOLEDisplayIcon

Nesnenin içeriği yerine ikonu gösterilir.

Yanda WordArt 2.0 nesnesinin içerik ve ikon hali görülmektedir.



HostName

Bir Ole nesnesi kendi uygulamasında düzenlenirken bu nesnenin hangi program tarafından açıldığını belirtmek için HostName özelliği kullanılır. Bu özelliğe vereceğiniz değer çoğunlukla kendi programınızın ismi olmalıdır.

Örneğin bu özelliğe

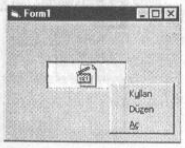
Ole1.HostName = "VB'de bir Ole Uygulaması"

değerini verdiğinizde nesnenizin başlığında bu ifade görülecektir.

AutoVerbMenu

Ole yi destekleyen uygulamaların Verb menüleri bulunur. Bu menüler aracılığı ile kullanıcı bir dosyayı çalabilir, düzenleyebilir veya buna benzer işlemleri yapabilir.

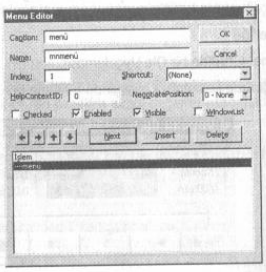
Bu özellikle, OLE kontrolü üzerinde iken sağ fare tuşunun kullanılması halinde Verb menüsünün otomatik olarak açılıp açılmayacağı belirlenir.



ObjectVerbsCount,ObjectVerbs(Index)

Ole uygulamasının desteklediği Verb sayısı ve işlemlerin içeriği bu özelliklerle öğrenilir. Çoğunlukla bir menüde göstermek için kullanılır.

ÖRNEK: Örnek olarak yandaki menü tasarımını formunuz için yapın. Buradaki alt menünün Index özelliğine 1 vererek bir dizi olmasını sağlayın. Böylece nesnenin desteklediği verbleri bu menüye çalışma zamanında yükleyebileceğiz.




```
Dim I
OLE1.InsertObjDlg
OLE1.FetchVerbs
On Local Error Resume Next
For I = 1 To OLE1.ObjectVerbsCount - 1
  Load mnmenü(I)
  mnmenü(I).Caption = OLE1.ObjectVerbs(I)
Next
```

ObjectVerbFlags(Index)

Yukarıdaki özelliklerle o nesnenin desteklediği işlemleri bir menüde gösterebiliyorduk. O özelliğin şu anda kullanılabilir olup olmadığını da bu özellikle öğreniyoruz. Bu özelliğin alacağı değerlere göre menünün özellikleri şöyle olacaktır.

&H0008,vbOLEFlagChecked :	Menü işaretli olacak
&H0002,vbOLEFlagDisabled :	Menü disabled olacak
&H0000,vbOLEFlagEnabled :	Menü enabled olacak
&H0001,vbOLEFlagGrayed :	Menü gizli olacak.
&H0800,vbOLEFlagSeparator :	Menü bir kesme çizgisi olacak.

Yukarıdaki örneğimizi buna göre düzenleyelim. Öncelikle İşlem menüsü açılmadan bu özellikleri kontrol ederek hangi menülerin aktif hangilerinin pasif olacağını belirlememiz gerekir.

```
Private Sub mnİşlem_Click()
    Dim i
    For i = 1 To OLE1.ObjectVerbsCount - 1
        Select Case OLE1.ObjectVerbFlags(i)
            Case &H8: mnmenü(i).Checked = True
            Case &H2: mnmenü(i).Enabled = False
            Case &H0: mnmenü(i).Enabled = True
        End Select
    Next
End Sub
```

ObjectGetFormatsCount, ObjectGetFormats(Index)

Nesnenin desteklediği format sayısı ve bu formatların içeriği bu özelliklerle öğrenilir. Format özelliği ile de hangi formatın kullanılacağı seçilir.

OLEType

Nesnenin içerdiği bilginin tipi bu özellikle öğrenilir.

0, vbOLELinked :	Bağlanmış nesne
1,vbOLEEmbedded :	Gömülmüş nesne
3,vbOLENone:	Ole nesne içermiyor.

OLETypeAllowed

Ole nesnesinin desteklediği bağlantı şekli bu özellikle öğrenilip değiştirilebilir.

0, vbOLELinked:	Bağlantılanmış nesneler olabilir.
1, vbOLEEmbedded :	Gömülmüş nesneler olabilir.
2, vbOLEEither :	Her ikisi de olabilir.

PasteOK

Panodaki bilginin Ole nesnesi tarafından desteklenip desteklenmediği bu özellikle öğrenilir. True değer dönüyorsa panodaki bilgi ole nesnesine yapıştırılabilir.

SizeMode

Ole nesnesinin nasıl boyutlanıdırılacağı bu özellikle belirlenir.

0, vbOLESizeClip

Varsayılan değer budur ve Ole nesnesi orijinal boyutlarında gösterilir. Ole nesnesine sığmıyorsa sığmayan kısım görülmez.

1,vbOLESizeStretch

Nesne Ole kontrolünün içeriğini kaplayacak şekilde boyutlanıdırılır. Nesnenin boyutları Ole kontrolünden büyükse nesne küçültülür, değilse büyültülür.

2,vbOLESizeAutoSize

Ole kontrolünün boyu içindeki nesnenin boyuna otomatik olarak ayarlanır.

3,vbOLESizeZoom

1 değeri gibidir. Ancak boyutlandırma sırasında nesnenin orijinal orantısı bozulmaz. Yani boyutları dikeyde yarıya indiyse yatayda da yarıya iner.

Methods

Close

Sadece gömme sonucundaki bağlantılar üzerinde etkili olan bu metod, OLE nesnesi ile kaynak arasındaki bağlantıyı keser.

```
Ole1.Close
```

Copy

OLE elemanı içindeki veriyi panoya kopyalar.

```
Ole1.Copy
```


CreateEmbed

Gömülü bir nesne oluşturur.

CreateLink

Bağlantılı bir nesne oluşturur.

Paste

OLE kontrol elemanına panodan veri yapıştırır.

```
Ole1.Paste
```

Update

OLE kontrolü içindeki veriyi yeniden günceller.

```
Ole1.Update
```

Close

Bir nesneyi kapatarak diğer uygulamalarla varolan bağlantıyı keser.

```
Ole1.Close
```

Delete

İlgili nesneyi silerek o nesne için ayrılan hafızayı boşaltır.

```
Ole1.Delete
```

DoVerb (İşlem)

Nesneye belirlenen işlemi uygular.

-1, vbOLEShow

Nesneyi düzenlemek için açar. Eğer nesne yerinde düzenlemeyi destekliyorsa form üzerinde düzenleme moduna geçer.

-2,vbOLEOpen

Nesneyi kendi penceresinde açar.

-3,vbOLEHide

Gömülü uygulamayı gizler.

-4, vbOLEUIActivate

Eğer nesne yerinde düzenlemeyi destekliyorsa, nesnenin kullanıcı arabirim form üzerinde yerini alır.

-5,VbOLEInPlaceActivate

Nesne kontrolü ele aldığında düzenleme moduna geçer.

-6,VbOLEDiscardUndoState

Nesne düzenleme moduna geçirildikten sonraki yapılan bütün değişimleri iptal eder.

SaveToFile(dosyano)

Nesneyi verilen dosyaya kaydeder. Bu dosyanın Binary modunda açılmış olması ve dosyanın numarasının bu metoda verilmesi gerekir.

```
Dim dosyaNo
Dosyano=Freefile
Open "dosya" for Binary As dosyano
Ole1.Filenumber dosyano
Ole1.Savetofile dosyano
Close #1
```

ReadFromFile(dosyano)

Bir dosyaya kaydedilen nesneyi yükler.

```
Dim dosyaNo
Dosyano=Freefile
Open "dosya" for Binary As dosyano
Ole1.Filenumber dosyano
Ole1.ReadFromFile  dosyano
Close #1
```

InsertObjDlg

Nesne yerleştirme diyalog kutusunu görüntüler.

```
Ole1.InsertObjDlg
```


PasteSpecialDlg:

Özel yapıştır diyalog kutusunu görüntüler.

```
Ole1.PasteSpecialDlg
```

FetchVerbs

Nesnenin desteklediği fiilleri günceller.

```
Ole1.FetchVerbs
```

SaveToOle1File

Bir nesneyi OLE1 formatında kaydeder.

```
Dim dosyaNo
Dosyano=Freefile
Ole1.Filenumber dosyano
Ole1.SavetoOle1File dosyano
```

Events

ObjectMove(left As Single, top As Single, width As Single, height As Single)

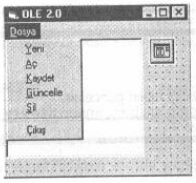
Ole kontrolü taşındığında veya boyutlandırıldığında bu olay meydana gelir. Olaydaki parametrelerle kontrolün yeni koordinatları ve boyutları öğrenilebilir.

Updated(code As Integer)

Ole kontrolünün içeriği güncellendiğinde bu olay meydana gelir. Code parametresi ile değişim hakkında bilgi alınabilir. Code parametresinin alabileceği değerler ve anlamları şöyledir.

0,vbOLEChanged:	Nesne içindeki bilgi değişti.
1,vbOLESaved:	Nesne içindeki bilgi kaydedildi.
2,vbOLEClosed :	Nesne kapatıldı.
3,vbOLERenamed :	Bağlantı kurulan dosyanın ismi değişti.

ÖRNEK: Şimdi Ole 2.0 ile ilgili iki örnek yapalım. Bunun için aşağıdaki form tasarımını kullanacağız:



Yukarıda görüldüğü gibi form üzerine Ole1 kontrolü ve CommonDialog1 yerleştiriliyor.

Yine yukarıda görülen menülerle bazı işlemler yapılmaktadır:

Yeni menü seçeneği ile yeni bir ole nesnesi oluşturulur.

Aç seçeneği ile daha önce kaydedilmiş ole nesneleri form üzerindeki ole1 kontrolüne yüklenir. Yüklenen bir ole nesnesinin çift tıklanmasıyla ole kaynağı ile otomatik bağlantı kurularak istenilen değişiklikler yapılabilir.

Kaydet seçeneği ile üzerinde çalışılan ole nesnesi kaydedilir. Güncelle seçeneği ile nesne güncellenir. Sil seçeneği ile Ole1'in içindeki nesne silinir.

Program çalıştırılıp istenilen dosya yüklendikten sonra Ole1 elemanı üzerinde iken farenin sağ tuşuna basıldığında aşağıdaki görüntü karşımıza çıkıyor:

Programın kodu aşağıdaki gibidir:

```
'Örnek:Ole 2.0
Option Explicit
Public Sub AcKaydet(menuklik As String)
    Dim Dosyano
```



```
    Dosyano = FreeFile
    ' Aç diyalog kutusunda çıkacak olan dosyaları filtrele
    frmOLE20.CommonDialog1.Filter = "Ole dosyaları
(*.OLE|*.OLE|Tüm      dosyalar (*.*)|*.*"
    frmOLE20.CommonDialog1.FilterIndex = 1
     frmOLE20.OLE1.FileNumber = Dosyano
On Error Resume Next
    Select Case menuklik
    Case "Kaydet"
     ' Kaydet diyalog penceresini aç
    frmOLE20.CommonDialog1.ShowSave
     ' Eğer kullanıcı cancel i seçmişse
     If Err = 32755 Then Exit Sub
     ' Diyalog kutusunu aç ve dosyayı kaydet
    Open frmOLE20.CommonDialog1.filename For Binary As Dosyano
     If Err Then Exit Sub
     frmOLE20.OLE1.SaveToFile Dosyano
     If Err Then MsgBox (Error)
    Case "Aç"
   ' Aç diyalog kutusunu aç
  frmOLE20.CommonDialog1.ShowOpen
  ' Eğer kullanıcı cancel i seçmişse
 If Err = 32755 Then Exit Sub
 ' dosya aç.
 Open frmOLE20.CommonDialog1.filename For Binary As Dosyano
If Err Then Exit Sub
frmOLE20.OLE1.ReadFromFile Dosyano
If (Err) Then
if Err = 30015 Then
MsgBox "Geçerli nesne değildir."
Else
MsgBox Error$
End If
Unload frmOLE20
End If
End Select
Close Dosyano
End Sub
```

ÖRNEK: Başka bir örnek olarak ta MDI formları kullanarak birden fazla nesneyi aynı pencerede yönetmeyi sağlayacak bir program yapalım.

Project menüsünden **Add MDI Form** seçeneği ile yeni bir MDI form oluşturun ve **Project** menüsündeki **Project Properties** seçeneği ile açılan pencereden **Startup Object** listesinde **MDIForm1** seçeneğini seçerek programın buradan çalışmaya başlamasını sağlayın.

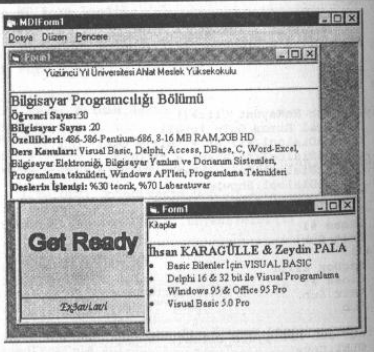
Ardından Form1 formunun **MDIChild** özelliğini **True** yapın ve bir **Ole** kontrolü ile bir **CommonDialog** kontrolünü form üzerin yerleştirerek aşağıdaki menü tasarımını Form1 için tasarlayın.


```
'Örnek:Ole
'MDI forma yazılacak
Private Sub MDIForm_Load()
    Form1.Show
End Sub
'Formle yazılacak
Private Sub Form_Load()
    Dim i
    OLE1.InsertObjDlg
    On Local Error Resume Next
    OLE1.Update
    OLE1.FetchVerbs
     For i = 1 To 10
         Unload mnmenü(i)'Daha önce yüklenmiş olanları sil
    Next
    For i = 1 To OLE1.ObjectVerbsCount - 1
        Load mnmenü(i)
        mnmenü(i).Caption = OLE1.ObjectVerbs(i)
    Next
End Sub
Private Sub MnAç_Click()
    On Local Error GoTo iptal
    CommonDialog1.Filter = "Bizim OLE|*.OLL||"
    CommonDialog1.filename = "*.OLL"
    CommonDialog1.ShowOpen
    Open CommonDialog1.filename For Binary As #1
    OLE1.ReadFromFile 1
    Close #1
    Exit Sub
iptal:
    MsgBox Error
    Close #1
    Exit Sub
End Sub
```



```
Private Sub MnBasamakla_Click()
    MDIForm1.Arrange vbCascade
End Sub

Private Sub MnÇık_Click()
    End
End Sub

Private Sub MnDikeyDöşe_Click()
    MDIForm1.Arrange vbTileVertical
End Sub

Private Sub mnDüzen_Click()
    Dim i
    On Local Error Resume Next
    For i = 1 To OLE1.ObjectVerbsCount - 1
        Select Case OLE1.ObjectVerbFlags(i)
            Case &H8: mnmenü(i).Checked = True
            Case &H2: mnmenü(i).Enabled = False
            Case &H0: mnmenü(i).Enabled = True
        End Select
    Next
    If OLE1.PasteOK Then
        MnYapıştır.Enabled = True
    Else
        MnYapıştır.Enabled = False
    End If
End Sub

Private Sub MnKaydet_Click()
    On Local Error GoTo iptal1
    CommonDialog1.Filter = "Bizim OLE|*.OLL||"
    CommonDialog1.filename = "*.OLL"
    CommonDialog1.DefaultExt = "*.OLL"
    CommonDialog1.ShowSave
    Open CommonDialog1.filename For Binary As #1
    OLE1.SaveToFile 1
    Close #1
    Exit Sub
iptal1:
    MsgBox Error
    Close #1
    Exit Sub
End Sub

Private Sub MnKopyala_Click()
    OLE1.Copy
End Sub

Private Sub mnmenü_Click(Index As Integer)
    On Local Error Resume Next
    OLE1.DoVerb Index
End Sub
```

```
Private Sub MnSil_Click()
    OLE1.Delete
End Sub

Private Sub MnSimgeleriYerleştir_Click()
    MDIForm1.Arrange vbArrangeIcons
End Sub

Private Sub MnYapıştır_Click()
    OLE1.Paste
End Sub

Private Sub MnYatayDöşe_Click()
    MDIForm1.Arrange vbTileHorizontal
End Sub

Private Sub MnYeni_Click()
    Dim Form As New Form1
    Form.Show
End Sub
```

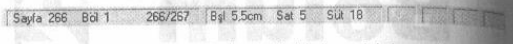


Bölüm 2

Windows 95 & 98 Kontrolleri

StatusBar (Durum Çubuğu)*

Bir çok programda pencerelerin en altında o anki durumu gösteren bazı bilgiler bulunur. Örneğin Word programının en alt satırında bazı bilgiler veren bir durum çubuğu bulunur.



Bu tip durum çubuklarını StatusBar kontrolünü kullanarak kolayca oluşturabilirsiniz.

Properties

Style

0,sbrNormal : Kontrol üzerinde birden fazla panel bulunabilir.

1,sbrSimple : Kontrol üzerinde sadece bir panel bulunabilir.

Panels

Kontrol üzerindeki paneller Panels özelliğinin aşağıdaki alt özellikleri ile yönetilir.

Panels.Add

Kontrole yeni paneller ekler. Eklenen bu panellere Panels(index) özelliği ile ulaşılır.

```
StatusBar1.Panels.Add
StatusBar1.Panels.Add
```

satırları ile iki tane yeni panel daha eklenebilir.

Panels.Remove (Index)

Index nolu paneli siler. İlk panelin numarası 1 dir.

*Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Windows Common Controls 6.0" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCOMCTL.OCX dosyasını bulup projeye eklemeniz gerekir.


```
StatusBar1.Panels.Remove 1
```

Satırı ile 1 numaralı panel silinebilir.

Panels.Clear

StatusBar kontrolündeki bütün panelleri siler.

Panels.Count

StatusBar kontrolündeki panel sayısını verir.

Panels(i).Style

0,sbrText : Text veya Resim içerebilen bir panel.

1,sbrCaps : Caps Lock tuşunu temsil eden bir panel

2,sbrNum : Number Lock tuşunu temsil eden bir panel

3,sbrIns : Insert tuşunu temsil eden bir panel

4,sbrScrl : Scroll Lock tuşunu temsil eden bir panel

5,sbrTime : Saati gösteren bir panel

6,sbrDate : Date gösteren bir panel

```
Dim i
    For i = 1 To 6
        StatusBar1.Panels.Add
        StatusBar1.Panels(i).Style = i
    Next
```

Yukandaki gibi bir kodla bütün paneller aşağıdaki gibi listelenebilir.

Panels(i).Alignment

Index numarası verilen panelin içeriğinin yerleşimini belirler.

0,sbrLeft : Sola dayalı

1,sbrCenter : Ortada

2,sbrRight : Sağa dayalı.

Panels(i).AutoSize

0,sbrNoAutoSize: Herhangi b,r boyutlandırma işlemi uygulanmaz.

1,sbrSpring: Form boyutlandırıldığında, statusbar kontrolünün kullanabileceği fazladan yer açılıyorsa her panel bu fazlalığı paylaşır.

2,sbrContents: Her panel gerekiyorsa içeriğini tam gösterecek şekilde genişler.

Panels(i).Bevel

Panellerin üç boyutlu görünümünü ayarlar.

0,sbrNoBevel: Üç boyut efekti uygulanmaz. Düz gösterilir.

1,sbrInset : İçe basık bir üç boyut efekti uygulanır.

2,sbrRaised: Dışa basık bir üç boyut efekti uygulanır.


```
Dim i
    For i = 1 To 6
        StatusBar1.Panels.Add
        StatusBar1.Panels(i).Style = i
        StatusBar1.Panels(i).Bevel = 1 + (i Mod 2)
    Next
```

Yukarıdaki kodla aşağıdaki gibi her bir paneli sırayla içe ve dışa basık olarak konumlandırabiliriz.

Panels(i).Visible

Index numarası verilen panel gösterilip gizlenebilir.

Panels(1).Enabled

Caps, Num, Ins ve Scrl tuşlarının durumlarını gösteren panellerde o tuş aktifse Enabled özelliği True, pasifse False olur. Bu özelliğe bakarak o tuşların durumu öğrenilebilir.

```
StatusBar1.Panels(1).Style = sbrCaps
DoEvents
If StatusBar1.Panels(1).Enabled Then
      Print "Capslock yaniyor"
Else
      Print "Capslock sönük"
End If
```


Panels(i).Key

Paneller açıklayıcı bir isim vermek için kullanılır.

Panels(i).MinWidth, Panels(i).MaxWidth, Panels(i). Width

Bu özelliklerle her bir panelin gelebileceği minimum ve maksimum boyutları ve aktif boyutu belirlenebilir.

Panels(i).Text

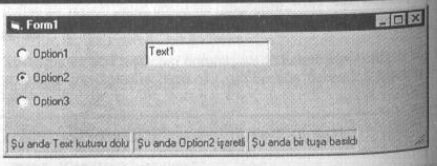
Panel üzerinde yazılacak metin bu özellikle belirlenir.

ÖRNEK: Örnek olarak form üzerindeki kontrollerle ilgili bilgi verecek bir program yazalım. Örneğimiz için form üzerine üç tane OptionButton, bir Text kutusu ve bir StatusBar kontrolü yerleştirin.

```
'Ornek:StatusBar
Private Sub Form_Load()
 StatusBar1.Panels(1).Style = sbrText
 StatusBar1.Panels(1).AutoSize = sbrContents
 StatusBar1.Panels(1).Text = "Şu anda Text kutusu dolu"
 StatusBar1.Panels.Add
 StatusBar1.Panels(2).Style = sbrText
 StatusBar1.Panels(2).AutoSize = sbrContents
 StatusBar1.Panels(2).Text = "Şu anda Option1 işaretli"
 StatusBar1.Panels.Add
 StatusBar1.Panels(3).Style = sbrText
 StatusBar1.Panels(3).AutoSize = sbrContents
 KeyPreview = True
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
 StatusBar1.Panels(3).Text = "Şu anda bir tuşa basıldı"
End Sub

Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
 StatusBar1.Panels(3).Text = "Tuş bırakıldı"
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, x
As Single, y As Single)
 StatusBar1.Panels(3).Text = "Mouse Form üzerinden geçiyor"
End Sub

Private Sub Option1_Click()
 StatusBar1.Panels(2).Text = "Şu anda Option1 işaretli"
End Sub
```

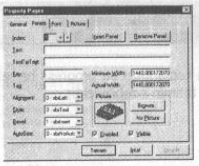


```
Private Sub Option1_MouseMove(Button As Integer, Shift As
Integer, x As Single, y As Single)
 StatusBar1.Panels(3).Text = "Mouse Option1 üzerinden geçiyor"
End Sub

Private Sub Option2_Click()
 StatusBar1.Panels(2).Text = "Şu anda Option2 işaretli"
End Sub

Private Sub Option2_MouseMove(Button As Integer, Shift As
Integer, x As Single, y As Single)
 StatusBar1.Panels(3).Text = "Bir Fare Option2 üzerinden geçiyor"
End Sub

Private Sub Option3_Click()
 StatusBar1.Panels(2).Text = "Şu anda Option3 işaretli"
End Sub

Private Sub Option3_MouseMove(Button As Integer, Shift As
Integer, x As Single, y As Single)
 StatusBar1.Panels(3).Text=" Bir Fare Option3 üzerinden geçiyor"
End Sub

Private Sub StatusBar1_MouseMove(Button As Integer, Shift As
Integer, x As Single, y As Single)
 StatusBar1.Panels(3).Text = " Bir Fare benim üzerinden geçiyor"
End Sub

Private Sub Text1_Change()
 If Len(Text1) = 0 Then
  StatusBar1.Panels(1).Text = "Şu anda Text kutusu boş"
 Else
  StatusBar1.Panels(1).Text = "Şu anda Text kutusu dolu"
 End If
End Sub

Private Sub Text1_MouseMove(Button As Integer, Shift As Integer,
x As Single, y As Single)
 StatusBar1.Panels(3).Text = " Bir Fare Text1 üzerinden geçiyor"
End Sub
```



Panels(i).Picture

Panellere metnin yanı sıra BMP,ICO ve WMF türündeki resimler de yüklenebilir. Bu resimler tasarım zamanında Custom özelliği ile açılan aşağıdaki pencereden yüklenebileceği gibi çalışma zamanında LoadPicture komutu ile de yüklenebilir.



Events

PanelClick(ByVal Panel As ComctlLib.Panel)

Panellerden biri tıklandığında PanelClick olayı meydana gelir. Buradaki Panel parametresi ile ilgili Panelin bütün özellikleri öğrenilebilir ve değiştirilebilir.

Örneğin bir panel tıklandığında panelin içeriği hakkında bilgi vermek için:

```
Private Sub StatusBar1_PanelClick(ByVal Panel As ComctlLib.Panel)
    MsgBox (Panel.Text)
End Sub
```

PanelDblClick(ByVal Panel As ComctlLib.Panel)

Panellerden biri çift tıklandığında ise PanelDblClick olayı meydana gelir.

Örneğin bir panel çift tıklandığında içeriğini sıfırlamak için:

```
Private Sub StatusBar1_PanelDblClick(ByVal Panel As
ComctlLib.Panel)
    Panel.Text = 0
End Sub
```

ProgressBar (İlerleme Çubuğu)*

Windows altında bir çok programda gördüğümüz, yapılan işlemin ilerleme durumunu gösteren kontrollerdendir. Bu kontrolün Min ve Max özellikleri ile aralık belirlendikten sonra Value özelliği ile yapılan işlemin o anki değeri belirlenir. Bu kontrol kullanılmadığı zaman ise çoğunlukla gizlenir.

Özellikle uzun süren kodlarınıza bir progress bar eklerseniz kullanıcı kodun daha ne kadar süreceğini tahmin edebilecektir.

Properties

Min, Max

ProgressBar'ın alacağı maksimum ve minimum değerleri belirler. Minimum değer olarak en az 0 verilebilir. Maximum değer olarak ise 10^{38} a kadar olan bir sayı verilebilir.

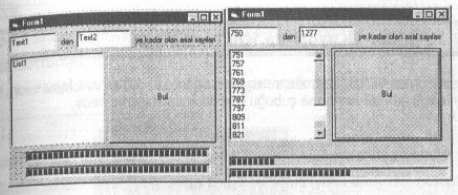
Value

ProgressBarın göstereceği değer bu özellikle belirlenir.

ÖRNEK: Örnek olarak kullanıcının girdiği iki aralıktaki asal sayıları bulan bir program yapalım.

Bir sayının asal olması için kendisinden ve 1 den başka hiçbir sayıya tam olarak bölünememesi gerekir. Yani 2'den kendisinin 1 eksiğine kadar olan hiç bir sayıya bölünememesi gerekir. Bu işlemi bir progress bar ile, şu anda asal olup olmadığına bakılan sayıyı ise diğer progress bar ile gösterelim.

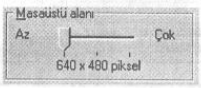
* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Windows Common Controls 6.0" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCOMCTL.OCX dosyasını bulup projeye eklemeniz gerekir.




```
'Örnek:ProgressBar
Private Sub Form_Load()
    ProgressBar1.Align = vbAlignBottom
    ProgressBar2.Align = vbAlignBottom
    ProgressBar1.Visible = False
    ProgressBar2.Visible = False
End Sub

Private Sub Command1_Click()
    Dim i, j, asal As Boolean
    On Local Error Resume Next
    List1.Clear
    ProgressBar1.Visible = True
    ProgressBar2.Visible = True
    ProgressBar1.Min = 0
    ProgressBar1.Max = Val(Text2) - Val(Text1)
    For i = Val(Text1) To Val(Text2)
        asal = True
        ProgressBar2.Min = 2
        ProgressBar2.Max = i
        ProgressBar1.Value = i - Val(Text1)
        For j = 2 To i - 1
            ProgressBar2.Value = j
            DoEvents
            If (i Mod j) = 0 Then
                'Tam bölünüyorsa asal değil.
                asal = False
                Exit For
            End If
        Next
        If asal Then List1.AddItem i
    Next
    ProgressBar1.Visible = False
    ProgressBar2.Visible = False
End Sub
```


Slider (Kaydırma Kontrolü)*

Windows-95/98 kontrollerinden olan Slider bir değeri ayarlamak için kullanılabileceği gibi bir kaydırma çubuğu gibi de kullanılabilmektedir.



Properties

Max,Min

Slider kontrolünün temsil edeceği değer aralıkları bu iki özellikle belirlenir.

Value

Slider kontrolünün şu anda temsil ettiği değer bu özellikle öğrenilip değiştirilebilir.

Orientation

Slider kontrolünün yerleşimini belirler. 0 için yatay, 1 için dikey olarak konumlandırılır.

TickFrequency

Slider kontrolünün üzerinde belli aralıklarla bir cetvel gibi çizgiler gösterilir. Bu çizgilerin sıklığı ise TickFrequency özelliği ile belirlenir. Örneğin Max özelliği 1000 olan bir Slider için bu değere 100 verilirse slider üzerinde 1000/100=10 tane çizgi gösterilir.

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Windows Common Controls 6.0" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCOMCTL.OCX dosyasını bulup projeye eklemeniz gerekir.

TickStyle

Slider üzerindeki çizgilerin yerleşim stili bu özellikle belirlenir.

0,sldBottomRight

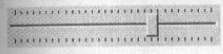
Çizgiler sadece Altta (yönü yatay ise) veya Sağda (yönü dikey ise) gösterilir.

1,sldTopLeft

Çizgiler sadece üstte (yönü yatay ise) veya Solda (yönü dikey ise) gösterilir.

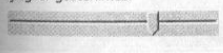
2, sldBoth

Çizgiler her iki yönde de gösterilir.



3, sldNoTicks

Çizgiler gösterilmez.



LargeChange, SmallChange

Kaydırma çubuklarında da gördüğümüz bu özellikler Slider üzerindeki değişimin hangi aralıklarla yapılacağını belirlerler. SmallChange, Slider kontrolünün en küçük değişim aralığını belirlerken, LargeChange özelliği de PageUp ve PageDown tuşları ile ne kadarlık bir değişimin gerçekleşeceğini belirler.

Events

Change( )

Slider kontrolünün değerinde bir değişme olduğunda Change olayı meydana gelir.

Scroll()

Slider kontrolünün değerin değişirken Scroll olayı meydana gelir. Change olayı sadece slider kontrolünün değeri değiştiği anda meydan gelir. Scroll olayı ise kullanıcı göstergeyi bir yerden bir yere sürüklerken sürekli olarak meydana gelir.

ÖRNEK: Örnek olarak bir formun yüksekliğini ve genişliğini Slider kontrolleri kontrol edelim. Bunun için formunuzun üzerine iki tane Slider kontrolü yerleştirin.

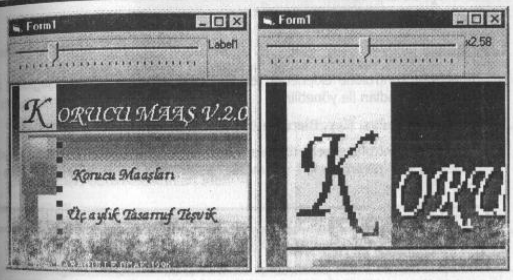
```
'Örnek:Slider
Private Sub Form_Load()
    Slider2.Orientation = sldVertical
    Slider1.Orientation = sldHorizontal
    Slider1.Max = Screen.Width
    Slider1.Value = Form1.Width
    Slider2.Max = Screen.Height
    Slider2.Value = Form1.Height
    Slider1.Left = 0
    Slider1.Top = 0
    Slider2.Left = 0
    Slider2.Top = Slider1.Height
End Sub

Private Sub Form_Resize()
    Slider1.Value = Form1.Width
    Slider2.Value = Form1.Height
End Sub

Private Sub Slider1_Scroll()
    Form1.WindowState = 0
    Form1.Width = Slider1.Value
End Sub

Private Sub Slider2_Scroll()
    Form1.WindowState = 0
    Form1.Height = Slider2.Value
End Sub
```

ÖRNEK: İkinci bir örnek olarakta bir PictureBox içindeki resmi slider kontrolü ile büyültüp küçültelim. Örneğimiz için içinde resim olan bir PictureBox, bir Label ve bir Slider yerleştirin.

```
'Örnek:Slider
Private Sub Form_Load()
    Slider1.Orientation = sldHorizontal
    Slider1.Max = 5 * Picture1.Width
    Slider1.TickFrequency = Picture1.Width / 5
    Slider1.Value = Picture1.Width
    Slider1.Left = Picture1.Left
```



```
    Slider1.Top = Picture1.Top - Slider1.Height
    Label1.Top = Slider1.Top
    Label1.Left = Slider1.Left + Slider1.Width
End Sub

Private Sub Slider1_Scroll()
    On Local Error Resume Next
    Picture1.PaintPicture Picture1.Picture, 0, 0, Slider1.Value,
Slider1.Value, 0, 0, Picture1.Width, Picture1.Height, vbSrcCopy
    Label1 = "x" & Format(Slider1.Value / Picture1.Width, "0.00")
End Sub
```



ImageList (Resim Kolleksiyonu)*

Birden fazla resmi bir arada tutmak için kullanılan bir kontroldür. Daha çok ListView, ToolBar, TabStrip, TreeView gibi birden fazla resimin bulunabileceği kontrollerdeki resimleri bir arada tutmak için kullanılır.

Properties

ListImages

ImageList kontrolüne eklenecek resimler ListImages özelliğinin aşağıdaki alt özellik ve metodları ile yönetilir.

ListImages.Add(Index, Key, Picture)

ImageList kontrolüne yeni resimler eklemek için bu metod kullanılır. Buradaki Index ve Key parametreleri verilmezse resim sona, verilirse verildiği yere eklenir.

```
ListImages.Add(, ,LoadPicture("Deneme.bmp")
```

satırı Deneme.Bmp dosyasını listedeki son resim olarak ekleyecektir.

Bu tip listelerde Index parametresi ile listenin bir elemanına ulaşabiliyorduk. Buradaki Key parametresi bize alternatif bir erişim şekli sunar. Programcı isterse index özelliği ile isterse String olan Key özelliği ile o elemana ulaşabilir.

```
ListImages.Add(1,"İlk Resim",LoadPicture("Deneme.bmp")
```

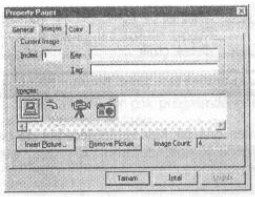
satırı ile eklenmiş bir resime

ListImages(1) satırı ile veya **ListImages("İlk Resim")** satırı ile ulaşılabilir.

Görüldüğü gibi Key özelliği resimleri yönetmek için sayılar yerine yazıları da kullanabilme imkanı verdiği için programlamayı kolaylaştırmaktadır.

Tasarım zamanında **Custom** özelliği ile açılan aşağıdaki pencereyi kullanarak gerekli resimleri kolayca ekleyip çıkarabilirsiniz.

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Windows Common Controls 6.0" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCOMCTL.OCX dosyasını bulup projeye eklemeniz gerekir.



ListImages.Count

Liste içindeki resim sayısı bu özellikle öğrenilir.

ListImages.Remove index

Liste içindeki index numaralı elemanı siler. İlk elemanın indexi 1 dir.

ListImages.Clear

Liste içindeki bütün elemanları siler.

ListImages(i).Picture

Listedeki I numaralı resimi öğrenmek için bu özellik kullanılır. Buradaki ilk elemanın indexi 1 dir. Index yerine Key için verilen metin de kullanılabilir.

```
Form1.Picture = ImageList1.ListImages(1).Picture
```

ListImages(i).ExtractIcon

Numarası verilen resimden bir ikon oluşturur. Buradan geriye dönen değer herhangi bir Icon özelliğine atanabilir.

```
Form1.Icon= ImageList1.ListImages(1).ExtractIcon
```

ListImages(i).Draw (hDC, x,y, style)

Liste içindeki bir resimi Picture özelliği ile alabileceğimiz gibi, Draw metodunu kullanarak başka bir kontrolün içine de çizdirebiliriz.

Buradaki hDC parametresine çizdirilecek yerin hDC numarası, x,y ile koordinatları Style ilede uygulanacak efect belirlenebilir. Çoğunlukla ImageList komut düğmeleri için düşünüldüğünden dolayı verilecek efektlerde bir komut düğmesinin alabileceği değişik durumları temsil eder. Style parametresinin alabileceği değerler şunlardır.

0,imlNormal:	Aynen çizilir.
1,imlTransparent:	Kopyalandığı yerin altını gösterecek şeklide çizilir.

2,imlSelected: Seçilmiş bir kontrol efekti verilerek çizilir.
3,imlFocus: Klavye kontrolünü elinde bulunduran bir kontrol efekti verilerek çizilir.

```
ImageList1.ListImages(1).Draw Form1.hDC, 0 0, imlSelected
```

ListImages(i).Key

Eleman eklenirken ona bir Key verilebileceğini de belirtmiştik. Bu değer daha sonra Key özelliği ile öğrenilip değiştirilebilir.

Methods

Overlay(index1, index2)

Verilen iki resmin geçişini oluşturur. Yani index2 ile belirtilmiş resmi, index1 ile belirtilmiş resmin üzerine çizer. Birden fazla resim bu şeklide arka arkaya gösterilerek yumuşak geçişli animasyonlar yapılabilir.

```
Picture1.Picture = ImageList1.Overlay(1, 2)
```

ÖRNEK: Örneğin bir animasyonun parçalarını oluşturan 7 tane resimi ListImage kontrolüne ekleyerek bir Timer aracılığıyla bunları arka arkaya göstererek, normal gösterimle, Overlay gösterim arasındaki farkı görebilirsiniz.

```
Private Sub Form_Load()
 Dim i
 For i = 1 To 7
 'Anim1.bmp-Anim7.Bmp dosyaları yükleniyor.
  Call ImageList1.ListImages.Add(, , LoadPicture("anim" & i &
".bmp"))
 Next
End Sub

Private Sub Timer1_Timer()
 Static i
 Dim j
 i = i Mod 6
 i = i + 1
 j = i + 1
 j = (i Mod 7) + 1
 'Overlay gösterim
 Picture1.Picture = ImageList1.Overlay(i, j).
 'Normal gösterim
 Picture2.Picture = ImageList1.ListImages(i).Picture
End Sub
```


ListView (Resimli Liste)*

Windows-95 ile birlikte gelen kontrollerden biri de ListView kontrolüdür. Bu kontrolleri Windows-95 ve 98 altında bir çok programda görebilirsiniz. Örneğin denetim masası programı bu kontrolü kullanır.

Bu listenin en önemli özelliği içindeki elemanları birer resimle temsil edebilmesi ve bunları farklı biçimlerde gösterebilmesidir.

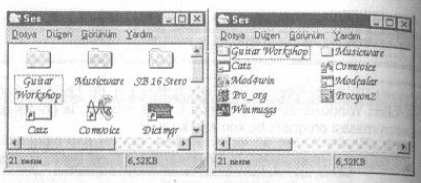
Properties

Icons, SmallIcons

ListView kontrolü her bir eleman için büyük ve küçük ikonlar gösterebilmektedir. ListView kontrolünde gösterilecek resimler bir ImageList kontrolüne yüklendikten sonra bu iki özelliğe o kontrollerin isimleri verilir. Daha sonra bir elemanın hangi ikonu kullanacağı o elaman eklenirken ImageList içindeki indexi ile belirlenir.

```
ListView1.Icons = ImageList1
ListView1.SmallIcons = ImageList2
```

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Windows Common Controls 6.0" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCOMCTL.OCX dosyasını bulup projeye eklemeniz gerekir.



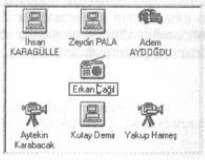
LabelEdit

0, lvwAutomatic

Herhangi bir koda gerek olmadan kullanıcı liste içindeki etiketleri aralıklı iki mouse tıklaması ile düzenleyebilir.

1, lvwManual

Kullanıcı labelleri otomatik olarak değiştiremez. Değiştirebilmesi için StartLabelEdit metodunun çağrılması gerekir.



Sorted, SortOrder,Sortkey

Liste içindeki elemanların sıralanıp sıralanmayacağı bu özelliklerle belirlenir. Sorted özelliğine True verildiğinde liste sırlanacaktır.

SortOrder özelliği ilede sıralamanın artan mı, azalan mı olacağı belirlenir.

0,lvwAscending: Artan sıralama. A-Z
1,lvwDescending : Azalan sıralama Z-A

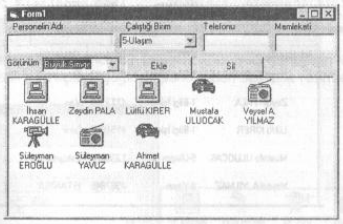
SortKey özelliği ilede sıralamanın neye göre yapılacağı belirlenir. Bu özelliğe 0 verilirse sıralama Text özelliğine göre, başka bir sayı verilirse Index (listedeki sıra numarası) özelliğine göre sıralanır.

View

ListView kontrolü içindeki elemanları farklı şekillerde gösterebilir. Bu özellikle elemanların nasıl gösterileceği belirlenir.

0,lvwIcon :

Elemanların Icon ve etiketleri gösterilir. Ikonlar büyük ve etiketler ikonun altındadır. Gösterilecek ikonlar Icons özelliği ile belirlenmiş ikonlardır.

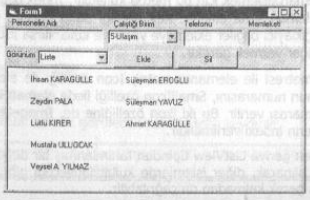


1,lvwSmallIcon:

Elemanların ikon ve etiketleri gösterilir. Fakat Ikonlar küçük ve etiketler ikonun yanındadır. Gösterilecek ikonlar SmallIcons özelliği ile belirlenmiş ikonlardır.

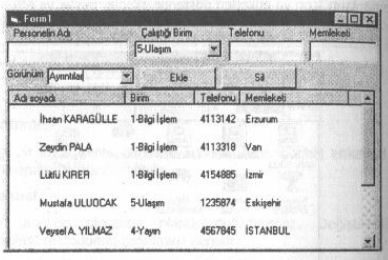
2,lvwList:

Elemanların ikon ve etiketleri liste şeklinde gösterilir. Ikonlar küçük ve etiketler ikonun yanındadır. Gösterilecek ikonlar SmallIcons özelliği ile belirlenmiş ikonlardır. Ayrıca elemanlar dikey olarak hizalanır.



3,lvwReport:

Elemanların ikon ve etiketleri ayrıntılı liste şeklinde gösterilir. Ikonlar küçük ve etiketler ikonun yanındadır. Gösterilecek ikonlar SmallIcons özelliği ile belirlenmiş ikonlardır. Ayrıntılar o elemanın SubItems(i) özelliği ile belirlenir. Ve her bir SubItems tanımlanmış kolonu altında gösterilir.



Büyük ve küçük ikonla gösterim modunda kullanıcı elemanların yerlerine fare ile değiştirebileceği gibi, textlerinide aralıklı olarak iki defa tıklayarak değiştirebilir.

ListItems

Listedeki elemanlar ListItems özelliğini, aşağıdaki alt özellik ve metodları ile yönetilir.

ListItems.Add(index, key, text, icon, smallIcon)

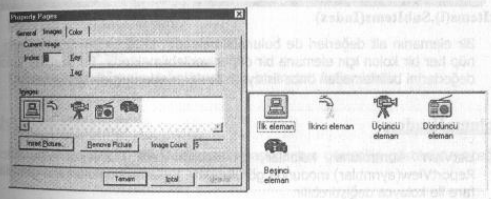
Listeye eleman eklemek için bu metod kullanılır. Index özelliği verilirse oraya verilmezse eleman sona eklenir. Key özelliği daha öncede gördüğümüz gibi elemanlara açıklayıcı bilgiler eklemeye yarar ve kullanılırsa listedeki her bir elemana farklı bir diğer vermek gerekir.

Text parametresi ile elemanın metni, Icon özelliği ile eleman için gösterilecek büyük ikonun numarasını, SmallIcon özelliği ilede eleman için gösterilecek küçük ikonun numarası verilir. Bu iki ikon özelliğine de, ImageList kontrolü ile yüklenmiş resimlerin indexi verilmelidir.

Bu özellikten geriye ListView tipinden tanımlanmış bir değer döner. Bu değer bir değişkene alınarak, diğer işlemlerde kullanılabileceği gibi Call ile bir değişkene aktarmaya gerek kalmadan da çağrılabilir.


```
ListView1.Icons=ImageList1'İkonlar ImageList1 kontrolünden
alınacak.
Call ListView1.ListItems.Add( , , "İlk eleman", 1)'1 numaralı
resimi ver
Call ListView1.ListItems.Add( , , "İkinci eleman", 2)'2 numaralı
resimi ver
Call ListView1.ListItems.Add( , , "Üçüncü eleman", 3)'3 numaralı
resimi ver
Call ListView1.ListItems.Add( , , "Dördüncü  eleman", 4)'4 nolu
resimi ver
Call ListView1.ListItems.Add( , , "Beşinci  eleman", 5)'3
numaralı resimi ver
```

Bu kodu çalıştırabilmek için önce bir ImageList kontrolü yerleştirin ve bu kontrolün Custom özelliği ile açılan aşağıdaki pencereden beş tane resim ekleyin.



ListItems.Remove Index

Indexi verilen elemanı listeden siler. İlk elemanın Index numarası 1'dir. Seçili elemanı silmek için ise SelectedItems özelliğinin Index özelliği kullanılabilir.

```
ListView1.ListItems.Remove 1'İlk elemanı sil
ListView1. ListItems.Remove ListView1.SelectedItems.Index 'Seçili
elemanı sil
```

ListItems.Clear

Listedeki bütün elemanları siler.

ListItems.EnsureVisible(Index)

Index numaralı eleman listede görülecek şekilde liste kaydırılır.

```
ListView1.ListItems(10).EnsureVisible'10 numaralı eleman
görülecek şekilde listeyi kaydır.
```


ListItems(i).Text, ListItems(i).Key

Eleman eklenirken elemana verilen text ve key özellikleri daha sonra bu özellikle öğrenilip değiştirebilir. Ayrıca text özelliğini kullanıcıda değiştirebilir.

ListItems(i).Selected

Listedeki i numaralı elemanın seçili olup olmadığını öğrenmek ve seçmek için bu özellik kullanılır.

```
ListView1.ListItems(1).Selected= True
```

ListItems(i).Ghosted

Listedeki i numaralı elemana pasif bir görünüm verir.

```
ListView1.ListItems(1).Ghosted = True
```

ListItems(i).SubItems(Index)

Bir elemanın alt değerleri de bulunabilmektedir. Liste birden fazla kolona bölünüp her bir kolon için elemana bir değer verilebilmektedir. Bu özellikle kolonların değerlerini belirlemeden önce listeye kolonların eklenmiş olması gerekir.

ColumnHeaders

ListView kontrolüne kolonlar eklenebilmektedir. Eklenen bu kolonlar ReportView(ayrıntılar) modunda görülecektir. Kullanıcı bu kolonların genişliklerini fare ile kolayca değiştirebilir.

Her bir elemanın kolonlarına değer eklemek için ListItems(i).SubItems(Index) özelliği kullanılır. Liste kontrolüne yeni kolonlar ekleyip-silmek için aşağıdaki alt özellik ve metodlar kullanılır.

ColumnHeaders.Add(index, key, text, width, alignment)

Listeye yeni kolonlar eklemek için bu metod kullanılır. Text özelliği ile kolonun başlığı, Width ile genişliği, alignment ilede yerleşim şekli belirlenir.

```
Call ListView1.ColumnHeaders.Add(, , "Adı soyadı",
ListView1.Width / 4)
Call ListView1.ColumnHeaders.Add(,,"Birim", ListView1.Width / 4)
Call ListView1.ColumnHeaders.Add(,,"Telefonu",ListView1.Width/4)
Call ListView1.ColumnHeaders.Add(,,"Memleketi",ListView1.Width/4)
ListView1.View = lvwReport'Ayrıntı moduna geç
Dim x As ListItem
Set x = ListView1.ListItems.Add(, , "İhsan KARAGÜLLE")
x.SubItems(1) = "Bilgi İşlem"
x.SubItems(2) = "4113142"
x.SubItems(3) = "Erzurum"
```


ColumnHeaders.Remove Index

Indexi verilen kolonu siler.

ColumnHeaders.Clear

Listedeki bütün kolonları siler.

ColumnHeaders(i).Count

Listedeki kolon sayısını verir.

ColumnHeaders(i).Text

Kolon başlığını öğrenip değiştirmek için bu metod kullanılabilir.

HideColumnHeaders

ReportView (ayrıntı) modunda kolon başlıklarının gösterilip gösterilmeyeceği bu özellikle belirlenir.

SelectedItem

Seçili elemana ait özelliklere ulaşmak için bu özellikten faydalanılır. ListItems özelliği gibi kullanılır.

Örneği seçili elemanın textini değiştirmek için

```
ListView1.SelectedItem.Text = "Merhaba"
```

Methods

FindItem(string, value, index, match)

Listede bir elemanı bulmak için bu metod kullanılabilir. Bu metoddan geriye ListItem tipinden bir değer döner. Geriye dönen bu değerin yukarıda gördüğümüz alt özellikleri ile bulunan eleman hakkında bilgi alınabilir. Elemanın buluna-

madığını ise **Is Nothing** ile anlaşılabilir. Ayrıca eleman bulunduktan sonra **EnsureVisible** metodu ile ekranda görünür duruma getirilebilir.

String

Aranacak eleman bu parametre ile belirlenir.

Value

Aramanın nerede yapılacağı bu parametre ile belirlenir. Bu parametreye aşağıdaki değerler verilebilir.

lvwText:	String text özelliğinde aranır.
LvwSubItem :	String alt elemanlarda (kolonlarda) aranır.
LvwTag:	String tag özelliğinde aranır.

Index

Bu özellik ile aramaya kaçıncı elemandan başlanacağı belirlenir. Verilmezse 0 elemandan başlanır. Bir eleman bulunduktan sonra aramaya kaldığı yerden devam etmek için bu özelliğe bulunan elemanın indexini vermek gerekir.

Kullanıcının Text1 kutusuna yazdığı metni listede bulmak için aşağıdaki gibi bir kod kullanılabilir.

```
Private Sub Command1_Click()
 Dim x As ListItem, c
 Set x = ListView1.FindItem(Text1, lvwText, 1)
 If x Is Nothing Then
  c = MsgBox("Böyle bir eleman adına rastlanmadı. Alt kolonlarda
aransınmı", vbYesNo, "Bul")
  If c = vbYes Then
   Set x = ListView1.FindItem(Text1, lvwSubItem, 1)
   If x Is Nothing Then
    c = MsgBox("Alt kolonlardada bulunamadı")
    Exit Sub
   End If
  Else
   Exit Sub
  End If
 End If
 x.EnsureVisible'bulunan elemanı ekranda görünecek şekilde ayarl
 x.Selected = True'Seç
 ListView1.SetFocus'kontrolü ver.
End Sub
```

Sonraki seferlerde arandığında en son kaldığı yerden aramaya devam etmesi sağlamak için ise bir static değişkenden faydalınabilir. Bu değişkene en son bulunan elemanın listedeki yeri atanarak sonraki seferlerde buradan aramaya devam etmesi sağlanabilir.


```
Private Sub Command3_Click()
 Dim x As ListItem, c
 Static b
 If IsEmpty(b) Then b = 0
 Set x = ListView1.FindItem(Text1, lvwText, b + 1)
 If x Is Nothing Then
  c = MsgBox("Böyle bir eleman adına rastlanmadı. Alt kolonlarda
aransınmı", vbYesNo, "Bul")
  If c = vbYes Then
   Set x = ListView1.FindItem(Text1, lvwSubItem, 1)
   If x Is Nothing Then
    c = MsgBox("Alt kolonlardada bulunamadı")
    b = 0
    Exit Sub
   End If
  Else
   b = 0
   Exit Sub
  End If
 End If
 b = x.Index
 x.EnsureVisible
 x.Selected = True
 ListView1.SetFocus
End Sub
```

GetFirstVisible

Listenin o anda ekranda görülen kısmındaki ilk eleman bu özellikle öğrenilebilir. Standart liste kutularının TopIndex özelliği gibidir. Farklı olarak elemanın sadece indexi değil bütün özellikleri geri döner.

```
Dim x As ListItem
Set x = ListView1.GetFirstVisible
MsgBox ("Ekrandaki ile eleman" & x.Text)
```

HitTest(x As Single, y As Single)

Bu metod ile verilen x,y koordinatlarındaki eleman öğrenilir. Çoğunlukla Drag-Drop işlemlerinde farenin bırakıldığı noktadaki elemanı öğrenmek için kullanılır.

Events

BeforeLabelEdit(cancel As Integer)

Kullanıcının ikon gösterim modunda elemanların etiketlerini değiştirebileceğini belirtmiştik. Kullanıcı bir etiketi değiştirmek üzereyken bu olay meydana gelir. Değişime izin verilmeyecekse Cancel parametresine True değeri atanabilir.

Bir eleman kullanıcı tarafından değiştirilirken önce seçileceği için hangi elemanın değiştirildiği SelectedItem özelliği ile öğrenilir.

AfterLabelEdit(cancel As Integer, newstring As String)

Kullanıcı bir etiketi değiştirdikten sonra bu olay meydana gelir. NewString parametresi ile kullanıcının girdiği yeni metni öğrenebilir ve Cancel özelliğine True atayarak ta değişimi iptal edebilirsiniz.

```
Private Sub ListView1_BeforeLabelEdit(Cancel As Integer)
 If ListView1.SelectedItem.Text = "İhsan" Then
  MsgBox ("Bu elemanın ismini değiştiremezsiniz")
  Cancel = True
 End If
End Sub

Private Sub ListView1_AfterLabelEdit(Cancel As Integer, NewString
As String)
 If NewString = "İhsan" Then
  MsgBox ("Bir elemana bu ismi sonradan veremezsiniz")
  Cancel = True
 End If
End Sub
```

Bu kod içeriği İhsan olan elemanın değiştirilmesine ve başka bir elemanın ismi nin İhsan olarak değiştirilmesine de müsaade etmeyecektir.

ColumnClick(ByVal columnheader As ColumnHeader)

Listedeki bir kolon tıklandığında bu olay meydana gelir. Hangi kolonun tıklandığı daha önce gördüğümüz ColumnHeader tipinden tanımlanmış columnheader parametresi ile öğrenilir.

ÖRNEK: Örneğin kullanıcının tıkladığı kolon hakkında bilgi vermek için:

```
Private Sub Form_Load()
 'Dört tane kolon ekle
 Call ListView1.ColumnHeaders.Add(,,"Adı soyadı",ListView1.Width
/4)
 Call ListView1.ColumnHeaders.Add(,,"Birim", ListView1.Width / 4)
 Call ListView1.ColumnHeaders.Add(,,"Telefonu",ListView1.Width/4)
 Call ListView1.ColumnHeaders.Add(,,"Memleketi",ListView1.Width
/4)
 ListView1.View = lvwReport
End Sub

Private Sub ListView1_ColumnClick(ByVal ColumnHeader As
ComctlLib.ColumnHeader)
 Select Case ColumnHeader.Index
  Case 1:MsgBox("Bu kolonda personelin isimleri listelenir")
  Case 2:MsgBox("Bu kolonda personelin çalıştığı birim
listelenir")
  Case 3:MsgBox("Bu kolonda personelin telefonları listelenir")
  Case 4:MsgBox("Bu kolonda personelin memleketleri listelenir")
 End Select
End Sub
```

Veya bir kolon tıklandığında artan sırada, tekrar tıklandığında ise azalan sırada listelemek için:

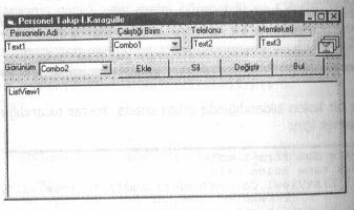
```
Private Sub Form_Load()
 'Dört tane kolon ekle
 Call ListView1.ColumnHeaders.Add(,,"Adı soyadı",ListView1.Width
/4)
 Call ListView1.ColumnHeaders.Add(,,"Birim",ListView1.Width / 4)
 Call ListView1.ColumnHeaders.Add(,,"Telefonu",ListView1.Width/4)
 Call ListView1.ColumnHeaders.Add(,,"Memleketi",ListView1.Width
/4)

 ListView1.View = lvwReport
 ListView1.Sorted = True
 ListView1.SortOrder = lvwAscending

 Dim x As ListItem
 Set x = ListView1.ListItems.Add(, , "İhsan KARAGÜLLE")
 x.SubItems(1) = "Bilgi İşlem"
 x.SubItems(2) = "4113142"
 x.SubItems(3) = "Erzurum"
 Set x = ListView1.ListItems.Add(, , "Zeydin PALA")
 x.SubItems(1) = "Bilgi İşlem"
 x.SubItems(2) = "4113318"
 x.SubItems(3) = "Van"
End Sub
```

```
Private Sub ListView1_ColumnClick(ByVal ColumnHeader As
ComctlLib.ColumnHeader)
 If ListView1.SortOrder = lvwAscending Then
  ListView1.SortOrder = lvwDescending
 Else
  ListView1.SortOrder = lvwAscending
 End If
End Sub
```

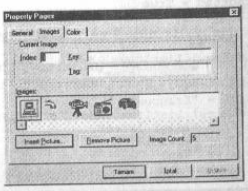
ItemClick(ByVal Item As ListItem)

Listedeki bir eleman tıklandığında ItemClick olayı meydana gelir. Tıklanan elemanın bilgileri de Item parametresi ile öğrenilip-değiştirilebilir.

ÖRNEK: Örnek olarak ListView kontrolünü kullanarak personel bilgilerini tutacak bir örnek yapalım. Örneğimiz için aşağıdaki formu oluşturun.



Programdaki **ImageList** kontrolünü, personelin çalıştığı birime uygun bir resim göstermek için kullanacağız. İşyerinde beş tane birim olduğunu kabul edelim ve bunlara uygun birer resim yükleyelim. **ImageList** kontrolünün **Custom** özelliği ile açılan aşağıdaki pencereden uygun beş tane resim ekleyin.




```
'Örnek:ListView
Private Sub Form_Load()
 'Dört tane kolon ekle
 Call ListView1.ColumnHeaders.Add(,,"Adı soyadı", ListView1.Width
/ 4)
 Call ListView1.ColumnHeaders.Add(,,"Birim", ListView1.Width / 4)
 Call ListView1.ColumnHeaders.Add(,,"Telefonu",ListView1.Width/4)
 Call ListView1.ColumnHeaders.Add(,,"Memleketi",ListView1.Width
/4)
 ListView1.Sorted = True
 ListView1.SortOrder = lvwAscending'Artan sıralama
 'Resimlerin alınacağı kontrolleri belirle
 ListView1.Icons = ImageList1
 ListView1.SmallIcons = ImageList1
 'ComboBoxlara değerleri gir
 Combo1.AddItem "1-Bilgi İşlem"
 Combo1.AddItem "2-Tamir"
 Combo1.AddItem "3-Haber"
 Combo1.AddItem "4-Yayın"
 Combo1.AddItem "5-Ulaşım"
 Combo2.AddItem "Büyük Simge"
 Combo2.AddItem "Küçük Simge"
 Combo2.AddItem "Liste"
 Combo2.AddItem "Ayrıntılar"
 Dim ad, tlf, mem, birim, i
 Dim x As ListItem
 If Dir("Personel.dat") <> "" Then
  Open "Personel.dat" For Input As #1
  While Not EOF(1)
   i = i + 1
   Input #1, ad, birim, tlf, mem
   Combo1.Text = birim
   Set x = ListView1.ListItems.Add(, , ad, Val(birim))
   x.SubItems(1) = birim
   x.SubItems(2) = tlf
   x.SubItems(3) = mem
  Wend
  Close #1
 End If
End Sub

Private Sub Combo2_Click()
 'Görünümü değiştir.
 ListView1.View = Combo2.ListIndex
End Sub
Private Sub Command1_Click()
 Dim x As ListItem
 Set x = ListView1.ListItems.Add(, , Text1, Combo1.ListIndex + 1)
 x.SubItems(1) = Combo1.Text
 x.SubItems(2) = Text2
 x.SubItems(3) = Text3
 Text1 = "": Text2 = "": Text3 = ""
End Sub
```

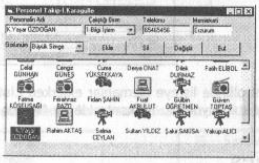


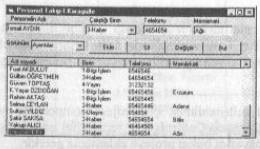
```
Private Sub Command2_Click() 'Seçili elemanı sil
 ListView1.ListItems.Remove ListView1.SelectedItem.Index
End Sub

Private Sub Command3_Click() 'Bul
 Dim x As ListItem, c
 Static b
 If IsEmpty(b) Then b = 0
 Set x = ListView1.FindItem(Text1, lvwText, b + 1)
 If x Is Nothing Then
  c = MsgBox("Böyle bir eleman adına rastlanmadı. Alt kolonlarda
aransınmı", vbYesNo, "Bul")
  If c = vbYes Then
   Set x = ListView1.FindItem(Text1, lvwSubItem, 1)
   If x Is Nothing Then
    c = MsgBox("Alt kolonlardada bulunamadı")
    b = 0
    Exit Sub
   End If
  Else
   b = 0
   Exit Sub
  End If
 End If
 b = x.Index
 x.EnsureVisible'Bulunan eleman görülecek şekilde listeyi kaydır
 x.Selected = True'Seç
 ListView1.SetFocus
End Sub

Private Sub Command4_Click() 'değiştir
 ListView1.SelectedItem.Text = Text1
 ListView1.SelectedItem.Icon = Combo1.ListIndex + 1
 ListView1.SelectedItem.SubItems(1) = Combo1.Text
 ListView1.SelectedItem.SubItems(2) = Text2
 ListView1.SelectedItem.SubItems(3) = Text3
End Sub

Private Sub ListView1_ColumnClick(ByVal ColumnHeader As
MsComctlLib.ColumnHeader)
 'Kolon tıklandığında sıralamayı değiştir.
 If ListView1.SortOrder = lvwAscending Then
  ListView1.SortOrder = lvwDescending
 Else
  ListView1.SortOrder = lvwAscending
 End If
End Sub

Private Sub ListView1_ItemClick(ByVal Item As
MsComctlLib.ListItem)
 'Seçilen elemanın bilgilerini göster
 Text1 = Item.Text
 Combo1.Text = Item.SubItems(1)
```



```
 Text2 = Item.SubItems(2)
 Text3 = Item.SubItems(3)
End Sub

Private Sub Form_Unload(Cancel As Integer)
 Dim i, ad, birim, mem, tlf
 Open "Personel.dat" For Output As #1
 For i = 1 To ListView1.ListItems.Count
  ad = ListView1.ListItems(i).Text
  birim = ListView1.ListItems(i).SubItems(1)
  tlf = ListView1.ListItems(i).SubItems(2)
  mem = ListView1.ListItems(i).SubItems(3)
  Write #1, ad, birim, tlf, mem
 Next
    Close #1
End Sub
```

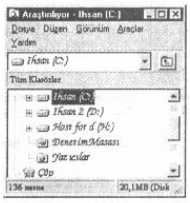
Programın farklı gösterim modlarındaki listeleri aşağıdaki gibi olacaktır.





TreeView*

Windows-95 ile birlikte gelen kontrollerden biri de TreeView kontrolüdür. Bu kontrolleri Windows-95 ve 98 altında bir çok programda görebilirsiniz. Örneğin Windows Gezgini programı bu kontrolü kullanır.



ListView kontrolünde listeye elemanlar eklerken **ListItems** özelliğini alt özellik ve metodlarını kullanıyorduk. Bu kontrolde ise **Nodes** özelliğini kullanacağız.

Properties

ImageList

TreeView kontrolü her bir eleman farklı resimler gösterebilir. ListView kontrolünde olduğu gibi bu kontrolde resimlerinin ImageList kontrolünden alabilir.

```
TreeView1.ImageLis = ImageList1
```

Sorted

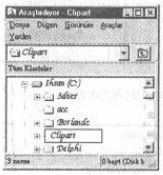
Liste içindeki elemanların sıralanıp sıralanmayacağı bu özelliklerle belirlenir. Sorted özelliğine True verildiğinde liste sırlanacaktır. Listedeki her bir düğümün alt elemanlarının sıralanıp sıralanmayacağı bu özellikle belirlenir.

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Windows Common Controls 6.0" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCOMCTL.OCX dosyasını bulup projeye eklemeniz gerekir.

LabelEdit

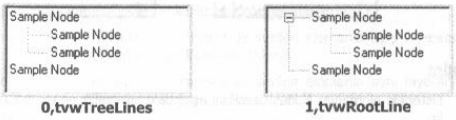
0, lvwAutomatic: Herhangi bir koda gerek olmadan kullanıcı ağaç içindeki etiketleri aralıklı iki mouse tıklaması ile düzenleyebilir.

1, lvwManual: Kullanıcı labelleri otomatik olarak değiştiremez. Değiştirebilmesi için StartLabelEdit metodunun çağrılması gerekir.



LineStyle

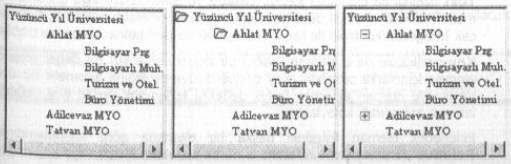
TreeView kontrolündeki elemanların nasıl gösterileceğini belirler.



0,tvwTreeLines **1,tvwRootLine**

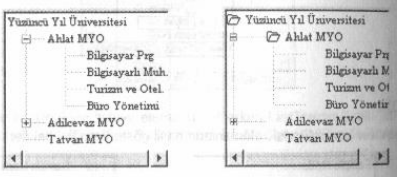
Style

0:Sadece metin 1:Resimler ve metin 2:+/- işaretleri ve metin



3:+/-işaretleri,resim,metin 4:Çizgiler ve metin 5:Çizgiler,resimler,metin

6:+/-işaretleri,çizgiler,metin 7:+/- işaretleri, çizgiler, resimler,metin



Nodes

Listedeki elemanlar Nodes özelliğini aşağıdaki alt özellik ve metodları ile yönetir.

Nodes.Add (relative, relationship, key, text, image, selectedimage)

Listeye eleman eklemek için bu metod kullanılır.

Text özelliği ile elemanın içeriği, **Image** ve **SelectedImage** özellikleri ile elemanın normal ve seçili durumları için gösterilecek resim belirlenir. Bu resim ancak ImageList kontrolü ile tanımlanmış resimlerden birinin numarası olabilir.

Key özelliği ile de o elaman tanıtıcı bir değer verilebilir. Bu değer listedeki her eleman için farklı olmalıdır. Key özelliği Index özelliğine alternatif bir özelliktir. Ve listeye eleman eklerken Index değerini kullanmak yerine Key değerini kullanmak anlaşılırlığı kolaylaştırır.

Eklenecek eleman listedeki başka bir elemana göre konumlandırılacaksa **Relative** parametresine referans alınacak elemanın indexi veya key değeri ver-

lir. Referansın nasıl kullanılacağı ise RelationShip parametresi ile belirlenir. Relative parametresi verilmezse eleman en üst hiyerarşide sona eklenir.

```
Call TreeView1.Nodes.Add(,,"YYÜni","Yüzüncü Yıl Üniversitesi",1,
1)
Call TreeView1.Nodes.Add(,,"UÜni","Uludağ Üniversitesi",1,1)
Call TreeView1.Nodes.Add(,,"İÜni","İstanbul Üniversitesi",1,1)
```

Burada Relative parametresini vermediğimiz için bütün elemanlar kök dizine eklenmektedir.

Bunlara ait **Key** değerlerini kullanarak bu elemanlara kolayca alt elemanlar ekleyebiliriz.

RelationShip parametresine verilebilecek değerler şunlardır.

0,tvwFirst: Relative parametresi ile verilen elemanla aynı hiyerarşide olacak şekilde ilk sıraya eklenir.

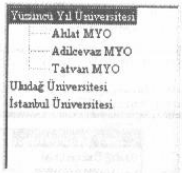
1,tvwLast : Relative parametresi ile verilen elemanla aynı hiyerarşide olacak şekilde son sıraya eklenir.

2,tvwNext: Relative parametresi ile verilen elemanla aynı hiyerarşide olacak şekilde onun altına eklenir.

3,tvwPrevious: Relative parametresi ile verilen elemanla aynı hiyerarşide olacak şekilde onun üstüne eklenir.

4,tvwChild: Relative parametresi ile verilen elemanın alt elemanı olacak şekilde eklenir.

```
Call TreeView1.Nodes.Add("YYÜni",tvwChild,"Amyo","Ahlat MYO",1,1)
Call TreeView1.Nodes.Add("YYÜni",tvwChild,"Admyo","AdilcevazMYO")
Call TreeView1.Nodes.Add("YYÜni",tvwChild,"Tmyo","Tatvan MYO")
```



Burada görüldüğü gibi daha önce Key özelliğine verdiğimiz değeri kullanarak ona alt elemanlar ekleyebiliyoruz.

```
Call TreeView1.Nodes.Add("Amyo",tvwChild,"AmyoBp","Bilgisayar
Prg")
Call TreeView1.Nodes.Add("Amyo",tvwChild,"AmyoBm","Bilgisayarlı
Muh.")
Call TreeView1.Nodes.Add("Amyo",tvwChild,"AmyoTo","Turizm ve
Otel.")
Call TreeView1.Nodes.Add("Amyo",tvwChild,"AmyoBy","Büro
Yönetimi")
Call TreeView1.Nodes.Add("Admyo",tvwChild,"AdmyoBp","Bilgisayar
Prg")
Call TreeView1.Nodes.Add("Admyo",tvwChild,"AdmyoIş","İşletme")
Call TreeView1.Nodes.Add("Admyo",tvwChild,"AdmyoTo","Turizm ve
Otel.")
Call TreeView1.Nodes.Add("Admyo",tvwChild,"AdmyoBy","Büro
Yönetimi")
```

Bu özellikten geriye Node tipinden tanımlanmış bir değer döner. Bu değer bir değişkene alınarak, diğer işlemlerde kullanılabileceği gibi Call ile bir değişkene aktarmaya gerek kalmadan da çağrılabilir.

Nodes.Remove Index

Indexi veya keyi verilen elemanı listeden siler. İlk elemanın Index numarası 1'dir. Seçili elemanı silmek için ise SelectedItem özelliğinin Index özelliği kullanılabilir.

```
TreeView1.Nodes.Remove 1'İlk elemanı sil
TreeView1.Nodes.Remove TreeView1.SelectedItem.Index 'Seçili
elemanı sil
```

Nodes.Clear

Listedeki bütün elemanları siler.

Nodes.Count

Listedeki bütün elemanların sayısını verir.

Nodes.EnsureVisible(Index)

Index numaralı eleman listede görülecek şekilde listenin alt elemanları açılır.

```
TreeView1.Nodes("AMyo").EnsureVisible'Amyo key'li eleman
görülecek şekilde listeyi aç
```

Nodes(i).Text, Nodes(i).Key, Nodes(i).Image, Nodes(i).SelectedImage

Eleman eklenirken elemana verilen text ve key özellikleri daha sonra bu özelliklerle öğrenilip değiştirebilir. Ayrıca text özelliğini kullanıcıda değiştirebilir.

Image ve SelectedImage özellikleri ile de o eleman için gösterilen normal ve seçili durumundaki resim değiştirilebilir. Yine bu özelliklere verilecek değer, ImageList kontrolü ile yüklenmiş resimlerden birinin numarası olmalıdır.

Nodes(i).Selected

Listedeki i numaralı elemanın seçili olup olmadığını öğrenmek ve seçmek için bu özellik kullanılır.

```
TreeView1.Nodes(1).Selected= True
```

veya key kullanarak

```
TreeView1.Nodes("YYÜni").Selected= True
```

Nodes(i).Children

Bir elemanın alt elemanlarının sayısı bu özellikle öğrenilebilir.

```
MsgBox TreeView1.Nodes("YYÜni").Children
```

Yukarıdaki örneğimize göre bu elamanın üç tane alt elemanı vardır.


```
MsgBox TreeView1.SelectedItem.Children
```

gibi bir satır ilede seçili elemanın alt elemanlarının sayısı öğrenilebilir.

Nodes(i).Child

Listedeki bir elemanın alt elemanlarına ulaşmak için bu özellik kullanılabilir.

TreeView1.SelectedItem.Text ile seçili elamanın textine

TreeView1.SelectedItem.Child.Text ilede seçili elamanın alt elemanının textine ulaşılabilir.

Buradaki Child özelliği Node tipinden bir özellik olduğu için Node tipine uygulanabilen bütün özellikler bu özelliğe de uygulanabilir.

```
MsgBox TreeView1.SelectedItem.Text
MsgBox TreeView1.SelectedItem.Child.Text
MsgBox TreeView1.SelectedItem.Child.Child.Text
MsgBox TreeView1.SelectedItem.Child.Child.Child.Text
```

Yukarıdaki gibi iç içe kullanılarak birden fazla alt hiyerarşiye ulaşılabilir. Bu yapısından dolayı da bir ağacın sonuna kadar Child özelliği kullanılarak bir döngü ile ulaşılabilir.

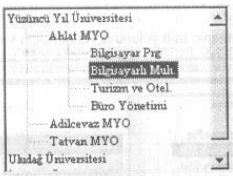
```
Dim n As Node
    Set n = TreeView1.SelectedItem
    Do
        List1.AddItem n.Child.Text
        Set n = n.Child
    Loop While n.Children > 0
```

Yukarıdaki gibi bir kod ile seçili elemanın altındaki bütün dalların ilk elemanları öğrenilebilir.

Nodes(i).FirstSibling, LastSibling, Previous, Parent, Next, Root

Bir düğümün alt elemanlarına Child özelliği ile ulaşabiliyorduk. Aynı şekilde bir elemanla aynı sıradaki diğer elemanları, onun altında bulunduğu elemanı ve kök dizindeki elemanı öğrenmek mümkündür.

Aşağıdaki listede "AmyoBm" key değeri ile eklediğimiz "Bilgisayar Muh." elemanını referans olarak alırsak.



TreeView1.Nodes("AmyoBm").Root satırı bize o elamanın bulunduğu kökün, yani "Yüzüncü Yıl Üniversitesi" elemanının referansını verir.

TreeView1.Nodes("AmyoBm").Parent satırı bize "Bilgisayarlı Muh." elemanının alt elemanı olduğu "Ahlat MYO" elemanının referansını verir.

TreeView1.Nodes("AmyoBm").Next satırı bize "Bilgisayarlı Muh." Elemanından sonraki elemanın yani "Turizm ve Otel." elemanının referansını verir.

TreeView1.Nodes("AmyoBm").Previous satırı bize "Bilgisayarlı Muh." elemanından önceki elemanın yani "Bilgisayar Prg." elemanının referansını verir.

TreeView1.Nodes("AmyoBm").FirstSibling satırı bize "Bilgisayarlı Muh." Elemanıyla aynı gruptaki ilk elemanın yani "Bilgisayar Prg." elemanının referansını verir.

TreeView1.Nodes("AmyoBm").LastSibling satırı bize "Bilgisayarlı Muh." Elemanıyla aynı gruptaki son elemanın yani "Büro Yönetimi." elemanının referansını verir.

Child özelliğinde olduğu gibi bu özelliklerde Node tipinden değer döndürdüğü için Nodes özelliğine uygulanan bütün işlemler bunlar içinde uygulanabilir.

TreeView1.Nodes("AmyoBm").Root.Text = "YY Üniversitesi" satırı AmyoBp elemanının kökünde bulunan "Yüzüncü Yıl Üniversitesi" elemanının textini "YY Üniversitesi" olarak değiştirir.

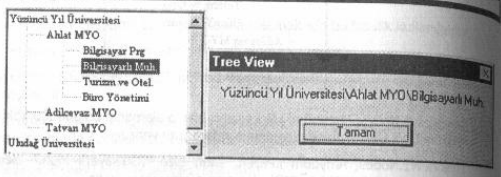
Örneğin seçili elemanla aynı seviyedeki bütün elemanların değerlerini toplamak için aşağıdaki gibi bir döngü kurulabilir.

```
Dim n As Node, i, t
Set n = TreeView1.SelectedItem.FirstSibling
While n.Index < TreeView1.SelectedItem.LastSibling.Index
    t = t + n.Text
    Set n = n.Next
Wend
t = t + n.Text
MsgBox t
```

Nodes(i).FullPath

Listedeki bir elemanın tam yolunu öğrenmek için kullanılır.

```
MsgBox TreeView1.Nodes("AmyoBm").FullPath
```



Nodes(i).Expanded

Listedeki index numaralı elemanın alt elemanları varsa onların açık olup olmadığı bu özellikle öğrenilip değiştirilebilir.

```
Dim i
For i = 1 to TreeView1.Nodes.Count
 TreeView1.Nodes(i).Expanded = True
Next
```

Yukarıdaki gibi bir kod ile listedeki bütün elemanların açılması sağlanabilir.

SelectedItem

Seçili elemana ait özelliklere ulaşmak için bu özellikten faydalanılır. Nodes özelliği gibi kullanılır.

Örneğin seçili elemanın textini değiştirmek için:

```
ListView1.SelectedItem.Text = "Merhaba"
```

Methods

GetVisibleCount

ListView kontrolünün şu anki boyutları ile ekranda aynı anda gösterebileceği eleman sayısını öğrenmek için kullanılır.

HitTest(x As Single, y As Single)

Bu metod ile verilen x,y koordinatlarındaki eleman öğrenilir. Çoğunlukla Drag-Drop işlemlerinde farenin bırakıldığı noktadaki elemanı öğrenmek için kullanılır.

Events

BeforeLabelEdit(cancel As Integer)

Kullanıcı listedeki elemanların etiketlerini aralıklı olarak iki defa tıklayarak değiştirebilir. Kullanıcı bir etiketi değiştirmek üzereyken bu olay meydana gelir. Değişime izin verilmeyecekse Cancel parametresine True değeri atanabilir.

Bir eleman kullanıcı tarafından değiştirilirken önce seçileceği için hangi elemanın değiştirildiği SelectedItem özelliği ile öğrenilir.

AfterLabelEdit(cancel As Integer, newstring As String)

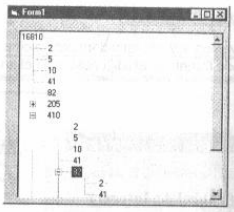
Kullanıcı bir etiketi değiştirdikten sonra bu olay meydana gelir. NewString parametresi ile kullanıcının girdiği yeni metni öğrenebilir ve Cancel özelliğine True atayarak ta değişimi iptal edebilirsiniz.

NodeClick(ByVal Node As ComctlLib.Node)

Listedeki bir eleman tıklandığında bu olay meydana gelir. Tıklanan elemanın özellikleri daha önce gördüğümüz Node tipinden tanımlanmış Node parametresi ile öğrenilir.

ÖRNEK: Örnek olarak kullanıcının girdiği sayının bütün çarpanlarını ve ona ait çarpanlarında çarpanlarını asal sayılara kadar gösterebilecek bir kodu ListView kontrolü ile yazalım.

İlk etap da kullanıcıya bir sayı sorarak başlayacağız. Daha sonra kullanıcı bu sayıyı çift tıkladığında ona ait çarpanları alt eleman olarak ekleyeceğiz. Bu alt elemanlardan biri çift tıklandığında yine ona ait alt elemanlar da listeye eklenecektir.



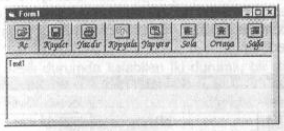

```
'Örnek:TreeView
Private Sub Form_Load()
 Dim x
 x = Val(InputBox("Bir sayı girin", "Sayı girişi", 16810))
 Call TreeView1.Nodes.Add(, , , x)
End Sub

Private Sub TreeView1_NodeClick(ByVal Node As MsComctlLib.Node)
 Dim i
 'daha önce eklenmişse çık
 If Node.Children > 0 Then Exit Sub
 For i = 2 To Node.Text - 1
  If (Node.Text Mod i) = 0 Then
   'Tam olarak bölünebiliyorsa alt eleman olarak ekle
   Call TreeView1.Nodes.Add(Node.Index, tvwChild, , i)
   DoEvents
  End If
 Next
End Sub
```

Expand, Collapse(ByVal Node As ComctlLib.Node)

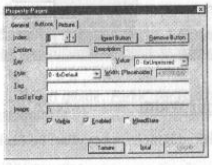
Listedeki bir elemanın alt elemanları açıldığında Expand olayı, kapatıldığında ise Collapse olayı meydana gelir.

ToolBar (Araç Çubuğu)*

Windows-95 ile birlikte gelen ve 98'de de kullanılan kontrollerden biri de ToolBar kontrolüdür. Programlardaki araç çubuklarını oluşturmak için kullanılır ve bir çok güzel özelliği vardır.



Tasarım zamanında Custom özelliği ile açılan aşağıdaki pencereden araç çubuğuna kolayca yeni düğmeler ekleyebilirsiniz.



Toolbar kontrolünde yeni düğmeler eklerken Buttons özelliğinin alt özellik ve metodlarını kullanacağız.

Properties

ImageList

ToolBar kontrolündeki düğmeler için kullanılacak resimler de diğer standart ActiveX kontrollerinde olduğu gibi ImageList kontrolü ile belirlenir.

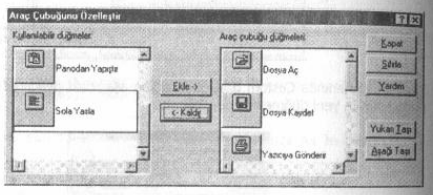
*Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Windows Common Controls 6.0" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCOMCTL.OCX dosyasını bulup projeye eklemeniz gerekir.

Form üzerine koyacağınız bir ImageList kontrolü ile kullanılacak resimleri belirledikten sonra bu özelliğe o kontrolün ismini vermeniz gerekir.

```
ToolBar1.ImageList=ImageList1
```

AllowCustomize

Bu özellikle kullanıcının araç düğmelerini özelleştirip özelleştiremeyeceği belirlenebilir. Eğer bu özelliğin değeri True ise kullanıcı araç çubuğunu çift tıkladığında aşağıdaki pencere açılacak ve kullanıcıya istediği düğmeleri ekleyip kaldırabilme özelliği verecektir.



ButtonHeight, ButtonWidth

Araç çubuğundaki düğmelerin boyutları bu özelliklerle belirlenir.

Wrappable

Bu özellik True ise, araç çubuğundaki düğmeler forma sığmadığında araç düğmeleri bir alttaki satırdan devam eder.

Buttons

Çalışma zamanında bu özelliğe ait aşağıdaki alt özellik ve metodlar kullanılarak toolbar içindeki düğmeler yönetilir.

Buttons.Add(index, key, caption, style, image)

Araç çubuğuna yeni düğmeler eklemek için kullanılır.

Index parametresi ile düğmenin nereye ekleneceği belirlenir. Verilmezse sona eklenecektir. **Key** parametresi ise diğer standart ActiveX kontrollerinde

(MSCOMCTL.OCX dosyasındaki diğer kontroller) olduğu gibi Index özelliğine alternatif erişim sağlayan bir özelliktir.

Caption parametresi ile komut düğmesinin metni, **Image** parametresi ilede bu düğme için kullanılacak resim belirlenir. Bu resim ImageList kontrolü ile yüklenmiş olan resimlerden birinin numarası olmalıdır.

Style parametresi ile düğmenin şekli belirlenir. Bu parametreye verilebilecek değerler şunlardır.

0,tbrDefault:	Normal bir komut düğmesi oluşturur. Aç, Kaydet, Yazdır gibi anlık komutlar için kullanılır.
1,tbrCheck:	Basılı durumda kalabilen iki durumlu bir düğme oluşturur. Kalın, Eğik, Altıçizili gibi iki durumu olan düğmeler için kullanılır. Bu düğmelere basıldığında ikinci kez basılana kadar basılı kalır.
2,tbrButtonGroup:	Grup olarak çalışabilen düğme oluşturur. Bir grupta ancak bir düğme basılı olabilir. Sola, Sağa, Ortaya, İki yana yasla gibi aynı anda sadece bir durumda bulunabilecek işlemler için kullanılır.
3,tbrSeparator:	Aslında bu bir düğme değildir. Düğmeler arasına boşluk bırakmak için kullanılır. 8 pixel genişliğinde bir boşluk oluşturur.
4,tbrPlaceholder:	Buda 3 değeri gibidir. Farklı olarak bırakılacak boşluk genişliği ayarlanabilir.

Buttons.Remove index

Index'i veya Key'i verilen düğmeyi siler.

Buttons.Clear

Araç çubuğundaki bütün düğmeleri siler.

Buttons.Count

Araç çubuğundaki düğme sayısını verir.

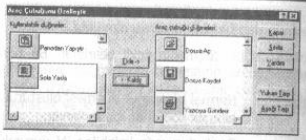
Buttons(i).Caption

Komut düğmesinin üzerindeki metni öğrenip değiştirmek için kullanılır.

Buttons(i).Description

Komut düğmesine açıklayıcı bir bilgi eklemeye yarar. Bu bilgiyi kullanıcı araç çubuğunu özelleştirirken görecektir.


```
Toolbar1.Buttons(1).Description= "Dosyaya kaydet"
Toolbar1.Buttons(2).Description= "Dosya Aç"
Toolbar1.Buttons(3).Description= "Aktif belgeyi yazdır"
```



Buttons(i).Image, Buttons(i).Key, Buttons(i).Style

Komut düğmesi eklenirken belirlenen Key,Style ve Image özellikleri daha sonra bu özelliklerle öğrenilip değiştirilebilir.

Buttons(i).Value

Araç düğmesi iki durumlu ise, yani basılı olma durumu da söz konusu ise bu özellikle düğmenin basılı olu olmadığı öğrenilebilir ve değiştirilebilir.

0,tbrPressed:	Basılı değil
1,tbrUnpressed:	Basılı

Buttons(i).MixedState

Araç düğmesinin temsil ettiği değerin belirsizliğini göstermek için bu değere true atanır. Örneğin bir düğme yazının Bold özelliğini temsil ediyorsa ve seçilen yazının bir kısmı kalın, bir kısmı değilse bu özelliğe True atanarak düğmenin şu anda belirsizliği bildirilebilir. CheckBox kontrolünde gördüğümüz Grayed değeri gibidir.

Methods

Customize

Bu metod kullanılarak araç çubuğunu özelleştirme penceresi aktif hale getirilebilir. AllowCustomize özelliği True iken kullanıcı araç çubuğunu çift tıklayarak ta bu pencereyi aktif hale getirilebilir.

SaveToolbar(key As String, subkey As String, value As String)

Toolbar kontrolünü kullanıcı özelleştirebiliyordu. Bu metodu kullanarak kullanıcının özelleştirdiği araç çubuğunun Registry dosyasına kaydedilmesini sağlayabilir-

siniz. Hatta SubKey anahtarına farklı isimler vererek programınızı kullanan her kullanıcının kendi araç çubuğunu oluşturmasını sağlayabilirsiniz.

Bu metoddaki üç parametreyi belirlemekte size kalmıştır. Genellikle ilk parametreye programınızın ismini, ikinci parametreye kullanıcı ismini, üçüncü parametreye de, programda birden fazla araç çubuğu olabileceği için araç çubuğunun ismini verebilirsiniz.

```
Ad=MsgBox("İsminiz")
Toolbar1.SaveToolbar "Bizim Program", Ad, "Toolbar1"
Toolbar2.SaveToolbar "Bizim Program", Ad, "Toolbar2"
MsgBox("Araç çubuğu sizin adınıza kaydedildi")
```

RestoreToolbar(key As String, subkey As String, value As String)

SaveToolbar metodu ile Registry dosyasına kaydedilen araç çubuğu bilgileri bu metod kullanılarak ta geri yüklenebilir.

```
Ad=MsgBox("İsminiz")
Toolbar1.RestoreToolbar "Bizim Program", Ad, "Toolbar1"
Toolbar2.RestoreToolbar "Bizim Program", Ad, "Toolbar2"
MsgBox("Size ait araç çubuğu geri yüklendi")
```

Events

Change ( )

Kullanıcı araç çubuklarını Customize Toolbar penceresi ile değiştirdiğinde Change olayı meydana gelir. Yapılan değişimleri SaveToolbar metodu ile kaydetmek için bu olay kullanılabilir.

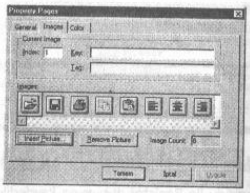
ButtonClick(ByVal button As Button)

Araç çubuğundaki bir düğme tıklandığında bu olay meydana gelir. Tıklanan düğmenin özellikleri button parametresi ile öğrenilip değiştirilebilir.

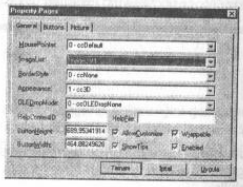
ÖRNEK: Toolbar'a düğmeler eklemek için çoğunlukla kod eklemeniz gerekmez. Her şeyi tasarım zamanında Customize özelliği ile açılan pencereden yapabilirsiniz. Sadece Change ve Click olaylarına kod yazarak düğmelerin görevlerini yapacak kodları yazabilirsiniz.

Yapmanız gerekenleri şöyle özetleyebiliriz.

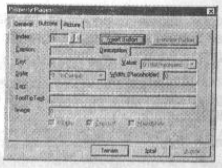
- Önce komut düğmelerinde kullanacağınız resimler için bir ImageList kontrolü yerleştirmek ve Custom özelliği ile açılan aşağıdaki pencereden gerekli resimleri yüklemek.



- Daha sonra ToolBar kontrolünün Custom özelliği ile açılan aşağıdaki pencereden ImageList özelliğine ImageList1 değerini vererek oradaki resimlerin kullanılabilmesini sağlamak.

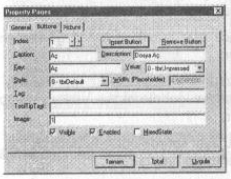


- Yukarıdaki pencerenin Buttons tabına geçip aşağıdaki pencereye ulaşıp yeni düğmeler oluşturmak.



- Burada Insert Button düğmesini kullanarak yeni düğmeler oluşturabilirsiniz. Oluşturduğunuz düğmelerin Caption, Description, Key ve Image özelliklerine uygun değerleri verin.

Sık kullanılmayan araç düğmelerini de ekleyip Visible özelliğini kaldırarak bunları, isterse kullanıcının Customize Toolbar penceresi ile gösterip gizleyebilmesini sağlayabilirsiniz.



Key özelliğine vereceğimiz değeri program içinde kullanacağız. Index parametresini kullanmaktansa Key parametresini kullanmak program kodunun daha anlaşılır olmasını sağlayacaktır.

```
Private Sub Form_Load()
 Call Toolbar1.RestoreToolbar("Prg adı","Kullanıcı adı","Toolbar
adı")
End Sub

Private Sub Toolbar1_Change()
 Call Toolbar1.SaveToolbar("Prg adı","Kullanıcı adı","Toolbar
adı")
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As
MsComctlLib.Button)
 Select Case Button.Key
  Case "Aç":  'Buraya açmak için gerekli kod
  Case "Kaydet":  'Buraya kaydetmek için gerekli kod
  Case "Yazdır":  'Buraya açmak için gerekli kod
  Case "Sola":
   If Button.Value = 1 Then
    'düğme basılı gerekli kod
   End If
  ..............
  ..............
 End Select
End Sub
```

Yukarıdaki gibi bir kod ile de basılan düğmeleri yönetmek mümkündür.

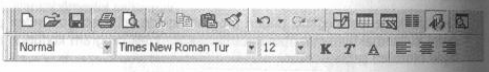
CoolBar*

Bir çok modern program, araç çubuklarını kullanıcının istediği gibi düzenlemesine imkan verir. Kullanıcı istediği araç çubuğunu alıp diğer çubukların istediği yerine bırakabilir. İşte bu işlemi hiç koda gerek kalmadan yapacak kontrol **CoolBar** kontrolüdür.

Bu işlemi sadece araç çubukları için değil hemen hemen bütün kontroller için yapabilmektedir.

Bu kontrol üzerine yerleştirdiğiniz kontrollerin genişlikleri otomatik olarak **CoolBar** genişliğine getirilecek ve yerleştirdiğiniz diğer kontroller de alt alta yerleştirilecektir.

Bir çok kontrolü aynı satırda bulundurabilmek için onları gruplama yapabilen bir kontrol içine yerleştirmeniz gerekir.

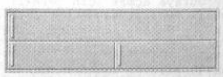


Coolbar kontrolünün kullanımı için genellikle kod yazmayacaksınız. Çünkü gerekli işlemleri kendisi yapmaktadır. Tasarım aşamasında CoolBar üzerinde bulunacak bant sayısını ve içinde bulunacak kontrolleri belirlemeniz yeterlidir.

CoolBar kontrolünün tasarımı için aşağıdaki adımları atabilirsiniz:

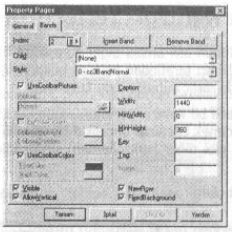
Bant Ekleme, Silme

CoolBar kontrolünü form üzerine yerleştirdiğinizde üç adet bant bulunduğunu göreceksiniz.



Bu sayıyı artırabileceğiniz gibi azaltabilirsinizde. **Custom** özelliğini çift tıklayarak aşağıdaki pencereyi açın ve açılan aşağıdaki pencerenin **Bands** kısmına geçin.

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Windows Common Controls-3 6.0" seçeneğini işaretlemeniz veya Browse düğmesi ile COMCT332.OCX dosyasını bulup projeye eklemeniz gerekir.



Yukarıdaki pencerede bulunan **Insert Band** düğmesi ile yeni bantlar ekleyebileceğiniz gibi, **RemoveBand** düğmesi ile de aktif bandı silebilirsiniz.

Bant Seçme

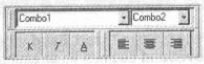
Custom özelliği ile açılan pencerenin **Bands** kısmında aktif banda ait özellikler gösterilmektedir. Aktif bandı değiştirmek ve ona ait özellikleri pencerede görebilmek için penceredeki **Index** kutusunun yanındaki okları kullanabilirsiniz.

Bantlara Kontrol Yerleştirme

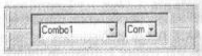
Coolbar kontrolünün asıl amacı diğer kontrolleri bir araya getirmek ve bunların yerleşiminin kullanıcı tarafından kolayca belirlenebilmesini sağlamaktır. Bu kontrol çoğunlukla da araç çubuklarında kullanılır.

Her banda sadece bir kontrol yerleştirilebilmektedir. Eğer bir banda birden fazla kontrol yerleştirilmek istenirse öncelikle bir Picture kontrolü (veya gruplama yapabilen, frame gibi diğer kontrollerden) yerleştirilmeli ve diğerleri bunun içine konmalıdır.

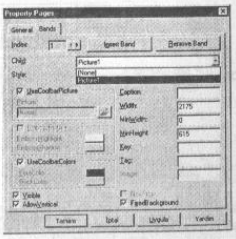
ÖRNEK: Örnek olarak aşağıdaki CoolBar'ı hazırlayalım:



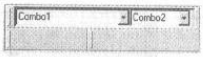
- Öncelikle CoolBar içine bir PictureBox yerleştirin ve onun içine de iki tane Combo kutusunu yerleştirin.



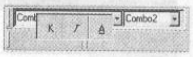
- **CoolBar** kontrolünün **Custom** özelliği ile açılan pencerenin **Bands** kısmına geçin ve **Child** kutusundan **Picture1** elemanını seçin.



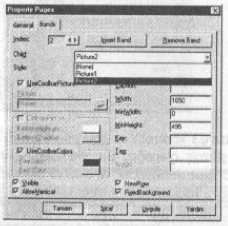
Böylece **Picture1** kontrolü ve dolayısıyla onun içindeki kontrolleri 1 numaralı Banda yerleştirmiş olacağız. Bu işlemden sonra Picture1 kontrolü otomatik olarak Band1 içine yerleşecek ve onunla birlikte hareket edecektir.



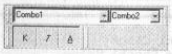
- İkinci bandımız için de CoolBar içine bir PictureBox daha yerleştirin ve içine üç tane CheckBox yerleştirin. Sırasıyla Caption özelliklerine K, T, A değerlerini verin ve her üçünün de Style özelliklerine 1 verin. Böylece bu CheckBox'lar birer düğme haline dönüşecektir.



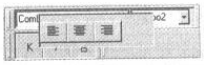
- **CoolBar** kontrolünün **Custom** özelliği ile açılan pencerenin **Bands** kısmına geçin, iki numaralı bandı aktif hale getirin ve **Child** kutusundan **Picture2** elemanını seçin.



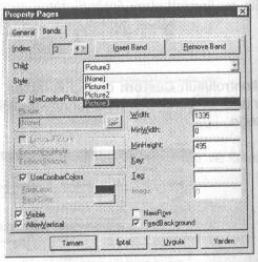
- Böylece **Picture2** kontrolü ve dolayısıyla onun içindeki kontrolleri 2 numaralı Banda yerleştirmiş olacağız. Bu işlemden sonra Picture2 kontrolü otomatik olarak Band2 içine yerleşecek ve onunla birlikte hareket edecektir.



- Üçüncü bandımız için de CoolBar içine bir PictureBox daha yerleştirin ve içine üç tane OptionButton yerleştirin. Caption özelliklerini silin çünkü bunlar için birer resim yerleştireceğiz. Her üçünün de Style özelliklerine 1 verin. Ve sırayla Picture özelliklerine COMMON\GRAPHICS\BITMAPS\TLBR_W95 dizini altında bulunan LFT.BMP, CTR.BMP ve RT.BMP dosyalarını verin. Böylece bu OptionButton'lar birer düğme haline dönüşecektir.



- **CoolBar** kontrolünün **Custom** özelliği ile açılan pencerenin **Bands** kısmına geçin, üç numaralı bandı aktif hale getirin ve **Child** kutusundan **Picture3** elemanını seçin.



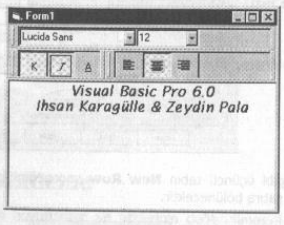
- Böylece **Picture3** kontrolü ve dolayısıyla onun içindeki kontrolleri 3 numaralı Banda yerleştirmiş olacağız. Bu işlemden sonra Picture3 kontrolü otomatik olarak Band3 içine yerleşecek ve onunla birlikte hareket edecektir.

Program için gerekli kodu da yazalım. Önce forma bir Text kutusu yerleştirin ve bu düğmelerle Text kutusunun yazıtipi özelliklerini değiştirelim.

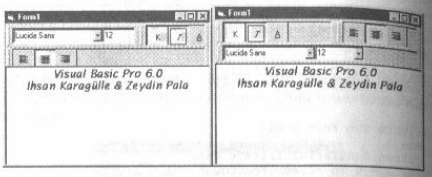
```
'ÖRNEK: CoolBar
Private Sub Check1_Click()
 Text1.FontBold = Check1.Value
End Sub
Private Sub Check2_Click()
 Text1.FontItalic = Check2.Value
End Sub

Private Sub Check3_Click()
 Text1.FontUnderline = Check3.Value
End Sub
```



```
Private Sub Combo1_Click()
 Text1.FontName = Combo1.Text
End Sub

Private Sub Combo2_Click()
 Text1.FontSize = Combo2.Text
End Sub

Private Sub Form_Load()
 Dim i
 'ekran fontlarını listeye ekle
 For i = 0 To Screen.FontCount - 1
  Combo1.AddItem Screen.Fonts(i)
 Next
 'fon boyutlarını listeye ekle
 For i = 8 To 100 Step 4
  Combo2.AddItem i
 Next
End Sub

Private Sub Option1_Click()
 Text1.Alignment = 0
End Sub

Private Sub Option2_Click()
 Text1.Alignment = 2
End Sub

Private Sub Option3_Click()
 Text1.Alignment = 1
End Sub
```

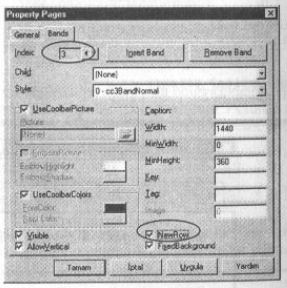


Programda kullanıcı araç çubuklarının yerleşimini istediği gibi değiştirebilir. Bantların başındaki çizgiden tutup istediği noktaya sürükleyerek bantların yeri kolayca değiştirilebilir.



Birden fazla satıra bölme

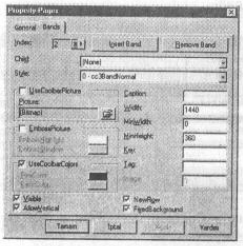
Form üzerine bir CoolBar yerleştirdiğinizde iki satıra bölünmüş olarak üç adet bant hazır olarak yerleştirilmekteydi. Daha fazla sayıda bandınız varsa satır sayısını da artırmak isteyebilirsiniz. Bu durumda aşağıdaki pencereden o tabın **New Row** seçeneğini işaretleyin.



Yukarıdaki gibi üçüncü tabın **New Row** seçeneğini işaretlediğimizde CoolBar kontrolü üç satıra bölünecektir.

Art alana Resim Yerleştirme ve Metin Yazma

CoolBar üzerine eklediğiniz her bandın zeminine farklı resimler verebileceğiniz gibi yazılar da yerleştirebilirsiniz.



Zeminine resim eklediğiniz bandı seçtikten sonra penceredeki **UseCoolbarPicture** seçeneğinin işaretini kaldırın ve hemen altındaki düğmeye basarak açılan pencereden zemine yerleştirmek istediğiniz resimi seçin. Bu resim belirlediğiniz bant üzerine döşenecektir.

Yazı yazmak için ise penceredeki **Caption** kutusunu kullanabilirsiniz. Bu yazının zemin ve yazı rengini değiştirmek için ise penceredeki **UseCoolbarColors** seçeneğinin işaretini kaldırdıktan sonra altındaki **ForeColor** ve **BackColor** seçeneklerini kullanabilirsiniz.

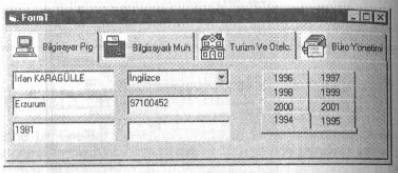
Böylece aşağıdaki gibi her banda farklı resim ve yazılar yerleştirebilirsiniz.

Bantları Sabitleme

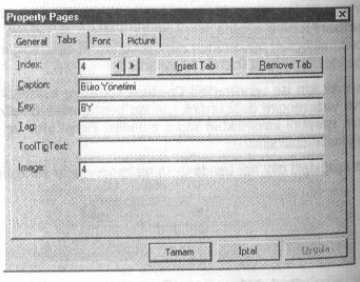
Eğer bazı bantların boyutlarının ve yerlerinin değiştirilmesini istemiyorsanız o bandı aktif hale getirdikten sonra penceredeki **Style** kutusundan **cc3BandFixedSize** elamanını seçin.

TabStrip*

Windows-95 ile birlikte gelen ve 98'de de kullanılan kontrollerden biri de TabStrip kontrolüdür.



Tasarım zamanında **Custom** özelliği ile açılan aşağıdaki pencereden gerekli tabları kolayca ekleyebilirsiniz.



*Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Windows Common Controls 6.0" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCOMCTL.OCX dosyasını bulup projeye eklemeniz gerekir.

Properties

ImageList

TabStrip kontrolündeki tablar için kullanılacak resimler de diğer standart ActiveX kontrollerinde olduğu gibi ImageList kontrolü ile belirlenir.

Form üzerine koyacağınız bir ImageList kontrolü ile kullanılacak resimleri belirledikten sonra bu özelliğe o kontrolün ismini vermeniz gerekir.

```
TabStrip1.ImageList=ImageList1
```

Tabs

Çalışma zamanında bu özelliğe ait aşağıdaki alt özellik ve metodlar kullanılarak TabStrip içindeki tablar yönetilir.

Tabs.Add (index, key, caption, image)

Yeni tablar eklemek için kullanılır.

Index parametresi ile Tabım nereye ekleneceği belirlenir. Verilmezse sona eklenecektir. **Key** parametresi ise diğer standart ActiveX kontrollerinde (MSCOMCTL.OCX dosyasındaki diğer kontroller) olduğu gibi Index özelliğine alternatif erişim sağlayan bir özelliktir.

Caption parametresi ile Tab metni, **Image** parametresi ilede bu Tab için kullanılacak resim belirlenir. Bu resim ImageList kontrolü ile yüklenmiş olan resimlerden birinin numarası olmalıdır.

```
TabStrip1.Tabs.Add  ( , "BP", "Bilgisayar Prg",1)
TabStrip1.Tabs.Add  ( , "BM", "Bilgisayarlı Muh",2)
TabStrip1.Tabs.Add  ( , "TO", "Turizm ve Otel",3)
TabStrip1.Tabs.Add  ( , "BY", "Büro Yönetimi",4)
```

Tabs.Remove index

İndexi veya Keyi verilen tabı siler.

Tabs.Clear

Bütün tabları siler.

Tabs.Count

Araç çubuğundaki düğme sayısını verir.

Tabs(i).Caption

Tab üzerindeki metni öğrenip değiştirmek için kullanılır.

Tabs(i).Image, Tabs(i).Key

Tab eklenirken belirlenen Key ve Image özellikleri daha sonra bu özelliklerle öğ renilip değiştirilebilir.

Tabs(i). Selected

Tab seçili ise bu özelliğin değeri True'dir. Bir tabın aktif olup olmadığını belirlemek için kullanılır.

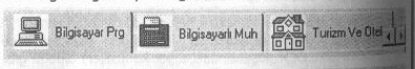
SlectedItem

Seçili tab üzerindeki işlemler bu özellikle yapılabilir.

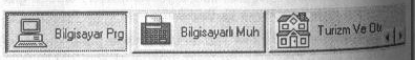
Style

Kontrollerin komut düğmesi gibi veya tab gibi olması bu özellikle belirlenir.

0,tabTabs:



1,tabButtons:



MultiRow

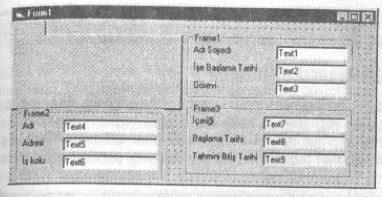
Bu özellik True ise sığmayan tablar bir alt satıra alınır. False ise kaydırma çubukları eklenir.

ClientHeight, ClientWidth, ClientLeft, ClientTop

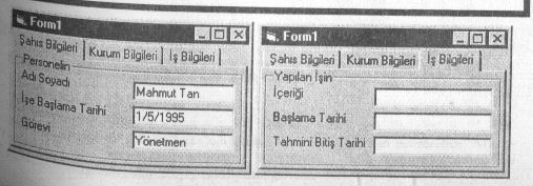
Tab kontrolünün kontroller için kullanılabilecek içi koordinatlarını verir.

Tab kontrollerinde her bir tab için farklı kontroller gösterilecekse, örneğin tablardan birinde kişiye ait nüfus bilgileri, diğerinde okul bilgiler vb farklı kontroller söz konusu ise her bir tab için kullanılacak kontroller bir Frame içine yerleştirilmeli ve ilgili Tab aktif hale geldiğinde, ilgili Frameyi bu özelliklerle tabın içine taşımalısınız.

ÖRNEK: Örnek olarak aşağıdaki gibi bir form hazırlayın.




```
'Örnek:TabStrip
Private Sub Form_Load()
 'Başlangıçta zaten bir tab var.Onu değiştir.
 TabStrip1.Tabs(1).Key = "PB"
 TabStrip1.Tabs(1).Caption = "Şahıs Bilgileri"
 Frame1.Caption = "Personelin"
 Call TabStrip1.Tabs.Add(, "KB", "Kurum Bilgileri")
 Frame2.Caption = "Kurumun"
 Call TabStrip1.Tabs.Add(, "İB", "İş Bilgileri")
 Frame3.Caption = "Yapılan İşin"
 With TabStrip1
  Frame1.Move .ClientLeft, .ClientTop, .ClientWidth,.ClientHeight
  Frame2.Move .ClientLeft, .ClientTop, .ClientWidth,.ClientHeight
  Frame3.Move .ClientLeft, .ClientTop, .ClientWidth,.ClientHeight
 End With
 Frame1.ZOrder 0'İlk frameyi öne getir.
End Sub

Private Sub TabStrip1_Click()
 'Seçilen taba ait frameyi öne getir.
 Select Case TabStrip1.SelectedItem.Key
  Case "PB": Frame1.ZOrder 0
  Case "KB": Frame2.ZOrder 0
  Case "IB": Frame3.ZOrder 0
 End Select
End Sub
```



Events

BeforeClick(cancel As Integer)

Bir tab tıklandığında henüz o taba geçmeden bu olay meydana gelir. Genel kontroller yapılarak şu anki tab içindeki bilgiler doğru değilse yeni taba geçi Cancel parametresine True atanarak iptal edilebilir. Ayrıca bir tabdan diğerine geçerken aktif tabdaki bilgiler kaydedilmesi gerekiyorsa yine bu olay kullanılabilir.

```
Private Sub TabStrip1_BeforeClick(Cancel As Integer)
 Dim c
 c = MsgBox(TabStrip1.SelectedItem.Caption & " bilgileri doğru
mu?", vbYesNo + vbQuestion, "Tab geçişi")
 If c = vbNo Then Cancel = True
End Sub
```

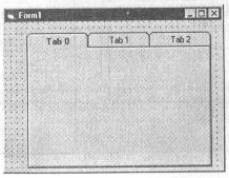
Click()

Yeni bir taba geçildiğinde bu olay meydana gelir. Seçilen tab hakkındaki bilgi SelectedItem özelliği ile öğrenilebilir.

SSTab*

Windows-95/98 ile birlikte gelen TabStrip kontrolünden daha kullanışlı bir kontroldür. Hatırlarsanız TabStrip kontrolünde her bir tab da gösterilecek kontrolleri frameler içine yerleştiriyor ve hangi tab aktif olursa ona göre ilgili frameyi görünür yapıyorduk. TabStrip'in bu özelliği nedeniyle özellikle üzerinde çok sayıda tab bulunacak formları hazırlamak zordur. Çünkü her bir tab için bir frame yerleştirmek ve kontrolleri bunun içine koymak gerekmektedir. Halbuki SSTab kontrolünün kendisi bir gruplayıcı eleman gibi davranır ve her tab için gerekli kontrolleri o tabın içine yerleştirmeye imkan vermektedir. Bu durum form tasarımında büyük kolaylıklar sağlar.

Bu kontrolü form üzerine yerleştirdiğinizde üzerinde hazır olarak üç tab bulunur.



Bu tablardan birini tıklayarak onun için gerekli tab içine kontrolleri yerleştirebilirsiniz. Yeni tablar eklemek için de **Tabs** özelliğini kullanabilirsiniz. Tabların başlıklarını belirlemek için de o tabı aktif yaptıktan sonra **Capiton** özelliği ile değiştirebilirsiniz.

Properties

Tabs

Kontrol üzerinde bulunacak tab sayısı bu özellikle belirlenir.

*Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Tabbed Dialog Control 6.0" seçeneğini işaretlemeniz veya Browse düğmesi ile TABCTL32.OCX dosyasını bulup projeye eklemeniz gerekir.

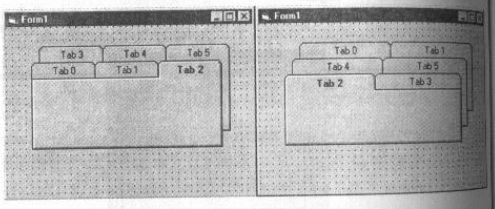
Tab

Aktif olan tab bu özellikle öğrenilebilir veya değiştirilebilir. İlk tabın numarası 0 dır.

TabsPerRow

Bir satırda bulunacak tab sayısı bu özellikle belirlenir. Normalde bu sayı üçtür ve üçten sonraki tablar alt satıra alınır.

Örneğin altı tabdan oluşan bir kontrolümüz olsun. **TabsPerRow** özelliğine 3 verdiğimizde soldaki gibi, 2 verdiğimde sağdaki resimdeki gibi olacaktır.

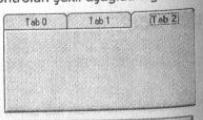


TabOrientation

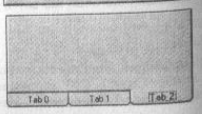
Tablar üstte olabileceği gibi kontrolün altında veya yanlarında da bulunabilir. Bu özellikle tabların yerleşimi belirlenir.

Bu özelliğin alabileceği değerler ve kontrolün şekli aşağıdaki gibidir.

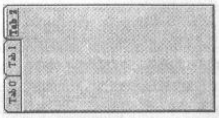
0



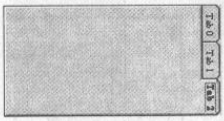
1



2



3

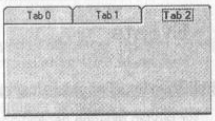


Tabları yanlara yerleştirecekseniz font olarak true type fontlardan birini seçmelisiniz. Aksi takdirde yazılar tam olarak yatay yazılamayabilir.

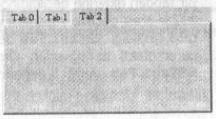
Style

Bu özelliğe 0 değeri verilirse tabların genişliği kontrolün genişliğine göre otomatik olarak belirlenir. 1 verilirse tabların genişliği sabit olur.

0



1

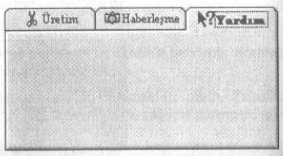


Caption

Aktif tabın başlığı bu özellikle değiştirilebilir. Komut düğmelerinde olduğu gibi & işareti kullanılarak da kısayol tuşu belirlenebilmektedir.

Picture

Her taba farklı bir resim verilebilmektedir. **Tab** özelliği ile aktif tabı belirledikten sonra bu özellikle o taba bir resim yüklenebilir.



TabEnabled(index)

Herhangi bir taba ulaşılamamasını istiyorsanız o tabın enabled özelliğini false yapabilirsiniz.

```
SSTab1.TabEnabled(1) = False
```

Events

Click(PreviousTab As Integer)

Bir tabdan diğerine geçildiğinde bu olay meydana gelir. Bu olaya yazılacak kodla tablar arası geçişte yapılması gereken kontroller veya işlemler yapılabilir. Hangi taba geçildiği **SSTab1.Tab** özelliği ile, hangi tabdan geçildiği ise **PreviousTab** parametresi ile öğrenilir. Örneğin 3 nolu tabdan 5 nolu taba geçildiğinde **SSTab1.Tab** özelliği 5 değerini, **PreviousTab** parametresi ise 3 değerini verir.

ImageCombo* (Resimli Combo)

Windows-95 ile birlikte gelen ve 98'de de kullanılan kontrollerden biri de ImageCombo kontrolüdür. Bu kontrol kullanılarak içerisinde resim bulunan Combo listeleri oluşturulabilir.

Properties

ImageList

ImageCombo kontrolündeki elemanların başında kullanılacak resimler de diğer standart ActiveX kontrollerinde olduğu gibi ImageList kontrolü ile belirlenir.

Form üzerine koyacağınız bir ImageList kontrolü ile kullanılacak resimleri belirledikten sonra bu özelliğe o kontrolün ismini vermeniz gerekir.

```
ImageCombo1.ImageList=ImageList1
```

ComboItems

ImageCombo içine eklenecek elemanlar bu özelliğin aşağıdaki alt özellik ve metodları ile düzenlenir.

ComboItems.Add(Index, Key, Text, Resim, SeçiliResim, Seviye)

Listeye eleman eklemek için bu metod kullanılır. **Index** ve **Key** parametreleri diğer kontrollerde olduğu gibidir.

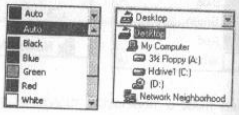
*Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Windows Common Controls 6.0" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCOMCTL.OCX dosyasını bulup projeye eklemeniz gerekir.

Text parametresi ile eklenecek elemanın metni belirlenir.

Resim parametresi ile eklenecek elemana bir resim verilebilir. ListImage kontrolüne resim yerleştirildikten sonra, ListImage kontrolündeki resimin numarası bu parametreye verilir.

SeçiliResim parametresi ile ise istenirse eleman seçili iken farklı bir resim gösterilmesi istenebilir. ListImage kontrolüne resim yerleştirildikten sonra, ListImage kontrolündeki resimin numarası bu parametreye verilir.

Seviye parametresi ile de eklenecek elemanın seviyesi belirlenebilir. Aşağıdaki resimlerden soldakinde seviyelendirme kullanılmamıştır. Sağdakinde ise seviyelendirme kullanılmıştır.

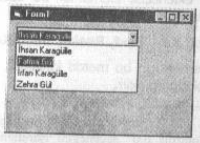


Bu özellikten geriye **ComboItem** tipinden bir değer döner. İstenirse bu değer bir değişkende saklanarak bunun üzerinde eklenen elemanla ilgili işlem yapılabilir. Bu gerekmiyorsa metod Call komutu ile çağrılarak geri dönen değer dikkate alınmaz.

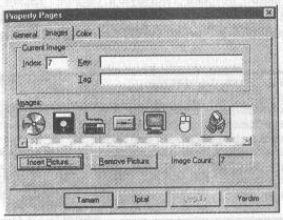
ÖRNEK: ImageCombo kontrolünü normal bir combo kutusu gibi kullanmak istersek Add metodunu şu şekilde kullanabiliriz:

```
Private Sub Form_Load()
 Call ImageCombo1.ComboItems.Add(, , "İhsan Karagülle")
 Call ImageCombo1.ComboItems.Add(, , "Fatma Gül")
 Call ImageCombo1.ComboItems.Add(, , "İrfan Karagülle")
 Call ImageCombo1.ComboItems.Add(, , "Zehra Gül")
End Sub
```

Bu işlem sonucunda normal bir ComboBox görünümünde bir liste oluşacaktır.



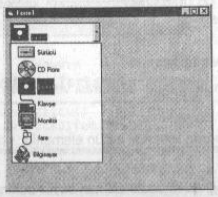
ÖRNEK: Listeye resim eklemek istersek formumuza bir **ImageList** kontrolü yerleştirip **Custom** özelliği ile açılan aşağıdaki pencereden kullanacağımız resimleri belirlememiz gerekir.



Bu işlemden sonra **ImageCombo** kontrolünün **ImageList** özelliğine **ImageList1** kontrolünün adı verilerek bu resimleri ImageCombo içinde kullanabiliriz.

Yukarıdaki pencerede belirlediğimiz her bir resmi ImageCombo içinde kullanabilmek için **Add** metodunda ona ait numarayı kullanabiliriz.

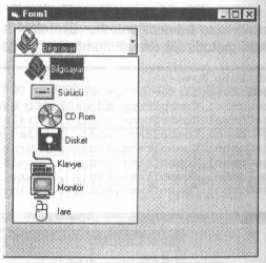
```
Private Sub Form_Load()
 ImageCombo1.ImageList = ImageList1
 Call ImageCombo1.ComboItems.Add(, , "CD Rom", 1)
 Call ImageCombo1.ComboItems.Add(, , "Disket", 2)
 Call ImageCombo1.ComboItems.Add(, , "Klavye", 3)
 Call ImageCombo1.ComboItems.Add(, , "Sürücü", 4)
 Call ImageCombo1.ComboItems.Add(, , "Monitör", 5)
 Call ImageCombo1.ComboItems.Add(, , "fare", 6)
 Call ImageCombo1.ComboItems.Add(, , "Bilgisayar", 7)
End Sub
```



ÖRNEK: İstersek elemanları eklerken seviyelendirme de yapabiliriz. Önceki örnekte bir bilgisayarda bulunan bazı parçaları resimleriyle birlikte ImageCombo kontrolüne ekledik. Belli bir seviyelendirme düzeyine göre de ekleme yapabiliriz. Örnekteki bütün parçalar bilgisayara bağlıdır. Disket ve Cd Rom ise sürücüye bağlıdır. Bunları hiyerarşik olarak eklemek istersek: Bilgisayar elemanına seviye olarak, diğerlerine 2, Sürücü ve Cd Rom elemanına ise 3 değerini vermemiz gerekir.

```
Private Sub Form_Load()
 ImageCombo1.ImageList = ImageList1
 Call ImageCombo1.ComboItems.Add(, , "Bilgisayar", 7, , 1)
 Call ImageCombo1.ComboItems.Add(, , "Sürücü", 4, , 2)
 Call ImageCombo1.ComboItems.Add(, , "CD Rom", 1, , 3)
 Call ImageCombo1.ComboItems.Add(, , "Disket", 2, , 3)
 Call ImageCombo1.ComboItems.Add(, , "Klavye", 3, , 2)
 Call ImageCombo1.ComboItems.Add(, , "Monitör", 5, , 2)
 Call ImageCombo1.ComboItems.Add(, , "fare", 6, , 2)
End Sub
```

Bu kod sonucunda bütün elemanlar hiyerarşik bir yapıda aşağıdaki gibi gösterilecektir.



ComboItems.Remove Index

Bu metod kullanılarak Index numarası verilen eleman listeden silinebilir.

ComboItems.Clear

Bu metod aracılığıyla listedeki bütün elemanlar silinebilir.

ComboItems.Count

Bu özellik aracılığı ile listedeki eleman sayısı bulunabilir.

ComboItems.Item(index)

Bu özellik aracılığı ile index numarası verilen elemana ait özellikler öğrenilebilir ve değiştirilebilir. (Combo kutusundaki List(index) özelliği gibidir)

ComboItems.Item(i) özelliği ile ComboItems(i) özelliği aynı anlamda kullanılır.

Bu özellik **ComboItem** tipinden tanımlanmıştır ve bu özelliğe ait aşağıdaki alt özellik bulunur.

ComboItems(index).Text

Index numarası verilen elemanın metni bu özellikle öğrenilebilir veya değiştirilebilir.

ComboItems(index).Image

Index numarası verilen elemanın kullandığı resimin numarası bu özellikle öğrenilebilir veya değiştirilebilir.

ComboItems(index).SelImage

Index numarası verilen elemanın seçili iken kullanacağı resimin numarası bu özellikle öğrenilebilir veya değiştirilebilir.

ComboItems(index).Indentation

Index numarası verilen elemanın seviyesi bu özellikle öğrenilebilir veya değiştirilebilir.

ComboItems(index).Selected

Index numarası verilen elemanın seçili olup olmadığı bu özellikle öğrenilebilir veya değiştirilebilir.

SelectedItem

Seçili elemanı bu özellik temsil eder. ComboItems(index) özelliğindeki bütün özellikler (text, image, selected, selimage, indentation) bu özellik için de geçerlidir. Seçili eleman üzerinde işlem yaparken bu özelliğin alt özellik ve metodları kullanılabilir.

Örneğin seçili elemanın metnini öğrenmek için **ImageCombo1.SelectedItem.Text** satırı kullanılabilir.

```
Private Sub ImageCombo1_Click()
 MsgBox ImageCombo1.SelectedItem.Text
End Sub
```

RichTextBox*

Windows-95 ile birlikte gelen ve 98'de de kullanılan kontrollerden biri de RichTextBox kontrolüdür. Diğer text kutuları formatlı bilgi girişi için uygun değildir. Örneğin yazının bir kısmı kalınken bir kısmı eğik olamaz. Standart text kutuları içindeki yazının tümü aynı özelliklerde olmak zorundadır. RichtextBox kontrolü ise bir kelime işlemci programı gibi bir çok özelliği destekler. Ayrıca bir çok popüler kelime işlemcinin desteklediği RTF formatını kullandığı için diğer kelime işlemcilerle kolayca dosya alış verişi yapılabilir.

Properties

Text, TextRTF

RichText kontrolü içindeki yazının tamamı bu iki özellikle öğrenilip değiştirilebilir. Text özelliği biçimsiz metin, SelRTF özelliği ise RTF biçimindeki metni temsil eder.

Örneğin RichtextBox içindeki yazıyı, standart bir text kutusuna alırken Text özelliğini, RTF formatını destekleyen başka bir kontrole alırken ise TextRTF özelliğini kullanmalısınız.

SelStart SelLength SelText SelRTF

Seçili kısmı öğrenip değiştirmek bu özelliklerle yapılır. Seçme işlemi Text kutusunda olduğu gibidir. Burada farklı olarak SelRTF özelliği seçili kısmı RTF formatında, SelText özelliği ise Text formatında verir. Yani SelRTF ile yazıya ait biçimde alınırken, SelText özelliğinde sadece metin alınacaktır.

SelFontName, SelFontSize, SelBold, SelItalic, SelColor, SelStrikethru, SelUnderline

RichText kontrolündeki yazılar farklı fontlarda olabileceği için seçili kısma ait font özellikleri bu özelliklerle öğrenilip değiştirilebilir. Bu özelliklere bir değer atandı-

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Rich TextBox Controls" seçeneğini işaretlemeniz veya Browse düğmesi ile RICHTX32.OCX dosyasını bulup projeye eklemeniz gerekir.

ğında, seçili bir kısım varsa orada ektili olur, seçili kısım yoksa kursörün bulunduğu nokta etkilenir.

Bu özelliklerin değerlerini öğrenirken IsNull fonksiyonunu kullanmanız gerekir. Örneğin seçili kısım kalın yazılmış ise SelBold özelliği True, kalın yazılmamışsa False, bir kısmı kalın bir kısmı değilse NULL değerini alacaktır. Dolayısıyla bir özelliğin değerinin karışık olduğunu Null değeri ile öğrenebiliriz.

Örneğin bir CheckBox kontrolünü yazının kalın olup olmadığını gösterirken:

```
If IsNull(RichText1.SelBold) Then
 Check1.Value=vbGrayed;
Else
 Check1.Value = Abs(RichText1.SelBold)
End If
```

Bu özelliklerin değerlerini bir başka özelliğe aktarmadan önce mutlaka yukarıdaki gibi Null olup olmadığını kontrol etmeniz gerekir. Aksi takdirde hata oluşacaktır.

SelAlignment

Yazının sola, sağa ve ortaya alınması bu özelliklerle yapılır.

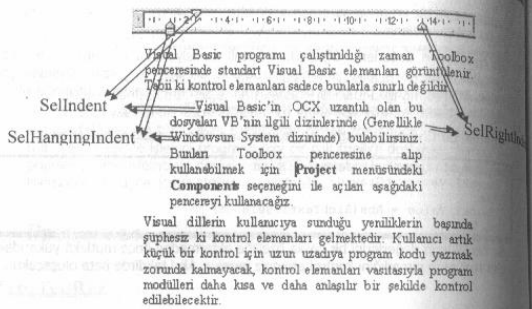
SelCharOffset

Seçili kısmın satır içindeki yüksekliği bu özelliklerle belirlenebilir. Yazıdaki her metin farklı yüksekliklerden başlayabilir.

SelCharOffset özelliği faklı değerlerde olan harflerden oluşan metin bunun gibi olacaktır.

SelIndent, SelHangingIndent, SelRightIndent

Seçili kısmın başlangıç ve bitiş noktaları bu özelliklerle belirlenir. Örneğin bir paragrafın daha içeriden başlaması için SelIndent özelliği, satır sonundan içeride bitmesi içinde SelRightIndent özelliği kullanılır. Bir paragrafın ilk satırı farklı yerden diğer satırları farklı yerden başlayabilir. SelIndent ilk satırın, SelHanging Indent ise diğer satırların başlayacağı noktayı belirler.



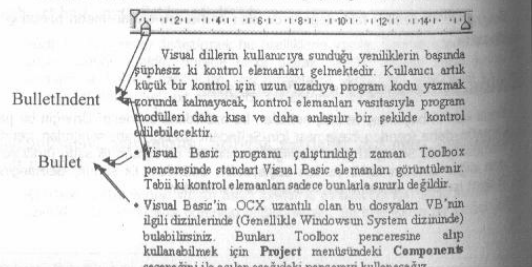
Visual Basic programı çalıştırıldığı zaman Toolbox penceresinde standart Visual Basic elemanları görüntülenir. Tabii ki kontrol elemanları sadece bunlarla sınırlı değildir.

Visual Basic'in .OCX uzantılı olan bu dosyaları VB'nin ilgili dizinlerinde (Genellikle Windowsun System dizininde) bulabilirsiniz. Bunları Toolbox penceresine alıp kullanabilmek için **Project** menüsündeki **Components** seçeneğini ile açılan aşağıdaki pencereyi kullanacağız.

Visual dillerin kullanıcıya sunduğu yeniliklerin başında şüphesiz ki kontrol elemanları gelmektedir. Kullanıcı artık küçük bir kontrol için uzun uzadıya program kodu yazmak zorunda kalmayacak, kontrol elemanları vasıtasıyla program modülleri daha kısa ve daha anlaşılır bir şekilde kontrol edilebilecektir.

SelBullet,BulletIndent

RTF formatında paragraf başlarına madde imleri de koyulabilmektedir. SelBullet özelliği true ise paragrafın başında im vardır. BullentIndent özelliği ise bu im metinlerdeki yazının ne kadar içeriden başlayacağını belirler.



Visual dillerin kullanıcıya sunduğu yeniliklerin başında şüphesiz ki kontrol elemanları gelmektedir. Kullanıcı artık küçük bir kontrol için uzun uzadıya program kodu yazmak zorunda kalmayacak, kontrol elemanları vasıtasıyla program modülleri daha kısa ve daha anlaşılır bir şekilde kontrol edilebilecektir.

- Visual Basic programı çalıştırıldığı zaman Toolbox penceresinde standart Visual Basic elemanları görüntülenir. Tabii ki kontrol elemanları sadece bunlarla sınırlı değildir.
- Visual Basic'in .OCX uzantılı olan bu dosyaları VB'nin ilgili dizinlerinde (Genellikle Windowsun System dizininde) bulabilirsiniz. Bunları Toolbox penceresine alıp kullanabilmek için **Project** menüsündeki **Components** seçeneğini ile açılan aşağıdaki pencereyi kullanacağız.

Methods

Find(String As String, [Start], [End],)

String parametresi ile belirlenen metni Start ve End parametreleri ile belirlenen aralıkta arar. Bulursa bulduğu yerin konumunu geri verir. Start ve End parametreleri kullanılmazsa baştan sona kadar arar.

```
Dim x
bul = InputBox("Aranacak ifade", "Bul", RichTextBox1.SelText)
x = RichTextBox1.Find(bul, RichTextBox1.SelStart)
If x < 0 Then
 MsgBox ("Bulunamadı")
End If
```

LoadFile, SaveFile(DosyaAdı)

RichTextBox içindeki metni RTF formatında kaydetmek ve RTF formatındaki bir dosyayı yüklemek için bu iki metod kullanılır.

```
RichTextBox1.SaveFile ('Deneme.RTF')
RichTextBox1.LoadFile ('Deneme.RTF')
```

SelPrint(hDC)

Seçili kısmı yazıcıya gönderir. Parametre olarak Printer.hDC verilmelidir.

GetLineFromChar(CharPos)

CharPos parametresi ile verilen pozisyonun kaçıncı satırda olduğunu verir. İlk satırın numarası sıfırdır.

```
Msgbox("Şu anda " & RichTextBox1.GetLineFromChar(RichTextBox1.
SelStart) + 1 & " nolu satırdasınız")
```

Events

SelChange()

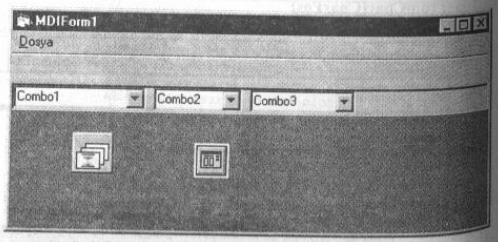
Seçili kısım değiştiğinde veya kursörün yeri değiştiğinde bu olay meydana gelir. Daha çok araç çubuklarının durumunu güncelleştirmek için kullanılır.

ÖRNEK: Örnek olarak RichEdit kontrolünü kullanarak bir kelime işlemci programı yapalım.

- Önce Project -Add MDIForm menü seçeneği ile yeni bir MDI form oluşturun.
- Project-Project Properties menüleriyle açılan pencereden Startup Object olarak MDIForm1 seçeneğini seçerek programın MDIForm ile çalışmaya başlamasını söyleyin.
- MDI Form üzerine MSCOMMCTL.OCX dosyasında bulunan bir ToolBar kontrolü, bir ImageList, COMDLG32.OCX dosyasında bulunan bir CommonDialog ve bir PictureBox yerleştirin.
- MDI Form üzerine yerleştirdiğiniz PictureBox içine üç tane ComboBox yerleştirin.
- ImageList kontrolünün Custom özelliği ile açılan pencereden aşağıdaki resimleri bulup ekleyin.

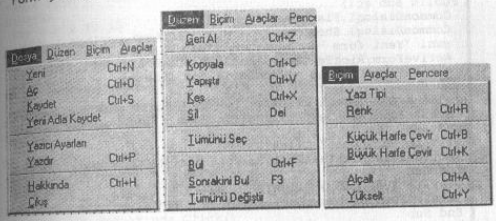
- MDI Form üzerinde yandaki menüleri oluşturun. (Menülere isim verirken Caption özelliğine verdiğiniz değerin başına Mn koyun. Yani Yeni menüsünün ismini MnYeni, Aç menüsünün ismini MnAç olarak verin)

MDI form aşağıdaki gibi olmalıdır.



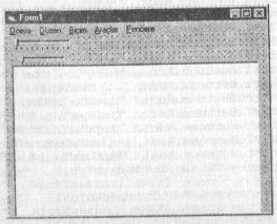
- Şimdi projenizdeki Form1 formunu seçin ve MDIChild özelliğini True yapın

- Formun üzerine bir RichTextBox yerleştirin ve HideSelection özelliğini False yapın.
- Sol ve sağ marjin ayarlarını yapabilmek için formunuzun üzerine, MSCOMCTL.OCX dosyasında bulunan iki tane Slider yerleştirin.
- Form için aşağıdaki menüleri tasarlayın.



- Pencere menüsünün WindowList özelliğini True yaparak Child formların listesini tutmasını sağlayın.

Form1 formu aşağıdaki gibi olmalıdır.




```
'Örnek:RichTextBox
'Bu kısmı MDIForm1 için yazılacak
Public Sub yeni()
 Dim Form As New Form1
 Form.Show
 Form.Caption = "Bizim Doküman " & Forms.Count - 1
End Sub
Public Sub aç()
 CommonDialog1.Flags = 0
 CommonDialog1.ShowOpen
 yeni 'Yeni form
 ActiveForm.RichTextBox1.LoadFile (CommonDialog1.filename)
 ActiveForm.Caption = CommonDialog1.filename
End Sub
Private Sub Combo1_Click()
 ActiveForm.RichTextBox1.SelFontName = Combo1.Text
End Sub
Private Sub Combo2_Click()
 ActiveForm.RichTextBox1.SelFontSize = Val(Combo2.Text)
End Sub
Private Sub Combo3_Click()
 ActiveForm.RichTextBox1.SelColor = QBColor(Combo3.ListIndex)
End Sub
Private Sub MDIForm_Load()
 Caption = "Bizim Word 1.0"
 CommonDialog1.Filter = "RTF dosyaları|*.rtf|Text dosyaları|
*.txt| Bütün dosyalar|*.*||"
 Toolbar1.ImageList = ImageList1
 Call Toolbar1.RestoreToolbar(App.EXEName,"İhsan Karagülle","Araç
Çubuğu")
 Call Toolbar1.Buttons.Add(, "Yeni", , tbrDefault, 1)
 Call Toolbar1.Buttons.Add(, "Aç", , tbrDefault, 2)
 Call Toolbar1.Buttons.Add(, "Kaydet", , tbrDefault, 3)
 Call Toolbar1.Buttons.Add(, , , tbrSeparator)
 Call Toolbar1.Buttons.Add(, "Yazdır", , tbrDefault, 4)
 Call Toolbar1.Buttons.Add(, , , tbrSeparator)
 Call Toolbar1.Buttons.Add(, "Kopyala", , tbrDefault, 5)
 Call Toolbar1.Buttons.Add(, "Yapıştır", , tbrDefault, 6)
 Call Toolbar1.Buttons.Add(, , , tbrSeparator)
 Call Toolbar1.Buttons.Add(, "Kalın", , tbrCheck, 7)
 Call Toolbar1.Buttons.Add(, "Eğik", , tbrCheck, 8)
 Call Toolbar1.Buttons.Add(, "AltÇ", , tbrCheck, 9)
 Call Toolbar1.Buttons.Add(, , , tbrSeparator)
 Call Toolbar1.Buttons.Add(, "Sola", , tbrButtonGroup, 10)
 Call Toolbar1.Buttons.Add(, "Ortaya", , tbrButtonGroup, 11)
 Call Toolbar1.Buttons.Add(, "Sağa", , tbrButtonGroup, 12)
 Call Toolbar1.Buttons.Add(, , , tbrSeparator)
 Call Toolbar1.Buttons.Add(, "Bullet", , tbrCheck, 13)
 Form1.Caption = "Bizim Doküman 1"
 Dim i
 For i = 0 To Screen.FontCount - 1
  Combo1.AddItem Screen.Fonts(i)
 Next
```



```
 For i = 8 To 100
  Combo2.AddItem i
 Next
 Dim renk
 renk = Array("Siyah", "Mavi", "Gri", "Cyan", "Kırmızı",
"Magenta", "Sarı", "Beyaz", "Koyu Gri", "Açık Mavi", "Açık Gri",
"Açık Cyan", "Açık Kırmızı", "Açık Magenta", "Açık Sarı", "Parlak
Beyaz")
 For i = 0 To 15
  Combo3.AddItem renk(i)
 Next
 Form1.Show
End Sub

Private Sub MnAç_Click()
 aç
End Sub

Private Sub MnÇık_Click()
 Unload Me
End Sub

Private Sub MnHakkında_Click()
 MsgBox ("Bizim Word 1.0 İhsan KARAGÜLLE Mart 1997 Ahlat/BİTLİS")
End Sub

Private Sub MnYeni_Click()
 yeni
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As
MsComctlLib.Button)
 With ActiveForm
  Select Case Button.Key
   Case "Yeni": yeni
   Case "Aç": aç
   Case "Kaydet": ActiveForm.RichTextBox1.SaveFile (Caption)
   Case "Yazdır": ActiveForm.RichTextBox1.SelPrint (Printer.hDC)
   Case "Kopyala": Clipboard.Clear
    Clipboard.SetText .RichTextBox1.SelRTF, vbCFRTF
   Case "Yapıştır":
    .RichTextBox1.SelRTF=Clipboard.GetText(vbCFRTF)
   Case "Kalın": .RichTextBox1.SelBold = Button.Value
   Case "Eğik": .RichTextBox1.SelItalic = Button.Value
   Case "AltÇ": .RichTextBox1.SelUnderline = Button.Value
   Case "Sola": .RichTextBox1.SelAlignment = Button.Value
   Case "Ortaya": .RichTextBox1.SelAlignment = Button.Value + 1
   Case "Sağa": .RichTextBox1.SelAlignment = Button.Value + 2
   Case "İkiYana": .RichTextBox1.SelAlignment = Button.Value + 3
   Case "Sola": .RichTextBox1.SelAlignment = 0
   Case "Ortaya": .RichTextBox1.SelAlignment = 2
   Case "Sağa": .RichTextBox1.SelAlignment = 1
   Case "Bullet": .RichTextBox1.SelBullet = Button.Value
```

```
   End Select
  End With
End Sub

'Bu kısım Form1 için yazılacak
Option Explicit
Dim bul, değişti

Private Sub Form_Load()
 değişti = False
 MDIForm1.Combo1.Text = RichTextBox1.SelFontName
 MDIForm1.Combo2.Text = RichTextBox1.SelFontSize
End Sub

Private Sub Form_Resize()
 Slider1.Move 0, 0, ScaleWidth'Formun en üstüne al
 Slider2.Move 0, Slider1.Height, ScaleWidth'Slide1'İn altına al
 RichTextBox1.Move 0,Slider2.Top+Slider2.Height,ScaleWidth,
ScaleHeight
 Slider1.Max = ScaleWidth
 Slider2.Max = ScaleWidth
 Slider1.TickFrequency = Slider1.Max / 100
 Slider2.TickFrequency = Slider2.Max / 100
End Sub

Private Sub Form_Unload(Cancel As Integer)
 Dim c
 If değişti Then
  c = MsgBox(Caption & " dosyasındaki Değişiklikler
kaydedilsinmi", vbYesNoCancel + vbQuestion, "Kapat")
  If c = vbYes Then MnKaydet_Click
  If c = vbCancel Then Cancel = True
 End If
End Sub

Private Sub MnAç_Click()
 MDIForm1.aç
End Sub

Private Sub MnAlçalt_Click()
 RichTextBox1.SelCharOffset = RichTextBox1.SelCharOffset - 10
End Sub

Private Sub MnAraçÇubukları_Click()
 MDIForm1.Toolbar1.Customize
 Call MDIForm1.Toolbar1.SaveToolbar(App.EXEName, "İhsan
Karagülle", "Araç Çubuğu")
End Sub

Private Sub MnBasamakla_Click()
 MDIForm1.Arrange vbCascade
End Sub
```



```
Private Sub MnBul_Click()
 Dim x
 bul = InputBox("Aranacak ifade", "Bul", RichTextBox1.SelText)
 x = RichTextBox1.Find(bul, RichTextBox1.SelStart)
 If x < 0 Then
  MsgBox ("Bulunamadı")
 End If
End Sub

Private Sub MnBüyükHarfeÇevir_Click()
 RichTextBox1.SelRTF = UCase(RichTextBox1.SelRTF)
End Sub

'Bu olay alt programının ismini de klavyeden yazmalısınız.
Private Sub MnDüzen_Click()
 'Menü açılmadan içindeki elemanları güncelleştirir
 If RichTextBox1.SelLength = 0 Then
  MnKopyala.Enabled = False
  MnKes.Enabled = False
  MnSil.Enabled = False
  MnTümünüDeğiştir.Enabled = False
 Else
  MnKopyala.Enabled = True
  MnKes.Enabled = True
  MnSil.Enabled = True
  MnTümünüDeğiştir.Enabled = True
 End If
 If bul = "" Then
  MnSonrakiniBul.Enabled = False
 Else
  MnSonrakiniBul.Enabled = True
 End If
End Sub

'Bu olay alt programının ismini de klavyeden yazmalısınız.
Private Sub MnBiçim_Click()
 'Menü açılmadan içindeki elemanları güncelleştirir
 If RichTextBox1.SelLength = 0 Then
  'Seçili kısım yoksa bazı menüleri pasif yap
  MnBüyükHarfeÇevir.Enabled = False
  MnKüçükHarfeÇevir.Enabled = False
 Else
  MnBüyükHarfeÇevir.Enabled = True
  MnKüçükHarfeÇevir.Enabled = True
 End If
End Sub

Private Sub MnÇıkış_Click()
 Unload MDIForm1
End Sub

Private Sub MnDikeyDöşe_Click()
 MDIForm1.Arrange vbTileVertical
```



```
End Sub

Private Sub MnGeriAl_Click()
 SendKeys "^z" 'Ctrl-Z tuşu
End Sub

Private Sub MnHakkında_Click()
 MsgBox ("Bizim Word 1.0 İhsan KARAGÜLLE Mart 1997 Ahlat/BİTLİS"
End Sub

Private Sub MnKaydet_Click()
 RichTextBox1.SaveFile (Caption)
 değişti = False
End Sub

Private Sub MnKes_Click()
 Clipboard.Clear
 Clipboard.SetText RichTextBox1.SelRTF, vbCFRTF
 RichTextBox1.SelRTF = ""
End Sub

Private Sub MnKopyala_Click()
 Clipboard.Clear
 Clipboard.SetText RichTextBox1.SelRTF, vbCFRTF
End Sub

Private Sub MnKüçükHarfeÇevir_Click()
 RichTextBox1.SelRTF = LCase(RichTextBox1.SelRTF)
End Sub

Private Sub MnRenk_Click()
 MDIForm1.CommonDialog1.Flags = 0
 MDIForm1.CommonDialog1.ShowColor
 RichTextBox1.BackColor = MDIForm1.CommonDialog1.Color
End Sub

Private Sub MnSil_Click()
 RichTextBox1.SelRTF = ""
End Sub

Private Sub MnSimgeleriYerleştir_Click()
 MDIForm1.Arrange vbArrangeIcons
End Sub

Private Sub MnSonrakiniBul_Click()
 Dim x
 x = RichTextBox1.Find(bul, RichTextBox1.SelStart + 1)
 If x < 0 Then MsgBox ("Bulunamadı")
End Sub

Private Sub MnTümünüDeğiştir_Click()
 Dim yer, sayi, değ, aranan
 aranan = RichTextBox1.SelText
```

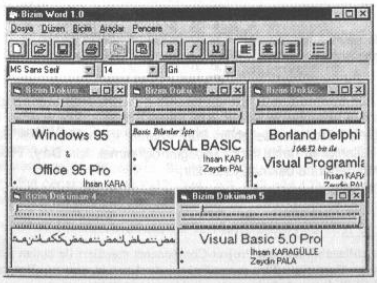


```
 değ = InputBox(aranan & " ifadesi ne ile değiştirilsin")
 yer = RichTextBox1.Find(aranan)
 While yer >= 0
  sayi = sayi + 1
  RichTextBox1.SelStart = yer
  RichTextBox1.SelLength = Len(aranan)
  RichTextBox1.SelText = değ
  yer = yer+Len(değ) - 1 'Yeni yer değiştirilen metnin son harfi
  yer = RichTextBox1.Find(aranan, yer + 1)
 Wend
 MsgBox (Str(sayi) + " tane değiştirme işlemi yapıldı")
End Sub

Private Sub MnTümünüSeç_Click()
 RichTextBox1.SelStart = 0
 RichTextBox1.SelLength = Len(RichTextBox1.TextRTF)
End Sub

Private Sub MnYapıştır_Click()
 RichTextBox1.SelRTF = Clipboard.GetText(vbCFRTF)
End Sub

Private Sub MnYatayDöşe_Click()
 MDIForm1.Arrange vbTileHorizontal
End Sub

Private Sub MnYazdır_Click()
 MDIForm1.CommonDialog1.Flags = 0
 MDIForm1.CommonDialog1.ShowPrinter
 If (MDIForm1.CommonDialog1.Flags And cdlPDSelection) = 0 Then
  'Tümünü yazdır seçildi ise tümünü seç
  MnTümünüSeç_Click
 End If
 RichTextBox1.SelPrint (Printer.hDC)
End Sub

Private Sub MnYazıcıAyar_Click()
 MDIForm1.CommonDialog1.Flags = cdlPDPrintSetup
 MDIForm1.CommonDialog1.ShowPrinter
End Sub

Private Sub MnYazıTipi_Click()
 MDIForm1.CommonDialog1.Flags = cdlCFEffects Or cdlCFBoth
 MDIForm1.CommonDialog1.ShowFont
 With RichTextBox1
  .SelFontName = MDIForm1.CommonDialog1.FontName
  .SelFontSize = MDIForm1.CommonDialog1.FontSize
  .SelBold = MDIForm1.CommonDialog1.FontBold
  .SelItalic = MDIForm1.CommonDialog1.FontItalic
  .SelUnderline = MDIForm1.CommonDialog1.FontUnderline
  .SelColor = MDIForm1.CommonDialog1.Color
 End With
End Sub
```

```
Private Sub MnYeni_Click()
 MDIForm1.yeni
End Sub

Private Sub MnYeniAdlaKaydet_Click()
 MDIForm1.CommonDialog1.Flags = 0
 MDIForm1.CommonDialog1.filename = Caption
 MDIForm1.CommonDialog1.ShowSave
 Caption = MDIForm1.CommonDialog1.filename
 RichTextBox1.SaveFile (Caption)
 değişti = False
End Sub

Private Sub MnYükselt_Click()
 RichTextBox1.SelCharOffset = RichTextBox1.SelCharOffset + 10
End Sub

Private Sub RichTextBox1_Change()
 değişti = True
End Sub

Private Sub RichTextBox1_SelChange()
 If Not IsNull(RichTextBox1.SelIndent) Then
  Slider1.Value = RichTextBox1.SelIndent
 End If
 If Not IsNull(RichTextBox1.SelRightIndent) Then
  Slider2.Value = Slider2.Max - RichTextBox1.SelRightIndent
 End If
 With MDIForm1.Toolbar1
  If IsNull(RichTextBox1.SelBullet) Then
   .Buttons("Kopyala").Enabled = False
  Else
   .Buttons("Kopyala").Value = Abs(RichTextBox1.SelBullet)
   .Buttons("Kopyala").Enabled = True
  End If
  If RichTextBox1.SelLength = 0 Then
   MnKopyala.Enabled = False
   .Buttons("Kopyala").Enabled = False
  Else
   MnKopyala.Enabled = True
   .Buttons("Kopyala").Enabled = True
  End If
  If IsNull(RichTextBox1.SelBold) Then
   .Buttons("Kalın").MixedState = True
  Else
   .Buttons("Kalın").Value = Abs(RichTextBox1.SelBold)
   .Buttons("Kalın").MixedState = False
  End If
  If IsNull(RichTextBox1.SelItalic) Then
   .Buttons("Eğik").MixedState = True
  Else
   .Buttons("Eğik").Value = Abs(RichTextBox1.SelItalic)
   .Buttons("Eğik").MixedState = False
```



```
  End If
  If IsNull(RichTextBox1.SelUnderline) Then
   .Buttons("AltÇ").MixedState = True
  Else
   .Buttons("AltÇ").Value = Abs(RichTextBox1.SelUnderline)
   .Buttons("AltÇ").MixedState = False
  End If
 End With
 If IsNull(RichTextBox1.SelFontSize) Then
  MDIForm1.Combo2.Text = ""
 Else
  MDIForm1.Combo2.Text = RichTextBox1.SelFontSize
 End If
 If IsNull(RichTextBox1.SelFontName) Then
  MDIForm1.Combo1.Text = ""
 Else
  MDIForm1.Combo1.Text = RichTextBox1.SelFontName
 End If
 Select Case RichTextBox1.SelAlignment
  Case 0: MDIForm1.Toolbar1.Buttons("Sola").Value = tbrPressed
  Case 1: MDIForm1.Toolbar1.Buttons("Sağa").Value = tbrPressed
  Case 2: MDIForm1.Toolbar1.Buttons("Ortaya").Value = tbrPressed
 End Select
End Sub

Private Sub Slider1_Click()
 RichTextBox1.SelIndent = Slider1.Value
End Sub

Private Sub Slider2_Click()
 RichTextBox1.SelRightIndent = Slider2.Max - Slider2.Value
End Sub
```



MonthView* (Takvim)

Bu kontrolü kullanarak programlarınıza takvim ekleyebilirsiniz. Bu kontrol aracılığı ile bir veya daha fazla aydan oluşan takvim gösterilebileceği gibi tarih ayarlaması da yaptırılabilir.

Kullanıcıdan tarih girmesini istediğiniz durumlarda bu kontrolü kullanabilirsiniz. Kullanıcı gireceği tarihi bu kontrolden görerek seçebilir.

Kullanıcı kontrol üzerindeki < > düğmeleri kullanarak sonraki ayları görebilir. Sonraki yıllara bu şekilde geçmek zor olduğu için yıl yazılı yeri tıkladığında bir spin açılacak ve istediği yıla gitmesini kolayca sağlayacaktır.

Ay kısmını tıkladığında ise bir liste açılarak ayları gösterecektir. Böylece kullanıcı istediği tarihi kolayca seçebilir. Bütün bunlar için kod yazmak da gerekmez. Sadece kullanıcının hangi tarihi seçtiğini öğrenmek için **Day, Month, Year** özelliklerinin değerine bakmak yeterlidir.

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Common Controls 2" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCOMCT2.OCX dosyasını bulup projeye eklemeniz gerekir.

Properties

Value

Bu günün tarihi bu özellikle öğrenilip değiştirilebilir.

Day, Month, Year, Week

Kullanıcının seçtiği tarih bu özelliklerle öğrenilip değiştirilebilir. Bu özellikler sırasıyla gün, ay, yıl ve yılın kaçıncı haftası olduğunu verir.

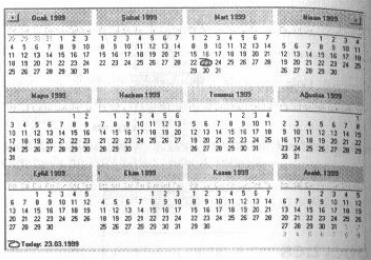
MinDate, MaxDate

İstenirse bu iki özellik kullanılarak kullanıcının seçebileceği tarihler sınırlandırılabilir. Normalde kullanıcı 1.1.1753 ile 31.12.9999 arasında bir tarih seçebilir.

MonthColumns, MonthRows

Normalde **MonthView** kontrolü üzerinde sadece bir aya ait takvim gösterilir. İstenirse birden fazla ay da aynı takvim üzerinde gösterilebilir. Bu iki özellikle kaç ayın gösterileceği belirlenebilir. Bu iki özelliğe verilecek tek başına veya birlikte değer 12'yi geçemez. Yani 1x12, 3x4, 2x6, 3x2 gibi 12'yi geçmeyecek şekilde bir değer verilebilir.

Aşağıdaki resimde **MonthColumns=4** ve **MonthRows=3** olan bir takvim görülmektedir.



StartOfWeek

Haftanın başlangıç gününün hangi gün olacağı bu özellikle belirlenir. Türkiye'deki sisteme uygun olan Pazartesi başlangıcı için bu özellik 2 değerinde olmalıdır.

ShowWeekNumber

Bu özellik **true** yapılırsa haftanın numarası da başına yazılır. Aşağıdaki resimlerden soldakinde hafta numarası yokken, sağdakinde bu özellik **true** yapılarak hafta numarasının da gösterilmesi sağlanmıştır.

MultiSelect, SelStart, SelEnd

Normalde kullanıcı takvimden tek bir tarih seçebilir. Ancak belli bir aralık seçebilmesini, yani birden fazla tarih seçebilmesini istiyorsanız **MultiSelect** özelliğini **True** yapabilirsiniz. Bu durumda seçilen başlangıç tarihini **SelStart** özelliği ile, seçilen sonuncu tarihi de **SelEnd** özelliği ile öğrenebilirsiniz.

Methods

HitTest(x, y, tarih)

Bu metod aracılığı ile x,y koordinatlarında hangi tarihin yazılı olduğu öğrenilebilir. Genellikle drag-drop işlemlerinde kullanıcı bir şeyi sürükleyip bu kontrolün üzerine bıraktığında bıraktığı noktanın hangi tarihe denk geldiğini bulmak için kullanılabilir. Genel olarak farenin geçtiği bir x,y koordinatındaki tarihi öğrenmek için kullanılır.

Bu metoddan **Tarih** parametresi ile x,y koordinatındaki tarih değeri döner. Ayrıca metoddan geriye aşağıdaki sayılardan biri dönerek bırakılan yer hakkında daha detaylı bilgi verir.

mvwCalendarBack	0 Takvimin zemini üzerinde bir yer
mvwCalendarDate	1 Takvimdeki bir tarihin bulunduğu yer
mvwCalendarDay	4 Gün isimlerinin yazılı olduğu yer.
mvwCalendarWeekNum	5 Hafta numarasının gösterildiği yer
mvwTitleMonth	10 Ay isminin yazılı olduğu yer
mvwTitleYear	11 Yıl isminin yazıldığı yer
mvwCalendarDay	4 Bu günün tarihini gösteren yer

ÖRNEK:Örnek olarak farenin geçtiği noktayla ilgili bilgi verecek kodu yazalım.

```
Private Sub MonthView1_MouseMove(Button As Integer, Shift As
Integer, x As Single, y As Single)
   Dim k, t As Date
   k = MonthView1.HitTest(x, y, t)
   Caption = t
   Select Case k
     Case 0: Caption = ""
     Case 4: Caption = "Bu gün:" & MonthView1.Value
     Case 5: Caption = MonthView1.Week & ".hafta"
     Case 10: Caption = MonthView1.Month & ".ay"
     Case 11: Caption = "Yıl " & MonthView1.Year
   End Select
End Sub
```

Events

DateClick, DataDblClick(ByVal DateClicked As Date)

Takvim üzerindeki bir tarih tıklandığında veya çift tıklandığında bu olay meydana gelir. **DateClicked** parametresi ile tıklanan tarih öğrenilebilir.

ÖRNEK: Örnek olarak çift tıklanan tarihleri bir listeye eklemek isteyelim.

```
Private Sub MonthView1_DateDblClick(ByVal DateDblClicked As Date)
  List1.AddItem DateDblClicked
End Sub
```

DateTimePicker * (Tarih Seçme Kutusu)

Programlarınızda tarih girilmesini istediğiniz yerlere bu kontrolü yerleştirerek kullanıcının istediği tarihi yazabilmesini veya kutuyu aşağı doğru açarak gösterilecek takvimden görsel olarak seçebilmesini sağlayabilirsiniz.

Bu kontrol normalde bir ComboBox gibi görünür ve kullanıcı içine bir tarih girebilir.

Ancak kullanıcı isterse Combo kutusunu aşağı doğru açar. Bu durumda ComboBoxla birlikte bir takvim görüntülenir ve kullanıcı bu takvimden de istediği tarihi seçebilir.

Properties

Value

Kontrolün göstereceği tarih bu özellikle belirlenebileceği gibi, kullanıcının seçtiği tarih de bu özellikle öğrenilir.

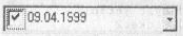
* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Common Controls 2" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCOMCT2.OCX dosyasını bulup projeye eklemeniz gerekir.

Day, Month, Year, DayOfWeek, Week

Kullanıcının girdiği tarih **Value** özelliği ile öğrenilebiliyordu. İstenirse tarihe ait gün, ay, yıl, haftanın günü ve yılın haftası bu özelliklerle ayrı ayrı öğrenilebilir.

CheckBox

Bu özelliğe True değeri verilirse tarihin başına bir işaret kutusu eklenir.

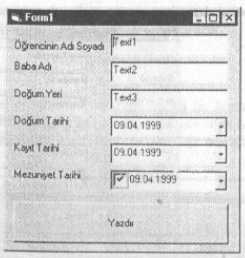


Örneğin mezuniyet tarihinin girilmesini istediğimiz bir alanda bu kontrolü kullandığımızı kabul edelim. Eğer öğrenci henüz mezun olmamışsa kullanıcı işaret kutusunu boş bırakacaktır ve böylece Value özelliği kutuda yazılı olan tarihi değil boş bir değer gönderecektir.

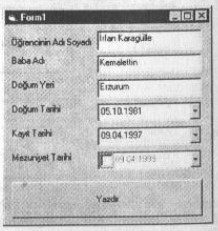
UpDown

Bu özellik True yapılırsa kontrol bir Combo kutusu gibi değil Text kutusu gibi davranır. Text kutusunun yanına otomatik olarak eklenen yukarı aşağı düğmeleri ile kullanıcı tarihi belirleyebilir.

ÖRNEK: Örnek olarak aşağıdaki gibi bir öğrenci bilgi formumuz olsun.

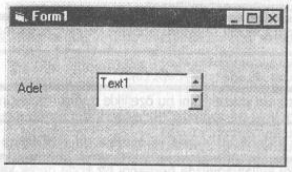


```
'ÖRNEK: DateTimePicker
Private Sub Command1_Click()
 Printer.Print "Öğrencinin"
 Printer.Print "Adı Soyadı",
 Printer.Print Text1
 Printer.Print "Baba Adı",
 Printer.Print Text2
 Printer.Print "Doğum Yeri",
 Printer.Print Text3
 Printer.Print "Doğum Tarihi",
 Printer.Print DTPicker1
 Printer.Print "Kayıt Tarihi",
 Printer.Print DTPicker2
 Printer.Print "Mezuniyet Tarihi",
 Printer.Print DTPicker3
 Printer.EndDoc
End Sub

Private Sub DTPicker2_Validate(Cancel As Boolean)
 If DTPicker2.Value < DTPicker1.Value Then
  MsgBox ("Öğrenci doğmadan mı kaydolmuş:)")
  Cancel = True
 End If
End Sub

Private Sub DTPicker3_Validate(Cancel As Boolean)
 If DTPicker3.Value < DTPicker2.Value Then
  MsgBox ("Kayıt tarihinden küçük olamaz")
  Cancel = True
 End If
End Sub

Private Sub Form_Load()
 DTPicker3.CheckBox = True
End Sub
```



UpDown*

VB'nin önceki versiyonlarında bulunan Spin kontrolüyle aynı işi yapan daha gelişmiş bir kontroldür. Bu kontrolün amacı bir değerin artırılıp azaltılmasında görsel bir arabirim sağlamaktır.



Bu kontrol başka bir kontrolle bağlantı kurarak onun içindeki değeri artırıp azaltabilmektedir. Programınızdaki text kutusu, label veya diğer kontrollerle bağlantısını kurarak onun içindeki değeri bu kontrol aracılığı ile artırıp azaltabilirsiniz.

Properties

Orientation

Bu kontrol hem yatay hem de dikey olarak yerleştirilebilir. Bu özelliğin değeri 0 ise kontrol yatay olarak, 1 ise dikey olarak görüntülenir.

BuddyControl

Bu kontrol aracılığı ile programınız içindeki başka bir kontrolün içeriğini değiştirebilirsiniz. Örneğin programınızdaki bir Text kutusu veya Label içindeki sayıyı bu kontrol aracılığı ile değiştirmek isterseniz bu özelliğe o kontrolün ismini vermeniz gerekir.

```
UpDown1.BuddyControl=Text1
```

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Common Controls 2" seçeneğini işaretlemeniz veya Browse düğmesi ile MSCOMCT2.OCX dosyasını bulup projeye eklemeniz gerekir.

BuddyProperty

BuddyControl özelliği ile UpDown kontrolünün kullanacağı kontrolü belirledikten sonra bu özellikle de o kontrolün hangi özelliğinin değiştirileceğini belirlemeniz gerekir.

Text1 kutusu içindeki sayıyı bu özellikle değiştirmek için:

```
UpDown1.BuddyControl=Text1
UpDown1.BuddyProperty="Text"
```

Label1 kutusu içindeki sayıyı bu özellikle değiştirmek için:

```
UpDown1.BuddyControl=Label1
UpDown1.BuddyProperty="Caption"
```

Text1 kutusunun yüksekliğini bu özellikle değiştirmek için:

```
UpDown1.BuddyControl=Text1
UpDown1.BuddyProperty="Height"
```

Bu iki özelliği kullandığınızda herhangi bir koda gerek kalmaksızın ilişki kurduğunuz kontrolün içeriği değiştirilecektir.

Alignment

BuddyControl özelliği ile bir kontrolle bağlantı kurulmuşsa bu özellikle UpDown düğmeleri o kontrolün sağına veya soluna yerleştirilebilir. Bu özelliğin değeri 0 ise kontrolün soluna, 1 ise sağına yerleşecektir.

Min, Max

UpDown kontrolünün değiştirebileceği değer aralığı bu iki özellikle belirlenir. Normalde bu değerler 0-10'dur ve 0 ile 10 arasında değer ayarlayabilir. Bu iki özellik değiştirilerek aralık yeniden belirlenebilir.

Increment

UpDown kontrolünün her seferinde ne kadarlık bir değişim gerçekleştireceği bu özellikle belirlenir. Normalde bu değer 1'dir ve birer birer artar veya azalır.

Value

UpDown kontrolünün temsil ettiği değer bu özellikle öğrenilebilir.

Events

UpClick()

Kontrol üzerindeki artırma düğmesi tıklandığında bu olay meydana gelir.

DownClick()

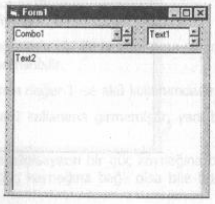
Kontrol üzerindeki azaltma düğmesi tıklandığında bu olay meydana gelir.

Change()

UpDown kontrolünün temsil ettiği değer değiştiğinde bu olay meydana gelir.

Buddy özellikleri aracılığıyla bu kontrol ile başka bir kontrol arasında bağlantı varsa herhangi bir kod yazmak gerekmez. Ancak bir bağlantı yoksa bu olaylar kullanılarak değer değişimlerinde yapılacak işlemler için gerekli kod bu olaylara yazılabilir.

ÖRNEK: Örnek olarak aşağıdaki formda font adı ve font boyutunu seçmeye yarayan kutularla UpDown kontrolü arasında bağlantı kurarak değişimlerin UpDown aracılığı ile de yapılabilmesini sağlayalım.




```
'ÖRNEK: UpDown
Private Sub Form_Load()
 Dim i
 UpDown1.BuddyControl = Combo1
 UpDown1.BuddyProperty = "ListIndex"

 UpDown2.BuddyControl = Text1
 UpDown2.BuddyProperty = "Text"
```

```
For i = 0 To Screen.FontCount - 1
 Combo1.AddItem Screen.Fonts(i)
Next

UpDown1.Min = 0
UpDown1.Max = Combo1.ListCount - 1
UpDown1.Increment = 1

UpDown2.Min = 0
UpDown2.Max = 100
UpDown2.Increment = 5
End Sub

Private Sub Combo1_Click()
 Text2.FontName = Combo1.Text
End Sub

Private Sub Text1_Change()
 Text2.FontSize = Val(Text1)
End Sub
```

Kullanıcı Combo kutusundan bir font adı seçebileceği gibi UpDown kontrolü ile de bu fontu değiştirebilir. Aynı şekilde Text1 kutusuna istediği bir değeri elle yazabileceği gibi UpDown aracılığı ile de artırıp azaltabilir.

SysInfo* (Sistem Bilgisi)

SysInfo kontrolünü kullanarak hem Windows hakkında bilgi alabilirsiniz hem de Windows çalışırken yapılan değişikliklerden programınızın haberi olur. SysInfo kontrolünü kullanabileceğiniz alanları şöyle özetleyebiliriz.

- Sistem ayarları değiştiğinde programınızın haberi olur. Örneğin ekran çözünürlüğü, saat veya buna benzer diğer ayarlarda bir değişiklik olduğunda bunu öğrenebilir ve programınızda gerekli değişiklikleri yapabilirsiniz.
- Özellikle notebook türü bilgisayarlarda akünün durumundan haberdar olabilir ve buna göre programınızı ayarlayabilirsiniz. Örneğin akü bitmek üzereyken bütün önemli verilerin kaydedilmesi sağlanabilir.
- Bilindiği gibi tak çalıştır uyumlu bazı aygıtlar Windows çalışırken de takılabilmekte ve çıkarılabilmektedir. Bu tür bir cihaz takıldığında veya çıkarıldığında programınızın bundan haberi olur.
- Windows'un aktif ayarlarını öğrenebilirsiniz.

Properties

ACStatus

Bilgisayar bir güç kaynağı veya akü ile besleniyorsa şu an kaynağın devrede olup olmadığı bu özellikle öğrenilebilir.

Bu özellikten geriye dönen değer 1 ise akü kullanımdadır.

Özelliğin değeri 0 ise akü kullanıma girmemiştir, yani bilgisayar bir prize bağlı olarak çalışıyordur.

Özelliğin değeri 255 ise bilgisayarın bir güç kaynağına bağlı olup olmadığı bilinmemektedir. Yani bir güç kaynağına bağlı olsa bile bunda Windows'un haberi yoktur.

ÖRNEK: Örnek olarak akünün veya güç kaynağının durumunu sürekli olarak bildirecek bir program yapalım. Örneğimiz için form üzerine bir Timer ve SysInfo kontrolü yerleştirin.

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft SysInfo Control 6.0" seçeneğini işaretlemeniz veya Browse düğmesi ile SYSINFO.OCX dosyasını bulup projeye eklemeniz gerekir.


```
'ÖRNEK: ACStatus
Private Sub Form_Load()
 Timer1.Interval = 1000
End Sub

Private Sub Timer1_Timer()
 Select Case SysInfo1.ACStatus
  Case 0
   Caption = "Akü veya güç kaynağı devrede değil"
  Case 1
   Caption = "Akü veya güç kaynağı devrede"
  Case 255
   Caption = "Akü veya güç kaynağı olup olmadığı bilinmiyor"
 End Select
End Sub
```

BatteryFullTime, BatteryLifeTime, BatteryLifePercent

BatteryFullTime: Bu özellik güç kaynağının tam dolu iken bilgisayarı ne kadar süre besleyebileceğini verir. Eğer bu özellikten geriye &HFFFFFFFF değeri dönüyorsa güç kaynağının tam dolu iken ne kadar süre çalıştırabileceği bilinmiyordur.

BatteryLifeTime: Bu özellik ise güç kaynağının kalan gücünün ne kadar süre bilgisayara güç verebileceğini bildirir. Yine bu özellikten geriye &HFFFFFFFF değeri dönüyorsa güç kaynağının ne kadar süre çalıştırabileceği bilinmiyordur.

BatteryLifePercent: Bu özellik ise kalan akü gücünün % olarak bildirir. Eğer bu özellikten 255 değeri dönüyorsa akü durumu bilinmiyordur.

ÖRNEK: Örnek olarak akünün veya güç kaynağının toplam süresini ve kalan süresini sürekli olarak bildirecek bir program yapalım. Örneğimiz için form üzerine bir Timer ve SysInfo kontrolü yerleştirin.

```
'ÖRNEK: BatteryFullTime, BatteryLifeTime, BatteryLifePercent

Private Sub Form_Load()
 Timer1.Interval = 1000
End Sub

Private Sub Timer1_Timer()
 Dim kalansüre, toplamsüre
 If SysInfo1.BatteryLifeTime <> &HFFFFFFFF Then
  kalansüre = Format(TimeSerial(0,0,SysInfo1.BatteryLifeTime),
"h:mm")
  Caption = "Kalan süre:" & kalansüre
 End If
 If SysInfo1.BatteryFullTime <> &HFFFFFFFF Then
  toplamsüre = Format(TimeSerial(0, 0, SysInfo1.BatteryFullTime),
"h:mm")
  Caption = Caption + " /Toplam süre:" & toplamsüre
 Else
```



```
  Caption = "Güç kaynağının veya akünün süresi bilinmiyor"
 End If
 If SysInfo1.BatteryLifePercent <> 255 Then
  Caption = Caption & " %" & SysInfo1.BatteryLifePercent
 End If
End Sub
```

BatteryStatus

Bu özellik aracılığı ile akünün veya güç kaynağının şarj durumu öğrenilebilir. Bu özellikten geri dönen değerler ve anlamları şunlardır:

1	Tam şarjlı
2	Düşük şarjlı
4	Şarj kritik durumda
8	Şarj ediliyor
128	Sisteme bağlı bir akü veya güç kaynağı yok veya bilinmiyor
255	Şarj durumu bilinmiyor

Şarj durumu değiştiğinde **PowerStatusChanged** olayı meydana gelir. Şarj durumunu kontrol etmek için gerekli kod da bu olaya yazılabilir.

```
'ÖRNEK: BatteryStatus
Private Sub SysInfo1_PowerStatusChanged()
 Select Case SysInfo1.BatteryStatus
  Case 1
   Caption = "Şarj: Tam"
  Case 2
   Caption = "Düşük şarj"
  Case 4
   Caption = "Şarj durumu kritik"
  Case 8
   Caption = "Şarj ediliyor"
  Case Else
   Caption = "Şarj durumu bilinmiyor"
 End Select
End Sub
```

OSPlatform,OSVersion,OSBuild

Bu özelliklerle kullanılan Windows hakkında bilgi edinilebilir.

OSPlatform: Bu özellik kullanılan Windows platformunu verir.

0	Win32s (Windows 3.1)

1 Windows 95/98

2 Windows NT

OSVersion: Bu özellikle kullanılan Windows'un tam versiyonu öğrenilebilir. (3.1, 4.0 vb) Windows 95'in versiyonu 4.0, 98'in versiyonu ise 4.1 olarak geçer.

OSBuild: Bu özellik aracılığı ile Windows'un yapım numarası öğrenilebilir. (95, 98 gibi)

```
'ÖRNEK: OSPlatform,OSVersion,OSBuild
Sub Form_Load
 Select Case SysInfo1.OSPlatform
  Case 0
   Caption = "Windows 3.1+Win32s" & "," & SysInfo1.OSBuild
  Case 1
   Caption = "Windows 9x, ver. " & SysInfo1.OSVersion & "," &
SysInfo1.OSBuild
  Case 2
   Caption = "Windows NT, ver. " & SysInfo1.OSVersion & "," &
SysInfo1.OSBuild
 End Select
End Sub
```

ScrollBarSize

Windows'ta bir çok ayarı olduğu gibi kaydırma çubuklarının genişlikleri de kullanıcı tarafından ayarlanabilmektedir. Bu özellik aracılığı ile kaydırma çubuklarının genişlikleri öğrenilebilir.

WorkAreaHeight, WorkAreaWidth, WorkAreaLeft, WorkAreaTop,

Bildiğiniz gibi Windows 9x ve Windows NT altında bulunan görev çubuğu masa sütünün bir kısmını kaplamakta ve görev çubuğunun yeri değiştirilebilmektedir. Görev çubuğundan geriye kalan ekran ise diğer programlar tarafından kullanılabilmektedir. Bu özellikler aracılığı ile masaüstünün kullanılabilir alanının koordinatları öğrenilebilir.

Ekran çözünürlüğü değiştiğinde DisplayChanged olayı meydana gelir. Bu olaya yazılacak kodla, gerekiyorsa program yeni koordinatlara uyarlanır.

Renk sayısı değiştirildiğinde ise SysColorsChanged olayı meydana gelir.

Bölüm 3

İnternet Kontrolleri

İnternet Kontrolleri

Visual Basic ile birlikte gelen kontrolleri kullanarak internet üzerinde çalışan uygulamalar yapmak mümkündür. Visual Basic ile gelen kontroller şunlardır.

Winsock: Bu kontrol MSWINSCK.OCX dosyası içinde bulunur. Bu kontrol kullanılarak internet üzerinden haberleşme yapacak programlar yazılabilir. Örneğin bir sohbet programı yapılabileceği gibi, internet üzerinden satış yapabilen programlar da bu kontrol aracılığı ile yapılabilir.

Inet: Bu kontrol MSINET.OCX dosyası içinde bulunur. Bu kontrol kullanılarak internetteki web ve ftp bölgelerine bağlanılabilir. Bir web sayfası alınıp görüntülenebileceği gibi, bir ftp sitesine bağlanılarak oradaki dosyalar görülebilir, oradan dosya alınabilir ve oraya dosya gönderilebilir.

MAPISession, MAPIMessages: Bu iki kontrol MSMAPI32.OCX dosyası içinde bulunur. Bu kontroller kullanılarak email alma ve gönderme işlemi yapılabilir.

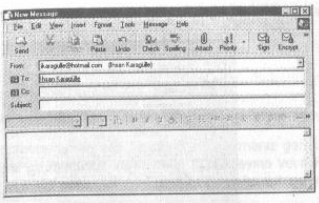
Bu dört kontrol aracılığı ile internet üzerinde işlem yapacak programları yapabilirsiniz. Ancak çok basit bazı işlemler için bu kontrolleri kullanmanız da gerekmez. Örneğin, yaptığınız programlara kendi email adresinizi ve varsa web adresini ekleyerek kullanıcının bu adreslere programınız içinden bağlanılabilmesini sağlayabilirsiniz. Bu işlemler için internet kontrollerini kullanmanız gerekmez. Çok basit bir satır ile kullanıcının size email gönderebilmesini veya web sayfanızı ziyaret edebilmesini sağlayabilirsiniz.

Windows'la birlikte gelen start isimli program email adresleri ve web sayfaları için gerekli programları çalıştırabilmektedir. Programınızın içinden Shell komut ile Start programını çalıştırmanız bu işlemi yapması için yeterlidir.

Örneğin

```
call Shell("start mailto:ikagulle@hotmail.com")
```

satırını programınız içinde kullanırsanız yüklü olan email programı (Outlook Express, Netscape Navigator veya diğerleri) otomatik olarak çalıştırılacak ve verilen email adresi gerekli kısma yazılacaktır.



Bu aşamadan sonra kullanıcıya kalan konuyu ve mesajı yazmaktır.

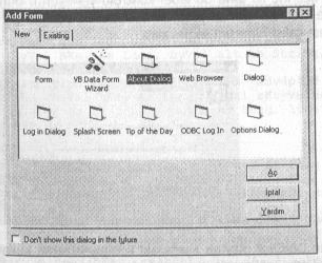
Aynı şekilde bir web sayfasına girebilmek için programınız içinden aşağıdaki satırı programınızda kullanabilirsiniz.

```
x = Shell("start www.turkmenkitabevi.com")
```

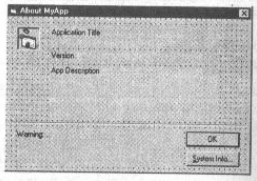
Bu işlemin sonucunda da yüklü olan Browser programı (Internet Exploer, Netscape Navigator veya diğerleri) otomatik olarak devreye girecek ve verdiğiniz adrese bağlanmaya çalışacaktır.

ÖRNEK:Örneğin, yaptığımız bir programın Hakkında kısmında bu bilgilere de yer verelim ve kullanıcının bu adreslere otomatik olarak ulaşabilmesini sağlayalım.

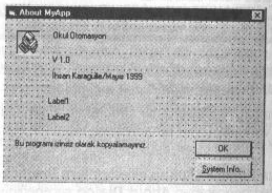
Programımıza hakkında kutusu eklemek için **Project-Add Form** menüleri ile açılan aşağıdaki pencereden **About Dialog** seçeneğini tıklayın.



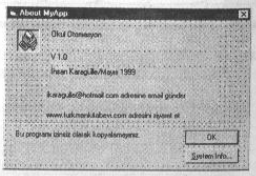
Bu işlem sonucunda aşağıdaki gibi Hakkında formu otomatik olarak oluşacaktır.



Formdaki Labellere programınızla ilgili bilgileri yazın ve resim kutusuna da uygun bir simge yerleştirin.



Formun üzerine iki tane de label kontrolü yerleştirin. Bunların içine email adresimizi ve web adresimizi yazarak kullanıcının bunları çift tıklaması halinde ilgili programın çalıştırılmasını sağlayalım.



Label'lerin DblClick olayına aşağıdaki kodları yazarak kullanıcının bunları çift tıklaması halinde ilgili programın çalıştırılmasını sağlayalım:


```
Private Sub Label1_Click()
 Call Shell("start mailto:ikaragulle@hotmail.com")
End Sub

Private Sub Label2_Click()
 x = Shell("start www.turkmenkitabevi.com")
End Sub
```

Ayrıca bu formu programınızın ana formundan çağırmanız gerektiğini unutmayın. Hakkında isimli bir menünüz varsa onun Click olayına veya Hakkında isimli bir düğmeniz varsa onun Click olayına **Form2.Show** komutunu yazmanız gerekir.

İnternet Bağlantısını Kontrol Etme

İnternet bağlantısının aktif olup olmadığını kontrol etmek için VB'de herhangi bir komut bulunmaz. Ancak internetle ilgili bir kontrol çalıştırıldığında bunu kontrol ederek aktif bağlantı yoksa, önce bağlantı kurmak için bağlantı penceresini açar. Yine de siz bir komutu kullanmadan önce internete aktif bağlantı olup olmadığını öğrenmek isterseniz bazı API'ler aracılığı ile regisrty dosyasında bulunan System\CurrentControlSet\Services\RemoteAccess yolu altındaki "Remote Connection" anahtarının değerini okuyarak aktif bağlantının olup olmadığını anlayabilirsiniz.

ÖRNEK: Bilgisayarın internete bağlı olup olmadığını anlayacak kod.

```
Option Explicit
Private Const HKEY_LOCAL_MACHINE = &H80000002
Private Declare Function RegCloseKey Lib "advapi32.dll" (ByVal
hKey As Long) As Long
Private Declare Function RegOpenKey Lib "advapi32.dll" Alias
"RegOpenKeyA" (ByVal hKey As Long, ByVal alt As String, hKey As
Long) As Long
Private Declare Function RegQueryValueEx Lib "advapi32.dll" Alias
"RegQueryValueExA" (ByVal hKey As Long, ByVal sKeyValue As
String, ByVal lpReserved As Long, lpType As Long, lpData As Any,
nSizeData As Long) As Long

Private Sub Form_Load()
   Dim hKey As Long, lpData As Long, nSizeData As Long
   Dim alt, anahtar
   alt = "System\CurrentControlSet\Services\RemoteAccess"
   anahtar = "Remote Connection"
   If RegOpenKey(HKEY_LOCAL_MACHINE, alt, hKey) = 0 Then
      lpData = 0&
      nSizeData = Len(lpData)
      If RegQueryValueEx(hKey, anahtar, 0&, 0&, lpData,
nSizeData) = 0 Then
```



```
         If lpData <> 0 Then
          MsgBox ("İnternet bağlantısı aktif")
         Else
          MsgBox ("İnternet bağlantısı aktif değil")
        End If
      End If
      RegCloseKey (hKey)
   End If
End Sub
```


Winsock (İnternet Kontrolü)*

Bu kontrol kullanılarak TCP/IP veya UDP protokolü aracılığıyla internetteki bir bilgisayara bağlantı kurulabilir, mesaj veya dosya alınıp gönderilebilir.

Ana Makine olarak Winsock kullanma

Winsock kontrolü Server olarak kullanılacaksa şu yöntemler izlenir:

1. LocalPort özelliğine server olarak kullanılacak programın kullanacağı bir port numarası verilir. Bu numara, o an kullanılmayan ve meşhur programlar** tarafından kullanılmayan herhangi bir numara olabilir.

```
Winsock1.LocalPort = 1251
```

2. Listen metodu ile Winsock aktif hale getirilir ve bu porttan gelecek mesajları dinlemesi sağlanır.

```
Winsock1.Listen
```

3. ConnectionRequest olayına yazılacak kodla server programına gelen bağlantı istekleri kabul edilir.

```
Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
 If Winsock1.State <> sckClosed Then Winsock1.Close
 Winsock1.Accept requestID
 Label1 = "Bir bağlantı isteği geldi"
End Sub
```

4. SendData metodu ile bağlantı sağlayan client programlara mesaj gönderilebilir.

```
Winsock1.SendData Text1.Text
```

5. Client makinalardan gelen mesajları almak için DataArrival olayına kod yazılır. GetData metodu ile gelen mesaj alınabilir.

```
Private Sub Winsock1_DataArrival (ByVal bytesTotal As Long)
 Dim s As String
 Winsock1.GetData s
```

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Winsock Control" seçeneğini işaretlemeniz veya Browse düğmesi aracılığı ile MSWINSCK.OCX dosyasını bulup projeye eklemeniz gerekir.

```
 Text2.Text = s
End Sub
```

6. Error olayına yazılacak kodla muhtemel hata mesajları takip edilebilir.

```
Private Sub Winsock1_Error(ByVal Number As Integer, Description
As String, ByVal Scode As Long, ByVal Source As String, ByVal
HelpFile As String, ByVal HelpContext As Long, CancelDisplay As
Boolean)
 MsgBox "Şu hata oluştu : " & Description
End Sub
```

Terminal olarak Winsock kullanma

Winsock kontrolü terminal olarak kullanılacaksa şu yöntemler izlenir:

1. RemoteHost özelliğine bağlanılacak serverin ip adresi veya domain adresi verilir.

```
Winsock1.RemoteHost = "195.112.5.14"
```

2. RemotePort özelliğine, bağlanılacak serverin LocalPort özelliğine verilen port numarası verilir. Yani bağlanacağınız program LocalPort olarak hangi portu kullanıyorsa, terminal programı da RemotePort olarak o portu kullanmalıdır.

```
Winsock1.RemotePort = 1251
```

3. Connect metodu ile bağlantı işlemi başlatılır.

```
Winsock1.Connect
```

4. SendData metodu ile server programa mesaj gönderilebilir.

```
Winsock1.SendData Text1.Text
```

5. DataArrival olayına yazılacak kodla ana makinadan gelen mesajlar GetData metodu atacılığıyla alınabilir.

```
Private Sub Winsock1_DataArrival (ByVal bytesTotal As Long)
 Dim s As String
 Winsock1.GetData s
 Text2.Text = s
End Sub
```

6. Error olayına yazılacak kodla muhtemel hata mesajları takip edilebilir.

** Standart olarak kullanılan bazı portların listesini görmek için WINDOWS dizini altındaki SERVICES dosyasına bakabilirsiniz.


```
Private Sub Winsock1_Error(ByVal Number As Integer, Description
As String, ByVal Scode As Long, ByVal Source As String, ByVal
HelpFile As String, ByVal HelpContext As Long, CancelDisplay As
Boolean)
 MsgBox "Şu hata oluştu : " & Description
End Sub
```

İnternet üzerinde client-server (anamakina, terminal) olarak çalışacak programlar yazarken bunları test etmeniz de gerekecektir. Test işlemi için iki ayrı bilgisayar kullanmanız çoğu zaman gerekmez. Her terminal programını, hem de server programını kendi bilgisayarınızda çalıştırdıktan sonra, Terminal programının bağlanacağı IP adresi olarak kendi IP numaranızı verirseniz iki program internet üzerinden birbirine bağlanmış olur. Böylece yaptığınız programları kolayca test edebilirsiniz.

Ayrıca bir program her server hem de client görevi görebilir. Yani hem kendisine bağlantı kabul edebilir, hem de başka bir yere bağlantı kurabilir. Bu işlem için iki tane Winsock kontrolü kullanılır. Birinci Winsock kontrolü Listen konumuna getirilerek bağlantıları dinlemeye alınır, diğeri ile de Connect metodu aracılığı ile bağlantı kurdurulabilir.

Properties

Protocol

Bu özellik kullanılarak bağlantı için hangi protokolün kullanılacağı belirlenir.

0-sckTCPprotocol: TCP/IP protokolü.

1-sckUDPprotocol: UDP protokolü.

TCP/IP protokolünde bir bilgisayarla haberleşebilmek için önce onunla bağlantı kurulması gerekir. UDP protokolünde ise herhangi bir bağlantı gerekmeksizin haberleşme yapılabilir. Ayrıca TPC/IP protokolünde bilginin karşı tarafa ulaşıp ulaşmadığı kontrol edilirken, UDP protokolünde bu kontrol yapılmaz. UDP protokolü hızlı, TCP/IP ise güvenlidir. Dosya, resim vb. bilgiler TCP/IP protokolü ile, mesaj gibi bilgiler ise UDP protokolü aracılığıyla gönderilmesi daha uygundur.

RemoteHost

Bağlantı kurulacak veya haberleşilecek bilgisayarın adresidir. Bu adres www.abc.com gibi bir dns adresi olabileceği gibi 195.12.11.101 gibi bir IP adresi de olabilir.

RemotePort, Local Port

Windows altında, internet altındaki haberleşme işlemleri portlar aracılığı ile yapılır. İnternette haberleşmek isteyen her program o an kullanılmayan bir portu kullanarak haberleşmeyi gerçekleştirebilir.

LocalPort ile kendi bilgisayarınızda kullanacağınız port adresini, **RemotePort** ile de bağlantı kuracağınız bilgisayarda kullanılacak port adresini verebilirsiniz.

```
Winsock1.Protocol=0 'TCP/Ip protokolü
Winsock1.RemoteHost="115.02.11.211"
Winsock1.LocalPort=1020
Winsock1.RemotPort=1010
```

İki program internet aracılığı ile bağlantı kurarken birinin (serverin) LocalPort özelliğine verilen değer diğerinin (clientin) RemotePort özelliğine verilmelidir.

State

Winsock kontrolünün durumu bu özellikle öğrenilir. Özellikle bazı komutları vermeden önce **State** özelliği ile durumu kontrol etmek ve durum o komuta uygunsa komutu kullanmak gerekir. Örneğin Listening durumunda iken Connect metodu ile bir bağlantı kurmaya çalışmak hata oluşturacaktır.

Bu özelliğin alabileceği değerler ve anlamları şunlardır.

sckClosed	0	Kapalı
sckOpen	1	Açık
sckListening	2	Port dinleniyor. (Bağlantı bekleniyor)
sckResolvingHost	4	Bağlantı için verilen adres çözülüyor.
sckHostResolved	5	Verilen adres çözüldü.
sckConnecting	6	Bağlanıyor.
sckConnected	7	Bağlandı.
sckClosing	8	Karşı taraf bağlantıyı kapatıyor.
sckError	9	Hata oluştu.

Örneğin karşı tarafa veri göndermeden önce bu özellik kontrol edilerek bağlantının olup olmadığı kontrol edilebilir.

```
If Winsock1.State = 7 Then 'bağlı ise
  Winsock1.SendData Text1.Text
End If
```

Aynı şekilde yeni bir bağlantı için Connect metodu kullanılmadan önce varolan bir bağlantının olup olmadığı, varsa kapatılması sağlanmalıdır. Bu ve buna benzer bir çok durumda oluşabilecek hataları önlemek için bu özellik kontrol edilmelidir.

Methods

Listen, Accept

Karşı taraftan gelecek bir bağlantıyı kabul edebilmek için **Listen** metodu ile port dinlemeye alınmalıdır. Bu metod verildikten sonra **LocalPort** özelliği ile belirlenen porta gelen bağlantı istekleri dinlenir ve bir bağlantı isteği geldiğinde **ConnectionRequest** olayı meydana gelir. Bu olaya yazılacak kodda **Accept** metodu kullanılarak bağlantı kabul edilmelidir.

```
Winsock1.LocalPort = 1024
Winsock1.Listen
```

Yukarıdaki kodlardan sonra 1024 nolu port dinlenmeye başlayacaktır. Bu porta bir bağlantı olayı geldiğinde **ConenctionRequest** olayına aşağıdaki gibi bir kod yazılarak bağlantı isteği kabul edilebilir.

```
Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
 If Winsock1.State <> sckClosed Then Winsock1.Close
 Winsock1.Accept requestID
 Label1 = "Bir bağlantı isteği geldi"
End Sub
```

Connect (RemoteHost, RemotePort)

TCP/IP protokolü kullanılıyorsa haberleşmeye başlayabilmek için karşı taraftaki bilgisayarla bağlantı kurulmalıdır. Bağlantı işlemi **Connect** metodu ile yapılır.

RemoteHost parametresi ile verilen adrese, **RemotePort** parametresi ile verilen port üzerinden bağlantı sağlanmaya çalışılır. Eğer bu iki parametre verilmezse **RemoteHost** ve **RemotePort** parametreleri ile belirlenen değerler kullanılır.

```
Winsock1.Protocol=0 'TCP/Ip protokolü
Winsock1.RemoteHost="115.02.11.211"
Winsock1.LocalPort=1020
Winsock1.RemotPort=1010
Winsock1.Connect
```

Bağlantı sağlandıktan sonra **Connect** olayı meydana gelir. Eğer bir hata oluşursa **Error** olayı meydana gelecektir. Bağlantı koptuğunda ise **Close** olayı meydana gelir.

Close

Bağlantıyı kesmek için bu metod kullanılır. Bu işlem başarılırsa **Close** olayı meydana gelecektir.

```
Private Sub Command1_Click()
 Winsock1.Close 'bağlantı varsa kapat
End Sub
```

SendData data

Bağlantı kurulan bilgisayara veri göndermek için bu metod kullanılır.

```
If Winsock1.State = 7 Then 'bağlı ise
 Winsock1.SendData Text1.Text
End If
```

GetData data, [tip,] [uzunluk]

Karşı taraftan bir veri geldiğinde bu metod kullanılarak gelen bilgi alınır. Karşı taraftan bir veri geldiğinde **DataArrival** olayı çalışacağı için bu metodu o olayda kullanmak gerekir.

```
Dim s
GetData s
```

Tek başına yukarıdaki gibi bir kodla gelen bilgi alınabileceği gibi istenirse son iki parametre ile veri tipi ve uzunluk da belirlenebilir.

Events

ConnectionRequest(ByVal requestID As Long)

Bir bağlantı isteği geldiğinde bu olay meydana gelir. Eğer bağlantı isteği kabul edilecekse **Accept** metodu ile bu bildirilmelidir.

Ayrıca **State** özelliği ile aktif bir bağlantının olup olmadığı kontrol edilmeli ve yeni bir bağlantı kabul edilmeden önce, önceki bağlantı **Close** metodu ile kapatılmalıdır.

```
Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
 'Yeni bir bağlantıyı kabul etmeden önce bağlantı varsa onu kes
 If Winsock1.State <> sckClosed Then Winsock1.Close
 Winsock1.Accept requestID
```



```
End Sub
```

Connect()

Bağlantı sağlandığında bu olay meydana gelir.

```
Private Sub Winsock1_Connect()
 Label1 = Winsock1.RemoteHost & "adresine bağlantı sağlandı"
End Sub
```

Error(ByVal Number As Integer, Description As String, ByVal Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As Long, CancelDisplay As Boolean)

Bir hata meydana geldiğinde bu olay meydana gelir. Bu olaydaki **Number** parametresi ile hatanın kodu öğrenilebileceği gibi **Description** parametresi ile de hatanın mesajı verilebilir.

```
Private Sub Winsock1_Error(ByVal Number As Integer, Description
As String, ByVal Scode As Long, ByVal Source As String, ByVal
HelpFile As String, ByVal HelpContext As Long, CancelDisplay As
Boolean)
 MsgBox "Şu hata oluştu : " & Description
End Sub
```

Close()

Bağlantı koptuğunda bu olay meydana gelir.

```
Private Sub Winsock1_Close()
 Label1 = Winsock1.RemoteHost & " Bağlantısı kesildi"
End Sub
```

DataArrival(ByVal bytesTotal As Long)

Bağlantı kurulan bilgisayardan bir veri geldiğinde bu olay meydana gelir. Bu olaydaki **bytesTotal** parametresi gelen bilginin uzunluğunu bildirir. Gelen bilginin kendisi ise **DetData** metodu ile alınır.

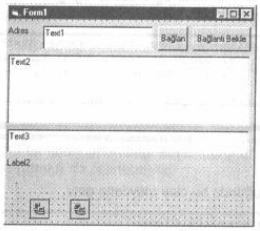
Örneğin karşıdan gelen bilgileri alıp bir text kutusunda gösterebilmek için aşağıdaki gibi bir kod yazabiliriz. (Gelen bilgilerin alt alta yazılabilmesi için Text kutusunun MultiLine özelliği True yapılmalıdır.)


```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
 Dim s As String
 enter = Chr(10) + Chr(13)
 Winsock1.GetData s
 Text2.Text = Text2.Text + s + enter
End Sub
```

Örnek: Sohbet programı

ÖRNEK: Örnek olarak internet üzerinde iki kişinin karşılıklı sohbet edebileceği bir program yapalım. Bu işlem için iki tane Winsock kontrolüne ihtiyacımız olacaktır. Bunlardan biri kendine gelecek bağlantıları kabul edecek, diğeri ise istene adrese bağlantı sağlayacaktır.

Örneğimiz için aşağıdaki formu hazırlayın ve **Text2** kontrolünün **MultiLine** özelliğini **True** yapın.




```
Option Explicit
Dim enter

Private Sub Command1_Click()
  Winsock2.RemoteHost = Text1
  Winsock2.RemotePort = 1024
  Winsock2.Connect
End Sub

Private Sub Command2_Click()
 If Winsock1.State = 2 Then 'bağlantı bekleniyorsa
  Winsock1.Close
  Command2.Caption = "Bağlantı Bekle"
 Else
  Winsock1.LocalPort = 1024
```

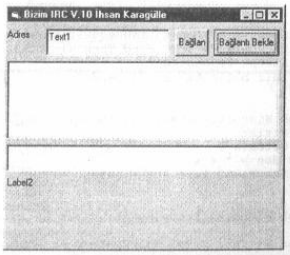


```
  Winsock1.Listen 'bağlantı bekle
  Command2.Caption = "Bağlantı Kes"
 End If
End Sub

Private Sub Form_Load()
 Caption = "Bizim IRC V.10 İhsan Karagülle"
 enter = Chr(13) + Chr(10)
 Text2 = ""
 Text3 = ""
End Sub

Private Sub Text3_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then 'entere basıldı ise gonder
  If Winsock1.State = 7 Then 'bağlı ise
   Winsock1.SendData Text3.Text
   Text2 = Text2 + "sizden > " + Text3 + enter
   Text3 = ""
  End If
   If Winsock2.State = 7 Then 'bağlı ise
   Winsock2.SendData Text3.Text
   Text2 = Text2 + "sizden > " + Text3 + enter
   Text3 = ""
  End If
 End If
End Sub
Private Sub Winsock1_Close()
 Label2 = "Bağlantı kesildi"
End Sub

Private Sub Winsock1_Connect()
 Label2 = "Bağlantı sağlandı"
End Sub

Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
 If Winsock1.State <> sckClosed Then Winsock1.Close
 Winsock1.Accept requestID
 Label2 = "Bir bağlantı isteği geldi"
End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
 Dim s As String
 Winsock1.GetData s
 s = "karsidan > " + s
 Text2.Text = Text2.Text + s + enter
End Sub

Private Sub Winsock1_Error(ByVal Number As Integer, Description
As String, ByVal Scode As Long, ByVal Source As String, ByVal
HelpFile As String, ByVal HelpContext As Long, CancelDisplay As
Boolean)
 MsgBox "Şu hata oluştu : " & Description
End Sub
```

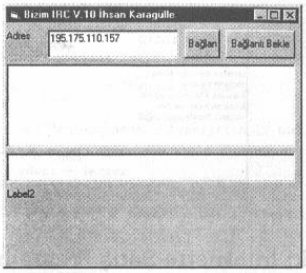
```
Private Sub Winsock2_Close()
 Label2 = "Bağlantı Kesildi"
End Sub

Private Sub Winsock2_Connect()
 Label2 = "Bağlantı sağlandı"
End Sub

Private Sub Winsock2_DataArrival(ByVal bytesTotal As Long)
 Dim s As String
 Winsock2.GetData s
 s = "karsidan > " + s
 Text2.Text = Text2.Text + s + enter
End Sub

Private Sub Winsock2_Error(ByVal Number As Integer, Description
As String, ByVal Scode As Long, ByVal Source As String, ByVal
HelpFile As String, ByVal HelpContext As Long, CancelDisplay As
Boolean)
 MsgBox "Şu hata oluştu : " & Description
End Sub
```

Programı çalıştırdıktan sonra, eğer birisinin size bağlanmasını istiyorsanız **Bağlantı Bekle** düğmesine basın. Eğer siz bir bağlantı kuracaksanız adres kutusuna bağlanacağınız kişinin Ip adresini yazın ve **Bağlan** düğmesine basın. Eğer karşı tarafta da bu program çalışıyorsa ve Bağlantı Bekle konumuna alınmışsa karşı tarafla bir bağlantı kurulacaktır.

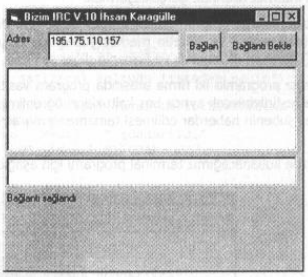
Örneği kendi bilgisayarınızda denemek isterseniz programı EXE haline getirin ve iki defa çalıştırın. Birinde **Bağlantı Bekle** düğmesine basın.



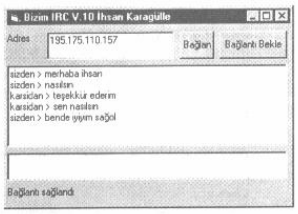
Diğerine ise kendi IP adresinizi yazıp bağlan düğmesine basın. (Ip adresinizi WINIPCFG programını çalıştırarak öğrenebilirsiniz)



İki program arasında bağlantı sağlanacak ve durum size label içinde bildirilecektir.



Bu işlemden sonra en alttaki Text kutusuna mesajınızı yazın ve Enter tuşuna basın. Yazdığınız mesaj karşıdaki programa gönderilecek ve onun gönderdiği mesajlar da ortadaki text kutusunda yazılacaktır.



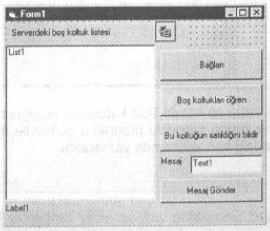
Örnek: İnternette Satış Programı

Şimdi İnternet üzerinden haberleşme yapacak bir program yazalım. Örneğimizdeki amacımız bir otobüs işletmesinin iki ayrı şehirdeki şubesi arasındaki bilet satışlarını internet üzerinde gerçekleştirmek.

Örneğimizde iki ayrı program kullanacağız. Bunlardan biri Server (ana makine) görevini üstelenecek ve şirketin merkez şubesinde bulunacak, diğer ise Client (terminal) görevini üstlenecek ve firmanın diğer şehirdeki şubesinde bulunacak.

Yapacağımız programla iki firma arasında program vasıtasıyla mesajlı haberleşme gerçekleştirilebilecek ayrıca boş koltukların öğrenilmesi ve bir bilet satıldığında da merkez şubenin haberdar edilmesi tamamen program aracılığı ile gerçekleştirilecek.

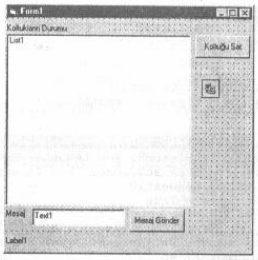
Örneğimizde kullanacağımız terminal programı için aşağıdaki formu hazırlayın.




```
'Örnek:Winsock
'Örnek terminal programı

Private Sub Command1_Click()
 Dim Server
 If Winsock1.State <> 7 Then 'kapalı değilse
  Winsock1.Close 'kapat
 End If
 Server = InputBox("Bağlanılacak bilgisayarın IP no", "IP:",
Server)
 If Len(Server) > 0 Then
  Winsock1.RemoteHost = Server
  Winsock1.RemotePort = 1024
  Winsock1.Connect
  Label1.Caption = Server + " adresine bağlanılmaya çalışılıyor"
 End If
End Sub

Private Sub Command2_Click()
 List1.Clear
 Winsock1.SendData "Koltuk Listesi"
 Label1.Caption = "Koltuk listesi istendi"
End Sub

Private Sub Command3_Click()
 Dim Mesaj As String
 If (List1.ListIndex < 0) Or (List1.ListIndex=List1.ListCount-1)
Then
  MsgBox ("Önce satılacak koltuğu listeden seçin")
 Else
  Mesaj = "Satıldı:" + List1.List(List1.ListIndex)
  Winsock1.SendData Mesaj
  Label1.Caption = Mesaj + " gönderildi"
  List1.RemoveItem List1.ListIndex
 End If
End Sub

Private Sub Command4_Click()
 Dim Mesaj As String
 If Len(Text1.Text) > 0 Then
  Mesaj = "Msg:" + Text1.Text
  Winsock1.SendData Mesaj
  Label1.Caption = "Mesaj gönderildi"
 End If
End Sub

Private Sub Form_Load()
 Caption = "EsInternet V1.0 Terminal İhsan Karagülle Mart 99"
 Winsock1.RemotePort = 1024
 List1.Clear
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
 Winsock1.Close
End Sub

Private Sub Winsock1_Close()
 Caption = Winsock1.RemoteHost + " adresinden bağlantı kesildi"
End Sub

Private Sub Winsock1_Connect()
 Label1.Caption = Winsock1.RemoteHost + " adresine bağlanıldı"
End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
 Dim s As String
 Static ListeGeliyor As Boolean
 Winsock1.GetData s
 If Len(s) > 5 Then
  If Mid(s, 1, 4) = "Msg:" Then
   Label1.Caption = s
   Exit Sub
  End If
 End If
 If Len(s) >= 10 Then
  If Mid(s, 1, 10) = "Liste Başı" Then
   s = Mid(s, 11)
   Open "liste.dat" For Output As #1
   Write #1, s
   Close #1
   Open "liste.dat" For Input As #1
   While Not EOF(1)
    Line Input #1, s
    List1.AddItem s
   Wend
   Close #1
  End If
 End If
End Sub

Private Sub Winsock1_Error(ByVal Number As Integer, Description
As String, ByVal Scode As Long, ByVal Source As String, ByVal
HelpFile As String, ByVal HelpContext As Long, CancelDisplay As
Boolean)
 Label1.Caption = Winsock1.RemoteHost + " adresine
bağlanılamıyor"
End Sub
```

Programın Server versiyonu için ise aşağıdaki formu hazırlayın.



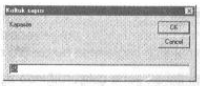

```
'Örnek:Winsock
'Örnek Server programı
Private Sub Command1_Click()
 Dim kn
 If List1.ListIndex < 0 Then
  MsgBox ("Önce satılacak koltuğu listeden seçin")
 Else
  kn = List1.ListIndex
  List1.List(kn) = "Satıldı " + Str(kn + 1) + " Kime:Buradan"
 End If
End Sub

Private Sub Command2_Click()
 Dim Mesaj As String
 If Len(Text1.Text) > 0 Then
  Mesaj = "Msg:" + Text1.Text
  Winsock1.SendData Mesaj
  Label1.Caption = "Mesaj gönderildi"
 End If
End Sub

Private Sub Form_Load()
 Dim ks, i
 Winsock1.LocalPort = 1024
 Winsock1.Listen 'bağlantı bekle
 ks = Val(InputBox("Kapasite", "Koltuk sayısı", "45"))
 List1.Clear
 Caption = "EsInternet V1.0 Ana Makina İhsan Karagülle Mart 99"
 For i = 1 To ks
  List1.AddItem i
 Next
 Label1.Caption = "Terminal Bağlantısı Yok"
 Command2.Enabled = False
End Sub
```



```
Private Sub Winsock1_Close()
 Label1.Caption = "Bağlantı kesildi"
 Command2.Enabled = False
End Sub

Private Sub Winsock1_Connect()
 Label1.Caption = Winsock1.RemoteHost + " bağlantı sağlandı"
End Sub

Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
  Label1.Caption = Winsock1.RemoteHost + " bağlantı isteği geldi"
  If Winsock1.State <> sckClosed Then Winsock1.Close
  Winsock1.Accept requestID
  Command2.Enabled = True
End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
 Dim s As String, i, kn
 Winsock1.GetData s
 If Len(s) > 5 Then
  If Mid(s, 1, 4) = "Msg:" Then
   Label1.Caption = s
  End If
 End If
 If s = "Koltuk Listesi" Then
  Label1.Caption = "Boş koltuk listesi gönderiliyor"
  Winsock1.SendData "Liste Başı" + Chr(13) + Chr(10)
  For i = 0 To List1.ListCount - 1
   If Val(List1.List(i)) > 0 Then
    Winsock1.SendData (Str(i + 1)) + Chr(13) + Chr(10)
   End If
  Next
  Winsock1.SendData "Liste Sonu"
  Label1.Caption = "Boş koltuk listesi gönderildi"
 End If
 If Mid(s, 1, 8) = "Satıldı:" Then
  kn = Val(Mid(s, 9, Len(s) - 8))
  List1.List(kn - 1) = "Satıldı " + Str(kn) + " Kime:" +
Winsock1.RemoteHostIP
  Label1.Caption = Str(kn)+" nolu koltuk "+Winsock1.RemoteHostIP
+ " adresine satıldı"
 End If
End Sub
```

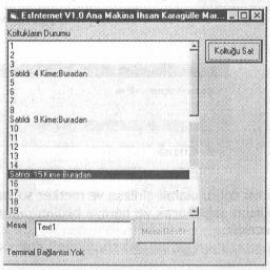
Şimdi programın çalışmasına gelelim. Öncelikle Server programını merkez şubede, terminal programını ise alt şubede çalıştırmak gerekiyor. Ayrıca her iki şubenin de internet bağlantısının açık olması gerekir.

Merkez şube server programını çalıştırdığında öncelikle koltuk sayısı sorulacaktır.



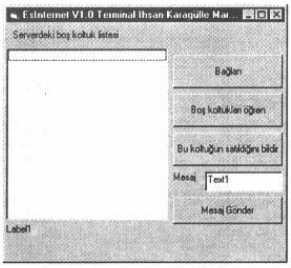
Koltuk sayısı girildikten sonra bütün koltuklar listeye eklenecektir.

Merkez şube satmak istediği koltuğu listeden seçerek **Koltuğu Sat** düğmesi ile istediği koltukları satabilecektir. Bu durumda koltuğun nereden satıldığı koltuk numarasının yanında aşağıdaki gibi yazılacaktır.



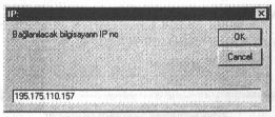
Merkez şubede çalışan yukarıdaki server programında başlangıçta herhangi bir terminal bağlantısının bulunmadığını programdaki Label içerisinde görebilmektesiniz. Eğer bir terminal size bağlanırsa bu durum da bildirilecektir.

Terminal programını ise alt şube çalıştırdığında aşağıdaki program açılacaktır.

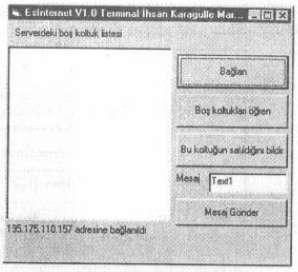


Terminalin yapması gereken, önce servere bağlanmaktır. Bu işlem için Serverin IP numarasını bilmesi gerekir. (Örneği kendi bilgisayarınızda deneyecekseniz her iki programı da çalıştırın ve ip numarası olarak kendi ip numaranızı girin. Kendi ip numaranızı Başlat-Çalıştır menüleri ile açılan pencereye WINIPCFG yazarak çalıştıracağınız programdan öğrenebilirsiniz.)

Programdaki **Bağlan** düğmesine basıldığında size serverin IP adresi sorulacaktır.

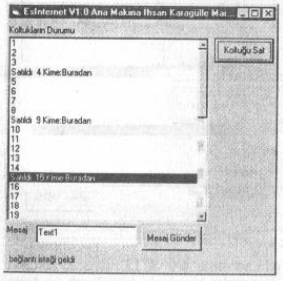


Eğer Ip numarası doğru olarak girilirse ve merkez şubede de server programımız çalışıyorsa bağlantı sağlanacak ve nereye bağlanıldığı aşağıdaki gibi Label içerisinde gösterilecektir.

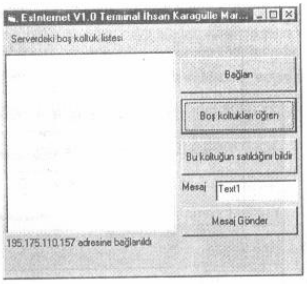


Şimdi boş koltukların numarasını serverden öğrenebilirsiniz. **Boş Koltukları Öğren** düğmesine bastığınızda program otomatik olarak servere internet üzerinde mesaj gönderecek ve hiç bir kullanıcı müdahalesi olmadan serverdeki programımız bu mesajı algılayıp otomatik olarak boş koltuk listesini terminale gönderecektir.

Serverdeki koltukların durumu aşağıdaki gibi olsun.

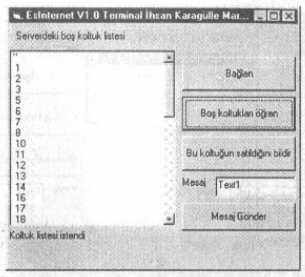


Alt şubedeki kullanıcı **Boş Koltukları Öğren** düğmesine bastığında:

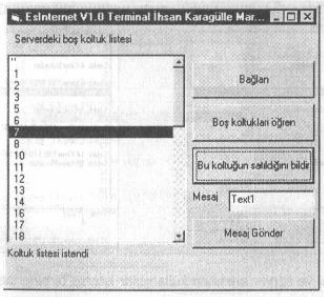


Serverdeki program bu mesajı algılayacak ve listeyi göndermeye başlayacaktır. Listeyi gönderdiğini Label içerisinde de kullanıcıya bildirecektir.

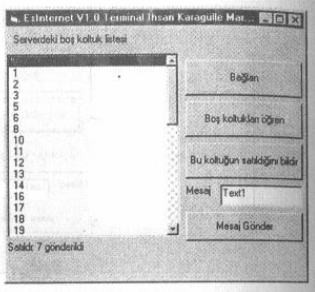
Görüldüğü gibi serverde 4-9 ve 15 numaralı koltuklar satılmıştır. Bu numaralar haricindeki numaralar terminale bildirilecek ve terminal programındaki listede sadece boş koltuk numaraları aşağıdaki gibi görülecektir.



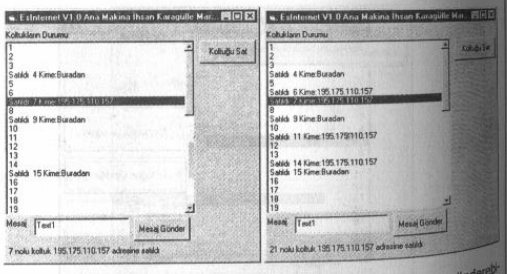
Şimdi alt şubedeki kullanıcı bu boş koltuklardan birini veya bir kaçını satabilir. Listeden bir koltuk numarası seçip **Bu koltuğun satıldığını bildir** düğmesine basın.



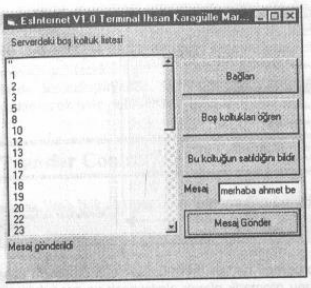
Satılan koltuk terminaldeki listeden çıkarılacaktır.



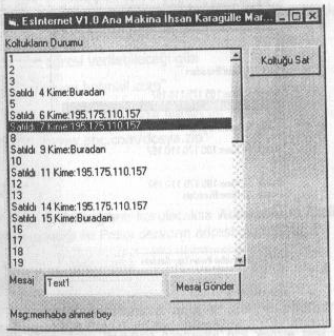
Aynı zamanda serverdeki makinada bu koltuğun satıldığı ve hangi adrese satıldığı aşağıdaki gibi anında listelenecektir.



Terminal ve server arasında kullanıcılar isterse birbirlerine mesaj da gönderebilirler.

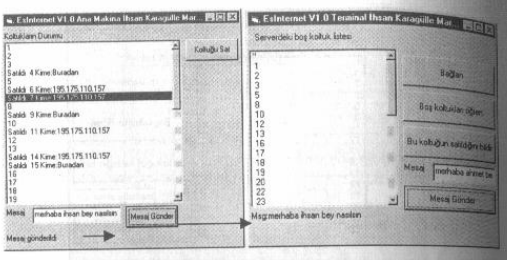


Yukarıdaki gibi terminal programını kullanan kişi **Mesaj** kutusuna mesajını yazıp **Mesaj Gönder** düğmesine bastığında:

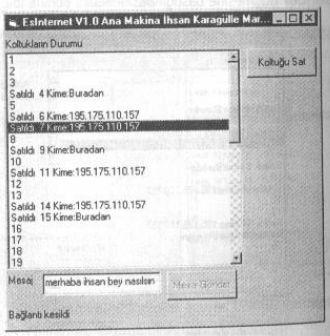


Server programında bu mesaj anında görülecektir. Yukarıda görüldüğü gibi Server programındaki Label içerisinde gelen mesaj görüntülenmektedir.

Server kullanıcısı da aynı şekilde terminaldeki şubeye mesaj gönderebilir.



Terminal programı hattan ayrıldığında veya programı kapattığında server programımız bunu otomatik olarak algılayacak ve aşağıdaki gibi Label içinde anında bildirecektir.



Programı test etmek için internete bağlı iki bilgisayara ihtiyacınız yok. Her iki programı da internet bağlantısı olan tek bir makinede çalıştırıp hem terminal hem de server olarak programın çalışmasını test edebilirsiniz. Kendi bilgisayarınızdaki Terminalden, yine kendi bilgisayarınızdaki servere bağlanırken kendi IP numaranızı yazmanız yeterlidir. Böylece iki program iletişimi internet üzerinden

gerçekleştirecektir. Böylece yazdığınız programları ikinci bir bilgisayara gerek kalmadan test edebilirsiniz.

Evet burada sadece iki bilgisayar arasındaki satış işlemini internet üzerinden yapabilecekleri ve mesajlaşabilecekleri örneği gördük. Aslında bu işlem yüzlerce bilgisayar arasında da gerçekleştirilebilir. Server programı düzenlenerek çok sayıda terminali destekleyecek hale getirilebilir.

Internet Transfer Control *

Bu kontrol kullanılarak Web bölgelerindeki sayfalar ve Ftp bölgelerindeki dosyalar transfer edilebilir.

Properties

URL

Bağlanılacak sitenin adresi bu özellikle belirlenir. Bağlantının başlaması için ise OpenUrl veya Execute metodu kullanılmalıdır.

Aşağıdaki gibi bir site adresi verilebileceği gibi

Inet1.URL = "HTTP://www.hotmail.com"

Aşağıdaki gibi, bir sitede bulunan dosya adresi de verilebilir.

Inet1.URL = "HTTP://www.abc.com/dosya.zip"

AccessType, Proxy

Eğer bir proxy aracılığı ile bağlantı kurulacaksa **AccessType** özelliğine 2 değeri verilmeli ve **Proxy** özelliği ile Proxy serverin adresi girilmelidir.

UserName, Password

Eğer bağlantı kurulacak site kullanıcı adı ve şifre gerektiriyorsa bu bilgiler **UserName** ve **Password** özelliği ile belirlenmelidir.

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Internet transfer Control" seçeneğini işaretlemeniz veya Browse düğmesi aracılığı ile MSINET.OCX dosyasını bulup projeye eklemeniz gerekir.

Normalde ftp servislerinin herkese açık olan ortak kısımlarına giriş için, kullanıcı adı olarak anonymous ve şifre olarak da email adresi verilmesi yeterlidir. Ancak bu iki özelliği boş bıraksanız da vb sizin yerinize bu iki bilgiyi gönderir. Ancak size ait özel kullanıcı adı ve şifreniz varsa bu iki özellikle, bağlanmadan önce bunları bildirmeniz gerekir.

Not: UserName ve Password özelliklerini belirledikten sonra URL özelliği ile adresi belirlerseniz belirlemiş olduğunuz UserName ve PassWord özellikleri silinecektir. Bu yüzden önce URL özelliğini, sonra UserName ve Password özelliğini belirleyin.

Protocol

Inet kontrolünün bağlantı için kullanacağı protokol bu özellikle belirlenir.

icUnknown	0	Bilinmeyen
icDefault	1	Varsayılan
icFTP	2	FTP
icHTTP	4	HTTP
icHTTPS	5	Secure HTTP

RemotePort

Bağlantı kurulacak port numarası bu özellikle belirlenir. Genellikle http için 80, ftp için ise 21 nolu portlar kullanılır.

RequestTimeout

Bağlantı için beklenecek süre bu özellikle belirlenebilir. Eğer bu özellikle belirlenen süre içerisinde bağlantı gerçekleşmezse **StateChanged** olayı meydana gelir.

Bu özelliğe verilecek değer saniye cinsindendir ve eğer 0 verilirse bu süre sonsuz olarak belirlenmiş olur. Genellikle bu özelliğe 60 değeri verilir. Yani 1 dakikalık bir süre içinde bağlantı sağlanamazsa işlem iptal edilir. Ancak bağlantı yaptığınız yer sürekli meşgulse bu süreyi artırabilirsiniz.

StillExecuting

Inet kontrolünün o anda bir iş yapıp yapmadığı (dosya indirmek gibi) bu özellikle öğrenilebilir. Eğer bu özelliğin değeri True ise Inet kontrolü meşguldür. Inet kontrolü bir işi yaparken başka bir işi yapması istenemez.

Methods

OpenURL (url [,datatype])

OpenURL metodu **url** parametresi ile belirlenen adresteki bilgiyi alır. Bu bilgi bir html sayfası olabileceği gibi, bir dosya veya ftp komutu sonucu oluşan bilgi de olabilir.

İstenirse **datatype** parametresi ile alınacak bilginin türü de belirlenebilir. Bu özelliğe 0 değeri verildiğinde (verilmezse de bu kabul edilir) bilgi string türünde, 1 değeri verildiğinde bayt dizisi türünde gelir.

ÖRNEK: Bir Web sayfasındaki HTML kodunu alıp-göstermek için:

```
Text1.Text = Inet1.OpenURL("http://www.microsoft.com")
```

ÖRNEK: Bir ftp sitesindeki dosyayı alıp-göstermek için:

```
Text1.Text=Inet1.OpenURL("ftp://ftp.intel.com/readme.txt")
```

ÖRNEK: Bir arama motoruna (CGI) arama yaptırıp sonucu göstermek için:

```
Text1.Text = Inet1.OpenURL("http://www.yahoo.com/cgi-bin/search.
exe?find=ihsan")
```

ÖRNEK: Bir ftp sitesindeki dosyayı binary formatta alıp-kaydetmek için:

```
Dim b () As Byte
b () = Inet1.OpenURL("ftp://ftp.intel.com/readme.txt", 1)
Open "C:\ readme.txt " For Binary Access Write As #1
Put #1, , b ()
Close #1
```

ÖRNEK: Bir web sayfasını alıp-kaydetmek için:

```
Open "c:\html" For Output As #1
Write #1, Inet1.OpenURL("http://www.intel.com/")
Close #I1
```


Execute url, operation

OpenURL metodu gibidir ancak ondan daha gelişmiş bir yapıya sahiptir. Bu metodla bir web bölgesindeki bilgiler alınabileceği gibi, bir ftp sitesine bağlanılarak dizin-dosya bilgileri, dizinlerde gezme, dosya alma, dosya gönderme gibi işlemler de yapılabilir.

URL parametresi ile adres belirlendikten sonra **operation** parametresi ile de çalıştırılmak istenen komut gönderilir.

FTP için kullanılabilecek komutlar

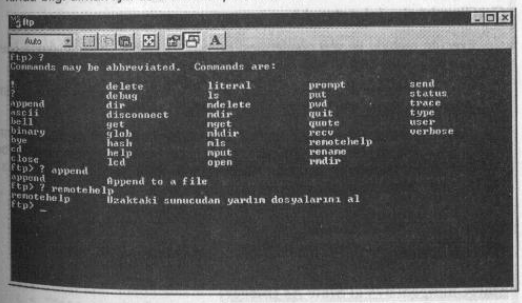
FTP bölgeleri tıpkı harddiskiniz gibidir. İçinde dizinler ve dosyalar bulunur. Bir dizine geçmek, dizindeki dosyaların listesini almak, dosya göndermek, dosya almak, dosya silmek gibi bir çok işlem yapılabilir. Bu işlemler için **Execute** metodunun ikinci parametresi olan **Operation** parametresine uygun FTP komutlarının verilmesi gerekir. Verilebilecek komutlar ve anlamları şöyledir:

Komut	Anlamı
CD dizin	Verilen dizine geçiş yapar. Inet1.Execute "ftp.intel.com", "CD pub" Inet1.Execute "ftp.intel.com", "CD msdos\driver"
CDUP	Bir üst dizine çıkar. CD.. komutu gibidir. Inet1.Execute "ftp.intel.com", "CDUP"
CLOSE	FTP bağlantısını kapatır. Inet1.Execute "ftp.intel.com", "CLOSE"
DELETE dosya	İsmi verilen dosyayı siler. Inet1.Execute "ftp.intel.com", "DELETE a.txt"
DIR dosya	Verilen isme uyan dosyaları listeler. Eğer bir dosya verilmezse bütün dosyaları listeler. İstenirse *, ? gibi joker karakterler de kullanılabilir. Gelen dizin bilgisini almak için **GetChunk** metodundan faydalanılır.
GET dosya1 dosya2	Dosya1 ismiyle belirlenen dosyayı FTP server'inden alarak dosya2 ismiyle belirlenen isimde bilgisayarınıza kaydeder. Inet1.Execute "ftp.intel.com", "GET a.txt c:\a.txt"
MKDIR dizin	İsmi verilen dizini ftp serverinde açar. Inet1.Execute "ftp.intel.com", "MKDIR yeni"
PUT dosya1 dosya2	FTP serverine dosya gönderek için kullanılır. Dosya1 gönderilecek dosyanın bilgisayarınızdaki yeri, dosya2 ise ftp serverindeki ismidir. Inet1.Execute "ftp.intel.com", "PUT c:\a.txt a.txt"

PWD	Aktif dizinin ismini verir. Gelen bilgiyi öğrenmek için **GetChunk** metodundan faydalanılır.
RECV dosya1 dosya2	GET komutu gibidir.
RENAME dosya1 dosya2	dosya1 isimli dosyanın ismini dosya2 olarak değiştirir.
RMDIR dosya1	İsmi verilen dizini siler.
SEND dosya1 dosya2	PUT metodu gibidir.
SIZE dosya	İsmi verilen dosyanın boyutunu gönderir.

Bazı komutları kullanabilmek için o ftp serverinde yetkinizin olması gerekir. herkese açık olan anonymous ftp sitelerinde silme, isim değiştirme gibi işlemler yapılamaz.

Diğer FTP komutlarını öğrenmek için Windows'la birlikte gelen ftp programını kullanabilirsiniz. Başlat-Çalıştır menülerine FTP yazarak ftp programını çalıştırın. Program çalışacak ve ftp> iletisi ile sizden bir komut bekleyecektir. **?** yazıp entere basarak kullanabileceğiniz komutların listesini öğrenebilir, bir komut hakkında bilgi almak için de **? komut** yazabilirsiniz.



HTTP için kullanılabilecek komutlar

Komut	Anlamı
GET	Url ile verilen sayfayı alır. Inet1.Execute "http://www.ab.com/index.htm","GET"
HEAD	Url ile verilen sayfanın sadece başlık kısmını alır. Inet1.Execute "http://www.ab.com/index.htm","HEAD"
POST	Form bilgisini postalar. Inet1.Execute "http://www.ab.com/anket.htm","POST",formbilgisi
PUT	Url ile verilen dosyanın üzerine yeni dosyayı gönderir. Inet1.Execute "http://www.ab.com/index.htm","PUT","indexyeni.htm"

GetChunk(size [,datatype])

Execute metodu ile veri göndermesi istenen bir komut çalıştırıldığında (get, dir, size vb.) gelen bilgi **GetChunk** metodu ile alınır.

Size parametresi ile alınacak bilginin uzunluğu belirlenir. **DataType** parametresi ile de gelen bilginin türü (0=string, 1=binary) belirlenir.

Execute metodu ile veri göndermesi istenen bir komut gönderdikten sonra, en uygun yöntem **StateChanged** olayına kod yazarak gelen bilgiyi burada **GetChunk** metodu ile almaktır. Çünkü istenen bilgi geldiğinde veya hata oluştuğunda **StateChanged** olayı meydana gelecektir.

Events

StateChanged(ByVal State As Integer)

Bağlantının durumunda değişiklik olduğunda bu olay meydana gelir. Bağlantının durumu ve istenen bilgilerin gelip gelmediği bu olaya yazılacak kodla takip edilebilir. Bu olaydaki **State** parametresi bağlantının durumu hakkında bilgi verir. Bu parametrenin alabileceği değerler ve anlamları şöyledir:

0-icNone	Bir değişme yok
1-icResolvingHost	Verilen adres araştırılıyor.
2-icHostResolved	Verilen adres bulundu.

3-icConnecting	Bağlantı yapılıyor.
4-icConnected	Bağlantı sağlandı.
5-icRequesting	Komut gönderiliyor.
6-icRequestSent	Komut gönderildi.
7-icReceivingResponse	Komuta cevap veriyor.
8-icResponseReceived	Komuta cevap verildi.
9-icDisconnecting	Bağlantı kesiliyor.
10-icDisconnected	Bağlantı kesildi.
11-icError	Hata oluştu.
12-icResponseCompleted	Komuta cevap verildi ve gerekli bilginin alınması tamamlandı.

ÖRNEK: Aşağıdaki gibi bir kodla meydana gelen değişiklikler takip edilerek bir Label içinde kullanıcıya bildirilebilir.

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
 Dim t
 Select Case State
  Case icNone:
   t = "Bir değişme yok"
  Case icResolvingHost:
   t = "Verilen adres araştırılıyor."
  Case icHostResolved:
   t = "Verilen adres bulundu."
  Case icConnecting:
   t = "Bağlantı yapılıyor."
  Case icConnected:
   t = "Bağlantı sağlandı."
  Case icRequesting:
   t = "Komut gönderiliyor."
  Case icRequestSent:
   t = "Komut gönderildi."
  Case icReceivingResponse:
   t = "Komuta cevap veriyor."
  Case icResponseReceived:
   t = "Komuta cevap verildi."
  Case icDisconnecting:
   t = "Bağlantı kesiliyor."
  Case icDisconnected:
   t = "Bağlantı kesildi."
  Case icError:
   t = "Hata oluştu."
   t = t & Inet1.ResponseCode & ":" & Inet1.ResponseInfo
  Case icResponseCompleted: t = "Komuta cevap verildi ve gerekli
bilginin alınması tamamlandı."
 End Select
 Label1.Caption = t
End Sub
```

Eğer bilgi **Execute** metodu ile istenmişse bilgiyi alacak kod **StateChanged** ola-

yına yazılmalıdır. Bu olaya yazılacak kodla **State** parametresinin değeri kontrol edilir. Bu değer **icResponseCompleted** (12) olduğunda istenen bilgi tamamlanmış demektir. **GetChunk** metodu ile tampondaki bilgi alınır.

Gelen bilgi binary türdense (program, resim gibi) gelen bilgi aşağıdaki gibi bir kodla dosyaya yazılabilir.

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
 Dim veri As Variant
 Select Case State
  Case icResponseCompleted ' 12
   Open "dosyaadi" For Binary Access Write As #1
   veri = Inet1.GetChunk(1024, icString)'1kblık kısmı al
   Do While LenB(veri) > 0'veri bitene kadar al
    Put #1, , veri'dosyaya yaz
    veri = Inet1.GetChunk(1024, icString)'sonraki 1kb al
   Loop
   Put #1, , veri
   Close #1
 End Select
End Sub
```

Gelen bilgi text türündense gelen bilgi aşağıdaki gibi bir kodla Text kutusu içinde gösterilebilir.

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
 Dim veri As Variant
 Select Case State
 Case icResponseCompleted ' 12
  veri = Inet1.GetChunk(1024, icString)'1kblık kısmı al
  Do While Len(veri) > 0'veri bitene kadar al
    Text1=Text1 & veri'Text1'e yaz
    veri = Inet1.GetChunk(1024, icString)'sonraki 1 kb al
  Loop
  Text1=Text1 & veri
End Select
```


Bölüm 4

Seri ve Paralel İletişim

Seri ve Paralel İletişim

Bilgisayarın dış dünyadaki cihazlarla iletişim kurmasının en yaygın yollarından ikisi seri ve paralel iletişimdir. Normal hayatta bildiğimiz bir çok cihaz bilgisayarla bu portlar üzerinden haberleşir.

Her bilgisayarda en az 2 Seri ve 1 Paralel port bulunur. Paralel porttan yapılan haberleşmeler daha hızlı, seri port ise daha yavaştır. Çünkü paralel portta bilgiler 8 bit'lik gruplar halinde gönderilirken, seri portta birer bitlik gruplar halinde peş peşe gönderilir. Bu yüzden paralel port seri porttan oldukça hızlıdır. Hız isteyen çevre birim cihazları paralel portu tercih ederken daha yavaş cihazlar veya hız ihtiyacı olmayan cihazlar seri portu tercih eder. Paralel portun bu hız avantajına rağmen önemli iki dezavantajı da vardır. Bunlardan biri paralel bağlantı da daha fazla kablo gerekmesi diğeri ise kablo mesafesinin 1.5 metreden uzun olamamasıdır.

Günlük hayatta kullandığımız bazı cihazların tercih ettikleri iletişim portları şunlardır: Yazıcı, tarayıcı, harici CD-Rom gibi hız gerektiren cihazlar paralel portu kullanır. Fare, Modem, Ethernet kartı, uzaktan kumanda algılayıcı gibi yüksek hız gerektirmeyen cihazlar ise seri portu kullanır. Aslında Modem ve Ethernet kartı için yüksek hız gerekir ancak paralel portun çok sayıda kablo gerektirmesi yüzünden mecburi olarak seri portu kullanırlar. Örneğin modemi paralel port ile bağlamak isteseydik 8 tane telefon hattı kullanmamız gerekirdi.

Visual Basic Seri/Paralel iletişim için iki farklı yöntem sunar:

Seri iletişim için MSComm kontrolü tasarlanmıştır. Bu kontrol aracılığı ile seri porta bağlı cihazları kontrol etmek oldukça kolaydır.

Paralel iletişim için ise Printer nesnesini sunar. Ancak bu genel bir paralel iletişim kontrolü değildir. Özel olarak bu nesne ile sadece yazıcı kullanılabilir.

Her iki iletişim türü için de VB'nin sunduğu diğer yol OPEN komutudur. Normalde dosyalama işlemlerinde kullandığımız bu komut aynı zamanda seri ve paralel iletişim için de kullanılabilmektedir.

Open komutu ile seri ve paralel iletişim*

Dosya açmak için kullandığımız OPEN komutunu seri ve paralel iletişim için de kullanabiliriz. Open komutunun formunu hatırlayalım:

Open "Dosya Adı" For ErisimModu As #DosyaNo

Buradaki dosya adı yerine seri portu temsil eden "COM1:" veya "COM2:", paralel portu temsil eden "LPT1:" veya "LPT2:" kullanılırsa bu portlar birer dosyaymış gibi kullanılabilir. Yani dosyaya yazmak için kullandığımız Print #, Write#, Put# gibi komutlar bu bilgileri porta gönderirken, dosyadan okumak için kullandığımız Input#, Get# gibi komutlar da porttan gelen bilgileri okur.

ÖRNEK: Örnek olarak **Open** komutu ile LPT1 portunu açalım ve **Write #** komutu ile bu porta bilgi gönderelim. Sonuçta gönderdiğimiz bilgiler eğer LPT1 üzerinde bir yazıcı varsa yazıcıdan çıkacaktır.

```
Private Sub Form_Load()
 Open "lpt1:" For Output As #1
  Write #1, "Open komutu ile yazıcı denemesi"
  Write #1, "1.Satır"
  Write #1, "2.Satır"
  Write #1, "3.Satır"
 Close #1
End Sub
```

Programı çalıştırdığınızda Write komut ile yazdırdığımız bilgiler yazıcıdan çıkacaktır. Eğer LPT1 üzerinde yazıcı değil de başka bir cihaz varsa bu bilgiler o cihaza gönderilecektir.

ÖRNEK: COM3'e bağlanmış bir modeme 4125308 nolu numarayı çevirmesini söyleyelim. Modeme bir numara çevirmesini söylemek için "ATDT telefonno" komutu kullanılır.

```
Private Sub Form_Load()
 Open "com3:" For Binary As #1
 Put #1, , "ATDT 4125308" & vbCr 'VbCr=enter tuşu
 Close #1
End Sub
```

* Open komutunun bu özelliği Dos ortamında çalışan Gwbasic, Quick Basic gibi dillerde bulunmasına rağmen Visual Basic dokümanlarında bahsedilmemekte, ancak bu yöntem şimdilik çalışmaktadır. Çünkü bu özellik, yani portların bir dosya gibi açılabilmesi DOS'un sağladığı bir özelliktir ve Windows, Dos ile olan bağlantısını tamamen kestiğinde bu yöntem de büyük bir ihtimalle çalışmayacaktır.

MSComm (Seri İletişim Kontrolü)*

Bu kontrol aracılığı ile Seri portlar kullanılabilir. Fare, Modem gibi seri portlara bağlanan aygıtlarla haberleşilebileceği gibi, daha çok endüstriyel amaçlı kullanılan ve seri port üzerinden bilgisayarla haberleşebilen ölçüm cihazı gibi cihazlar da bu kontrol aracılığı ile kumanda edilebilir.

Bu kontrolün kullanım adımlarını şu şekilde özetleyebiliriz.

1. **CommPort** özelliği ile kullanılacak olan portun numarası belirlenir.

```
MSComm1.CommPort = 2 'COM2'yi kullan
```

2. **Settings** özelliği ile kullanılacak portun ayarları yapılır.

```
MSComm1.Settings = "9600,N,8,1" Hız, parity, data, stop bit ayarı
```

3. **PortOpen** özelliğine True verilerek belirlenen port kullanıma açılır.

```
MSComm1.PortOpen = True'Portu aç
```

4. **Input** metodu ile porttan bilgi okunur

```
Text1.Text = MSComm1.Input
```

5. **OutPut** metodu ile porta bilgi yazılır.

```
MsComm1.Output = "Merhaba"
```

6. Gerekiyorsa **CommEvent** olayına yazılacak kodla portun durumu kontrol edilir.
7. **PortOpen** özelliğine False verilerek belirlenen port kapatılır.

```
MSComm1.PortOpen = False'Portu kapat
```

* Bu kontrolü kullanabilmek için Project-Components menüleri ile açılan pencereden "Microsoft Comm Control 6.0" seçeneğini işaretlemeniz veya Browse düğmesi aracılığı ile MSCOMM32.OCX dosyasını bulup projeye eklemeniz gerekir.

Properties

CommPort

Kullanılacak seri portun numarası bu özellikle belirlenir. Genellikle her bilgisayarda en az iki seri port bulunur ve bunlar COM1, COM2 şeklinde adlandırılır. Çoğu zaman da COM1 fare tarafından kullanılmaktadır. Genellikle COM2 portunu kullanmak daha az problem çıkmasına sebep olacaktır.

```
MSComm1.CommPort = 2 'COM2'yi kullan
```

Settings

Port kullanıma açılmadan önce bazı ayarlamalar yapılmalıdır. Bu özellik dört ayarı içinde bulundurur ve araya virgül koyularak bu dört ayar birlikte yapılır.

Settings="Baud Rate, Parity, Data Bit, Stop Bit"

Bu ayarlamalar şunlardır:

Baud Rate: Haberleşme için kullanılacak hızı belirler. Birim bps (bit Per second) cinsindendir ve saniyede gönderilecek bit sayısını belirler. Örneğin 9600 olarak belirlendiğinde saniyede 9600 bit ile haberleşileceğini gösterir. Verilebilecek hız değerleri şunlardır:

110, 300, 600, 1200, 2400, 9600(Default), 14400, 19200, 28800, 38400, 56000, 128000, 256000

Parity: Bilgi ile birlikte hata kontrolü için ek bir bit (parity biti) kullanılabilir. Parity'nin tek ve çift olmak üzere en çok kullanılan iki çeşidi vardır. Tek parity kullanılıyorsa gönderilen mesajdaki bitlerdeki 1'lerin sayısına bakılır. Eğer bu sayı çift ise parity biti 1 yapılarak 1'lerin sayısı tek yapılır. Eğer çift parity kullanılıyorsa mesajdaki 1'lerin sayısına bakılır. Bu 1'lerin sayısı tek ise parity biti 1 olarak bunların sayısını çift yapar. Diğer durumda 0 olarak 1'lerin sayısının çift olarak kalmasını sağlar.

Gönderilecek mesajın 10010110 olduğunu kabul edelim. Bu mesajdaki 1'lerin sayısı 4'tür.

Eğer çift parity kullanıyorsak 1'lerin sayısı zaten çift olduğu için parity biti 0 olur ve mesajın sonuna eklenir. 100101100 (çift parity).

Eğer tek parity kullanıyorsak 1'lerin sayısı çift olduğu için parity biti 1 olarak 1'lerin sayısının tek olmasını sağlar ve parity biti mesajın sonuna eklenir. 100101101(tek parity).

Sonuçta karşı taraf bu mesajı aldığında mesajdaki 1'lerin sayısı ile parity'yi kontrol eder. Eğer bir uyuşmazlık söz konusu ise yani tek parity kullanıldığı halde 1'lerin sayısı tek değilse veya çift parity kullanıldığı halde 1'lerin sayısı çift değilse bir hata olduğunu anlar.

Kullanılabilecek değerler ve anlamları şöyledir:

E	Çift Parity
M	Mark
N	(Default) Parity yok
O	Tek Parity
S	Space

Data Bit: Haberleşmede kullanılacak bit sayısını belirler. Şu değerlerden birini alabilir. 4,5,6,7,8 bit.

Stop Bit: Mesajın bittiğini belirtmek için kullanılacak bit sayısını belirler. Şu değerlerden birini alabilir. 1, 1.5, 2 bit

```
MSComm1.Settings = "9600,N,8,1" Hız, parity, data, stop bit ayarı
```

Yukarıdaki satır ile 9600 bps hızında, parity biti olmadan, 8 bitlik veriler ve 1 bitlik stop biti ile haberleşme yapılacağı bildirilmektedir.

PortOpen

CommPort özelliği ile belirlenen portu kullanıma açmak için bu özelliğe **True** değeri verilir. İşlem bittikten sonra portu serbest bırakmak için de bu özelliğe **False** değeri verilerek port kapatılır.

Eğer CommPort özelliği ile verilen port yoksa veya o an kullanımda ise hata oluşacaktır. Bu yüzden PortOpen özelliği ile portu açmadan önce hata yakalama kodları kullanmak yerinde olacaktır.

```
'ÖRNEK: PortOpen
Private Sub Form_Load()
 MSComm1.CommPort = 1
 MSComm1.Settings = "9600,N,8,1"
 On Local Error GoTo hata
 MSComm1.PortOpen = True
 Exit Sub
hata:
 MsgBox ("Port açılamadı: " & Error)
 Exit Sub
End Sub
```


OutPut

Açılan porta bilgi göndermek için bu özellik kullanılır. Portun ucunda bir modem varsa bu özellikle modeme komut da gönderilebilir.

Eğer string türü bilgiler gönderiliyorsa variant tipi değişkenler, dosya gibi binary türü bilgiler gönderiliyorsa byte diziler kullanılmalıdır.

ÖRNEK: Örneğin porta "Merhaba" mesajı göndermek için:

```
MSComm1.Output = "Merhaba"
```

ÖRNEK: Örneğin kullanıcının bastığı tuşları porta göndermek için:

```
Private Sub Form_KeyPress (KeyAscii As Integer)
  Dim x as Variant
  x= Chr$(KeyAscii)
  MSComm1.Output = x
End Sub
```

ÖRNEK: Açılan porta bir modem bağlı ise modeme bir numarayı çevirmesini söylemek için:

```
MSComm1.Output = "ATDT 4125139" & vbCr 'vbCr=Enter
```

InputLen

Porta gelen bilgiler alınıncaya kadar buffer'de tutulur. **InputLen** özelliği ile okuma yapılırken kaç karakterlik bloklar halinde okuma yapılacağı belirlenir. Eğer bu özelliğe 0 değeri verilirse buffer'deki bütün karakterlerin tek seferde okunacağı belirlenmiş olur.

InBufferCount

Buffer'de bekleyen karakter sayısı bu özellikle öğrenilir. Okuma yapılmadan önce bu özelliğe bakılarak buffer'de bekleyen karakter olup olmadığı anlaşılabilir. Ayrıca buffer boşaltılmak isteniyorsa bu özelliğe 0 değeri verilebilir.

InBufferSize

Normalde buffer'in uzunluğu 1024 baytdır ve buffer dolmadığı sürece problem çıkmaz. Ancak eğer bu değer yeterli gelmiyorsa **InBufferSize** özelliği ile buffer belleğini artırabilirsiniz.

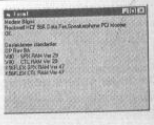
Input

Porta gelen ve buffer'de bekleyen bilgi bu özellikle öğrenilir. Okunan bilginin miktarı **InputLen** özelliği ile belirlenmiş karakter sayısı kadardır. Eğer **InputLen** özelliğine 0 değeri verilmişse buffer'de bekleyen bütün bilgiler alınır. Ayrıca Input özelliği ile gelen bilgi alınmadan önce **InBufferCount** özelliği ile buffer'de bilgi olup olmadığı kontrol edilmelidir.

```
Private Sub Command1_Click()
 Dim x as String
 MSComm1.InputLen = 0' Bufferdeki bütün karakterleri al
 If MSComm1.InBufferCount>0 Then 'Bufferde veri varsa
  x= MSComm1.Input 'Bilgiyi al
 End If
 Print x 'Ekrana yaz
End Sub
```

ÖRNEK: COM3 portuna bağlı bir modeme ATI4 komutunu göndererek modem ismini soralım ve gelen bilgiyi Input özelliği ile alalım. Ayrıca ATI6 komutu göndererek modemin desteklediği standartları öğrenelim.

```
Private Sub Form_Load()
 Show
 MSComm1.CommPort = 3
 MSComm1.PortOpen = True
 'Modem ismini sor
 MSComm1.Output = "ATI4" & vbCr
 MSComm1.InBufferCount = 0
 'Gelen bilginin tümünü tek seferde al
 MSComm1.InputLen = 0
 Do
  'Bilgi gelene kadar bekle
 Loop While MSComm1.InBufferCount <= 0
 Print "Modem Bilgisi:" & MSComm1.Input
 'Desteklenen standartları sor
 MSComm1.Output = "ATI6" & vbCr
 MSComm1.InBufferCount = 0
 Do
  'Bilgi gelene kadar bekle
 Loop While MSComm1.InBufferCount <= 0
 Print "Desteklenen standartlar:" & MSComm1.Input
End Sub
```



Bölüm 5

Visual Basic'te Kullanılan Diğer Nesneler

VB'DE KULLANILAN DİĞER NESNELER

VB'de görsel bir kontrole bağlı olmayan nesnelerde vardır. Screen, Printer, App gibi nesneler her uygulamada vardır ve bunlar için herhangi bir kontrolün forma yerleştirilmesi gerekmez. Ayrıca bir kontrole bağlı olmayan Menü kontrolünü de bu sınıftan sayabiliriz. Diğerlerinden farklı olarak menü bir kontrol olarak yerleştirilmez. Tasarım esnasında Menü editörü ile tasarlanır.

Screen nesnesi ekranla ilgili bazı ayarlamalar ve işlemler için kullanılır.

Printer nesnesi yazıcıyla ilgili bazı ayarlamalar ve işlemler için kullanılır. Form üzerine metodları kullanarak yaptığımız yazım ve çizimlerin aynısı yazıcıya da yapılabilmektedir. Bütün bu işlemler için **Printer** nesnesi kullanılır.

App nesnesi ise yapılan programla ilgili bazı ayarlamalar ve işlemler için kullanılır.

Menüler ise her Windows programının vazgeçilmez bileşenlerindendir. VB, hem uygulama menüleri hem de sağ fare tuşu ile açılan menülerin oluşturulmasına imkan verir.

Screen (Ekran)

Adından da anlaşılabileceği gibi ekranla ilgili temel işlemlerin temsil edildiği bir nesnedir.

Properties

ActiveForm

Ekran üzerindeki aktif olan formu gösterir. Bu form Windows altında çalışan diğer programların formları değil yalnız programınızdaki formlardan aktif olanıdır.

ÖRNEK: Örneğin bir çok formla çalışan bir programımız olsun ve hangi form aktif ise o formun başlığında saatin sürekli olarak gösterilmesini isteyelim. Bunun için üç tane form oluşturun ve ilk forma **interval**'i 1000 olan bir **timer** kontrolü koyun. Ve sadece ilk forma aşağıdaki kodları girin.


```
'ÖRNEK : ActiveForm
Private Sub Form_Load()
   Form2.Show
   Form3.Show
End Sub

Private Sub Timer1_Timer ()
 Screen.ActiveForm.Caption = Time
End Sub
```

Programı çalıştırdığınızda hangi form aktif ise o formun başlığında saatin yazıldığını göreceksiniz.

ActiveControl

Ekran üzerindeki aktif olan kontrolün ismi gibi davranır, dolayısıyla bu değerin tipi *Control* tipindendir.

ÖRNEK: Örnek olarak ekran üzerindeki kontrollerle ilgili yardım verecek bir programımız olsun. Bunun için bir TextBox, bir FileListBox ve bir tane de ComboBox oluşturun. Daha sonra "Yardım" Caption'lu ve MnYardim isimli bir menü oluşturun. Amacımız kullanıcı bir kontrolün üzeride iken yardım menüsünü seçerse o kontrolle ilgili kısa bir mesaj vermek ve o işin nasıl yapılacağını göstermek olsun.

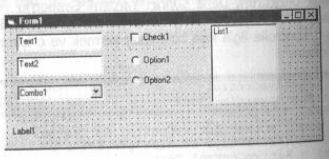
```
'ÖRNEK : ActiveControl
Private Sub mnyardim_Click ()
  Dim eski, i, j
  If TypeOf  screen.ActiveControl Is TextBox Then
    MsgBox ("Bu kutuya adınızı girin")
    eski = ActiveControl.Text
    Dim ad
    ad = "Muhammed Mustafa"
    For i = 1 To Len  (ad)
      ActiveControl.Text = Left (ad, i)
       For j = 1 To 50000: Next'bekle
    Next
    ActiveControl.Text = eski
  End If
  If TypeOf screen.ActiveControl Is FileListBox Then
    MsgBox ("Bu Listeden bir dosya seçin")
    eski = ActiveControl.ListIndex
    For i = 0 To ActiveControl.ListCount - 1
        ActiveControl.ListIndex = i
        For j = 1 To 5000: Next
    Next
    ActiveControl.ListIndex = eski
  End If
  If TypeOf screen.ActiveControl Is ComboBox Then
    MsgBox ("Bu Listeden mesleğinizi seçin")
```



```
    eski = ActiveControl .ListIndex
    For i = 0 To ActiveControl.ListCount - 1
        ActiveControl.ListIndex = i
        For j = 1 To 50000: Next
    Next
    ActiveControl.ListIndex = eski
  End If
End Sub
```

ÖRNEK: İkinci bir örnek olarak ta formdaki bir paletle kullanıcının istediği rengi seçebilmesini sağlayalım. Palet olarak Labelleri kullanacağız. Kullanıcı bir Label sol tuşla tıklarsa ekrandaki aktif kontrolün zemin rengi sağ tuşla tıklarsa yazı rengi değişsin.

Örneğimiz için aşağıdaki formu oluşturun ve Label1 kontrolünün Index özelliğini Properties penceresinden 0 değerini verin. Böylece bu Labeli bir dizi olarak tanımlamış olacağız. Program çalışmaya başlayınca da Formun Load olayına yazacağımız kodla bunlardan 15 tane daha oluşturup bunların her birinde farklı bir rengi kullanacağız.

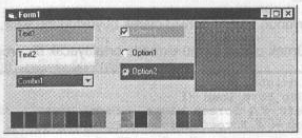



```
'ÖRNEK : ActiveControl
Private Sub Form_Load()
 Dim i
 Label1(i).BackColor = QBColor(0)
 Label1(i) = ""
 Label1(i).Width = 300
 For i = 1 To 15
  Load Label1(i)
  Label1(i).Left = Label1(i - 1).Left + Label1(i - 1).Width + 15
  Label1(i).Visible = True
  Label1(i).BackColor = QBColor(i)
 Next
End Sub

Private Sub Label1_MouseDown(Index As Integer, Button As Integer,
Shift As Integer, X As Single, Y As Single)
 On Local Error Resume Next
 If Button = 1 Then 'Sol tuş ile basılmışsa
  ActiveControl.BackColor = QBColor(Index)
```



```
 Else
  ActiveControl.ForeColor = QBColor(Index)
 End If
End Sub
```

Örneği çalıştırırsanız kursör hangi kontrolde ise, paletten seçtiğiniz renk o kontrole uygulanacaktır. Burada palet olarak bir Label kontrolünü değil de bir Text kutusu veya PictureBox gibi Focus alabilen kontrol kullanırsanız program doğru olarak çalışmaz. Çünkü Label tıklandığında Focus almadığı için kontrol hala kursörün bulunduğu yerdedir. Dolayısıyla ActiveControl özelliği Label'i değil kursörün bulunduğu kontrolü temsil edecektir.



FontCount

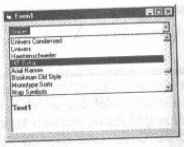
Ekranda gösterilebilecek fontların sayısını verir.

Fonts(Index)

Ekranda kullanılabilen index numaralı fontun ismini verir.

ÖRNEK: Ekranda kullanılabilen fontları bir Combo kutusunda veren ve kullanıcının text kutusu içerisindeki yazının fontunu seçebilmesine imkan tanıyan bir program yapalım. Form üzerine bir TextBox ve bir ComboBox yerleştirelim.

```
'ÖRNEK : FontCount, Fonts
Private Sub Form_Load ()
  Dim i
  For i = 0 To screen.FontCount - 1
    Combo1.AddItem screen.Fonts( i )
  Next
  Combo1.ListIndex = 0
  text1.FontName = Combo1.Text
End Sub
Private Sub Combo1_Click ()
  text1.FontName = Combo1.Text
End Sub
```



Height, Width

Ekranın boyunun ve eninin twip olarak büyüklüğünü verir.

ÖRNEK: Örnek olarak formu ekranda ortalayacak bir program yazalım.

```
'ÖRNEK : Height, Width
Private Sub Form_Load ()
  Form1.Left = ( Screen.Width - Form1.Width ) /2
  Form1.Top = ( Screen.Height - Form1. Height ) /2
End Sub
```

Böylece formumuz hangi grafik modunda olursa olsun ortalanacaktır. (Formun WindowState özelliği *0-normal* olmak şartıyla)

TwipsPerPixelX, TwipsPerPixelY

Ekranda, yatayda ve düşeyde bir twipin kaç pixele eşit olduğunu verir. Bu değer kullanılan grafik moduna göre farklılık gösterecektir. Dolayısıyla bu değere bakarak kullanılan grafik modu hakkında bir fikir elde edebilirsiniz. Ayrıca Windows API'leri genellikle pixel moduyla ölçekleme yaparken VB -scalemodu değiştirmemişse- twip moduyla ölçekleme yapar, API çağrısı yaparken bu değer dönüşüm işlemlerinizi kolayca yapmanızı sağlayacaktır.

ÖRNEK: Bu özellikleri kullanarak ekran çözünürlüğünü bulan bir program yazalım.

```
'ÖRNEK : TwipsPerPixelX, TwipsPerPixelY
Private Sub Form_Load ()
  text1.Text = "X:" & screen.Width / screen.TwipsPerPixelX
  text1 = text1 & "y:" & screen.Height / screen.TwipsPerPixelY
End Sub
```

MousePointer

Ekrandaki mouse göstericisinin şeklini belirler. Alabileceği değerler daha önce verilmişti.

Printer (Yazıcı)

Printer nesnesi kullanılarak VB'den ekrana çıktı yapıyormuş gibi yazıcıya çıktı yapılabilir. Printer nesnesine uygulanan bir çok yöntem ve metod Form veya Picture nesnesine uygulanıyormuş gibi uygulanabilir.

Properties

DeviceName

Yazıcının ismi bu özellikle öğrenilir.

DriverName

Yazıcı sürücüsünün ismi bu özellikle öğrenilir. Geriye dönen değer yazıcı için kullanılan sürücü dosyasının ismidir. Buradan geriye dönen değere .DRV uzantısı eklenirse sürücünün tam ismi öğrenilmiş olur.

Port

Yazıcının hangi porta bağlı olduğu bu özellikle öğrenilir. Geriye dönen değer string olarak bağlı olduğu portun ismini verir.

FILE:, LPT1:, LPT2: COM1:, COM2

PaperBin

Kağıt besleme kaynağı bu özellikle öğrenilip değiştirilebilir. Aşağıdaki değerlerden birini alır.

Sayısal	Sembolik	Sayısal	Sembolik
1	vbPRBNUpper	7	vbPRBNAuto
2	vbPRBNLower	8	vbPRBNTractor
3	vbPRBNMiddle	9	vbPRBNSmallFmt
4	vbPRBNManual	10	vbPRBNLargeFmt
5	vbPRBNEnvelope	11	vbPRBNLargeCapacity
6	vbPRBNEnvManual	14	vbPRBNCassette

ColorMode

Yazdırmanın renkli mi siyah beyaz mı olacağı bu özellikle öğrenilip değiştirilebilir. Aşağıdaki değerlerden birini alır.

2:vbPRCMColor: Renkli

1:vbPRCMMonochrome: Siyah beyaz

0: Geriye dönen değer 0 ise yazıcı renkli değildir.

Copies

Aynı sayfadan yazdırılacak kopya sayısı bu özellikle öğrenilip değiştirilebilir.

Duplex

Kullanılan yazıcı çift yönlü yazdırmayı destekliyorsa bu özellikle yazdırma işleminin nasıl yapılacağı öğrenilip değiştirilebilir.

1: vbPRDPSimplex:	Tek yönlü yazdırma
2: vbPRDPHorizontal:	Çift yönlü yatay yazdırma.
3: vbPRDPVertical:	Çift yönlü dikey yazdırma.
0:	Geriye dönen değer 0 ise kullanılan yazıcı çift yönlü yazdırmayı desteklemiyordur.

Orientation

Yazdırma işleminin yatay mı, dikey mi yaptırılacağı bu özellikle öğrenilip değiştirilebilir.

1: vbPRORPortrait:	Dikey
2:vbPRORLandscape:	Yatay

PaperSize

Sayfa boyutları bu özellikle öğrenilip değiştirilebilir.

Sayısal	Sembolik	Boyut
1	vbPRPSLetter	Letter, 8 1/2 x 11 inç
2	vbPRPSLetterSmall	+A611Letter Small, 8 1/2 x 11 inç
3	vbPRPSTabloid	Tabloid, 11 x 17 inç
4	vbPRPSLedger	Ledger, 17 x 11 inç

Sayısal	Sembolik	Boyut
5	vbPRPSLegal	Legal, 8 1/2 x 14 inç
6	vbPRPSStatement	Statement, 5 1/2 x 8 1/2 inç
7	vbPRPSExecutive	Executive, 7 1/2 x 10 1/2 inç
8	vbPRPSA3	A3, 297 x 420 mm
9	vbPRPSA4	A4, 210 x 297 mm
10	vbPRPSA4Small	A4 Small, 210 x 297 mm
11	vbPRPSA5	A5, 148 x 210 mm
12	vbPRPSB4	B4, 250 x 354 mm
13	vbPRPSB5	B5, 182 x 257 mm
14	vbPRPSFolio	Folio, 8 1/2 x 13 inç
15	vbPRPSQuarto	Quarto, 215 x 275 mm
16	vbPRPS10x14	10x14 inç
17	vbPRPS11x17	11x17 inç
18	vbPRPSNote	Note, 8 1/2 x 11 inç
19	vbPRPSEnv9	Envelope #9, 3 7/8 x 8 7/8 inç
20	vbPRPSEnv10	Envelope #10, 4 1/8 x 9 1/2 inç
21	vbPRPSEnv11	Envelope #11, 4 1/2 x 10 3/8 inç
22	vbPRPSEnv12	Envelope #12, 4 1/2 x 11 inç
23	vbPRPSEnv14	Envelope #14, 5 x 11 1/2 inç
24	vbPRPSCSheet	C size sheet
25	vbPRPSDSheet	D size sheet
26	vbPRPSESheet	E size sheet
27	vbPRPSEnvDL	Envelope DL, 110 x 220 mm
28	vbPRPSEnvC5	Envelope C5, 162 x 229 mm
29	vbPRPSEnvC3	Envelope C3, 324 x 458 mm
30	vbPRPSEnvC4	Envelope C4, 229 x 324 mm
31	vbPRPSEnvC6	Envelope C6, 114 x 162 mm
32	vbPRPSEnvC65	Envelope C65, 114 x 229 mm
33	vbPRPSEnvB4	Envelope B4, 250 x 353 mm
34	vbPRPSEnvB5	Envelope B5, 176 x 250 mm
35	vbPRPSEnvB6	Envelope B6, 176 x 125 mm
36	vbPRPSEnvItaly	Envelope, 110 x 230 mm
37	vbPRPSEnvMonarch	Envelope Monarch, 3 7/8 x 7 1/2 inç
38	vbPRPSEnvPersonal	Envelope, 3 5/8 x 6 1/2 inç
39	vbPRPSFanfoldUS	U.S. Standard Fanfold, 14 7/8 x 11 inç
40	vbPRPSFanfoldStdGerman	German Standard Fanfold, 8 1/2 x 12 inç
41	vbPRPSFanfoldLglGerman	German Legal Fanfold, 8 1/2 x 13 inç
256	vbPRPSUser	Kullanıcı tanımlı

Geriye dönen değer 256 ise yani kullanıcı tanımlı boyut ise, kağıt boyutları **Height** ve **Width** özellikleri ile öğrenilir.

PrintQuality

Yazdırma kalitesi bu özellikle öğrenilip değiştirilebilir.

-1: vbPRPQDraft:	Taslak çıktı
-2: vbPRPQLow:	Düşük kaliteli çıktı
-3: vbPRPQMedium:	Orta kaliteli çıktı.
-4: vbPRPQHigh:	En kaliteli çıktı

TrackDefault

Geriye dönen değer True ise, seçili yazıcı denetim masasından varsayılan olarak belirlenmiş yazıcıdır.

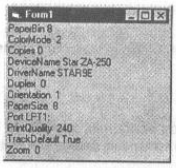
Zoom

Kullanılan yazıcı ölçeklemeyi destekliyorsa bu özellik vasıtası ile çıktının yüzde kaç büyültülüp küçültüleceği bu özellikle öğrenilip değiştirilebilir. Bu özellikten geriye 0 değeri dönüyorsa yazıcı ölçeklemeyi desteklemiyordur.

Yukarıdaki özelliklerin her biri CommonDialog kontrolü ile de kullanıcı tarafından değiştirilebilir.

```
'ÖRNEK : Printer özellikleri
Private Sub Form_Load()
 Show
 Print "PaperBin"; Printer.PaperBin
 Print "ColorMode "; Printer.ColorMode
 Print "Copies"; Printer.Copies
 Print "DeviceName "; Printer.DeviceName
 Print "DriverName "; Printer.DriverName
 Print "Duplex "; Printer.Duplex
 Print "Orientation "; Printer.Orientation
 Print "PaperSize "; Printer.PaperSize
 Print "Port "; Printer.Port
 Print "PrintQuality "; Printer.PrintQuality
 Print "TrackDefault "; Printer.TrackDefault
 Print "Zoom "; Printer.Zoom
End Sub
```

Programın ekran çıktısı aşağıdaki gibi olacaktır:



Page

Yazdırılan aktif sayfanın numarasını verir.

Printer nesnesine ait diğer Propertiesler daha önce anlatıldığı için burada tekrar vermiyoruz.

Methods

Print

Formatı normal Print komutu gibidir. Yazıcıdan Font özellikleriyle belirlenen fontta çıktı almaya yarar.

Circle, Line, Pset

Yazıcıda çizim yapmaya yarar. Bu metodları daha önce anlattığımız için tekrar vermiyoruz.

EndDoc

Dokümanın bittiğini belirten komutu yazıcıya gönderir ve yazıcıdan sayfalar basılır. Yazdırma işlemi sona erdiğinde bu komut kullanılmalıdır. Aksi takdirde programdan çıkıncaya kadar yazıcıya çıktı gönderilmez.

NewPage

Aktif sayfayı bitirir ve yeni bir sayfaya geçer.

KillDoc

Yazdırma işlemini iptal eder ve yazıcı kuyruğundan siler.

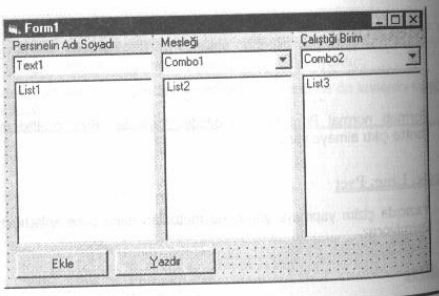
Scale (x1,y1,x2,y2)

Scale metodu formda olduğu gibi yazıcıda da koordinat sistemini yeniden belirlemeye yarar. Çizim kısmındaki örneğe bakabilirsiniz.

TextWidth, TextHeight

Bu özellikler seçilen font özelliklerine göre bir yazının genişliğini ve yüksekliğini verir. Bu metodlar Çizim kısmında detaylı olarak anlatılmıştır.

ÖRNEK: Örnek olarak aşağıdaki formda girilen personelin listesini yazıcıdan bir tablo olarak çıkarmak isteyelim.




```
'ÖRNEK : Printer
Private Sub Form_Load()
    With Combo1
        .AddItem "İşçi"
        .AddItem "Memur"
        .AddItem "Hizmetli"
        .AddItem "Mühendis"
        .AddItem "İşletmeci"
        .AddItem "Muhasebeci"
        .AddItem "Basın Yayın"
    End With
```



```
    With Combo2
        .AddItem "Reklam"
        .AddItem "Pazarlama"
        .AddItem "Muhasebe"
        .AddItem "Satış"
        .AddItem "Üretim"
        .AddItem "Yönetim"
    End With
End Sub

Private Sub Command1_Click()
    List1.AddItem Text1
    List2.AddItem Combo1.Text
    List3.AddItem Combo2.Text
End Sub

Private Sub Command2_Click()
 Dim i, x1, x2, x3, t, y1
 With Printer
  .FontSize = 14
  .FontBold = True
  x1 = (.ScaleWidth - .TextWidth("Personel Listesi")) / 2
  .CurrentX = x1 'Yazıyı ortala
  Printer.Print "Personel Listesi"
  .FontSize = 12
  x1 = .TextWidth(String(20, "M"))'Adı soyadı için 20 harflik alan
  x2 = x1 + .TextWidth(String(15, "M"))
  x3 = x2 + .TextWidth(String(15, "M"))
  y1 = .CurrentY
  Printer.Line (1, .CurrentY)-(x3, .CurrentY)
  .CurrentX = 1
   Printer.Print "Personelin Adı Soyadı";
  .CurrentX = x1
   Printer.Print "Mesleği";
  .CurrentX = x2
   Printer.Print "Çalıştığı Birim"
   Printer.Line (1, .CurrentY)-(x3, .CurrentY)
   .FontBold = False
   For i = 0 To List1.ListCount - 1
    .CurrentX = 1
    Printer.Print List1.List(i);
    .CurrentX = x1
    Printer.Print List2.List(i);
    .CurrentX = x2
    Printer.Print List3.List(i)
    Printer.Line (1, .CurrentY)-(x3, .CurrentY)
   Next
   Printer.Line (1, y1)-(1, .CurrentY)
   Printer.Line (x1, y1)-(x1, .CurrentY)
   Printer.Line (x2, y1)-(x2, .CurrentY)
   Printer.Line (x3, y1)-(x3, .CurrentY)
  End With
End Sub
```

Clipboard (Pano)

Windows'un kesme ve yapıştırma işlemleri için Clipboard (Pano)'yu kullandığını biliyorsunuz. VB' de Clipboard nesnesinin aşağıdaki yöntemlerini kullanarak panoyla ilgili işlemler yapılabilir.

Methods

Clear

Panonun içeriğini siler. Panoya bir bilgi kopyalanmadan önce bu metod kullanılarak önce panonun içeriği silinmelidir.

GetFormat

Panonun içindeki nesnenin formatını almaya yarar. Clipboard'ın içinden alınabilecek nesneler şunlardır. BMP, WMF ve DIP formatındaki resimler, Renk Paleti, Text ve DDE bilgisi Bu metoddan dönen değerler ve anlamı şöyledir.

Sayısal	Sembolik	Anlamı
-16640	vbCFLink	Bağlantı Bilgisi, DDE bağlantı bilgisi
-16639	vbCFRTF	RTF metni, Formatlı metin bilgisi
1	vbCFText	Text, Formatsız metin bilgisi
2	vbCFBitmap	BMP, Bitmap türü resim bilgisi
3	vbCFEMetafile	WMF, Windows Metafile türü resim bilgisi
8	vbCFDIB	DIB, DIB formatlı resim bilgisi
9	vbCFPalette	Palet bilgisi
15	vbCFFiles	Windows Explorer Dosya adı

GetData

Panonun içindeki nesneyi almaya yarar. Clipboardın içinden bu metodla alınabilecek nesneler şunlardır. BMP, WMF ve DIP formatındaki resimler ve Renk Paleti

ClipBoard.GetData [format]

Format parametresini kullanmazsanız pano içindeki nesne otomatik olarak belirlenecektir. Format parametresinin alabileceği değerler şöyledir.

Sembolik	Sayısal	Anlamı
vbCFBitmap	2	BMP
vbCFMetafile	3	WMF
vbCFDIB	8	DIB
vbCFPalette	9	Palet

```
Picture1.Picture = ClipBoard.GetData
```

Yukarıdaki kodla, bir Picture kutusuna panodaki resmi alabilirsiniz.

GetText

Panonun içindeki Text veya DDE bilgisini almaya yarar.

ClipBoard.GetText [format]

Format parametresini kullanmazsanız pano içindeki nesne otomatik olarak belirlenecektir. Format parametresinin alabileceği değerler şöyledir.

Sembolik	Sayısal	Anlamı
vbCFLink	1	Text
vbCFLink	&HBF00	DDE Bilgisi
vbCFRTF	&HBF01	Rich Text Format

```
If ClipBoard.GetFormat = 1 Then
 Print ClipBoard.GetText
End If
```

SetData

Panoya BMP, WMF ve DIP formatındaki resimler veya Renk Paletini kopyalamaya yarar. Panoya herhangi bir şey kopyalanmadan önce panonun silinmesi gerekir.

ClipBoard.SetData data, [format]

Data parametresiyle panoya kopyalanacak nesne ve format parametresiyle de nesnenin formatı verilebilir. Format verilmezse yine nesnenin formatı otomatik olarak bulunacaktır.

```
ClipBoard.Clear
ClipBoard.SetData Picture1.Picture
```


SetText

Panoya text veya DDE bilgisi kopyalamaya yarar. Formatı SetData gibidir.

```
ClipBoard.Clear
ClipBoard.SetData  Text1.Text
```

ÖRNEK: Örnek olarak panoyu yakın takibe alacak bir program yazalım. Örneğimiz için formunuzun üzerine bir timer, bir text kutusu ve bir PictureBox yerleştirin.

```
'ÖRNEK : ClipBoard
Private Sub Form_Load()
 Timer1.Interval = 1000
End Sub

Private Sub Timer1_Timer()
 With Clipboard
  If .GetFormat(vbCFLink) Then
   Text1 = "Bağlantı bilgisi:" & .GetText()
  End If
  If .GetFormat(vbCFRTF) Then
   Text1 = "RTF metni:" & .GetText()
  End If
  If .GetFormat(vbCFText) Then
   Text1 = "Salt Metin:" & .GetText()
  End If
  If .GetFormat(vbCFPalette) Then
   Text1 = "Palet bilgisi içeriyor bu bilgi görüntülenemez!"
  End If
  If .GetFormat(vbCFBitmap) Then
   Text1 = "Bitmap resim bilgisi"
   Picture1.Picture = .GetData()
  End If
  If .GetFormat(vbCFEMetafile) Then
   Text1="Metafile resim bilgisi"
   Picture1.Picture = .GetData()
  End If
  If .GetFormat(vbCFDIB) Then
   Text1 = "DIB resim bilgisi"
   Picture1.Picture = .GetData()
  End If
  If .GetFormat(vbCFFiles) Then
   Text1="Explorer dosya bilgisi" & .GetText()
   Picture1.Picture = .GetData()
  End If
 End With
End Sub
```

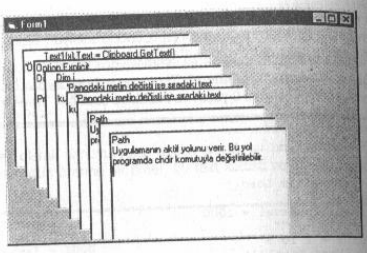
ÖRNEK: İkinci bir örnek olarak ta Windows'un panosunun eksikliğini giderecek bir alternatif pano programı hazırlayalım. Biliyorsunuz Windows tarafından sağ-

lanan panoya aynı anda sadece bir bilgi aktarabiliyorsunuz. Yani panoda bir bilgi varken başka bir bilgiyi panoya alırsanız önceki siliniyor. Bizim yapacağımız pano 10 tane metni panoya aynı anda atabilme imkanı versin.

Örneğimiz için formun üzerine bir Timer yerleştirin. Ayrıca bir Text kutusu yerleştirerek Index özelliğini 0 yapın ve MultiLine özelliğini True yapın.

```
'ÖRNEK : ClipBoard
Option Explicit
Dim x

Private Sub Form_Load()
    Dim i
    Timer1.Interval = 1000
    Text1(0) = ""
    For i = 1 To 9
        Load Text1(i)
        Text1(i).Top = Text1(i - 1).Top + 200
        Text1(i).Left = Text1(i - 1).Left + 200
        Text1(i).Visible = True
    Next
End Sub

Private Sub text1_Click(Index As Integer)
    'tıklananı öne getir
    Text1(Index).Zorder
End Sub

Private Sub text1_DblClick(Index As Integer)
    'Çift tıklananın içeriğini panoya at
    Clipboard.SetText (Text1(Index).Text)
End Sub

Private Sub Timer1_Timer()
    'Panodaki metin değişti ise sıradaki text kutusuna al
    If Clipboard.GetText() <> Text1(x).Text Then
        x = (x + 1) Mod 10
        Text1(x).Text = Clipboard.GetText()
        'Öne getir
        Text1(x).Zorder
    End If
End Sub
```

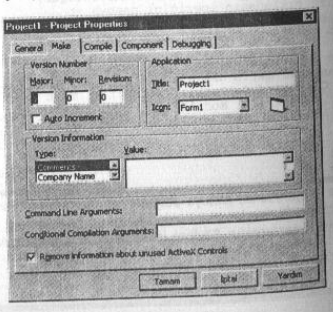
Programı çalıştırırsanız, panoya attığınız her metin programdaki bir text kutusuna alınacaktır. Bunlardan birini tıkladığınızda o öne gelecek ve çift tıkladığınızda ise panoya aktarılacaktır. Böylece istediğiniz kısmı programlarınıza yapıştırabilirsiniz.



App (Uygulamanızla ilgili özellikler)

Uygulamanızla ilgili özellikleri App nesnesi ile belirleyebilirsiniz.

Windows, program dosyalarına tanıtıcı kimliklerin eklenmesine imkan verir. Visual Basic'te de uygulamalarınıza tanıtıcı özellikler verebilirsiniz. Örneğin program dosyalarının versiyonları ne iş yaptıkları gibi. App nesnesinin aşağıdaki bazı özellikleri ile çalışan programın bu özelliklerini belirlemek mümkündür.



Programla ilgili bilgilerin belirlenmesi işlemi, Project menüsündeki Project Properties seçeneği ile açılan yukarıdaki pencereden yapılır ve programın çalışması esnasında da aşağıdaki özelliklerle öğrenilir.

Properties

Comments

Yukarıdaki pencerede de görüldüğü gibi programa ait bir açıklamayı Comments kutusuna yazmak mümkündür. Bu, çoğunlukla programın ne işe yaradığını açıklayan ve sizin belirlediğiniz bir bilgidir. Bu bilgi program çalışırken Comments özelliği ile öğrenilebilir.

CompanyName, ProductName

Yine yukarıdaki pencereden programcı tarafından belirlenen şirket ismi bu özellikle öğrenilebilir.

FileDescription

Program dosyasına ait açıklama bu özellikle öğrenilir.

LegalCopyright, LegalTrademarks

Programın yasal telif hakkının kimde olduğu bu özelliklerle öğrenilir.

Major, Minor, Revision

Program tasarlanırken mutlaka versiyon numarasını yukarıdaki pencereden verin. Böylece programınızın yeni versiyonlarını çıkardığınızda, programın önceki versiyonlarını belirlemeniz kolay olur. Örneğin programınızı kuran birisinde programın daha yeni bir versiyonu zaten kurulu ise kullanıcıyı uyarabilir veya sadece daha yeni versiyona sahip dosyalar kopyalanabilir. Versiyonlar genellikle x.yy şeklindedir. Bunlardan x önemli gelişmeleri yy ise daha çok güncellemeleri ifade eder. Major ve Minör özellikleriyle de bu versiyon numarasını öğrenebilirsiniz. Major.Minör şeklinde değerlendirmeniz gerekir.

StartMode

Uygulamanın standart bir uygulama mı yoksa bir ActiveX uygulaması mı olduğu bu özellikle öğrenilir.

1:vbSModeAutomation: OLE uygulaması, ActiveX

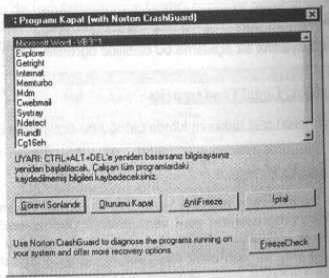
0:vbSModeStandalone: Standart uygulama

TaskVisible

Uygulamanızın görev listesinde görülmesini istemiyorsanız bu özelliğe false değerini atayabilirsiniz.

```
Private Sub Form_Load()
 App.TaskVisible = False 'Uygulamayı gizle
 Form1.Hide 'Formu da gizle
End Sub
```

Yukarıdaki kodla, uygulamanızın Windows görev listesinde gözükmemesini sağlamış olursunuz. Ctrl+Alt+Del tuşları ile karşınıza gelen aşağıdaki görev listesinde programınıza ait isim bulunmayacaktır.



UnattendedApp

Program herhangi bir kullanıcı arabirimine sahip değilse (form gibi) True değeri döner.

NonModalAllowed

Program modal olarak çalışıyorsa bu özellik False değerini alır. (Modal olarak çalışan formlarda o form kapatılmadan programdaki diğer formlara geçiş yapılamıyordu)

hInstance, ThreadID

Çalışan uygulamanın tanıtıcı numaraları bu özelliklerle öğrenilir. Bu değerler daha çok API çağrılarında kullanılır.

Path

Uygulamanın aktif yolunu verir. Bu yol programda **chdir** komutuyla değiştirilebilir.

EXEName

Uygulamanın EXE dosya ismini verir.

HELPFile

Programın Help dosyasının ismini belirlemeye yarar. Eğer bu özellikle bir dosya ismi belirlenmişse kullanıcının F1 tuşuna basılmasıyla VB WinHelp programını bu help dosyası ile açar ve **HelpContextID** özelliği ile verilen numaralı konuya gider.

PrevInstance

Uygulamanızın bir kopyasının sistemde şu anda çalışıp çalışmadığını bildirir. Eğer programınızın bir kopyası daha sistemde çalışıyorsa bu özellikten dönen değer True'dir. Bu özelliği programınızın aynı anda tek bir kopyasının çalışmasını istiyorsanız kullanabilirsiniz. Bu komutu programınızın Form_Load olayına yerleştirerek daha önce zaten çalışıyorsa programın tekrar çalışmasını önleyebilirsiniz.

```
Private Sub Form_Load
 If App.PrevInstance Then
  MsgBox ("Uygulama zaten çalışıyor")
  End
 End If
End Sub
```

Title

Programınızın başlığını belirlemeye yarar. Programınız bu özelliğe verdiğiniz isim ile Windows Görev Yöneticisinde temsil edilir.

ÖRNEK: Örnek olarak programa ait bazı bilgileri verecek bir program yazalım. Programınızın bazı özelliklerini Project menüsündeki Project Properties seçeneği ile açılan pencereden ayarlayın ve aşağıdaki kodu yazın.

```
'ÖRNEK : App
Private Sub Form_Load()
 Show
 Print "Programın dizini:"; App.Path
 Print "EXE dosya adı:"; App.EXEName
 Print "Versiyon:"; App.Major; "."; App.Minor; "."; App.Revision
 Print "Başlık:"; App.Title
 Print "Comment:"; App.Comments
 Print "CompanyName:"; App.CompanyName
 Print "Dosya Açıklaması:"; App.FileDescription
 Print "Yasal Telif hakkı:"; App.LegalCopyright
 Print "Trade Marks:"; App.LegalTrademarks
 Print "Product Name:"; App.ProductName
 Print "Help dosyası adı:"; App.HelpFile
 Print "Program "; IIf(App.NonModalAllowed, "Modal değil",
"Modal")
 Print "Programın bir kopyası daha "; IIf(App.PrevInstance,
"Çalışıyor", "Çalışmıyor")
 Print "Başlama Modu:"; IIf(App.StartMode = vbSModeStandalone,
"Standart Uygulama", "ActiveX")
 Print "Uygulama "; IIf(App.TaskVisible, "görünür", "gizli")
 Print "Kullanıcı ara birimi "; IIf(App.UnattendedApp, "yok",
"var")
 Print "ThreadID:"; Hex(App.ThreadID)
 Print "Handle Instance No:"; Hex(App.hInstance)
End Sub
```

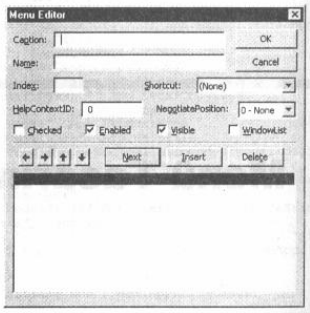
Bölüm 6

Menü Tasarımı

MENÜ TASARIMI

Visual Basic'te oluşturacağınız menülerin de bir kontrol gibi Properties ve Eventsleri vardır. Bu özellikleri vermeden önce menü tasarımı nasıl yapılır onu görelim.

Menü oluşturmak veya oluşturulmuş menüleri değiştirmek VB'nin Tools menüsü altındaki Menu Editor seçeneği ile yapılır. Kısaca bu seçeneğe Ctrl-E tuşları ile ulaşabilirsiniz. Bu seçenek seçildiğinde aşağıdaki gibi bir pencere açılır.



Name

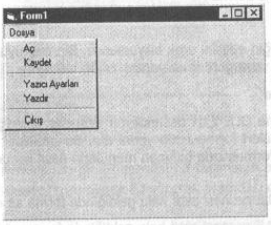
Bu kutuya menünün ismi yazılır. Diğer kontrollerin Name özelliğinden bir farkı yoktur. Prensip olarak bu ismin önüne Mn gibi bir tanıtıcı harf koyarsanız bunun bir menü ismi olduğunu hatırlamanız daha kolay olur. Örneğin Kaydet yerine MnKaydet ismini vermek bu ismin bir menüye ait olduğunu hatırlamak için daha uygundur.

Caption

Menü başlığıdır. Bu kutuya yazılan yazı menü üzerinde görüntülenir. Bu aynı zamanda menünün bir propertiesidir. Çalışma zamanı da değiştirilebilir. Buraya yazılacak metnin herhangi bir harfinin başına & işaretinin konması o harfin altı çizili harf olmasını sağlar.

Farklı olarak bu özelliğe – (eksi işareti) verilirse bu durumda o eleman bir menü elemanı olarak değil bir kesme çizgisi olarak gösterilir.

Aşağıdaki menüde, biri Kaydet menüsü altında, diğeri Yazdır menüsü altında olmak üzere iki tane kesme çizgisi görmektesiniz. Bu kesme çizgilerini oluşturmak için Caption olarak – verilir.



Index

Bu da bir propertiesdir. Ve menü bir dizi olarak tanımlanacaksa yani aynı isimli bir kaç menü oluşturulacaksa bu kutuya index numarası verilir. Bu kutu içine herhangi bir sayı yazılırsa menü dizi olarak tanımlanmış kabul edilir ve bu menüye ulaşım menü ismiyle birlikte parantez içinde bu numara verilerek yapılır.

Shortcut

Menüye bir kısa yol tuşu bu kutudan seçilebilir. Örneğin F2 tuşuna basıldığında kaydet menüsünün aktif hale gelmesini sağlayabilirsiniz.

WindowList

Bu özellik Form'un bir MDIChild form olması durumunda kullanılır. Bir menü elemanına bu özellik verildiğinde o anda yüklü olan MDIChild formların bir listesi bu menünün alt elemanları olacak şekilde otomatik olarak eklenir. Bu durumu Program Yöneticisinin Pencere menüsünde görebilirsiniz. (Örnek için MDI Formlar bölümüne bakınız)

Enabled, Visible

Bu iki özelliğin etkisini artık biliyorsunuz. Biri menüyü aktif-pasif yaparken diğeri de görünür veya görünmez yapar.

NegotiatePosition

Form üzerine OLE 2.0'ı destekleyen nesneler eklediğinizde, eklediğiniz uygulamanın menüleri formunuzda yerini alır. Bu özellikle, bir OLE nesnesi aktif hale geldiğinde formunuzda bulunan menülerin nasıl konumlandırılacağını belirleyebilirsiniz.

0: Bir OLE nesnesi aktif hale geldiğinde forma ait menüler gösterilmeyecek.

1: Bir OLE nesnesi aktif hale geldiğinde forma ait menüler menü çubuğunun solunda yer alacak.

2: Bir OLE nesnesi aktif hale geldiğinde forma ait menüler menü çubuğunun ortasında yer alacak.

3: Bir OLE nesnesi aktif hale geldiğinde forma ait menüler menü çubuğunun sağında yer alacak.

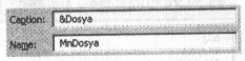
Menü tasarımı

Bir menü bir başka menünün alt menüsü olarak tanımlanacaksa pencere üzerindeki → düğmesine basılır. Bunu yaptığınızda bu menünün başına ---- işareti konacaktır. Bu durumda bu menü bir üstteki menünün alt menüsü olur. Bir menüyü başka bir menünün alt menüsü olmaktan çıkarmak için de ← düğmesine basılır.

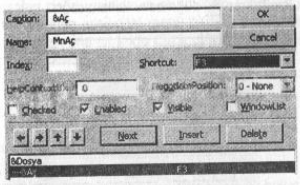
Araya bir menü eklemek için Insert, silmek için de Delete düğmesini kullanabilirsiniz.

ÖRNEK: Örnek olarak aşağıdaki menüyü adım adım oluşturalım.

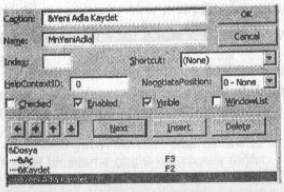
Önce **Dosya** menüsü için aşağıdaki değerleri girip Entere basın.



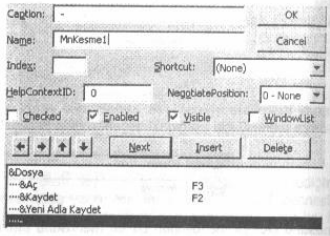
Aç menüsünü **Dosya** menüsünün bir alt menüsünü tanımlayacağımız için önce → düğmesine basın ve kısayol tuşu olarak F3 seçerek aşağıdaki değerleri girin.



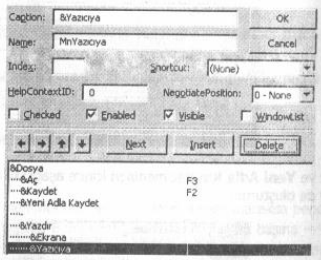
Kaydet ve **Yeni Adla Kaydet** menüleri içinde aşağıdaki değerleri girerek bu iki menüyü de oluşturun.



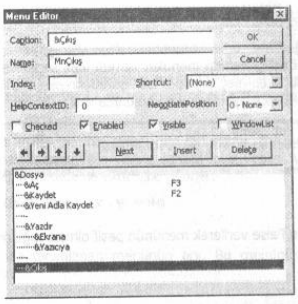
Şimdi aradaki kesme çizgisini oluşturmamız gerekiyor. Bunun için menünün başlığına aşağıdaki gibi - vermemiz gerekiyor.



Yazdır menüsünü de diğerleri gibi oluşturduktan sonra buna ait alt menüyü oluşturmak için tekrar → düğmesini kullanacağız.



Yazdır menüsünün alt menüsü olmaktan çıkmak için ← düğmesine basın ve kesme çizgisini de yukarıdaki gibi tanımladıktan sonra **Çıkış** menüsünü de tamamlayın. Menü tasarım penceremiz aşağıdaki gibi olmalıdır.



Yukarıdaki elemanları doğru şekilde oluşturduktan sonra Ok düğmesine basarsanız formunuzun üzerinde aşağıdaki menü oluşacaktır. Bu aşamadan sonra istediğiniz menüyü seçerek ona ait kodu **Click** olayına yazabilirsiniz.

Ne yazık ki VB'de menülerin sadece **Click** olayı bulunuyor. En azından MouseMove gibi bir olayın bulunması o menüye ait açıklamaların gösterilmesinde bir kolaylık sağlayabilirdi.

Pencere üzerindeki diğer seçenekler birer propertiesdir ve aşağıda anlatılmıştır.

Properties

Caption

Bu özellik menü üzerindeki yazıyı belirler. Özel olarak bu değere "-" verilirse bu bir menü elemanı değil bir kesme çizgisi oluşmasını sağlar.

Checked

Bu özelliğe True verilerek menünün seçili (işaretli) olması sağlanabilir. Genellikle Checked özelliği verilen menüler, işaretli iken seçildiklerinde işaretinin kalkması, işaretli değilken seçildiğinde ise işaretlenmesi istenir. Bunu yapmak için o menünün Click olayında şu kod bulunmalıdır.

```
Private Sub Menu_Click()
  Menu.Checked = Not Menu.Checked
End Sub
```

Enabled

Bu özelliğe False verilerek menünün pasif olması sağlanır.

Visible

Bu özelliğe False verilerek menünün görülmez olması sağlanır.

Menü nesnesinin diğer özellikleri daha önce anlatılan kontrollerdeki gibidir.

Events

Click( )

Menünün sadece click olayı vardır ve menü seçildiğinde bu olay aktif hale gelir.

Bu olay koduna ulaşabilmek için tasarladığınız menüyü form üzerinden seçmeniz gerekir. Alt menüleri olan bir menünün ise Click olayına ulaşabilmek için Vb bir yol sunmaz. Bu kodu kendiniz yapabilirsiniz.

Alt menüleri olan bir menünün Click olayı, daha çok alt menüler açılmadan önce gerekli menüleri aktif-pasif duruma getirmek için kullanılır.

Örneğin Düzen menüsü ve altında Kes-Kopyala-Yapıştır seçenekleri varsa, Düzen menüsünün Click olay kodunu yazarak, panoda bilgi yoksa, menüler henüz açılmadan Yapıştır menüsünü pasif hale getirebilir, seçili bir kısım yoksa da Kes ve Kopyala seçeneklerini pasif hale getirebilirsiniz.

Popup Menü

Sağ fare tuşuna basınca ekranın herhangi bir yerinde açılan menüleri bir çok programda görmüşsünüzdür. VB'de de tek komutla bu tip menüleri oluşturmak mümkündür.

Popup menü tasarımı aynen normal menü gibi tanımlanır. Bu menünün menü çubuğunda gösterilmesi istenmiyorsa visible özelliği false yapılır. Ve açılması istenen yerde PopupMenu metodu ile aktif hale getirilir.

PopupMenu menuadı , flags , x , y, bold

menuadı:

Menü Design kısmında tanımlanan menünün adı. Bu menünün mutlaka alt menüleri bulunmalıdır.

x,y:

Menünün gösterileceği koordinatlar

flags:

Flags parametresi, x koordinatının nasıl kullanılacağını ve farenin hangi tuşunun click olayını meydana getireceğine dair iki bilgi içerir. **Flags** parametresini şöyle formülize edersek

Flags = Koorx + Fare

Burada Koorx in alacağı değerler ve anlamları şöyledir.

0: vbPopupMenuLeftAlign: Menü x koordinatının solunda açılacak

4: vbPopupMenuCenterAlign: Menü x koordinatını ortalayacak.

8: vbPopupMenuRightAlign :Menü x koordinatının sağında açılacak

Fare değişkeninin alacağı değerler ise şöyledir.

0: vbPopupMenuLeftButton: Sol fare tuşu ile menüden eleman seçilecek.

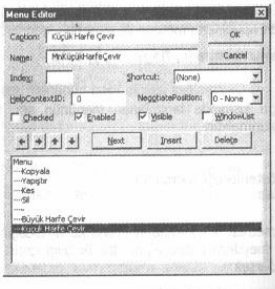
2: vbPopupMenuRightButton: Sağ fare tuşu ile menüden eleman seçilecek.

Bold:

Bu parametre ile açılacak menüdeki seçeneklerden biri kalın yapılabilir. Kalın olarak yazılması istenen menünün ismi bu özelliğe verilirse popup menü açıldığında o menü bold olarak yazılacaktır.

ÖRNEK: Örnek olarak bir text kutusu içinde kesme, kopyalama, silme, yapıştırma ve büyük/küçük harfe çevirme özelliklerini destekleyecek bir menü tasarlayalım.

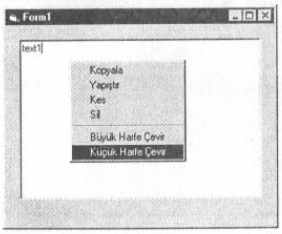
Örneğimiz için formunuzun üzerine bir text kutusu yerleştirin ve **Tools-Menu Editor** menü seçeneği yardımı ile aşağıdaki menü yapısını oluşturun. (Caption'larının önüne Mn koyarak Name özelliklerine yazın. Yani Menu Caption'lu menünün ismini MnMenu olarak verin)




```
'ÖRNEK : PopupMenü
Private Sub Form_Load ()
 MnMenu.Visible = False
End Sub

Private Sub Text1_MouseDown (button As Integer, Shift As Integer,
x As Single, y As Single)
 'Sağ fare tuşuna basıldı ise
 If button = 2 Then
    PopupMenu MnMenu, 4, text1.Left + x, text1.Top + y
 End If
End Sub

Private Sub MnKes_Click ()
   'Seçili kısmı panoya al
  clipboard.SetText text1.SelText
  've sil
  text1.SelText = ""
End Sub

Private Sub mnkopyala_Click ()
   'Seçili kısmı panoya kopyala
  clipboard.SetText text1.SelText
```



```
End Sub

Private Sub mnsil_Click ()
  'Seçili kısmı sil
  text1.SelText = ""
End Sub

Private Sub Mnyapistir_Click ()
  'Seçili kısma panodakini yapıştır
  text1.SelText = clipboard.GetText()
End Sub

Private Sub MnBüyükHarfeÇevir_Click()
    If Text1.SelLength = 0 Then
    'seçili kısım yoksa tümünü çevir
        Text1.Text = UCase(Text1.Text)
    Else
        Text1.SelText = UCase(Text1.SelText)
    End If
End Sub

Private Sub MnKüçükHarfeÇevir_Click()
    If Text1.SelLength = 0 Then
    'seçili kısım yoksa tümünü çevir
        Text1.Text = LCase(Text1.Text)
    Else
        Text1.SelText = LCase(Text1.SelText)
    End If
End Sub
```



Bölüm 7

ÇİZİM

ÇİZİM

Visual Basic kullanarak çizim yapmak için circle, line ve pset metodları kullanılır. Çizime ait renk, desen vb işlemler için ise propertiesler kullanılmaktadır. Ayrıca kontrole bağımlı olmadan yazı yazmak için de Print metodu kullanılır. Yazıma ait renk,i yazı tipi ayarları ise yine propertiesler aracılığı ile ayarlanır.

Şimdi vereceğimiz metodlar hem Form hem PictureBox hem de yazıcı (VB'de ismi Printer olarak geçer) için geçerlidir. Bu işlemleri PictureBox içinde yapacaksanız sadece komutların başına Picture1. eklemeniz yeterlidir. Örneğin Circle(50,100),70 komutu form üzerine bir daire çizer. Eğer Picture1 içine çizdirmek isterseniz Picture1.Circle(50,100),70 şeklinde kullanmanız gerekir. Aynı işlemi yazıcıda yaptırmak için ise Printer.Circle(50,100),70 şeklinde kullanmanız gerekir.

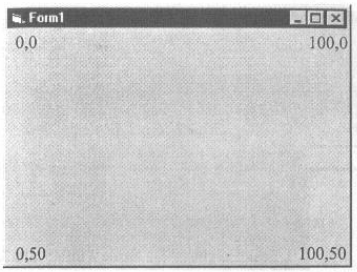
- Metodlar kullanılarak yapılan çizim (Line, Circle, Pset) ve yazımların (Print) ekranda süreklilik kazanabilmesi, dosyaya kaydedilebilmesi ve yazdırılabilmesi için **AutoReDraw** özelliğini **True** yapmanız gerekir.

Koordinat Sistemi

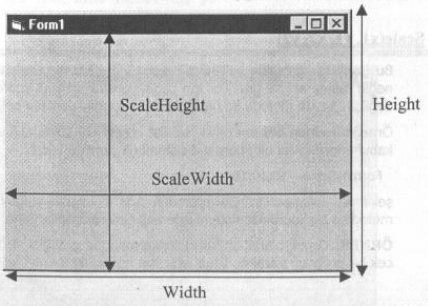
VB'deki genelde de programlama dillerindeki koordinat sistemi matematikteki koordinat sistemine terstir. Özellikle Y koordinatı aşağı doğru giderken artar, yukarı doğru giderken ise azalır. Bildiğiniz gibi matematikte Y koordinatı aşağı doğru azalır.

Formun orta noktası değil sol üst noktası orijindir yani 0,0 noktasıdır. İleri doğru gittikçe x artar, aşağı doğru gittikçe ise y artar.

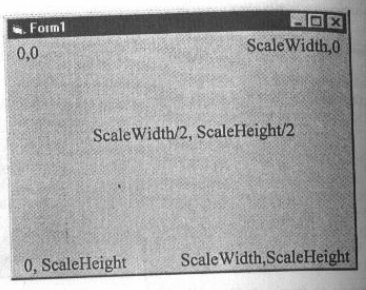
Formun genişliğini 100, yüksekliğini 50 kabul edersek köşe koordinatları aşağıdaki gibi olacaktır:



Form içinde çizim yapılabilecek koordinatlar formun iç bölgesidir. Yani formun başlık kısmı ve çerçeveleri bu çizim alanının dışındadır. Dolayısıyla **Width** özelliğiyle çizim yapılacak alanın genişliği, **Height** özelliğiyle de çizim yapılacak alanın yüksekliği tam olarak bulunamaz. Çerçeve genişliği ve başlık yüksekliği bu değerlerden çıktıktan sonra çizim alanı belirlenebilir. Bu yüzden bu iki değeri tam olarak verecek iki özellik bulunur. **ScaleWidth** özelliği çizim yapılabilecek bölgenin genişliğini, **ScaleHeight** özelliği ise çizim yapılabilecek bölgenin yüksekliğini tam olarak verir.



Buna göre koordinat sistemini yazalım:



Koordinat Sistemini Değiştirme

Bütün programlama dillerinde yukarıda bahsettiğimiz koordinat sistemi kullanılır ve çizimler buna göre yapılır. Ancak eğer matematiksel fonksiyonların grafiğini çizdirmek isterseniz bu grafik eksenlerine bazı dönüşümler uygulamanız gerekir ki bu da işlemleri karıştırır. Bunlarla uğraşmak istemiyorsanız **Scale** metodu ile koordinat sistemini değiştirebilirsiniz.

Scale(x1, y1, x2, y2)

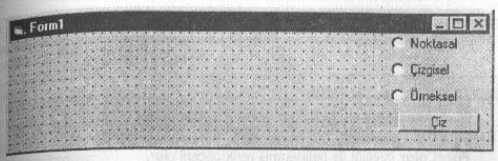
Bu özellikte kontrolün kullanacağı koordinat sistemi yeniden tanımlanabilir. Örneğin matematiksel grafikler için bilgisayarda kullanılan koordinat sistemi uygun değildir. **Scale** metodu ile bu koordinat sistemini yeniden belirleyebilirsiniz.

Örneğin formun ortasını orijin, sol üst köşeyi -10,10 ve sağ alt köşeyi de 10,-10 kabul edecek yeni bir koordinat sistemi oluşturmak için :

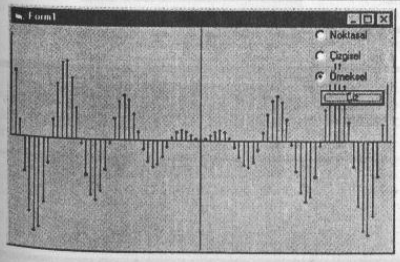
```
Form1.Scale (-10,10,10,-10)
```

şeklinde kullanılabilir. Bu işlemden sonra uygulayacağınız yazım ve çizim metodları bu koordinat sistemi referans alınarak yapılacaktır.

ÖRNEK: Örnek olarak x*Sin(x) fonksiyonunun grafiğini değişik şekillerde çizdirecek bir program yazalım. Örneğimiz için aşağıdaki formu oluşturun.



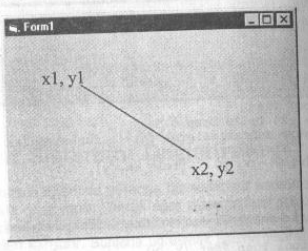

```
'ÖRNEK : Scale
Private Sub Command1_Click()
 Cls
 Scale (-20, 20)-(20, -20)'koordinat sistemini değiş
 Line (0, 20)-(0, -20) 'y eksenini çiz
 Line (-20, 0)-(20, 0) 'x eksenini çiz
 Dim x, y
 If Option3 Then
  For x = -20 To 20 Step 0.5
   'bu satır değiştirilerek farklı fonksiyonlar çizilebilir.
   y = x * Sin(x)
   Line (x, 0)-(x, y): Circle (x, y), 0.1
  Next
 Else
  For x = -20 To 20 Step 0.01
   'bu satır değiştirilerek farklı fonksiyonlar çizilebilir.
   y = x * Sin(x)
   If Option1 Then PSet (x, y)
   If Option2 Then Line (x, 0)-(x, y)
  Next
 End If
End Sub
```



Çizgi Çizme

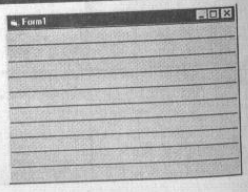
Line(x1, y1)-(x2, y2),renk

Line komutunun bu formuyla x1,y1 noktasından x2,y2 noktasına verilen renkte bir çizgi çizdirilebilir. İstenirse renk parametresi verilmeyebilir. Bu durumda **ForeColor** özelliği ile belirlenmiş renk geçerli olur.



ÖRNEK: Formumuza satır çizgileri çizdirelim.

```
'ÖRNEK : Line
Private Sub Form_Load()
 Show
 For i = 0 To ScaleHeight Step ScaleHeight / 10
  Line (0, i)-(ScaleWidth, i)
 Next
End Sub
```



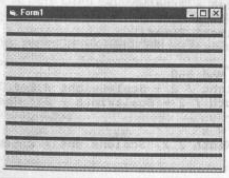
Yukarıdaki işlemi yazıcıya yapmak isterseniz, yani yazıcıya bu çizgileri çizdirmek istersenız ScaleHeight, ScaleWidth ve Line satırlarının başına **Printer.** yazmanız yeterlidir.

```
'ÖRNEK : Line
Private Sub Form_Load()
 For i = 0 To Printer.ScaleHeight Step Printer.ScaleHeight / 10
  Printer.Line (0, i)-( Printer.ScaleWidth, i)
 Next
 Printer.EndDoc
End Sub
```

- İstenirse çizgi kalınlığı **DrawWidth** özelliği ile değiştirilebilir.

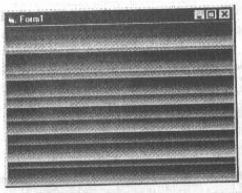
ÖRNEK: Yukarıdaki örneği çizgi kalınlığını değiştirerek yapalım

```
'ÖRNEK : Line
Private Sub Form_Load()
 DrawWidth=5 'Çizgi kalınlığı
 For i = 0 To Printer.ScaleHeight Step Printer.ScaleHeight / 10
  Printer.Line (0, i)-( Printer.ScaleWidth, i)
 Next
 Printer.EndDoc
End Sub
```



ÖRNEK: Farklı renklerde sık çizgiler çizdirerek formlarımızın arka planına, güzel renk geçişleri oluşturabiliriz. Aşağıdaki kod her satıra farklı bir renkle çizgi çizdirmektedir.

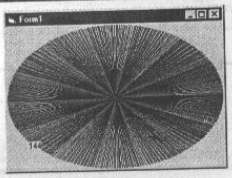
```
'ÖRNEK : Line
Private Sub Form_Paint()
 ScaleMode = 3 'Pixel moduna geçir
 For i = 0 To ScaleHeight
  Line (0, i)-(ScaleWidth, i), i * 1800
 Next
End Sub
```



Örnekteki i*1800 değerini değiştirerek değişik renk geçişleri elde edebilirsiniz. Örneğin i*256 yaparsanız yeşilin tonları, i*65536 yaparsanız mavinin tonları sadece i yaparsanız kırmızının tonlarını elde edersiniz. İsterseniz renk değerlerini RGB fonksiyonundan da aldırabilirsiniz.

ÖRNEK: Çizgilerle renkli bir daire çizdirelim. Örneğimiz için bir Timer ve bir Label yerleştirin.

```
'ÖRNEK : Line
Private Sub Form_Load()
 Timer1.Interval = 10
 AutoRedraw = True
 Label1.AutoSize = True
End Sub
Private Sub Timer1_Timer()
  Dim x, y
  Static i
  i = (i + 1) Mod 360
  y = ScaleHeight / 2 + ScaleHeight / 2 * Sin(i * 3.1415 / 180)
  x = ScaleWidth / 2 + ScaleWidth / 2 * Cos(i * 3.1415 / 180)
  Label1.Top = y
  Label1.Left = x
  Line (ScaleWidth / 2, ScaleHeight / 2)-(x, y), i*10
  Label1 = i
End Sub
```



ÖRNEK: Örnek olarak basit bir analog saat yapalım. Saatimizde bulunacak saniye, yelkovan ve akrep'i Line komutlarıyla oluşturalım.

Saatin çizgilerini çizdirmek için Timer kontrolünü kullanacağız. Intervali 1000 yani 1saniye olan bir Timer işimizi görecektir. Saniye, dakika ve saat bilgisini ise **Time** komutu ile öğrenebiliriz.

Çizgileri ekrana çizdirmek için Cos ve Sin fonksiyonlarından yararlanabiliriz. Çünkü saatteki çizgiler dairesel bir hareket yapmaktadır. Sadece bunlara ait açıları bulmamız gerekir.

Dairede toplam 360 derece vardır bir dakikada ise 60 saniye. O halde bir saniyenin kaç dereceye denk geldiğini **360 derece / 60 saniye = 6 derece** formülü ile bulabiliriz. Tek fark saat açı yönünün tersi yönde ve 90 derece faz farkıyla ilerler. O halde merkezden (-cos(açı)*6+90, -cos(açı)*6+90) koordinatlarına çizilecek bir çizgi işimizi görecektir.

Dakika da saniye gibidir. Çünkü bir saatte 60 dakika vardır. Saatte ise formülün biraz değişmesi gerekiyor. Çünkü bir dilimde 12 saat var. Bunu da **360 derece / 12 saat = 30 derece** formülü ile bulabiliriz. O halde -cos(açı)*30+90 formülü işimizi görecektir.

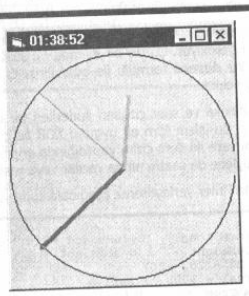
Ayrıca saniye, dakika ve saat çizgileri ilerlerken bir önceki bulundukları noktayı silmeleri gerekir. Bu işlem için en uygunu XOR ile çizim yapmaktır. Çünkü XOR kullanılarak aynı yere iki defa çizim yapıldığında orijinal haline döner, yani önceki silinmiş olur. Böylece de saatin altına yazılar veya resimler de çizdirilebilir.

Form üzerine bir Timer yerleştirerek aşağıdaki kodu yazın:

```
'ÖRNEK:Line
Private Sub Form_Load()
 Width = ScaleHeight
 Scale (-20, 20)-(20, -20)
 Timer1.Interval = 1000 '1 saniye
 AutoRedraw = True
 Circle (0, 0), 20
 DrawMode = 7 'xor
End Sub

Private Sub Timer1_Timer()
 Dim aci, saniye, dakika, saat
 Static sx, sy, dx, dy, stx, sty
 Caption = Time
 DrawWidth = 1
 Line (0, 0)-(sx, sy), QBColor(10) 'Öncekinin üzerine çizerek sil
 saniye = Second(Time) 'saniyeyi saatten al
 aci=-saniye*6+90 'her bir saniye 6 derecedir.360 derece/60saniye
 sx = 20 * Cos(aci * 3.1415 / 180)
 sy = 20 * Sin(aci * 3.1415 / 180)
 Line (0, 0)-(sx, sy), QBColor(10) 'saniyeyi çiz
```



```
 DrawWidth = 3
 Line (0, 0)-(dx, dy), QBColor(11) 'Öncekinin üzerine çizerek sil
 dakika = Minute(Time) 'dakikayı saatten al
 aci=-dakika*6+90 'her bir dakika 6 derecedir.360 derece/60dakika
 dx = 20 * Cos(aci * 3.1415 / 180)
 dy = 20 * Sin(aci * 3.1415 / 180)
 Line (0, 0)-(dx, dy), QBColor(11) 'yelkovanı çiz

 DrawWidth = 3
 Line(0,0)-(stx, sty), QBColor(12) 'Öncekinin üzerine çizerek sil
 saat = Hour(Time) 'saati saatten al
 aci=-saat*30+90'her saat 30 derecedir. 360 derece / 12 saat = 30
 stx = 12 * Cos(aci * 3.1415 / 180)
 sty = 12 * Sin(aci * 3.1415 / 180)
 Line (0, 0)-(stx, sty), QBColor(12) 'akrebi çiz
End Sub
```



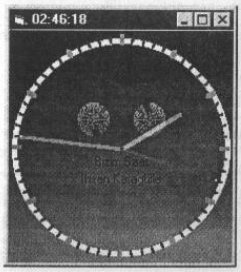
ÖRNEK: Yukarıdaki saatin daha gelişmiş bir versiyonu. Bu örneğimiz için formunuza iki tane Timer yerleştirin.

```
'ÖRNEK: Line
Private Sub Form_Load()
 Dim aci, i, t
 AutoRedraw = True
 Timer1.Interval = 1000 '1 saniye
 Timer2.Interval = 10 '1 salise
 'zemin desenini çiz
 ScaleMode = 3 'pixel moduna geçir
 For i = 0 To ScaleHeight
  Line (0, i)-(ScaleWidth, i), i * 256
 Next
 ScaleMode = 1 'normal moda dön
```



```
 'form yüksekliğini ve genişliğini aynı yap
 Width = ScaleHeight
 'matematiksel koordinatlara göre yeniden ölçekle
 Scale (-20, 20)-(20, -20)

 t = "Bizim Saat"
 CurrentX = -TextWidth(t) / 2 'orta noktayı bul
 CurrentY = -1
 Print t
 t = "İhsan Karagülle"
 CurrentX = -TextWidth(t) / 2
 CurrentY = -4
 Print t

 'saatin yuvarlağını çiz
 DrawWidth = 5
 Circle (0, 0), 19, 65535

 DrawWidth = 2

 'saniye çizgilerini çiz
 For aci = 0 To 360 Step 6
  Line (18 * Cos(aci * 3.1415 / 180), 18 * Sin(aci * 3.1415 /
180))-(19 * Cos(aci * 3.1415 / 180), 19 * Sin(aci * 3.1415 /
180)), QBColor(5)
 Next

 'Saat çizgilerini çiz
 DrawWidth = 4
 For aci = 0 To 360 Step 6 * 5
 Line (18 * Cos(aci * 3.1415 / 180), 18 * Sin(aci * 3.1415 /
180))-(19 * Cos(aci * 3.1415 / 180), 19 * Sin(aci * 3.1415 /
180)), QBColor(8)
 Next
 DrawMode = 7 'xor
End Sub

Private Sub Timer1_Timer()
 Dim aci, saniye, dakika, saat, i
 Static sx, sy, dx, dy, stx, sty
 Caption = Time
 DrawWidth = 2
 Line (0, 0)-(sx, sy), QBColor(10) 'saniyeyi çiz
 saniye = Second(Time) 'saniyeyi saatten al
 aci=-saniye*6+90 'her bir saniye 6 derecedir.360 derece/60saniye
 sx = 18 * Cos(aci * 3.1415 / 180)
 sy = 18 * Sin(aci * 3.1415 / 180)
 Line (0, 0)-(sx, sy), QBColor(10) 'saniyeyi çiz

 DrawWidth = 3
 Line (0, 0)-(dx, dy), QBColor(11) 'yelkovanı çiz
 dakika = Minute(Time) 'dakikayı saatten al
```

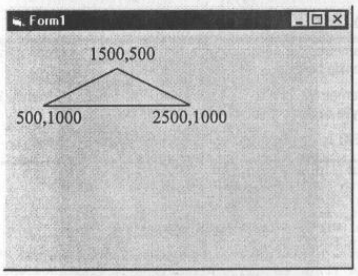
```
 aci=-dakika*6+90'her bir dakika 6 derecedir. 360derece/60dakika
 dx = 18 * Cos(aci * 3.1415 / 180)
 dy = 18 * Sin(aci * 3.1415 / 180)
 Line (0, 0)-(dx, dy), QBColor(11) 'yelkovanı çiz

 DrawWidth = 3
 Line (0, 0)-(stx, sty), QBColor(12) 'akrebi çiz
 saat = Hour(Time) 'saati saatten al
 aci = -saat*30+90 'her bir saat 30 derecedir.360 derece/12 saat
 stx = 12 * Cos(aci * 3.1415 / 180)
 sty = 12 * Sin(aci * 3.1415 / 180)
 Line (0, 0)-(stx, sty), QBColor(12) 'akrebi çiz

 'Saat başı ise zil çal
 If Minute(Time) = 0 Then Beep
End Sub

Private Sub Timer2_Timer()
 Static sls
 sls = (sls + 1) Mod 360
 Dim aci
 Dim sx, sy, dx, dy, stx, sty
 DrawWidth = 1
 aci = -sls*3.6+90 'her bir saniye 3.6 derecedir. 360 derece/100
 sx = 3 * Cos(aci * 3.1415 / 180)
 sy = 3 * Sin(aci * 3.1415 / 180)
 Line (5, 5)-(5 + sx, 5 + sy), QBColor(10) 'sağdaki küçük ibre
 Line (-5, 5)-(-5 - sx, 5 - sy), QBColor(10) 'soldaki küçük ibre
End Sub
```



Line -(x2,y2), renk

Line komutu bu haliyle kullanıldığında en son kalınan noktadan x2,y2 noktasına bir çizgi çizer. Dikkat ederseniz bu formda x1,y1 başlangıç koordinatları bulunmamaktadır. Bu koordinatlar bir önceki çizim komutunun kaldığı noktalar olarak kabul edilir ve o noktadan x2,y2 noktasına bir çizim yapar. Çok kenarlı şekilleri çizdirirken bu komut kolaylık sağlar.

Örnek olarak aşağıdaki üçgeni çizdirmek istediğimizi düşünelim.



Yukarıdaki üçgeni **Line (x1,y1)-(x2,y2)** komutlarıyla oluşturmak istesek aşağıdaki kodu kullanmamız gerekecektir.

```
Private Sub Form_Load()
 Show
 Line (1500, 500)-(500, 1000)
 Line (500, 1000)-(2500, 1000)
 Line (2500, 1000)-(1500, 500)
End Sub
```

Halbuki dikkat ederseniz her çizginin bitiş noktası diğerinin başlangıç noktasını oluşturmaktadır. O halde başlangıç koordinatını vermeden **Line –(x2,y2)** formunu kullanarak aynı işi daha kısa bir şekilde yapabiliriz:

```
Private Sub Form_Load()
 Show
 Line (1500, 500)-(500, 1000)
 Line -(2500, 1000)
 Line -(1500, 500)
End Sub
```


Dikdörtgen Çizme

Line (x1,y1)-(x2,y2),renk,B

Line komutunda en son parametre olarak B verilirse x1,y1 ve x2,y2 koordinatlarını köşe olarak kabul eden bir dikdörtgen çizer.

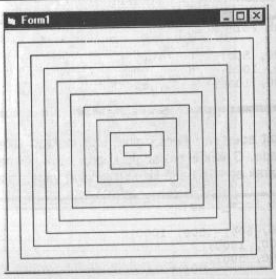
Line (x1,y1)-(x2,y2),renk,BF

F parametresi ile kullanıldığında ise dikdörtgenin içi renk parametresi ile belirlenen renkle boyanır.

F parametresi olmaksızın kullanıldığında ise dikdörtgenin için **ForeColor** ve **FillStyle** özellikleri ile belirlenen modda boyanır.

ÖRNEK: İç içe dikdörtgenler çizecek bir program yazalım.

```
'ÖRNEK : Line
Private Sub Form_Load()
 Show 'Önce formu göster
 Dim i, stp
 Form1.Width = Form1.Height
 stp = Form1.ScaleWidth / 20
 For i = 0 To Form1.ScaleHeight / 2 Step stp
  Line (i, i)-(Form1.ScaleWidth - i, Form1.ScaleHeight - i), , B
 Next
End Sub
```



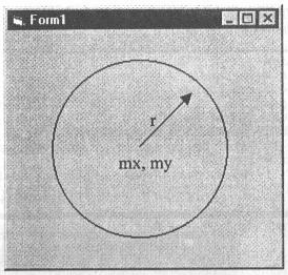
ÖRNEK: Aynı örneği içini değişik renklerde boyatarak da yapabiliriz. Boyama rengini belirlemek için **ForeColor** özelliğini kullanabiliriz.

```
'ÖRNEK : Line
Private Sub Form_Load()
 Show 'Önce formu göster
 Dim i, stp
 Form1.Width = Form1.Height
 stp = Form1.ScaleWidth / 20
 For i = 0 To Form1.ScaleHeight / 2 Step stp
  ForeColor = i * 200
  Line (i, i)-(Form1.ScaleWidth - i, Form1.ScaleHeight - i), , BF
 Next
End Sub
```

Çember Çizme

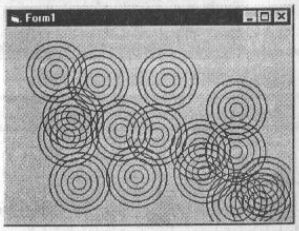
Circle(mx, my), r, renk

Circle komutu bu formuyla; merkezi mx, my olan r yarıçaplı çemberi verilen renkle çizer.



ÖRNEK: Kullanıcının fareyi tıkladığı noktaya içi içe daireler çizdirelim.

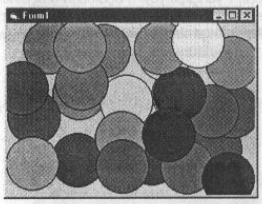
```
'ÖRNEK: Circle
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X
As Single, Y As Single)
 For i = 1 To 5
  Circle (X, Y), i * 100, i * 1500
 Next
End Sub
```



- Eğer içinin doldurulması isteniyorsa **FillStyle** özelliği 0 yapıldıktan sonra iç boyama rengi **FillColor** özelliği ile belirlenir.

ÖRNEK: Kullanıcının tıkladığı noktalara farklı renklere boyanmış daireler çizdirelim.

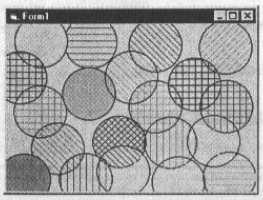

```
'ÖRNEK: Circle
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X
As Single, Y As Single)
 FillStyle = 0
 FillColor = Rnd * 16777216'rasgele bir renk
 Circle (X, Y), 500
End Sub
```



- İstenirse **FillStyle** özelliğine 0-7 arası bir değer verilerek iç boyaması desenli de yapılabilir.

ÖRNEK: Kullanıcının fareyle tıkladığı noktalara farklı renklerde ve farklı desenlerde daireler çizdirelim.

```
'ÖRNEK: Circle
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X
As Single, Y As Single)
 FillStyle = Rnd * 7
 FillColor = Rnd * 16777216
 Circle (X, Y), 500
End Sub
```



Yay Çizme

Circle (mx,my),r,renk,a,b

mx, my merkezli, r yarıçaplı yayı a açısından b açısına kadar verilen renkle çizer. Buradaki a ve b açıları radyan cinsindendir. Bu değerleri 0, 2pi verilirse yay tamamlanmış olur yani çember olur.

Circle (500, 500), 450, ,0, 3.1415 / 2 '0,Pi/2

Circle (500, 500), 450, , 0, 3.1415 '0,Pi

Elips Çizme

Circle (mx,my),r,renk,,,basıklık

mx,my merkezli r yarıçaplı elipsi verilen renkle çizer. Son parametre elipsin basıklık oranını belirler. Bu parametreye 1 verilerek çember, 1 den küçük veya büyük bir değer verilerek yatay veya dikey basık elips çizilir.

Circle (500, 500), 450, , , , 1.5

Circle (500, 500), 450, , , , 0.5

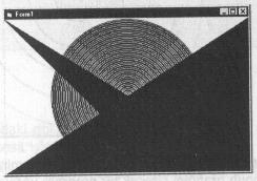
Circle (mx,my),r,renk,a,b,basıklık

Bu formda ise a açısından b açısına kadar bir elips yayı çizilir.

```
'ÖRNEK : Circle,Line
Private Sub Form_Load()
    Show'Önce formu göster
    Dim a, b, i
```



```
    a = Form1.ScaleWidth
    b = Form1.ScaleHeight
    For i = 0 To a
        Line (i, i)-(a, b)
        Line (a, i)-(i, b)
    Next
    For i = 1 To b / 2 Step b / 200
        Circle (a / 2, b / 2), i
    Next
End Sub
```



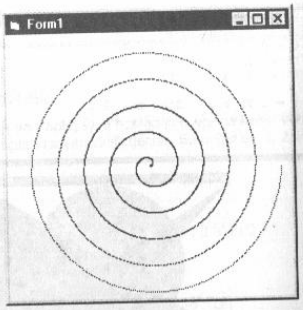
Noktasal Çizim

Pset (x,y),renk

x,y koordinatlarına verilen renkle bir nokta koyar. Bu noktanın kalınlığı **DrawWidth** özelliği ile değiştirilebilir. Bu metod daha çok matematiksel fonksiyonların çiziminde kullanılır.

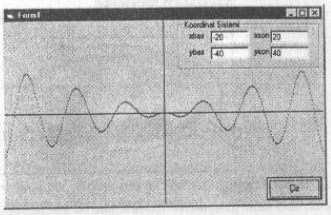
ÖRNEK: Örnek olarak bir spiral çizdirelim.

```
'ÖRNEK : Pset
Private Sub Form_Load()
    Show
    Dim x, y, i, r
    r = ScaleHeight / 2
    While r > 0
        i = (i + 1) Mod 360
        r = r - 1'Yarı çapı sürekli azalt
        y = ScaleHeight / 2 + r * Sin(i * 3.1415 / 180)
        x = ScaleWidth / 2 + r * Cos(i * 3.1415 / 180)
        PSet (x, y)
    Wend
End Sub
```



ÖRNEK: Örnek olarak kullanıcının koordinat sistemini belirleyebileceği matematiksel fonksiyonun grafiğini çizecek bir program yazalım.

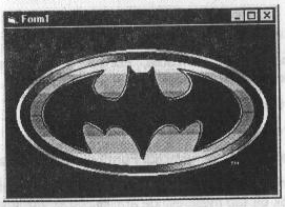
```
ÖRNEK: pset
Private Sub Form_Load()
 Text1 = -20
 Text2 = 20
 Text3 = -20
 Text4 = 20
End Sub

Private Sub Command1_Click()
 Dim xbas, xson, ybas, yson
 xbas = Val(Text1)
 xson = Val(Text2)
 ybas = Val(Text3)
 yson = Val(Text4)
 Scale (xbas, ybas)-(xson, yson) 'koordinat sistemini değiş
 Cls 'önce ekranı sil
 Line (0, ybas)-(0, yson) 'y eksenini çiz
 Line (xbas, 0)-(xson, 0) 'x eksenini çiz
 Dim x, y
 For x = xbas To xson Step 0.1
   'bu satır değiştirilerek farklı fonksiyonlar çizilebilir.
   y = x * Sin(x)
   PSet (x, y)
 Next
End Sub
```

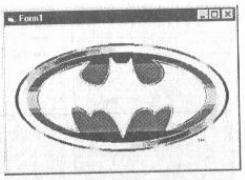
renk = Point (x,y)

x,y koordinatlarındaki noktanın rengini verir. Bu metod kullanılarak ekrandaki görüntü üzerinde basit işlemler yapmak mümkündür.

Örneğin form üzerindeki resimin negatifini alacak bir program yazalım. Formunuzun **Picture** özelliği ile bir resim belirleyin ve aşağıdaki kodu yazın.




```
'ÖRNEK : Point, Pset
Private Sub Form_Load()
 Show
 Dim i, j, r
 ScaleMode = 3 'pixel
 For i = 0 To ScaleWidth
  For j = 0 To ScaleHeight
   r = Point(i, j) 'i,j noktasının rengini öğren
   r = Not r ' tersini al
   PSet (i, j), r 'aynı noktaya tekrar koy
  Next
 Next
End Sub
```

Yazı Yazma

Print

Form üzerine veya yazıcıya kontrolden bağımsız olarak yazı yazmak için kullanılır.

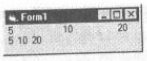
Yazının;

- Koordinatları **CurrentX, CurrentY** özellikleri ile
- Biçimi **FontName, FontSize, FontBold, FontItalic, FontUnderLine, FontStrikeThru** özellikleriyle
- Boyu **FontSize** özelliği ile
- Rengi **ForeColor** özelliği ile
- Zemin renginin olup olmayacağı **FontTransparent** özelliği ile belirlenebilir.

Print metodunda birden fazla değişken araya noktalı virgül koyularak yan yana, virgül koyularak bir tab mesafesinde aralıklı olarak yazdırılabilir.

```
Dim x,y,z
Show
x=5:y=10:z=20
Print x,y,z
Print x;y;z
```

Yukarıdaki program satırlarıyla oluşacak ekran görüntüsü aşağıdaki gibidir. Görüldüğü gibi ilk satırdaki sayılar arasında daha fazla boşluk vardır. İlk satır araya virgül koyularak yazdırılmıştır. İkinci satırdakiler arasında ise sadece bir boşluk vardır. Bunlar ise noktalı virgül kullanılarak yazdırılmıştır.

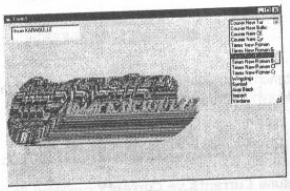


Print komutunun sonunda noktalı virgül veya virgül kullanılarak bir sonraki **Print** ifadesinin de aynı satırdan devam etmesi sağlanabilir.

Yazım Koordinatlarını Belirleme

Print komutunu **CurrentX** ve **CurrentY** özellikleri ile kullanarak metin istenen koordinatlara yazdırılabilir. Örnek olarak formunuzun üzerine bir text kutusu, ve bir liste kutusu koyarak aşağıdaki kodu yazalım.

```
'ÖRNEK : Print
Private Sub Form_Load ()
  fontsize = 30
  windowstate = 2'Maximize
  fontunderline = True
  Text1 = "İhsan KARAGÜLLE"
  Dim I
  For I = 0 To screen.FontCount - 1
      list1.AddItem screen.Fonts(I)'Ekran fontlarını listeye ekle
  Next
End Sub

Private Sub List1_Click ()
    fontname = list1.Text
    Cls
    Dim I
    Randomize Timer
    For I = 0 To 3000
        currentx = 1000 + I * Sin(I / 205)
        currenty = 2500 + I * Cos(I / 190)
        forecolor = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
        Print Text1
    Next
End Sub
```



Harf Boyutlarını Öğrenme

TextHeight(Metin), TextWidth(Metin)

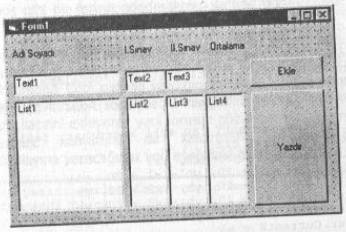
Bu özellikler, **Metin** parametresiyle verilen yazının form, picturebox veya yazıcıya **Print** metoduyla yazılması durumunda, yazının yüksekliğini ve genişliğini nesnenin **ScaleMode** özelliğiyle belirlenen birim cinsinden verir. Bu değerlere nesnenin **FontName, Fontsize, FontItalic, FontBold** gibi font özellikleri etki eder.

Bu özellikleri, özellikle yazıcıda tablo gibi aynı hizada bulunması gereken çıktılarda kullanmak gerekli olabilir.

ÖRNEK: Örnek olarak yazıcıdan aşağıdaki gibi bir liste almak istediğimizi düşünelim:

Buradaki yazıları **Printer.Print** komutu ile, ara çizgileri ise **Printer.Line** komutu ile çizdirebiliriz. Ancak çizgilerin çizdirileceği koordinatları belirleyebilmek için yazının yüksekliğini ve genişliğini de bilmemiz gerekir. Bunları da **Printer.TextWidth** ve **Printer.TextHeight** komutları ile öğrenebiliriz.

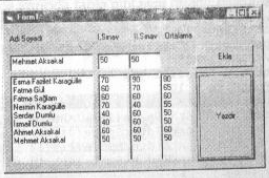
Örneğimiz için aşağıdaki formu oluşturalım:

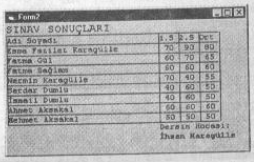



```
Private Sub Command1_Click()
 List1.AddItem Text1
 List2.AddItem Text2
 List3.AddItem Text3
 List4.AddItem (Val(Text2) + Val(Text3)) / 2
End Sub
Private Sub Command2_Click()
 Dim t, yb, ys, x1, x2, x3, x4, i
 Printer.FontSize = 14
 Printer.FontName = "Courier New Tur"
 t = "SINAV SONUÇLARI"
 Printer.Print t 'başlığı yaz
  Printer.FontSize = 10
 yb = Printer.CurrentY 'şu anki koordinatı sakla
 x1 = Printer.TextWidth(Space(30)) '30 harflik adı soyadı
 x2 = x1 + Printer.TextWidth(Space(4)) '4 harflik sınav notu
 x3 = x2 + Printer.TextWidth(Space(4)) '4 harflik sınav notu
 x4 = x3 + Printer.TextWidth(Space(4)) '4 harflik sınav notu
 'ilk yatay çizgiyi çiz
 Printer.Line (20, yb)-(x4 - 20, yb)
 'Başlığı yaz
 Printer.CurrentX = 20
 Printer.Print "Adı Soyadı";
 Printer.CurrentX = x1 + 20
 Printer.Print "1.S";
 Printer.CurrentX = x2 + 20
 Printer.Print "2.S";
 Printer.CurrentX = x3 + 20
 Printer.Print "Ort"
  'İkince yatay çizgiyi çiz
 Printer.Line (20, Printer.CurrentY)-(x4 - 20, Printer.CurrentY)
```



```
 'verileri yazdır
For i = 0 To List1.ListCount - 1
 Printer.CurrentX = 20
 Printer.Print List1.List(i); 'adı soyadı
 Printer.CurrentX = x1 + 20
 Printer.Print Val(List2.List(i)); '1. sınav
 Printer.CurrentX = x2 + 20
 Printer.Print Val(List3.List(i)); '2. sınav
 Printer.CurrentX = x3 + 20
 Printer.Print Val(List4.List(i)) 'ortalama
 'ara çizgi
 Printer.Line (20, Printer.CurrentY)-(x4 - 20, Printer.CurrentY)
Next
ys = Printer.CurrentY 'şu anki koordinatı sakla
 'dikey çizgileri çiz
Printer.Line (20, yb)-(20, ys)
Printer.Line (x1 + 20, yb)-(x1 + 20, ys)
Printer.Line (x2 + 20, yb)-(x2 + 20, ys)
Printer.Line (x3 + 20, yb)-(x3 + 20, ys)
Printer.Line (x4 + 20, yb)-(x4 + 20, ys)
Printer.CurrentX = x1
Printer.Print "Dersin Hocası:"
Printer.CurrentX = x1
Printer.FontBold = True
Printer.Print "İhsan Karagülle"
Printer.EndDoc
End Sub
```





Bu tür kodlar yazarken kodu denemek gerekir. Her seferinde yazıcıdan bir sayfa çıkararak kodu denemek kağıt ve zaman israfına sebep olacaktır. Print, Line, Circle gibi metodlar hem yazıcı, hem de formda bulunduğu için; sonuçlar yazıcıya gönderiyormuş gibi bir forma gönderilebilir ve kod doğru olarak çalıştıktan sonra forma ait komutlar yazıcıya çevrilerek kod tamamlanabilir.

Yukarıdaki örneği bir form üzerinde de deneyebiliriz. Project-Add form menüleri ile programınıza bir form daha yerleştirin. Bu formu yazıcıymış gibi kullanacağız. Yazdırma için Command_Click olayına yazdığımız koddaki bütün printer. ifadelerini form2. şekline çevirirseniz sonuçlar forma çizdirilecektir. Tabi ki kodun başında form2.show satırını ekleyerek yeni formun gözükmesini sağlamalısınız. Ayrıca printer.enddoc komutunu da kaldırmanız gerekir. Bu haliyle command2_click olayına yazacağımız kod aşağıdaki gibi olacaktır.

```
'ÖRNEK: Sonucu yazıcı yerine forma gönderme
Private Sub Command2_Click()
 Dim t, yb, ys, x1, x2, x3, x4, i
 Form2.Show
 Form2.FontSize = 14
 Form2.FontName = "Courier New Tur"
 t = "SINAV SONUÇLARI"
 Form2.Print t 'başlığı yaz

 Form2.FontSize = 10
 yb = Form2.CurrentY 'şu anki koordinatı sakla
 x1 = Form2.TextWidth(Space(30)) '30 harflik adı soyadı
 x2 = x1 + Form2.TextWidth(Space(4)) '4 harflik sınav notu
 x3 = x2 + Form2.TextWidth(Space(4)) '4 harflik sınav notu
 x4 = x3 + Form2.TextWidth(Space(4)) '4 harflik sınav notu
 'ilk yatay çizgiyi çiz
 Form2.Line (20, yb)-(x4 - 20, yb)
 'Başlığı yaz
 Form2.CurrentX = 20
 Form2.Print "Adı Soyadı";
 Form2.CurrentX = x1 + 20
 Form2.Print "1.S";
 Form2.CurrentX = x2 + 20
 Form2.Print "2.S";
 Form2.CurrentX = x3 + 20
 Form2.Print "Ort"

 'İkince yatay çizgiyi çiz
 Form2.Line (20, Form2.CurrentY)-(x4 - 20, Form2.CurrentY)

 'verileri yazdır
 For i = 0 To List1.ListCount - 1
  Form2.CurrentX = 20
  Form2.Print List1.List(i); 'adı soyadı
  Form2.CurrentX = x1 + 20
  Form2.Print Val(List2.List(i)); '1. sınav
  Form2.CurrentX = x2 + 20
```

```
  Form2.Print Val(List3.List(i)); '2. sinav
  Form2.CurrentX = x3 + 20
  Form2.Print Val(List4.List(i)) 'ortalama
  'ara çizgi
  Form2.Line (20, Form2.CurrentY)-(x4 - 20, Form2.CurrentY)
Next
ys = Form2.CurrentY 'şu anki koordinatı sakla

'dikey çizgileri çiz
Form2.Line (20, yb)-(20, ys)
Form2.Line (x1 + 20, yb)-(x1 + 20, ys)
Form2.Line (x2 + 20, yb)-(x2 + 20, ys)
Form2.Line (x3 + 20, yb)-(x3 + 20, ys)
Form2.Line (x4 + 20, yb)-(x4 + 20, ys)
Form2.CurrentX = x1
Form2.Print "Dersin Hocası:"
Form2.CurrentX = x1
Form2.FontBold = True
Form2.Print "İhsan Karagülle"
End Sub
```

Kursörlü Yazma

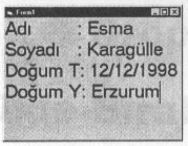
Dos altında program yapmış çoğu programcı ekrana direkt yazma işinin olmadığından şikayet ederler. Print komutu ile forma yazı yazdırılabilmesine rağmen bu ancak program kodu ile olabilmektedir. Peki kullanıcı forma yazı yazamaz mı? Basit bir kodla bu yapılabilir. Yapılması gereken formun KeyPress olayına yazılacak kodla basılan tuşları ekrana yazdırmaktır.

```
Private Sub Form_Load()
 KeyPreview = True
 AutoRedraw = True
End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)
 'basılan tuşu yaz
 Print Chr(KeyAscii);
End Sub
```

Yukarıdaki kod bu işi yapacaktır. Programı çalıştırırsanız forma yazı yazabileceğini görürsünüz. Tabi ki bir noksanlık var burada. O da kursorün olmaması. Kullanıcı bastığı harfin nereye yazılacağını göremeyecektir. Ancak basit bir kodla frma kursör de eklenebilir.

Kursör olarak kullanmak için forma düz bir **Line** kontrolü yerleştirin. Ayrıca kursörün yanıp sönmesini sağlayabilmek için de bir Timer yerleştirin.


```
Option Explicit
Dim kursorboyu

Private Sub Form_Load()
 KeyPreview = True
 Timer1.Interval = 70
 kursorboyu = 25 * FontSize
 AutoRedraw = True
End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)
 'basılan tuşu yaz
 Print Chr(KeyAscii);
 'kursörü yeni koordinatlara getir
 Line1.X1 = CurrentX
 Line1.X2 = CurrentX
 Line1.Y1 = CurrentY
 Line1.Y2 = CurrentY + kursorboyu
End Sub

Private Sub Timer1_Timer()
 'Kursorun yanıp sönmesini sağla
 Line1.Visible = Not Line1.Visible
End Sub
```



Artık forma giriş yaptırılabilir. Ayrıca silme işlemleri için gerekli kodu da yamanız gerekecektir.

SÜRÜKLE & BIRAK (Drag & Drop)

Windows'da bildiğimiz drag-drop (sürükle ve bırak) olayı bir nesnenin bir yerden kaldırılıp bir yere taşınması olayıdır. Bu olayları takip edebilmek için VB'de bir çok kontrolün Drag özellikleri, olayları ve metodları vardır.

Bir kontrolün taşınması işleminin otomatik olarak yapılıp yapılmayacağı o kontrolün **DragMode** özelliği ile, taşınırken alacağı şekil ise **DragIcon** özelliği ile belirlenir.

Taşıma gerçekleşirken, taşıdığınız şey diğer kontrollerin üzerinden geçecek ve sonuçta birinin üzerine bırakılacaktır. Bu işlemler sırasında üzerinden geçilen nesne **DragOver** olayı, üzerine bırakıldığı nesne ise **DragDrop** olayı meydana gelir.

Eğer **DragMode** özelliği ile taşıma işleminin otomatik olarak yapılmayacağı belirtilmişse bu durumda taşıma işlemi **Drag** metodu ile yönlendirilir.

Ayrıca VB'nin 5.0 versiyonu ile gelen ve VB 6.0'da da bulunan çok güzel bir özellik daha var. Bu da **Ole Drag & Drop** olaylarıdır. Bu olayları kullanarak sadece kendi programınızdaki kontroller arasında değil diğer programlar arasında da sürükleme ve bırakma yapabilirsiniz. Örneğin Word'de seçtiğiniz kısmı sürükleyip programınızdaki bir Text kutusuna bırakabilir veya tersini yapabilirsiniz.

STANDART DRAG & DROP

Properties

DragIcon

Windows Gezgininde bir dosyayı başka bir dizine mouse ile taşımak istediğimizde mouse şeklinin değiştiğini fark etmişsinizdir. İşte bu özellik de o nesneyi mouse ile taşırken alacağı şekli belirten icondur. Fareyle taşıma yaparken nesne, burada seçtiğiniz icona dönüşecektir. **LoadPicture** komutu ile program çalışırken de icon değiştirilebilir.

DragMode

Bu özelliğin alabileceği iki değer vardır.

0 : vbManual : Bu durumda kullanıcı nesneyi taşımak istediğinde DragDrop olayında yazdığınız kodla nesnenin koordinatlarını değiştirerek sürüklenmesi ve o nesnenin bırakılması sağlanmalıdır.

1 : vbAutomatic : Bu durumda ise nesne otomatik olarak sürüklenir sadece bırakma işlemi için kod yazılması gerekir.

Ancak drag modunun automatic olması bazı problemlere sebep olur. Çünkü bu modda nesnenin **Click** olayı meydana gelmez. Yani bir text kutusunun **DragMode** özelliğine bu değer verilmişse kullanıcı Text kutusundaki yazıyı mouse ile seçemez. Bir listenin **DragMode** özelliğine bu değer verilmişse kullanıcı mouse kullanarak bir elemanı seçemez.

Genel olarak kullanıcının fare ile seçim yaptığı TextBox, ListBox gibi kontrollerin **DragMode** özelliğine **Manual**, Label, PictureBox gibi kontrollerinkine ise **Automatic** verilebilir.

DragMode özelliği **Manual** olarak belirlenmiş kontrollerde **Drag** olayı, o kontrolün **MouseDown** olayında, **Drag** metodu ile başlatılır.

```
Private Sub Text1_MouseDown(Button As Integer, Shift As Integer,
X As Single, Y As Single)
 'Farenin sol tuşuna basıldı ise Drag işlemini başlat
 If Button = 1 Then Text1.Drag
End Sub
```

Events

DragDrop(Source As Control, X As Single, Y as Single)

Taşınan nesnenin (Source) bir kontrol üzerine bırakılması durumunda, üzerine bırakılan kontrolde DragDrop olayı meydana gelir. Sürükle-Bırak işleminde bırakmak için gerekli kod bu olaya yazılmalıdır.

Source : Taşıma sonucu bu nesnenin üzerine bırakılan nesne

X,Y: Mouse koordinatları (dolayısıyla sürüklenen elemanın bırakıldığı noktanın koordinatları)

Windows Gezgininde bir dosyayı mouse ile bir başka dosya veya dizin üzerine bıraktığınızda, hedef dosyanın buna tepki gösterdiğini fark etmişsinizdir. Bu işlemi hedef nesnenin **dragdrop** olayı yapar.

ÖRNEK: Örnek olarak bir text kutusunu başka bir text kutusu üzerine taşıdığımızda içeriği direk olarak transfer edilsin. Ancak taşınan nesnenin text kutusu olmayan bir nesne olması halinde ise bip sesi versin.

Bu örnek için **dragmode** özellikleri **1 (Automatic)** olan iki adet **Text** kutusu ve bir adet **Label** kontrolü kullanalım.

```
'ÖRNEK : DragDrop
Private Sub Text1_DragDrop (Source As Control, X As Single, Y As
Single)
  If TypeOf Source Is TextBox Then
   ' Sürüklenen kaynağın tipi textbox ise
   Text1 = Source 'Kaynağın içeriğini text1'e kopyala
  Else
   Beep' değilse sesli olarak uyar
  End If
End Sub

Private Sub Text2_DragDrop (Source As Control, X As Single, Y As
Single)
  If TypeOf Source Is TextBox Then
   ' Sürüklenen kaynağın tipi textbox ise
   Text2 = Source 'Kaynağın içeriğini text2'ye kopyala
  Else
   Beep' değilse sesli olarak uyar
  End If
End Sub
```

Programı çalıştırıp Text1 kutusunu Text2 üzerine bırakırsanız, Text1 içeriğinin Text2'ye aktarıldığını, ancak Label1 kutusunu taşıdığınızda aynı işlemi yapmadığını ve bip sesiyle uyardığını göreceksiniz.

DragOver(Source As Control, X As Single, Y as Single, Status As Integer)

Taşıma sırasında hedef nesne üzerinden geçilirken bu olay meydana gelir.

Source : Taşınan nesne

X,Y: Mouse koordinatları (dolayısıyla sürüklenen elemanın bırakıldığı noktanın koordinatları).

State : Taşınan nesnenin, hedef üzerinden geçiş durumunu gösterir. Şu üç değerden birini alır:

0 : vbEnter: Taşınan nesnenin hedef üzerine giriş yaptığını,

1 : vbLeave: Taşınan nesnenin hedef üzerinden ayrıldığını,

2 : vbOver: Taşınan nesnenin hedef üzerinde geçmekte olduğunu ifade eder.

0 ve 1 durumu anlık bir olaydır. 2 durumu daha uzun bir süre meydana gelecektir.

Windows Gezgininde bir dosyayı mouse ile taşırken bırakılması mümkün olmayan bir yerden geçtiğinde, buna mouse şeklinin değiştirilerek tepki gösterdiğini biliyoruz. Bu işlemi, üzerinden geçilen nesnenin **dragover** olayı gerçekleştirir.

ÖRNEK: Yukarıdaki örneğimizi geliştirerek text kutusu üzerinden text kutusu olmayan bir nesne geçtiğinde, text kutusunun zemin renginin değişmesini sağlayalım.

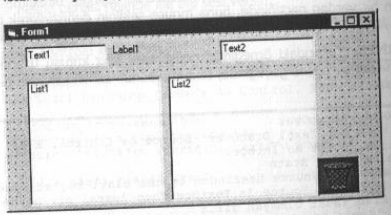
```
'ÖRNEK : DragOver
Private Sub Text1_DragOver (Source As Control, X As Single, Y As
Single, State As Integer)
  Select Case State
   Case 0 : 'Source üzerinden taşıma olayı başladı ise
     If TypeOf Source Is TextBox Then 'yasal bir nesne ise
     Else 'Yasal olmayan giriş
      'Zemin rengini değiştirerek tepki göster.
      Text1.Backcolor = Text1.Backcolor XOR Text1.ForeColor
     End If
   Case 1: 'Source üzerinden çıkış yapıldı ise
     If TypeOf Source Is TextBox Then 'yasal bir nesne ise
     Else 'Yasal olmayan elemanın çıkışı
      'zemin rengini orijinal haline getir.
      Text1.Backcolor = Text1.Backcolor XOR Text1.ForeColor
     End If
  End Select
End Sub

Private Sub Text1_DragDrop (Source As Control, X As Single, Y As
Single)
```



```
  If TypeOf Source Is TextBox Then' Sürüklenen kaynağın tipi
textbox ise
   Text1 = Source 'Kaynağın içeriğini text1'e kopyala
  Else
   Beep' değilse sesli olarak uyar
   Text1.Backcolor = Text1.Backcolor XOR Text1.ForeColor
  End If
End Sub
```

Yukarıdaki örnekte, Text kutusu olmayan bir nesnenin kutu üzerinden geçmesi durumunda, nesne giriş yapmışsa zemin rengi değiştirilir. Xor işlemine tabi tutulması Xor'un karakteristik özelliğindendir. Çünkü bir sayı iki defa aynı sayı ile Xor işlemine tabi tutulursa orijinal hali elde edilir. Dikkat edilirse nesne çıkış yaparken de aynı işlem tekrar uygulanarak zemin rengi orijinal hale getirilmektedir. Burada problem nesne giriş yapar da çıkış yapmaz ise yani bırakılırsa orijinal renge geri dönülemeyecektir. Bunun için de DragDrop'a eklenen bir kodla zemin rengi orijinal haline getirilir.

ÖRNEK: Başka bir örnek olarak ta ekrandaki bütün kontrollerin drag drop yöntemi ile değiştirilebileceği bir program yazalım. Örneğimiz için aşağıdaki formu oluşturup PictureBox içine Çöp kutusu resmi yerleştirin.



Amacımız ekrandaki bütün kontrollerin birbirleriyle Drag-Drop yöntemi ile veri transferi yapabilmelerini sağlamak. Ayrıca çöp kutusuna sürükleyerek de silmek.

```
'ÖRNEK : DragOver,DragDrop,Drag
Private Sub Form_Load()
    Label1.DragMode = 1
End Sub
Private Sub picture1_DragDrop(Source As Control, X As Single, Y
As Single)
    On Local Error Resume Next
    Dim i, j
    If TypeOf Source Is ListBox Then
        'Silme animasyonu yap
```



```
        For i = 30 To 8 Step -1
            Picture1.FontSize = i
            Picture1.Print Source.Text
            For j = 1 To 10000: Next
            DoEvents
            Picture1.Cls
        Next
        Source.RemoveItem Source.ListIndex
    End If
    If TypeOf Source Is TextBox Then Source = ""
End Sub
Private Sub Text1_DragDrop(Source As Control, X As Single, Y As
Single)
    On Local Error Resume Next
    If TypeOf Source Is TextBox Then Text1 = Source
    If TypeOf Source Is Label Then Text1 = Source
    If TypeOf Source Is ListBox Then Text1 = Source.Text
End Sub
Private Sub Text1_MouseDown(button As Integer, Shift As Integer,
X As Single, Y As Single)
    'Farenin sol tuşuna basıldı ise Drag işlemini başlat
    If button = 1 Then Text1.Drag
End Sub
Private Sub Text2_MouseDown(button As Integer, Shift As Integer,
X As Single, Y As Single)
    'Farenin sol tuşuna basıldı ise Drag işlemini başlat
    If button = 1 Then Text2.Drag
End Sub
Private Sub Text2_DragDrop(Source As Control, X As Single, Y As
Single)
    On Local Error Resume Next
    If TypeOf Source Is TextBox Then Text2 = Source
    If TypeOf Source Is Label Then Text2 = Source
    If TypeOf Source Is ListBox Then Text2 = Source.Text
End Sub
Private Sub List1_DragDrop(Source As Control, X As Single, Y As
Single)
    On Local Error Resume Next
    If TypeOf Source Is TextBox Then List1.AddItem Source
    If TypeOf Source Is Label Then List1.AddItem Source
    If TypeOf Source Is ListBox Then List1.AddItem Source.Text
End Sub
Private Sub List1_MouseDown(button As Integer, Shift As Integer,
X As Single, Y As Single)
    If button = 1 Then List1.Drag
End Sub
Private Sub List2_DragDrop(Source As Control, X As Single, Y As
Single)
    On Local Error Resume Next
    If TypeOf Source Is TextBox Then List2.AddItem Source
    If TypeOf Source Is Label Then List2.AddItem Source
    If TypeOf Source Is ListBox Then List2.AddItem Source.Text
End Sub
```

```
Private Sub List2_MouseDown(button As Integer, Shift As Integer,
X As Single, Y As Single)
    If button = 1 Then List2.Drag
End Sub
```

Methods

Drag

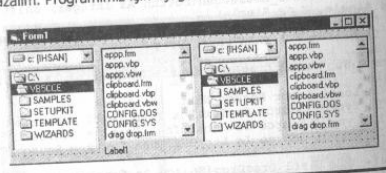
Eğer DragMode özelliğine **Manual** verilmişse Drag-Drop olaylarının **Drag** metodu ile yönlendirilmesi gerekir. Şu parametrelerle birlikte kullanılır.

0:VbCancel: Sürükleme işlemini iptal et

1:vbBeginDrag: Sürükleme işlemini başlat

2:vbEndDrag: Sürükleme işlemini sona erdir.

ÖRNEK: Örnek olarak sürükle-bırak yöntemi ile dosya kopyalayabilecek bir program yazalım. Programımız için aşağıdaki formu oluşturun.




```
'ÖRNEK : DragOver,Drag,DragDrop
Option Explicit
Dim d1, d2

Public Function dosyaadi(p, d)
    If Right(p, 1) = "\" Then
        'yol içerisinde \ varsa
        dosyaadi = p & d
    Else
        dosyaadi = p & "\" & d
    End If
End Function

Private Sub Form_Load()
    Label1.Visible = 0
    File1.Pattern = "*.*"
End Sub
```



```
Private Sub File2_MouseDown(Button As Integer, Shift As Integer,
X As Single, Y As Single)
    d1 = dosyaadi(File2.Path, File2.filename)
    Label1.Move File2.Left, File2.Top + Y
    Label1.Drag
End Sub

Private Sub File1_MouseDown(Button As Integer, Shift As Integer,
X As Single, Y As Single)
    d1 = dosyaadi(File1.Path, File1.filename)
    Label1.Move File1.Left, File1.Top + Y
    Label1.Drag
End Sub

Private Sub File1_DragDrop(Source As Control, X As Single, Y As
Single)
    On Local Error GoTo hata1
    d2 = dosyaadi(File1.Path, File2.filename)
    FileCopy d1, d2
    File1.Refresh
    Exit Sub
hata1:
    MsgBox ("Bu işlem başarılamadı. " & Error)
    Exit Sub
End Sub

Private Sub File2_DragDrop(Source As Control, X As Single, Y As
Single)
    On Local Error GoTo hata2
    d2 = dosyaadi(File2.Path, File1.filename)
    FileCopy d1, d2
    File2.Refresh
Exit Sub
hata2:
    MsgBox ("Bu işlem başarılamadı. " & Error)
    Exit Sub
End Sub

Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub

Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub
Private Sub Dir2_Change()
    File2.Path = Dir2.Path
End Sub
Private Sub Drive2_Change()
    Dir2.Path = Drive2.Drive
End Sub
```


OLE DRAG & DROP

Properties

OLEDragMode

Taşıma işleminin nasıl yapılacağını belirler.

0, vbOLEDragAutomatic: Bu modda sürükleme işlemi otomatik olarak başlatılır.

1,vbOLEDragManual: Bu modda sürükleme işlemini başlatmak için OLEDrag metodu kullanılmalıdır.

OLEDropMode

Bırakma işleminin nasıl yapılacağını belirler.

0,vbOLEDropNone: Nesne bırakma işlemini desteklemeyecek.

1: vbOLEDropManual: Bırakma işlemi program kodu ile yapılacak.

0: vbOLEDropAutomatic: Bu modda hiç bir koda gerek kalmadan bırakma işlemi gerçekleştirilebilir.

Eğer bir kontrolün hem **OleDragMode** ve hem de **OleDropMode** özellikleri otomatik değerleri destekliyorsa bu kontrollere hiç bir kod yazmanız gerekmez. Sadece bu iki özelliğe **Automatic** değerlerini vermeniz yeterlidir.

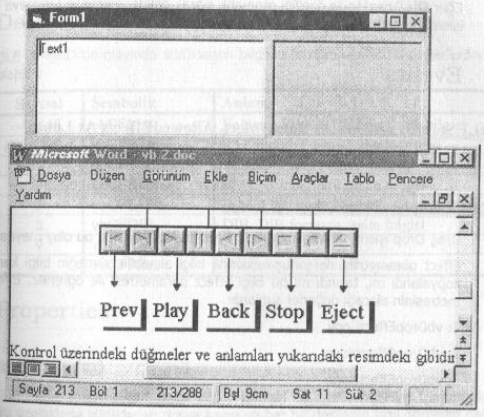
ÖRNEK: Örnek olarak programınıza iki tane Text kutusu yerleştirin ve her ikisinde de bu özelliklerini Automatic olarak ayarlayıp programı çalıştırın. Bu iki kutu arasında seçtiğiniz kısmı taşıyabileceğiniz gibi bu kutulardan seçip OLE Drag&Drop işlemlerini destekleyen diğer uygulamalara da taşıyabileceğinizi veya oradan bu kutulara taşıyabileceğinizi göreceksiniz.

Formunuzun üzerine bir FileListBox koyup OleDragMode özelliğini Automatic yaparsanız artık buradaki dosyalardan seçip istediğiniz bir yere taşıyarak oraya kopyalayabilirsiniz. Örneğin masa üstüne veya Explorer de bir dizine bırakarak oraya kopyalanmasını sağlayabilirsiniz. Hatta FileListBox'un MultiSelect özelliğini 1 veya 2 yaparak birden fazla dosyayı aynı anda istediğiniz yere kopyalayabilirsiniz. Tabi bunlar için hiç bir kod yazmadan.

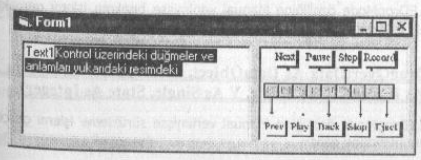
Aynı şekilde bir PictureBox'un bu iki özelliğine Automatic değerini vererek diğer uygulamalardaki resimleri getirip PictureBox üzerine bırakabilirsiniz.

Bu işlemler sırasında Ctrl tuşuna basılı tutarsanız sürüklediğiniz şey karşı tarafa kopyalanır, bu tuşa basılı tutmadan sürüklerseniz oraya taşınır.

Aşağıda VB'de tasarlanmış ve OleDrag/DropMode özellikleri Automatic yapılmış bir TextBox ve bir PictureBox görülmektedir.



Word programını da çalıştırıp formunuzdan Word'e resim/yazı alabileceğiniz gibi Word'den de formunuza resim ve yazı alabilirsiniz.



Methods

OLEDrag()

Eğer **OleDragMode** özelliği otomatik olarak belirlenmemişse taşımaya başlama işlemi bu metodla yapılır.

Events

OLEStartDrag(Data As DataObject, AllowedEffects As Long)

Sürükleme işlemi başladığında bu olay meydana gelir.

OLECompleteDrag(Effect As Long)

Drag-Drop işlemi tamamlandığında veya iptal edildiğinde bu olay meydana gelir.

Effect parametresi ile sonuç hakkında bilgi alınabilir. Örneğin bilgi karşı tarafa kopyalandı mı, taşındı mı bu bilgi Effect parametresi ile öğrenilir. Effect parametresinin alacağı değerler şunlardır.

1: vbDropEffectCopy

2: vbDropEffectMove

3: vbDropEffectNone

OLEDragDrop(Data As DataObject, Effect As Long, Button As Integer, Shift As Integer, X As Single, Y As Single)

OLEDropMode özelliğine Manual verilmişse bırakma işlemi gerçekleştiğinde bu olay meydana gelir.

OLEDragOver(Data As DataObject, Effect As Long, Button As Integer, Shift As Integer, X As Single, Y As Single, State As Integer)

OLEDropMode özelliğine Manual verilmişse sürükleme işlemi gerçekleşirken bu olay meydana gelir.

OLEGiveFeedback(Effect As Long, DefaultCursors As Boolean)

Bu olay, OLE kaynağının sürükle ve bırak işlemi esnasında fare şeklinin değişmesi gerektiği durumlarda meydana gelir.

DataObject

OLE Drag&Drop olayında sürüklenen bilginin formatı aşağıdaki değerlerden biri olabilir.

Sayısal	Sembolik	Anlamı
-16640	vbCFLink	Bağlantı Bilgisi, DDE bağlantı bilgisi
-16639	vbCFRTF	RTF metni, Formatlı metin bilgisi
1	vbCFText	Text, Formatsız metin bilgisi
2	vbCFBitmap	BMP, Bitmap türü resim bilgisi
3	vbCFEMetafile	WMF, Windows Metafile türü resim bilgisi
8	vbCFDIB	DIB, DIB formatlı resim bilgisi
9	vbCFPalette	Palet bilgisi
15	vbCFFiles	Windows Explorer Dosya adı

Properties

Count,Files(Index)

Sürükleme işlemindeki Datalar dosya isimleri ise, yani vbCFFiles formatında ise bu özelliklerle sürüklenen dosya sayısı ve her bir dosyanın ismi öğrenilip değiştirilebilir.

Methods

Clear()

Datanın içeriği silinir.

GetData(Format As Integer)

Format parametresi ile belirlenen bilgiyi öğrenmek için kullanılır.

GetFormat(Format As Integer) As Boolean

Datanın Format parametresi ile belirlenen formatı destekleyip desteklemediği bu özellikle öğrenilir. Geriye dönen değer True ise bilgi o formattadır.

SetData([Bilgi], [Format])

Taşınacak bilgi ve bu bilginin formatı bu metodla belirlenebilir.

Bölüm 9

DDE & Link

DDE & LINK

Windows altındaki programlar DDE denen teknoloji sayesinde birbirleri ile haberleşebilirler. Örneğin bir program, program yöneticisine bir program grubu oluşturmasını söyleyebilir. Veya bir program Excel'deki bir hücre ile bağlantı kurarak oradaki bütün değişimlerden haberdar olabilir.

Bütün bu işlemler VB de **Link** özellikleri, metodları ve olayları ile desteklenir. Bu bölümde VB'den bir uygulama ile nasıl bağlantı kurulur, nasıl mesaj gönderilir ve gelen mesajlar nasıl değerlendirilir, bunları göreceğiz.

Properties

LinkItem

Bağlantının kurulacağı kontrol bu özellikle belirlenir.

LinkTopic

Bağlantı kurulan kontrol bir formun veya başka bir gruplayıcı elemanın parçası olacağı için bu özelliğe o kontrolün bulunduğu programın ve formun ismi araya | karakteri konarak verilir.

Örneğin Excel içindeki Sayfa1 tablosu ile bağlantı kurarken bu özelliğe

```
Text1.LinkTopic="Excel|Sayfa1"
```

değerini vermek gerekir. Bu durumda LinkItem özelliğine bağlantı kurulacak hücrenin adresi verilir.

```
Text1.LinkItem="R1C1"
```

Her uygulamada ikinci parametre bulunmayabilir. Bu durumda ise birincisi ile aynı değeri vermek gerekir. Örneğin Program yöneticisi (ProgMan) ile bağlantı kurarken her iki parametreye de aynı değeri vermek gerekir.

```
"Program|Progman"
```

Eğer bağlantıyı VB ile yapılmış bir programla yapacaksanız LinkTopic özelliğine "Programınİsmi|FormunİsmiI" değerini vermelisiniz. Buradaki programın ismi program derlenirken verilen isimdir. Eğer program henüz derlenmemişse proje dosyasının ismidir. Formun ismi ise formun Name özelliği ile belirlenen değerdir. Bu durumda LinkItem özelliğinde bağlantı kurulacak kontrolün ismini vermeniz

gerekir. VB'de sadece bir Label, TextBox veya PictureBox ile bağlantı kurabilirsiniz.

Örneğin VB'de yapılmış ABC isimli programın Form1 isimli formu üzerindeki Text1 isimli Text kutusu ile bağlantı kuracaksanız

```
Text1.LinkTopic= "ABC|Form1"
Text1.LinkItem= "Text1"
```

vermeniz gerekir.

Burada programın ismi her zaman programın Exe dosyasının ismi olmayabilir. Program derlenirken verilen isimdir. Bu isim App.ExeName özelliği ile temsil edilir. Kullanıcı programın Exe dosyasının ismini değiştirmiş olsa bile orijinal isim geçerlidir.

Excel'den kendi programınızdaki bir kontrol ile bağlantı kurmak isteyelim. Programımızın ismi ABC, formumuzun ismi Form1 ve bağlantı kuracağımız kontrolün ismi ise Text1 olsun. Bu kontrolün içeriğini Excel'den alıp işlem yapmak için Excel'deki bir hücreye

```
=Abc|Form1!Text1
```

formülünü yazmak gerekir.

LinkMode

Yapılacak bağlantının ne zaman güncelleneceği bu özellik vasıtası ile belirlenir.

0, vbLinkNone:	Herhangi bir bağlantı yok.
1, vbLinkAutomatic:	Bağlantı yapılan bilgi değiştiğinde, otomatik olarak kontrolün içeriği de değişecek.
2, vbLinkManual:	Bağlantı yapılan bilginin güncelleştirilmesi sadece **LinkRequest** metodu ile yapılacak.
3, vbLinkNotify:	Bağlantı yapılan bilgi değiştiğinde **LinkNotify** olayı meydana gelecek ve **LinkRequest** metodu kullanılırsa bilgi güncelleştirilecek.

LinkTimeout

Bağlantı kurulan uygulamanın cevap vermesi için maksimum ne kadar bekleneceği bu özellik ile belirlenir. Bu özelliğe verilecek değer saniyenin onda biridir. Yani 100 verilmesi 10 saniye anlamına gelir. Maksimum 65535 verilebilir ki buda yaklaşık 2 saat yapar. Bu süre içinde kullanıcı ESC tuşu ile bağlantıyı iptal eder.

Methods

LinkRequest

Eğer **LinkMode** özelliği ile bağlantı şekli otomatik olarak belirlenmemişse kontrolün içeriğinin güncellenmesi için **LinkRequest** metodu kullanılır. Örneğin bir Text kutusu Excel'deki bir hücre ile bağlantısı kurulmuşsa Text kutusunun içeriğini güncelleştirmek için

```
Text1.LinkRequest
```

metodu kullanılmalıdır.

LinkPoke

Kontrolün içeriği bağlantı kurulan uygulamadaki kontrole bu metod ile gönderilir. Eğer bir PictureBox ile bağlantı kurulmuşsa bu metod picture box içindeki resimi, bir Label veya Text kutusu ile bağlantı kurulmuşsa onların içeriğindeki metin gönderilecektir.

```
Picture1.LinkPoke
```

şeklinde kullanılır.

LinkExecute

Bağlantı kurulan uygulama ile haberleşme bu metodla yapılır. Bu metoda karşı tarafın anlayacağı bir string gönderilir.

Bu metodda kullanacağınız stringin içeriği bağlantı kurduğunuz uygulamaya göre değişir. Örneğin Word, Excel gibi bazı programlar kendilerine ait makroları bu metodla kullanabilmenizi sağlar. Word veya Excel ile bağlantı kurarken onlara ait makroları [ ] parantezleri içinde verebilirsiniz.

ÖRNEK: Bir çok install programı, programı kurduktan sonra program yöneticisinde (Win95/98'de Başlat-Programlar menüsü altında) bir program grubu oluşturur ve programa ait simgeleri bu gruba yerleştirir. Sizde yaptığınız programların Install programlarını yazarken bu işlemlerin yapılmasını gerçekleştirebilirsiniz.

Önce programınıza bir **Text** kutusu yerleştirin. Program yöneticisi ile bu kontrol üzerinden bağlantı kuracağız.

İlk yapmamız gereken LinkTopic özelliğini aşağıdaki gibi belirlemek olacaktır.

```
Text1.LinkTopic = "ProgMan|ProgMan"
```


Program yöneticisinde herhangi bir kontrolle bağlantı kurmayacağımız için **LinkItem** özelliğini boş bırakacağız.

```
Text1.LinkItem = ""
```

Daha sonra **LinkItem** özelliği ile bağlantı modunu **Manual** olarak belirleyelim.

```
Text1.LinkMode = 2
```

Son olarak da **LinkExecute** metodu ile Program yöneticisine komutlar göndereceğiz.

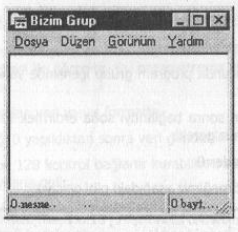
Program yöneticisinde bir Program grubu oluşturmak için:

"[CreateGroup(grup adı)]"

şeklinde kullanmamız gerekiyor. Buradaki grup adı yerine oluşturulmasını istediğimiz grubun adını gireceğiz. Örnek olarak bu grubun adı "Bizim grup" olsun.

```
Text1.LinkExecute "[CreateGroup(Bizim Grup)]"
```

Bu işlem aşağıdaki gibi boş bir program grubu oluşturacaktır:



Bunun içine program öğeleri yerleştirmek için ise

"[AddItem(DosyaAdı, Tanımı)]"

şeklinde kullanacağız.

Örneğimizde VB.EXE dosyasını Visual Basic tanımlı bir program öğesi olarak oluşturmak isteyelim.

```
Text1.LinkExecute "[AddItem(VB.exe, Visual Basic)]"
```

Burada dosya adını dosyanın bulunduğu sürücüyü ve yolu da belirtmeniz gerekir. Yukarıdaki örnekte VB.EXE dosyası c:\VB dizininde ise

"[AddItem(VB.exe, Visual Basic)]"

satırını

"[AddItem(c:\vb\VB.exe, Visual Basic)]"

şeklinde kullanmanız gerekir.

Bu işlem sonucunda program grubu içerisinde yandaki gibi program öğesi de oluşturulacaktır.

Bu işlemlerden sonra bağlantıyı sona erdirmek için tekrar **LinkMode** özelliğini **None** yapmamız gerekir.

```
Text1.LinkMode=0
```

Sonuç olarak örneğimiz aşağıdaki gibi olacaktır.

```
'ÖRNEK : LinkMode,LinkTopic,LinkItem,LinkExecute
Private Sub Form_Load()
 Text1.LinkMode = 0
 Text1.LinkTopic = "ProgMan|ProgMan"
 Text1.LinkItem = ""
 Text1.LinkMode = 2
 Text1.LinkExecute"[CreateGroup(Bizim Grup)]"
 Text1.LinkExecute"[AddItem(c:\msoffice\vb\VB.exe,Visual Basic)]"
 Text1.LinkMode = 0
End Sub
```

LinkSend

Bir PictureBox ile bağlantı otomatik olarak kurulmuş bile olsa Picture kutularındaki resimler çok büyük olabileceği ve işlemleri yavaşlatabileceği için bunlar otomatik olarak güncellenmez. **LinkSend** metodu kullanarak değişimlerin güncellenmesi gerekir.

```
Picture1.LinkSend
```


Events

LinkClose ( )

Yapılan bağlantı sona erdiğinde bu olay meydana gelir. **LinkMode** özelliğine 0 haricinde bir değer verildiğinde bağlantı kurulu ve **LinkMode** özelliğine 0 verildiğinde ise bağlantı sona erer ve bu olay meydana gelir. Bağlantı kurulduktan sonra bu bağlantı iptal edilmese bile programınız sona erdiğinde VB tarafından bu bağlantı da sona erdirilir.

Bu olay daha çok kullanıcının bağlantıyı kopardığı durumlarda yapılması gereken işlemler için kullanılır.

LinkError (LinkErr As Integer)

Bağlantı sırasında bir hata oluştuğunda bu olay meydana gelir. Hatanın ne olduğu ise LinkErr parametresi ile öğrenilir. Bu parametrenin alabileceği değerler ve oluşan hatalar şunlardır.

1: Bağlantıda kullanılan veri yanlış formatta

6: LinkMode özelliği 0 yapıldıktan sonra veri gönderilmeye çalışıldı.

7: Aynı anda en çok 128 kontrol bağlantı kurabilir. Bu değer aşıldığına 7 numaralı hata meydana gelir.

8: Kontrolün içeriği güncellenirken bir hata oluşuyor.

11: Bağlantı için yeterli bellek yok.

LinkNotify

Bağlantı modu olarak **LinkMode** özelliğine 3 (Notify) verilmişse bağlantı kurulan kontrolün içeriği değiştiğinde bu olay meydana gelir. Bu durumda güncelleştirme için gerekli kod bu olaya yazılmalıdır.

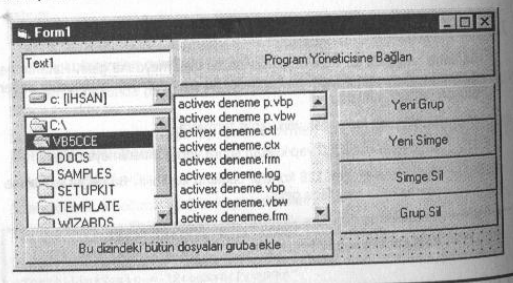
LinkOpen(cancel As Integer)

Bir bağlantı kurulduğunda bu olay meydana gelir. Eğer bağlantı kabul edilmeyecekse Cancel parametresine True atanarak bağlantı iptal edilebilir.

LinkExecute(cmdstr As String, cancel As Integer)

Hep siz mesaj gönderecek değilsiniz. Size de bir uygulama mesaj gönderebilir. Bir uygulama size mesaj gönderdiğinde (LinkExecute metodu ile mesaj gönderebilir) bu olay meydana gelir. Gönderilen mesajın içeriği cmdstr parametresi ile öğrenilir. Eğer mesaj kabul edilmeyecekse Cancel parametresine True değeri atanarak karşı tarafa bu bildirilir. Gelen mesajın ne olacağı konusunda hiç bir standart yoktur. Bu yüzdende iletişim kuran uygulamalar birbirlerinin kabul edeceği mesajları bilmesi gerekir.

Şimdi program yöneticisi (explorer) ile bağlantı kurup, yeni program grupları ve öğeleri oluşturup silecek bir program yazalım. Örneğimiz için aşağıdaki formu hazırlayalım.



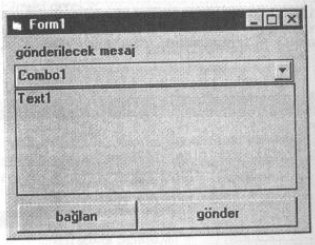

```
'ÖRNEK : Link
Private Sub Form_Load()
    File1.Pattern = "*.exe;*.bat;*.com;*.pif"
End Sub

Private Sub Command1_Click()
    Text1.LinkTopic = "Progman|Progman"
    Text1.LinkItem = ""
    Text1.LinkMode = 2
End Sub
```



```
Private Sub Command2_Click()
    Dim x
    On Local Error GoTo hata2
    x = InputBox("Program grubunun ismini gir", "BİZİM GRUP")
    Text1.LinkExecute "[CreateGroup(" & x & ")]"
Exit Sub
hata2:
    MsgBox (x & ":yapılamadı :" & Error)
    Exit Sub
End Sub

Private Sub Command3_Click()
    Dim x, y
    On Local Error GoTo hata3
    x = InputBox("Program öğesinin ismini gir", "Calc.exe")
    y = InputBox("Program öğesinin tanımını GİR", "Hesap
Makinesi")
    Text1.LinkExecute "[AddItem(" & x & "," & y & ")]"
hata3:
    MsgBox (x & ":yapılamadı :" & Error)
    Exit Sub
End Sub

Private Sub Command4_Click()
    On Local Error GoTo hata
    Dim x
    x = InputBox("Silinecek elemanın İSMİNİ GİR")
    Text1.LinkExecute "[deleteitem(" & x & ")]"
Exit Sub
hata:
    MsgBox (x & ":böyle bir şey yok :" & Error)
    Exit Sub
End Sub

Private Sub Command5_Click()
    On Local Error GoTo hata1
    Dim x
    x = InputBox("Silinecek grubun İSMİNİ GİR")
    Text1.LinkExecute "[deleteGroup(" & x & ")]"
Exit Sub
hata1:
    MsgBox (x & ":böyle bir şey yok :" & Error)
    Exit Sub
End Sub

Private Sub Command6_Click()
    Dim x, y
    On Local Error GoTo hata4
    Dim i
    For i = 0 To File1.ListCount - 1
        x = File1.Path & "\" & File1.List(i)
        y = File1.List(i)
        Text1.LinkExecute "[AddItem(" & x & "," & y & ")]"
```

```
    Next
    Exit Sub
hata4:
    MsgBox (x & ":yapılamadı :" & Error)
    Resume Next
End Sub

Private Sub Drive1_Change()
    ChDrive Drive1.Drive
    Dir1.Path = Drive1.Drive
End Sub
```

ÖRNEK: Şimdi hem mesaj alan hem de mesaj gönderen bir program yapalım. Örneğimiz için aşağıdaki formu hazırlayın.



Formun **LinkMode** özelliğine 1, **LinkTopic** özelliğine de **Form1** değerlerini verin. **Text1** kutusunun **Multiline** özelliğini de **True** yapın ve aşağıdaki kodu yazın.

```
'ÖRNEK : Link
Private Sub Form_Load ()
  caption = APP.EXEName
  command2.Default = True
  combo1.AddItem "MERHABA"
  combo1.AddItem "SELAMÜN ALEYKÜM"
  combo1.AddItem "NASILSINIZ"
  combo1.AddItem "BENDE İYİYİM"
  combo1.AddItem "ÇIK"
  combo1.AddItem "SİMGE DURUMUNA GEL"
  combo1.AddItem "EKRAN KAPLA"
  combo1.AddItem "BANA SEVDİĞİNİ SÖYLE"
  combo1.AddItem "APTAL"
  combo1.AddItem "ORADA HAVALAR NASIL"
```

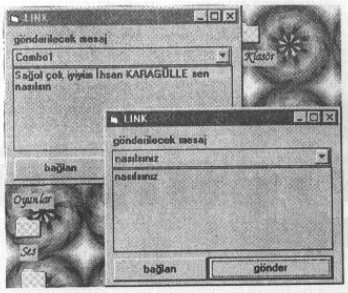


```
  combo1.AddItem "HANGİ TAKIMI TUTUYORSUN"
  combo1.AddItem "SEN KİMSİN"
End Sub

Private Sub Combo1_Click ()
    text1 = combo1.Text
End Sub

Private Sub Form_LinkOpen (cancel As Integer)
  cancel = False
  MsgBox ("İletişim kuruldu")
End Sub

Private Sub Text1_LinkClose ()
  MsgBox ("Bağlantımız kesildi")
End Sub

Private Sub Command1_Click ()
 Dim prg
 On Local Error GoTo hata
 prg = InputBox("Bağlantı kurulacak programın adı", "LİNK",
APP.EXEName & "|" & "FORM1")
 text1.LinkMode = 0
 text1.LinkTopic = prg
 text1.LinkItem = "Text1"
 text1.LinkMode = 2
Exit Sub
hata:
  MsgBox ("Bağlantı kurulamadı." & Error)
  Exit Sub
End Sub

Private Sub Command2_Click ()
  If text1.LinkMode = 0 Then
    MsgBox ("Hiç bir bağlantımız yok. Önce bağlantı kurun")
  Else
    text1.LinkExecute text1
  End If
End Sub

Private Sub Form_LinkExecute (cmdstr As String, cancel As
Integer)
 Static ilk, AD, MSG
 ReDim Takim(4)
 Takim(0) = "FENERBAHÇE"
 Takim(1) = "GALATASARAY"
 Takim(2) = "BEŞİKTAŞ"
 Takim(3) = "HİÇBİRİ"
  cancel = False
  Select Case UCase(cmdstr)
    Case "MERHABA": ilk = 1
                    MSG = "Merhaba. İsminiz nedir"
                    text1.Text = MSG
```



```
                    text1.LinkPoke
                    Exit Sub
    Case "SELAMÜN ALEYKÜM": ilk = 1
                    MSG = "Ve aleykümüs selam. İsminiz nedir"
                    text1.Text = MSG
                    text1.LinkPoke
                    Exit Sub
    Case Else
      If ilk = 0 Then
        text1 = "Selamsız konuşma yok"
        text1.LinkPoke
        Exit Sub
      End If
      If ilk = 1 Then
        ilk = 2
        AD = cmdstr
        text1 = "Merhaba " & AD
        text1.LinkPoke
        Exit Sub
    End If
  End Select
  Select Case UCase(cmdstr)
    Case "NASILSINIZ": MSG = "SAĞOL. ÇOK İYİYİM " & AD & " SEN
NASILSIN"
    Case "BENDE İYİYİM": MSG = "OH OH ALLAH İYİLİK VERSİN"
    Case "ÇIK": End
    Case "SİMGE DURUMUNA GEL": WINDOWSTATE = 1: MSG = "SİMGE
DURUMUNA GELDİM"
    Case "EKRANI KAPLA": WINDOWSTATE = 2: MSG = "EKRAN ŞU ANDA
BENİM"
    Case "BANA SEVDİĞİNİ SÖYLE": MSG = "SENİ ÇOKKK SEVİYORUM"
    Case "APTAL": Beep: MSG = "ANLAŞILMADI"
    Case "ORADA HAVALAR NASIL": MSG = "PARÇALI BULUTLU"
    Case "HANGİ TAKIMI TUTUYORSUN": MSG = "TABİKİ " &
Takim(Int(Rnd * 4))
    Case "SEN KİMSİN": MSG = "BEN BİR LINK ÖRNEĞİYİM. VISUAL
BASIC KİTABINA KONMAK İÇİN 12 OCAK 1997 TARİHİNDE İHSAN KARAGÜLLE
TARAFINDAN PROGRAMLANDIM "
    Case Else: MSG = "MESAJ ANLAŞILMADI"
  End Select
  text1.Text = MSG
  text1.LinkPoke
End Sub
```

Programı yazın ve Exe dosyaya çevirin. Exe'ye çevirdiğiniz dosyayı çalıştırın ve ekranın sol üst köşesine alın. Bu programı kapatmadan aynı programı tekrar çalıştırın ve bunuda ekranın sağ alt köşesine alın. Artık bu iki programımız arasında bağlantı kurabiliriz. Önce Bağlan düğmesi ile bağlantıyı kurun. Daha sonra text kutusuna mesajlar yazarak karşı tarafın ne cevaplar verdiğini izleyin.



Uygulamalara mesaj göndermek ve alıp onları değerlendirmek önemli bir olaydır. Bu örnek size bir fantazi gibi gelmiş olabilir. Benim yazdığım programa kim mesaj göndermek isteyebilir ki diye düşünmeyin. Siz kendi yazdığınız değişik programlar arasında bu şekilde bağlantılar kurarak birinin yapmadığı işi diğerine yaptırabilir veya programlardan biri çalışırken diğerinin işlevlerini kısıtlayabilirsiniz.

Bölüm 10

Ekran Koruyucu Yazma

EKRAN KORUYUCU YAZMA

Aslında ekran koruyucu (screen saver) programların normal programlardan pek farkı yoktur. Ekran koruyucuları normal bir program gibi yazılır ancak derlenirken uzantı yerine EXE değil de SCR verilir. Windows ekran koruyucuyu çalıştırırken bazı özel komut satırı parametreleri kullanır. Bu parametreleri programınızdan işlerseniz ekran koruyucu programı uygun şekilde çalıştırmış olursunuz.

Bir ekran koruyucu programda bulunması gereken işlemleri şu adımlarla özetleyebiliriz.

1. Ekran koruyucunun işlevini yerine getirecek kod. Örneğin ekrana rasgele çizimler yapacak kod. Bunu genellikle bir Timer kontrolünün Timer olayına yazabilirsiniz.
2. Ekran koruyucunun iki defa çalışmasını önleyecek kod.
3. Alt+Tab ve Ctrl+Alt+Del tuşlarını önleyecek kod.
4. Fare veya Klavyeden bir tuşa basıldığıdan ekran koruyucuyu sonlandırma.
5. /s, /p , /c, /a parametrelerini işleyerek Windowstan gelen mesajlara göre ekran koruyucuyu çalıştıracak kod.
6. Parola korumasının etkinleştirilmesi
7. Formun tam ekran haline getirilmesi ve başlığının kaldırılması.
8. Farenin gizlenmesi
9. Ekran koruyucu uygulamasının SCR uzantısıyla derlenmesi ve Windows'a tanıtılması.

Şimdi bu adımları sırasıyla anlatarak bir örnekte uygulayalım.

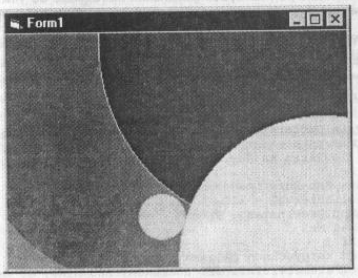
1-Ekran koruyucunun işlevini yerine getirecek kod

Bu kısım ekran koruyucunuzun ekranda yapacağı işlemleri içerir. Ekranda animasyonlar, müzik veya çizimler yaptırabilirsiniz.

Örnek olarak ekranda rasgele daireler çizecek bir ekran koruyucu yapalım. bunun için formumuza bir Timer yerleştirin ve Interval özelliğini 100 yaparak aşağıdaki kodu yazın.


```
Private Sub Timer1_Timer()
 FillColor = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
 FillStyle = 0
 Circle (Rnd * Width, Rnd * Height), Rnd * Width, RGB(Rnd * 255,
Rnd * 255, Rnd * 255)
End Sub
```

Bu kodumuz, form üzerine aşağıdaki gibi ragele daireler çizcektir.



2- Ekran koruyucunun iki defa çalışmasını önleyecek kod.

Ekran koruyucunun iki defa çalışmasını önlemek için VB'deki **App** nesnesinin **PrevInstance** özelliğini kullanabiliriz. Eğer uygulama zaten çalışıyorsa bu özellik true değerini alacaktır. Bu özelliği kontrol ederek, zaten çalışıyorsa tekrar çalışmamasını sağlayabiliriz.

```
Private Sub Form_Load()
 Timer1.Interval = 100
 If App.PrevInstance Then
  Unload Me
 End If
End Sub
```

3- Alt+Tab ve Ctrl+Alt+Del tuşlarını önleyecek kod.

Ekran koruyucu çalışırken Alt+Tab ve Ctrl+Alt+Del gibi Windows'a ait özel tuşların görevlerini yerine getirmemesi gerekir. Bunu yapabilmek için SystemParametersInfo api'sini kullanabiliriz. Ekran koruyucu çalıştığında bu Api'yi kullanarak ekran koruyucunun çalışmaya başladığını Windows'a bildirmemiz, ekran koruyucu çalışmasını bitirdiğinde de yine aynı Api ile uygulamanın sona erdiğini bildirmemiz gerekir.

Bu işlem için formun Load ve UnLoad olaylarında durumu Windows'a aşağıdaki gibi bildirebiliriz.

```
Option Explicit
 Private Const SPI_SCREENSAVERRUNNING = 97
 'Api tanımı
 Private Declare Function SystemParametersInfo Lib "user32" Alias
"SystemParametersInfoA" (ByVal uAction As Long, ByVal uParam As
Long, lpvParam As Any,ByVal fuWinIni As Long) As Long

Private Sub Form_Load()
 Timer1.Interval = 100
  If App.PrevInstance Then
  Unload Me
 End If
 'Ekran koruyucunun çalışmaya başladığını bildir
 SystemParametersInfo SPI_SCREENSAVERRUNNING, 1, ByVal 1&, False
End Sub

Private Sub Form_Unload(Cancel As Integer)
 'Ekran koruyucunun bittiğini bildir
 SystemParametersInfo SPI_SCREENSAVERRUNNING, 0, ByVal 1&, False
End Sub
```

4-Fare veya Klavye hareketlerini işleme

Ekran koruyucu çalışırken bir tuşa basıldığında, fare hareket ettirildiğinde veya fare tıklandığında ekran koruyucunun sona ermesi gerekir. Formun KeyDown, Click, DblClick ve MouseMove olaylarına aşağıdaki gibi çıkış için gerekli kodu yazmak gerekir.

```
Private Sub Form_Click()
 Unload Me
End Sub

Private Sub Form_DblClick()
 Unload Me
End Sub
```



```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
Unload Me
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X
As Single, Y As Single)
 Unload Me
End Sub
```

Bu kodun normal çalışması beklenir ancak MouseMove olayında problem çıkacaktır. Çünkü form çalıştığı anda formun MouseMove olayı meydana gelir ve program sona erer. Ayrıca farenin en ufak hareketinde sona ermemesi için belli bir aralık konabilir. Mouse belli bir miktar hareket ettikten sonra programın sona ermesi istenir. Bu işlem için MouseMove olayı aşağıdaki gibi değiştirilmelidir.

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, x
As Single, y As Single)
 Static ox, oy
 'İlk defa çalışıyorsa veya önceki koordiatlardan 5 birimden daha
az hareket etmişse etkilenme
 If ((ox = 0) And (oy = 0)) Or ((Abs(ox - x) < 5) And (Abs(oy -
y) < 5)) Then
  'koordinatları sakla
  ox = x'
  oy = y
  Exit Sub
 Else
  Unload Me
 End If
End Sub
```

5- /s, /p , /c, /a parametrelerini işleyecek kod

Windows bir ekran koruyucuya çalıştırırken bazı parametreler gönderebilir. Bunlar komut satırı parametreleridir ve ekran koruyucuya yapması gereken işi bildirir. Komut satırı parametrelerini VB'de Command fonksiyonuyla öğrenerek gelen parametreleri yorumlayabiliriz.

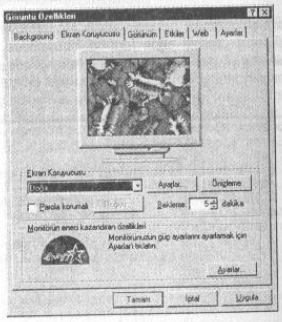
Windows'un göndereceği parametreler şunlardır:

/s : Ekran koruyucuyu normal çalıştır.

/c : Kullanıcı ekran koruyucu ayarlarını yapmak istemiştir. Varsa ayar formunuzu gösterin.

/a handle: Şifre değiştirme penceresini çalıştır. Pencerenin handle numarası parametreyle birlikte gelir.

/p handle: Preview-Önizleme modunu çalıştır. Denetim Masasındaki Görüntü simgesini çift tıkladığınızda açılan pencerenin **Ekran Koruyucu** kısmına geçip bir ekran koruyucu seçerseniz penceredeki küçük ekranda ekran koruyucunun çalıştığını görürsünüz. İşte bu parametre ile o pencerenin handle numarası gelir. Bu handle numarasını kullanarak çiziminizi o pencere içine yaptırabilirsiniz.



/s parametresi

Command fonksiyonu ile komut satırında /s parametresi kullanılıp kullanılmadığını kontrol edebilirsiniz. Eğer /s parametresi kullanılmışsa ekran koruyucunun normal olarak çalışması gerekir.

Örneğimizde Timer1 kontrolüne yazdığımız kod ekran koruyucunun işlevini yerine getiriyordu. Fomun Load olayında /s parametresini kontrol ederek Timer'i aktif hale getirebiliriz.

```
Private Sub Form_Load()
 Timer1.Interval = 100
 Timer1.Enabled = False
 If App.PrevInstance Then
  Unload Me
 End If
```



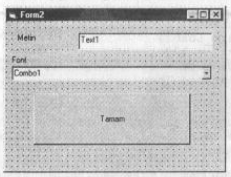
```
 'Ekran koruyucunun çalışmaya başladığını bildir
 SystemParametersInfo SPI_SCREENSAVERRUNNING, 1, ByVal 1&, False

 If Command = "/s" Then Timer1.Enabled = True
End Sub
```

/c parametresi

Command fonksiyonu ile komut satırında /c parametresi kullanılıp kullanılmadığını kontrol edebilirsiniz. Eğer /c parametresi kullanılmışsa ekran koruyucunun ayarlarının yapılabileceği yeni bir formu gösterebilirsiniz.

Örneğimizde ayar olarak kullanıcının Text kutusuna bir şeyler yazabilmesini ve yazdığı metnin çizimle birlikte yazılmasını sağlayalım. Ayrıca kullanıcı fontu da belirleyebilsin.

Bunun için programınıza **Project-Add Form** menüleri ile yeni bir form ekleyip aşağıdaki gibi hazırlayın.



Formdaki Combo1 içinde font isimlerini gösterebilmek ve Tamam düğmesi ile formu kapatıp ayarları Registry içine saklamak için de aşağıdaki kodları Form2'nin kod penceresine yazalım.

```
'Bu kod Form2'ye yazılacak
Private Sub Command1_Click()
 SaveSetting "BizimEkranKoruyucu", "Ayarlar", "Metin", Text1
 SaveSetting "BizimEkranKoruyucu", "Ayarlar", "Font", Combo1.Text
 Unload Me
End Sub

Private Sub Form_Load()
 Dim i
 For i = 0 To Screen.FontCount - 1
  Combo1.AddItem Screen.Fonts(i)
 Next
End Sub
```

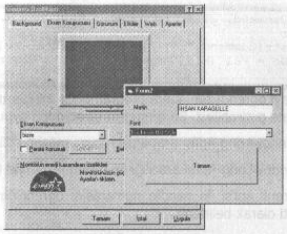
Şimdi Form1 içinde gerekli düzenlemeleri yaparak /c paramatresi kullanılmışsa ayar kutusunun gösterilmesini sağlayalım.

```
Private Sub Form_Load()
 Timer1.Interval = 100
 Timer1.Enabled = False
 If App.PrevInstance Then
  Unload Me
 End If
 'Ekran koruyucunun çalışmaya başladığını bildir
 SystemParametersInfo SPI_SCREENSAVERRUNNING, 1, ByVal 1&, False
 Caption = Command
 If Command = "/s" Then Timer1.Enabled = True
 If Left(Command, 2) = "/c" Then Form2.Show: Unload Me
End Sub
```

Ayrıca yaptığımız ayarlara göre kullanıcının Text kutusuna yazdığı yazıyı çizimle birlikte ekrana yazabilmek için Registry'den ilgili ayarları okumamız gerekir. Bunun için Form1'deki kodu aşağıdaki gibi değiştirelim.

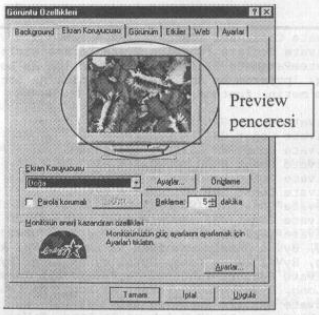
```
Dim metin, fontadi
Private Sub Form_Load()
 Timer1.Interval = 100
 Timer1.Enabled = False
 If App.PrevInstance Then
  Unload Me
 End If
 'Ekran koruyucunun çalışmaya başladığını bildir
 SystemParametersInfo SPI_SCREENSAVERRUNNING, 1, ByVal 1&, False
 Caption = Command
 If Command = "/s" Then Timer1.Enabled = True
 If Left(Command, 2) = "/c" Then Form2.Show: Unload Me

 metin = GetSetting("BizimEkranKoruyucu", "Ayarlar", "Metin",
"İhsan Karagülle")
 fontadi = GetSetting("BizimEkranKoruyucu", "Ayarlar", "Font",
"Times New Roman")
End Sub

Private Sub Timer1_Timer()
 FillColor = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
 FillStyle = 0
 Circle (Rnd * Width, Rnd * Height), Rnd * Width, RGB(Rnd * 255,
Rnd * 255, Rnd * 255)
 FontName = fontadi
 FontSize = Rnd * 50 + 8
 Print metin
End Sub
```

Böylece kullanıcı ayar penceresi ile ekran koruyucu programımıza ait bazı ayarları yapabilecektir.



/p parametresi

Eğer ekran koruyucu /p parametresi ile çalıştırılmışsa aşağıdaki penceredeki küçük bölgede ekran koruyucuyu çalıştırmamız gerekir.



Bu bölgenin handle numarası /p parametresi ile birlikte gelir. Formun Load olayına yazacağımız kodla komut satırı parametresinin ilk iki harfinin /polup olmadığını kontrol edebilir ve /p paramtersi varsa boluktan sonraki sayıyı öğrenebiliriz. Bu sayı bize yukarıdaki küçük pencerenin handle numarasını bildirecektir.


```
If Left(Command, 2) = "/p" Then
 dim yer
 yer=Instr(Command," ") 'Boşluğun yerini bul
 handleNo = Val (Mid(Command, yer+1)) 'Boşluktan sonraki değer
End If
```

Bu numarayı öğrendikten sonra programımıza ait formu bu küçük alanın bir child formu haline getirebiliriz. Bu işlem için şu adımları uygulamamız gerekir.

1. GetWindowLong Api'si aracılığıyla formumuzun stili alınır.
2. SetWindowLong Api'si aracılığı ile formumuz Child form haline getirilir.
3. SetParent Api'si ile handle numarasını öğrendiğimiz pencere formumuzun parenti olarak belirlenir.
4. SetWindowLong Api'si ile bu değişiklik aktif hale getirilir.
5. GetClientRect Api'si ile preview penceresinin boyutları öğrenilir.
6. SetWindowPos Api'si ile formumuzun boyutları preview penceresinin boyutlarını küçültülür.

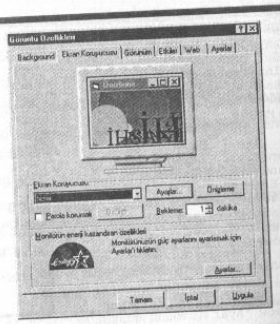
Bu işlemde kullanılacak Api ve diğer sabit tanımlarını formumuzun en başına yazın.

```
Private Const SPI_SCREENSAVERRUNNING = 97
 Private Declare Function SystemParametersInfo Lib "user32" Alias
"SystemParametersInfoA" (ByVal uAction As Long, ByVal uParam As
Long, lpvParam As Any, ByVal fuWinIni As Long) As Long
 Dim metin, fontadi, handleno

 Private Const GWL_STYLE = -16
 Private Const GWL_HWNDPARENT = -8
 Private Const WS_CHILD = &H40000000
 Private Const HWND_TOPMOST = -1&
 Private Const HWND_TOP = 0&
 Private Const SWP_NOZORDER = &H4
 Private Const SWP_NOACTIVATE = &H10
 Private Const SWP_SHOWWINDOW = &H40
 Private Declare Sub SetWindowPos Lib "user32" (ByVal hwnd As
Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As
Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long)
 Private Declare Function SetParent Lib "user32" (ByVal hWndChild
As Long, ByVal hWndNewParent As Long) As Long
 Private Declare Function SetWindowLong Lib "user32" Alias
"SetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal
dwNewLong As Long) As Long
 Private Declare Function GetWindowLong Lib "user32" Alias
"GetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long) As
Long
 Private Declare Function GetClientRect Lib "user32" (ByVal hwnd
As Long, lpRect As RECT) As Long
```



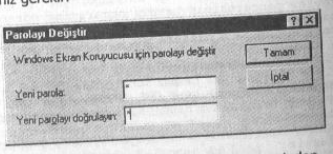
```
 Private Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
 End Type
 Dim DispRec As RECT

Private Sub Form_Load()
 Timer1.Interval = 100
 Timer1.Enabled = False
 If App.PrevInstance Then
  Unload Me
 End If

 '/s parametresi. Ekran koruyucuyu çalıştır
 If Command = "/s" Then
  Timer1.Enabled = True
  'Ekran koruyucunun çalışmaya başladığını bildir
  SystemParametersInfo SPI_SCREENSAVERRUNNING, 1, ByVal 1&, False
  'formu en üstte tut
  SetWindowPos Form1.hwnd, -1,0,0,0,0, &H10 Or &H40 Or &H1 Or &H2
 End If

'/c parametresi. Ayar penceresini göster
 If Left(Command, 2) = "/c" Then
  Form2.Show 'Ayar formunu göster
  Unload Me
 End If

'/p parametresi. Preview yap.
 If Left(Command, 2) = "/p" Then
  Dim yer
  yer = InStr(Command, " ") 'Boşluğun yerini bul
  'Preview penceresinin handle numarasını öğren
  handleno = Val(Mid(Command, yer + 1)) 'Boşluktan sonraki değer
  Form1.Caption = "Önizleme"
  Dim style
  style=GetWindowLong(Form1.hwnd, GWL_STYLE) ' Form stilini öğren
  style = style Or WS_CHILD 'Child özelliğini ver
  SetWindowLong Form1.hwnd,GWL_STYLE,style'Yeni özelliği aktifyap
  SetParent Form1.hwnd,handleno'preview penceresini formumuza
parent yap
  SetWindowLong Form1.hwnd,GWL_HWNDPARENT,handleno'Bu değişikliği
etkinleştir
  GetClientRect handleno,DispRec'Preview penceresinin boyutlarını
al
  'Formumuzu preview penceresine yerleştir.
  SetWindowPos Form1.hwnd, HWND_TOP, 0&, 0&, DispRec.Right,
DispRec.Bottom, SWP_NOZORDER Or SWP_NOACTIVATE Or SWP_SHOWWINDOW
  Timer1.Enabled = True
 End If
```

```
metin = GetSetting("BizimEkranKoruyucu", "Ayarlar", "Metin",
"İhsan Karagülle")
fontadi = GetSetting("BizimEkranKoruyucu", "Ayarlar", "Font",
"Times New Roman")
End Sub
```



/a parametresi

Eğer ekran koruyucu /a parametresi ile çalıştırılmışsa şifre değiştirme penceresini görüntülemeniz gerekir.



Bu pencerenin handle numarası /a parametresinden sonra gelir. **PwdChangePassword** api'si aracılığı ile gelen handle numarasını kullanıp şifre değiştirme penceresini göstermemiz gerekir.

Api'nin tanımı aşağıdaki gibidir. Bu tanım ofrmun General-Declaration kısmına yazılır.


```
Private Declare Function PwdChangePassword& Lib "mpr" Alias
"PwdChangePasswordA" (ByVal lpcRegkeyname$, ByVal hWnd&, ByVal
uiReserved1&, ByVal uiReserved2&)
```

Formun Load olayına yazılacak kodla /a parametresi kullanılıp kullanılmadığı kontrol edilir. Eğer bu parametre kullanılmışsa boşluktan sonra da şifre penceresinin handle numarası gelecektir. Bu bilgiyi API içinde kullanarak şifre değiştirme penceresini aktif hale getirebiliriz.

```
'şifre penceresi
If Left(Command, 2) = "/a" Then
  yer = InStr(Command, " ") 'Boşluğun yerini bul
  'password penceresinin handle numarasını öğren
  handleno = Val(Mid(Command, yer + 1)) 'Boşluktan sonraki değer
  Dim x As Long
  x = PwdChangePassword("SCRSAVE", handleno, 0&, 0&)
End If
```

6-Formun tam ekran haline getirilmesi

Eğer parola koruması aktif hale gelmişse çıkmadan önce şifreyi sormak ve şifre bilinmiyorsa ekran koruyucudan çıkılmasını engellemek gerekir. Şifre sorma işlemini **VerifyScreenSavePwd** api'si yapar. Bu apiden geriye false değeri dönmüşse şifre doğru girilmemiş demektir. Formun Query Unload olayına yazılacak kodla bu API çağrılır. Eğer kullanıcı şifreyi bilememişse çıkış iptal edilir.

Bu api'nin tanımı şöyledir:

```
Private Declare Function VerifyScreenSavePwd Lib "password.cpl"
(ByVal hWnd As Long) As Boolean
```

QueryUnload olayına yazılacak kod ise şöyledir:

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As
Integer)
 'şifre bilinemediyse
 If VerifyScreenSavePwd(Me.hWnd) = False Then
  Do:
   DoEvents
  Loop Until ShowCursor(False) < 0 'kursorü gizle
  Cancel = True     'Çıkışı iptal er
 End If
End Sub
```

7-Formun tam ekran haline getirilmesi

Ekran koruyucunun tam ekran olarak ve başlıksız çalışması gerekir. Bunun için formun BorderStyle özelliğine 0, WindowState özelliğine tam ekran vermemiz ve

Capiton özelliğini de kaldırmamız gerekir. Bütün bu işlemleri formun Load olayında yapabiliriz.

```
BorderStyle = 0 'Çerçevesiz form
WindowState = 2 'tam ekran
Caption = "" 'Başlıksız
```

8-Farenin Gizlenmesi

Ekran koruyucu çalışırken fare kursorünün gözükmesi istenmez. Bunu gizlemek ve göstermek için **ShowCursor** API'sinden faydalanılır.

Bu api'nin tanımı aşağıdaki gibidir:

```
Private Declare Function ShowCursor Lib "user32" (ByVal fShow As
Integer) As Integer
```

Bu tanım formun General-Declaration kısmına yazıldıktan sonra, fareyi gizlemek için:

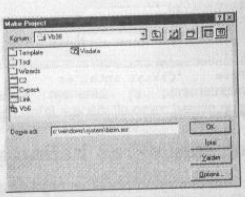
```
call showcursor(0)
```

Fareyi göstermek için ise aşağıdaki kod kullanılır.

```
call showcursor(-1)
```

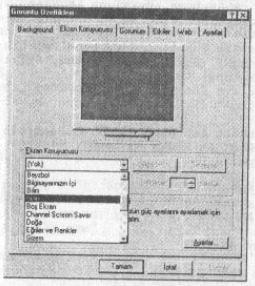
9-Ekran Koruyucuyu Derleme

Ekran koruyucu programı hazırladıktan sonra onu derleyerek SCR uzantılı hale getirmemiz ve Windows\System dizinine kopyalamamız gerekir. SCR uzantısının EXE uzantısından hiç bir farkı yoktur. Bu yüzden **File** menüsündeki **Make Project1.EXE** komutunu kullanacağız. Bu komutu seçtiğinizde aşağıdaki pencere ile programın derlenecek adı sorulacaktır.



Buraya c:\windows\system\bizim.scr yazarak uygulamamızın bizim.scr ismiyle Windows\System dizinine derlenmesini sağlayabiliriz.

Şimdi ekran koruyucuyu Windows'ta izleyelim **Denetim** masasından **Görüntü** kısmına girin ve açılan aşağıdaki pencerenin **Ekran Koruyucu** kısmına geçin.



Penceredeki ComboBox'u aşağı doğru açarsanız bizim ekran koruyucuyu listede görebilirsiniz.

Yaptığımız ekran koruyucunun tam kodu aşağıdaki gibidir:

```
'ÖRNEK: Ekran Koruyucu
Option Explicit
 Private Declare Function ShowCursor Lib "user32" (ByVal fShow As
Integer) As Integer
 Private Declare Function PwdChangePassword& Lib "mpr" Alias
"PwdChangePasswordA" (ByVal lpcRegkeyname$, ByVal hWnd&, ByVal
uiReserved1&, ByVal uiReserved2&)
 Private Declare Function VerifyScreenSavePwd Lib "password.cpl"
(ByVal hWnd As Long) As Boolean
 Private Const SPI_SCREENSAVERRUNNING = 97
 Private Declare Function SystemParametersInfo Lib "user32" Alias
"SystemParametersInfoA" (ByVal uAction As Long, ByVal uParam As
Long, lpvParam As Any, ByVal fuWinIni As Long) As Long
 Dim metin, fontadi, handleno

 Private Const GWL_STYLE = -16
 Private Const GWL_HWNDPARENT = -8
 Private Const WS_CHILD = &H40000000
 Private Const HWND_TOPMOST = -1&
```

```
Private Const HWND_TOP = 0&
Private Const SWP_NOZORDER = &H4
Private Const SWP_NOACTIVATE = &H10
Private Const SWP_SHOWWINDOW = &H40
Private Declare Sub SetWindowPos Lib "user32" (ByVal hwnd As
Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As
Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long)
Private Declare Function SetParent Lib "user32" (ByVal hWndChild
As Long, ByVal hWndNewParent As Long) As Long
Private Declare Function SetWindowLong Lib "user32" Alias
"SetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal
dwNewLong As Long) As Long
Private Declare Function GetWindowLong Lib "user32" Alias
"GetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long) As
Long
Private Declare Function GetClientRect Lib "user32" (ByVal hwnd
As Long, lpRect As RECT) As Long
Private Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type
Dim DispRec As RECT

Private Sub Form_Click()
 Unload Me
End Sub

Private Sub Form_DblClick()
 Unload Me
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
Unload Me
End Sub

Private Sub Form_Load()
 call showcursor(0)'fareyi gizle
 Timer1.Interval = 100
 Timer1.Enabled = False
 Timer2.Interval = 100
 Timer2.Enabled = False
 BorderStyle = 0 'Çerçevesiz form
 WindowState = 2 'tam ekran
 Caption = "" 'Başlıksız
 If App.PrevInstance Then
  Unload Me
  Exit Sub
 End If
```



```
'/s parametresi. Ekran koruyucuyu çalıştır
If Command = "/s" Then
 Timer1.Enabled = True
 'Ekran koruyucunun çalışmaya başladığını bildir
 SystemParametersInfo SPI_SCREENSAVERRUNNING, 1, ByVal 1&, False
 'formu en üstte tut
 SetWindowPos Form1.hwnd, -1, 0, 0, 0, 0, &H10 Or &H40 Or &H1 Or
&H2
End If

'/c parametresi. Ayar penceresini göster
If Left(Command, 2) = "/c" Then
 Form2.Show 'Ayar formunu göster
 Unload Me
End If

'p parametresi. Preview yap.
If Left(Command, 2) = "/p" Then
 Dim yer
 yer = InStr(Command, " ") 'Boşluğun yerini bul
 'Preview penceresinin handle numarasını öğren
 handleno = Val(Mid(Command, yer + 1)) 'Boşluktan sonraki değer
 Form1.Caption = "Önizleme"
 Dim style
 style = GetWindowLong(Form1.hwnd, GWL_STYLE) ' Form stilini
öğren
 style = style Or WS_CHILD 'Child özelliğini ver
 SetWindowLong Form1.hwnd, GWL_STYLE, style 'Yeni özelliği
aktifyap
 SetParent Form1.hwnd, handleno 'preview penceresini formumuza
parent yap
 SetWindowLong Form1.hwnd, GWL_HWNDPARENT, handleno 'Bu
değişikliği etkinleştir
 GetClientRect handleno, DispRec 'Preview penceresinin
boyutlarını al
 'Formumuzu preview penceresine yerleştir.
 SetWindowPos Form1.hwnd, HWND_TOP, 0&, 0&, DispRec.Right,
DispRec.Bottom, SWP_NOZORDER Or SWP_NOACTIVATE Or SWP_SHOWWINDOW
 Timer1.Enabled = True
End If

'şifre penceresi
If Left(Command, 2) = "/a" Then
 yer = InStr(Command, " ") 'Boşluğun yerini bul
 'password penceresinin handle numarasını öğren
 handleno = Val(Mid(Command, yer + 1)) 'Boşluktan sonraki değer
 Dim x As Long
 x = PwdChangePassword("SCRSAVE", handleno, 0&, 0&)
End If

Metin = GetSetting("BizimEkranKoruyucu", "Ayarlar", "Metin",
"İhsan Karagülle")
```



```
fontadi = GetSetting("BizimEkranKoruyucu", "Ayarlar", "Font",
"Times New Roman")
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As
Integer)
 'şifre bilinemediyse
 If VerifyScreenSavePwd(Me.hWnd) = False Then
  Do:
   DoEvents
  Loop Until ShowCursor(False) < 0 'kursörü gizle
  Cancel = True    'Çıkışı iptal et
 End If
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, x
As Single, y As Single)
 Static ox, oy
 If ((ox = 0) And (oy = 0)) Or ((Abs(ox - x) < 5) And (Abs(oy -
y) < 5)) Then
  ox = x
  oy = y
  Exit Sub
 Else
  Unload Me
 End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
 'Ekran koruyucunun bittiğini bildir
 SystemParametersInfo SPI_SCREENSAVERRUNNING, 0, ByVal 1&, False
 call showcursor(-1)'Fareyi göster
End Sub

Private Sub Timer1_Timer()
 FillColor = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
 FillStyle = 0
 Circle (Rnd * Width, Rnd * Height), Rnd * Width, RGB(Rnd * 255,
Rnd * 255, Rnd * 255)
 FontName = fontadi
 FontSize = Rnd * 50 + 8
 Print metin
End Sub

Private Sub Timer2_Timer()
 Unload Me
End Sub
```

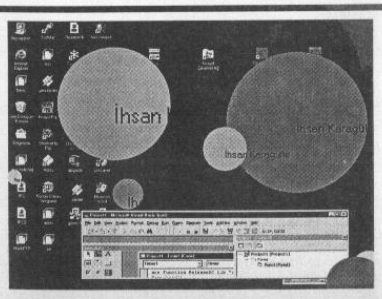
Ekran Görüntüsü Üzerinde Çalışma

Bazı ekran koruyucuların kendi formu üzerinde değil de ekrandaki görüntü üzerinde işlem yaptıklarını görürsünüz. Aslında bu o kadar da zor bir işlem değil. Ekran koruyucu başlamadan önce ekrandaki görüntüyü alıp formunuza yerleştirirseniz ekran koruyucunuz sanki o ekran üzerinde çalışıyormuş gibi olur.

Ekran görüntüsünü alabilmek için önce **GetDesktopWindow** apisi ile masaüstünün handle numarasını almanız ve bu numaradan faydalanarak **GetDC** apisi ile bir hdc numarası almanız gerekir. Bu hdc numarasını aldıktan sonra **BitBlt** apisi yardımıyla ekran görüntüsünü formunuzun üzerine alabilir ve işiniz bittikten sonra da **ReleaseDc** apisi ile hdc numarasını serbest bırakabilirsiniz.

Yaptığınız ekran koruyucunun, ekran görüntüsü üzerinde çalışmasını isterseniz aşağıdaki kodu programınıza eklemeniz yeterlidir.

```
Option Explicit
Private Declare Function GetDesktopWindow Lib "user32" () As Long
Private Declare Function BitBlt Lib "GDI32" (ByVal hDestDC As
Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long,
ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal XSrc As Long,
ByVal YSrc As Long, ByVal dwRop As Long) As Long
Private Declare Function GetDC Lib "user32" (ByVal hWnd As Long)
As Long
Private Declare Function ReleaseDC Lib "user32" (ByVal hWnd As
Long, ByVal hDC As Long) As Long
Private Sub Form_Load()
  Dim hd, w, h
  AutoRedraw = True
  w = Screen.Width / Screen.TwipsPerPixelX
  h = Screen.Height / Screen.TwipsPerPixelY
  hd = GetDC(GetDesktopWindow())
  Call BitBlt(Form1.hDC, 0, 0, w, h, hd, 0, 0, &HCC0020) 'srccopy
  Call ReleaseDC(GetDesktopWindow(), hd)
End Sub
```



Bölüm 11

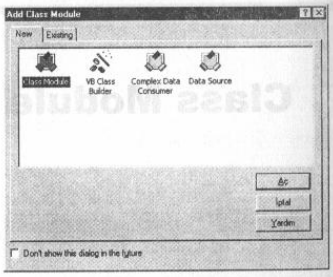
Class Module

Class Module

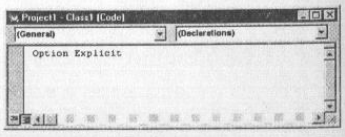
C++ gibi nesneye dayalı diller yeni sınıflar tanımlamanıza izin verir. Sınıflar (Class) kullanıcı tanımlı tiplere benzer ancak onlar gibi sadece değişkenlerden ibaret değildir. Class'lar da değişkenlerle birlikte fonksiyonlar da bulunabilmektedir. VB 6 içerisinde de kendi sınıflarınızı tanımlayabilirsiniz. Bu bölümde bir örneği adım adım geliştirerek VB ile bir Class tanımının nasıl yapılacağını ve nasıl kullanılacağını göreceğiz.

Class Module dosyaları **Project** menüsündeki **Add Class Module** seçeneği ile projeye eklenir ve bunlar CLS uzantılı dosyalara kaydedilirler.

Project menüsünden **Add Class Module** seçeneği ile projenize yeni bir Class Module ekleyin. Karşınıza aşağıdaki gibi bir pencere çıkacaktır.



Penceredeki Class Module seçeneğini çift tıklayın. Aşağıdaki gibi boş bir kod penceresi gelecektir.

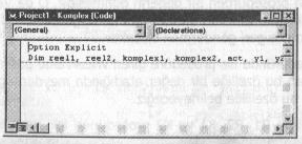


F4 tuşuna basarsanız buna ait özellikleri görebiliriz.

Örnek olarak komplex sayılar üzerinde işlem yapacak bir sınıf oluşturalım. Bunun içinde modülümüzün ismini Komplex olarak değiştirelim.

Komplex sayılarda sayının reel ve imajiner kısımları vardır. Bunun için R1, R2 ve K1, K2 isimli dört tane properties tanımlayalım. Bunlarla üzerinde işlem yapılacak iki ayrı komplex sayının reel ve komplex kısmı belirlensin ve yapılacak işlemin sonucu da bu propertieslerle öğrenilsin. Yazım1 ve Yazım2 isimli iki ayrı propertiesimiz daha olsun. Kullanıcı isterse de bunlarla tek seferde komplex sayıyı r + ki formatında girebilsin ve sonucu öğrenebilsin. Ayrıca birde Action özelliği olsun, bununla da yapılacak işlem belirlensin.

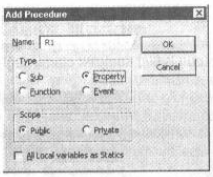
Önce properties değerlerini tutmak için gerekli değişkenlerimizi Class Modülün General Declarations kısmında tanımlayalım.



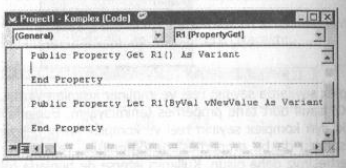
Propertiesleri Belirleme

Şimdi R1 isimli propertiesi tanımlayalım. Kullanıcı bu özellikle birinci komplex sayının reel kısmını belirleyecek.

Tools menüsündeki Add Procedure seçeneği ile açılacak aşağıdaki pencereden isim olarak R1 ve tip olarak ta Propertiesi seçelim.



Şimdi iki tane prosedür oluştu. Bunlardan biri Get diğeri de Let prosedürleri.



Buradaki Get prosedürü kullanıcı bu propertiesin değerini öğrenmeye çalıştığında, Let prosedürü ise bu propertiese bir değer atadığında meydana gelecek.

Yukarıda görüldüğü gibi Get prosedüründe herhangi bir giriş parametresi yok. Çünkü kullanıcı bir özelliğin değerini öğrenmek istemektedir. Dolayısıyla söz konusu olan prosedürden bir değerin dönmesidir. O da Variant olarak tanımlanmış bir değişkendir. O halde buraya yazmamız gereken kod özelliğin değerini fonksiyonun ismiyle geri göndermek olacaktır.

Set prosedüründe ise prosedüre giren vNewValue parametresi var. Set prosedürü kullanıcı bu özelliğe bir değer atadığında meydana geleceği için hangi değeri atadığını bu özellikle belirleyeceğiz.

Örneğimizde R1, R2, K1 ve K2 prosedürleri için yazmamız gereken kod sadece atanan değeri tanımladığımız değişkenlerde tutmak ve bunları istendiğinde vermek. Ayrıca Yazım1 ve Yazım2 isimli propertieslerimiz R1, K1 veya R2, K2 nin değişiminden etkileneceği için onları da Let prosedüründe güncellememiz gerekecektir.

```
Public Property Get R1() As Variant
 R1 = reel1
End Property

Public Property Let R1(ByVal vNewValue As Variant)
```



```
 reel1 = vNewValue
 y1 = reel1 & IIf(komplex1 > 0, "+", "-") & Abs(komplex1) & "i"
End Property
```

R2, K1 ve K2 özelliklerini de aynı şekilde oluşturarak aynı şeyleri onun içinde yazalım.

```
Public Property Get R1() As Variant
 R1 = reel1
End Property

Public Property Let R1(ByVal vNewValue As Variant)
 reel1 = vNewValue
 y1 = reel1 & IIf(komplex1 > 0, "+", "-") & Abs(komplex1) & "i"
End Property

Public Property Get r2() As Variant
 r2 = reel2
End Property

Public Property Let r2(ByVal vNewValue As Variant)
 reel2 = vNewValue
 y2 = reel2 & IIf(komplex2 > 0, "+", "-") & Abs(komplex2) & "i"
End Property

Public Property Get k1() As Variant
 k1 = komplex1
End Property

Public Property Let k1(ByVal vNewValue As Variant)
 komplex1 = vNewValue
 y1 = reel1 & IIf(komplex1 > 0, "+", "-") & Abs(komplex1) & "i"
End Property

Public Property Get k2() As Variant
 k2 = komplex2
End Property

Public Property Let k2(ByVal vNewValue As Variant)
 komplex2 = vNewValue
 y2 = reel2 & IIf(komplex2 > 0, "+", "-") & Abs(komplex2) & "i"
End Property
```

Kullanıcı isterse Yazım1 ve Yazım2 isimli özelliklerle de komplex sayıyı string formatında girebileceği için bu propertieslerin Get kısmında komplex sayıyı string formatında verecek ve Let kısmında da reel ve komplex değişkenlerini güncelleyecek kodu yazmamız gerekir.

Yazım1 ve Yazım2 isimli properties prosedürlerinide diğerleri gibi oluşturduktan sonra aşağıdaki kodu yazalım.

```
Public Property Get Yazım1() As Variant
```



```
 Yazım1 = y1
End Property

Public Property Let Yazım1(ByVal vNewValue As Variant)
 reel1 = Val(vNewValue)
 Dim yer 'imajiner kısmı bulmak için kullanacağız
 'sayı a+bi veya a-bi şeklinde olacağı için aradaki işareti
arıyoruz
 yer = InStr(vNewValue, "+")
 If yer = 0 Then yer = InStr(vNewValue, "-")'+ yoksa - ara
 If yer = 1 Then 'reel kısmın işareti var ise tekrar bul. Yer 1
ise -a + bi şeklindedir
  yer = InStr(2, vNewValue, "+")
  If yer = 0 Then yer = InStr(2, vNewValue, "-")
 End If
 If yer > 1 And Right(vNewValue, 1) = "i" Then 'imajiner kısım
var ise
  komplex1 = Val(Mid(vNewValue, yer))
 Else
  komplex1 = 0
 End If
 y1 = reel1 & IIf(komplex1 > 0, "+", "-") & Abs(komplex1) & "i"
End Property
Public Property Get Yazım2() As Variant
 Yazım2 = y2
End Property
Public Property Let Yazım2(ByVal vNewValue As Variant)
 reel2 = Val(vNewValue)
 Dim yer 'imajiner kısmı bulmak için kullanacağız
 'sayı a+bi veya a-bi şeklinde olacağı için aradaki işareti
arıyoruz
 yer = InStr(vNewValue, "+")
 If yer = 0 Then yer = InStr(vNewValue, "-")
 If yer = 1 Then 'reel kısmın işareti var ise tekrar bul
  yer = InStr(2, vNewValue, "+")
  If yer = 0 Then yer = InStr(2, vNewValue, "-")
 End If
 If yer > 1 And Right(vNewValue, 1) = "i" Then 'imajiner kısım
var ise
  komplex2 = Val(Mid(vNewValue, yer))
 Else
  komplex2 = 0
 End If
 y1 = reel2 & IIf(komplex2 > 0, "+", "-") & Abs(komplex2) & "i"
End Property
```

Son olarak ta Action özelliğini tanımlayalım. Bu özelliği kullanıcı işlem yapmak için kullanacak. Kullanıcı bu özelliğe 1 değerini verirse iki koplex sayıyı toplayalım, 2 verirse çıkaralım, 3 verirse çarpalım, 4 verirse de bölelim.

```
Public Property Let Action(ByVal vNewValue As Variant)
 Dim rel, kol
 Select Case Val(vNewValue)
```



```
  Case 1: rel = reel1 + reel2
   kol = komplex1 + komplex2
  Case 2: rel = reel1 - reel2
   kol = komplex1 - komplex2
  Case 3: rel = reel1 * reel2 - komplex1 * komplex2
   kol = reel1 * komplex2 + reel2 * komplex1
  Case 4: rel = (reel1 * reel2 + komplex1 * komplex2) / (reel2 ^
2 + komplex2 ^ 2)
   kol = (reel2 * komplex1 - reel1 * komplex2) / (reel2 ^ 2 +
komplex2 ^ 2)
  Case Else
   MsgBox ("Tanımsız işlem. Action özelliğine 1-4 arası bir değer
verin")
   Exit Property
 End Select
 reel1 = rel
 komplex1 = kol
 y1 = reel1 & IIf(komplex1 > 0, "+", "-") & Abs(komplex1) & "i"
End Property
```

Sonuç olarak Class Modülümüzdeki kodun aşağıdaki gibi olması gerekir.

```
'ÖRNEK: ClassModule
Option Explicit
Dim reel1, reel2, komplex1, komplex2, act, y1, y2
Public Property Get R1() As Variant
 R1 = reel1
End Property

Public Property Let R1(ByVal vNewValue As Variant)
 reel1 = vNewValue
 y1 = reel1 & IIf(komplex1 > 0, "+", "-") & Abs(komplex1) & "i"
End Property

Public Property Get r2() As Variant
 r2 = reel2
End Property

Public Property Let r2(ByVal vNewValue As Variant)
 reel2 = vNewValue
 y2 = reel2 & IIf(komplex2 > 0, "+", "-") & Abs(komplex2) & "i"
End Property

Public Property Get k1() As Variant
 k1 = komplex1
End Property

Public Property Let k1(ByVal vNewValue As Variant)
 komplex1 = vNewValue
 y1 = reel1 & IIf(komplex1 > 0, "+", "-") & Abs(komplex1) & "i"
End Property

Public Property Get k2() As Variant
```

```
 k2 = komplex2
End Property

Public Property Let k2(ByVal vNewValue As Variant)
 komplex2 = vNewValue
 y2 = reel2 & IIf(komplex2 > 0, "+", "-") & Abs(komplex2) & "i"
End Property

Public Property Get Yazım1() As Variant
 Yazım1 = y1
End Property

Public Property Let Yazım1(ByVal vNewValue As Variant)
 reel1 = Val(vNewValue)
 Dim yer 'imajiner kısmı bulmak için kullanacağız
 'sayı a+bi veya a-bi şeklinde olacağı için aradaki işareti
arıyoruz
 yer = InStr(vNewValue, "+")
 If yer = 0 Then yer = InStr(vNewValue, "-")
 If yer = 1 Then 'reel kısmın işareti var ise tekrar bul
  yer = InStr(2, vNewValue, "+")
  If yer = 0 Then yer = InStr(2, vNewValue, "-")
 End If
 If yer > 1 And Right(vNewValue, 1) = "i" Then 'imajiner kısım
var ise
  komplex1 = Val(Mid(vNewValue, yer))
 Else
  komplex1 = 0
 End If
 y1 = reel1 & IIf(komplex1 > 0,"+","-") & Abs(komplex1) & "i"
End Property

Public Property Get Yazım2() As Variant
 Yazım2 = y2
End Property

Public Property Let Yazım2(ByVal vNewValue As Variant)
 reel2 = Val(vNewValue)
 Dim yer 'imajiner kısmı bulmak için kullanacağız
 'sayı a+bi veya a-bi şeklinde olacağı için aradaki işareti
arıyoruz
 yer = InStr(vNewValue, "+")
 If yer = 0 Then yer = InStr(vNewValue, "-")
 If yer = 1 Then 'reel kısmın işareti var ise tekrar bul
  yer = InStr(2, vNewValue, "+")
  If yer = 0 Then yer = InStr(2, vNewValue, "-")
 End If
 If yer > 1 And Right(vNewValue, 1) = "i" Then 'imajiner kısım
var ise
  komplex2 = Val(Mid(vNewValue, yer))
 Else
  komplex2 = 0
 End If
```



```
 y1 = reel2 & IIf(komplex2 > 0, "+", "-") & Abs(komplex2) & "i"
End Property

Public Property Get Action() As Variant
   'Geriye değer dönmeyecek
End Property

Public Property Let Action(ByVal vNewValue As Variant)
 Dim rel, kol
 Select Case Val(vNewValue)
  Case 1: rel = reel1 + reel2
    kol = komplex1 + komplex2
  Case 2: rel = reel1 - reel2
    kol = komplex1 - komplex2
  Case 3: rel = reel1 * reel2 - komplex1 * komplex2
    kol = reel1 * komplex2 + reel2 * komplex1
  Case 4: rel = (reel1 * reel2 + komplex1 * komplex2) / (reel2 ^
2 + komplex2 ^ 2)
    kol = (reel2 * komplex1 - reel1 * komplex2) / (reel2 ^ 2 +
komplex2 ^ 2)
  Case Else
    MsgBox ("Tanımsız işlem. Action özelliğine 1-4 arası bir
değer verin")
    Exit Property
 End Select
 reel1 = rel
 komplex1 = kol
 y1 = reel1 & IIf(komplex1 > 0, "+", "-") & Abs(komplex1) & "i"
End Property
```

Class tipinden değişken tanımlama

Artık kendimize ait Komplex isimli bir sınıfımız var ve programın istediğimiz yerinden bu tipten değişkenler tanımlayıp onlar üzerinde işlemler yapabiliriz.

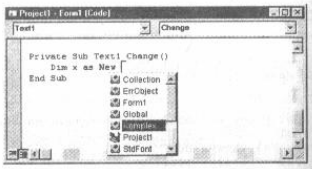
Class olarak tanımlanan tipten değişkenler tanımlamak için yine **Dim** deyimini **As New** parametresi ile kullanacağız.

Dim deg As New Sınıf

Örneğimizden bir x değişkeni tanımlamak için

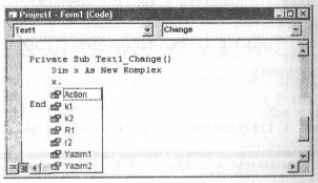
```
Dim x As New Komplex
```

şeklinde kullanacağız. Dikkat ederseniz Dim x As New yazdıktan sonra açılan listede bizim tanımladığımız sınıfında ismini görebiliyoruz.



Aynı anda isterseniz birden fazla değişkende tanımlayıp bağımsız olarak bunlar arasında işlemler yapabilirsiniz.

Tanımladığımız bu x değişkeninin propertieslerine diğer propertiesler gibi ulaşabileceksiniz. Zaten siz **x.** yazdığınızda sınıfımıza ait özellikler gösterilecektir.

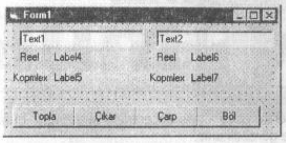


Propertieslerin doğru çalışıp çalışmadığını kontrol etmek için formun Load olayına aşağıdaki kodları yazalım.

```
Private Sub Form_Load()
 Show
 Dim z As New Komplex
 z.R1 = 2
 z.k1 = 4
 Print z.Yazım1'sonuç:2+4i
 z.r2 = 5
 z.k2 = 1
 Print z.Yazım2'sonuç:5+1i
 z.Action = 1 'topla
 Print z.Yazım1 'sonuç: 7+5i
 z.Yazım1 = "7+3i"
 z.Yazım2 = "6+8i"
 z.Action = 3 'çarp
 Print z.Yazım1 'sonuç:18+74i
End Sub
```


Artık komplex sayılar üzerinde işlemler yapmak için bu Class Modülünü kullanabilirsiniz. Tabi Action özelliğine ekler yaparak çok farklı işlemleri de yaptırabilirsiniz.

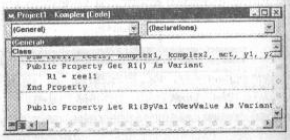
Örneğimizdeki forma aşağıdaki kontrolleri yerleştirip komplex işlemleri yaptıralım.



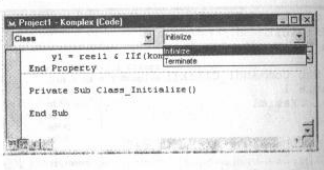

```
'ÖRNEK: ClassModule
Option Explicit
Dim k As New Komplex

Private Sub Command1_Click()
 k.Action = 1
 Text1 = k.Yazım1
End Sub

Private Sub Command2_Click()
 k.Action = 2
 Text1 = k.Yazım1
End Sub
Private Sub Command3_Click()
 k.Action = 3
 Text1 = k.Yazım1
End Sub

Private Sub Command4_Click()
 k.Action = 4
 Text1 = k.Yazım1
End Sub

Private Sub Text1_Change()
 k.Yazım1 = Text1
 Label4 = k.R1
 Label5 = k.k1
End Sub

Private Sub Text2_Change()
 k.Yazım2 = Text2
 Label6 = k.r2
 Label7 = k.k2
End Sub
```

Eventsler

Class Modülüne ait iki tanede olay bulunmaktadır.



Yukarıdaki pencereden Class seçeneğini seçerseniz buna ait iki tane olayı kod penceresinde görebilirsiniz.



Bunlardan biri Initialize olayı ki bu olay tanımladığınız sınıftan yeni bir değişken tanımlandığında oluşur. Değişken bellekten atılırken de Terminate olayı oluşur. Bu olayları daha çok kaynak ayıran sınıflarda kullanabilirsiniz. Tanımladığınız sınıftan yeni bir değişken tanımlandığında ona ait kaynakları ayırabilir ve o değişken sona erdiğinde de bu kaynakları serbest bırakabilirsiniz. Örneğin tanımladığınız sınıf bir dosya kullanıyorsa bu dosyayı Initialize olayında açabilir ve Terminate olayında da kapatabilirsiniz.

```
Sub Form_Load()
  Dim x As New Komplex
  komutlar
End Sub
```

Yukarıdaki gibi bir olay alt programında Komplex tipindeki değişkeni tanımladığınız anda Initialize olayı meydana gelir. Alt program çalışmasını bitirdiğinde ise değişken bellekten atılacağı için Terminate olayı meydana gelecektir. Tabi ki bu değişken yerel bir değişken değil de global bir değişken olarak tanımlanırsa program kapanırken Terminate olayı meydana gelecektir.

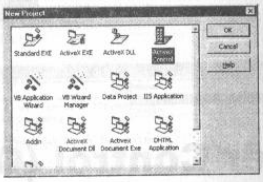
Bölüm 12

User Control (OCX) Property Page

User Control (OCX)

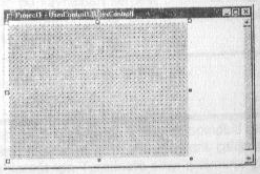
Visual Basic'in ve diğer visual dillerin en önemli özellikleri OCX türü dosyalarda tanımlanmış olan componentleri kullanabilmeleridir. Zaten kitap boyunca gördüğümüz kontrollerin bir çoğu özel OCX dosyalarında bulunmakta idi. İşte biz de yeni kontroller hazırlayıp bunları OCX dosyası olarak derleyebiliriz.

File menüsünden **New Project** seçeneği ile açılan aşağıdaki pencereden **ActiveX Control** seçeneğini kullanarak kendi ActiveX dosyalarınızı oluşturabilirsiniz.



Bu dosyalar CTL uzantılı dosyalara kaydedilirler. File menüsündeki Make ..... OCX seçeneği ile de OCX dosyası olarak derlenebilirler. Kendi OCX dosyalarınızı bu şekilde oldukça rahat oluşturabilirsiniz.

Bir ActiveX projesi başlattığınızda aşağıdaki gibi bir pencere açılır. Bu pencereyi kullanarak kendi kontrolünüzü oluşturabilirsiniz.

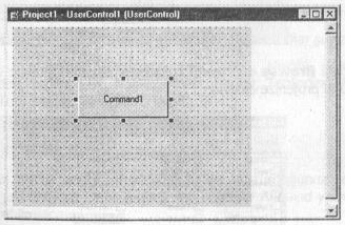


ActiveX projenizdeki kontrolü oluştururken VB'nin ToolBox penceresindeki hazır kontrollerden bir çoğunu kullanabilirsiniz. Kendi kontrolünüzü bir form oluşturuyormuş gibi rahatça oluşturabilirsiniz. Oluşturduğunuz proje bir OCX dosyası olacağı için kendine özgü bazı özellik, olay ve metodlarınında bulunması gerekecektir. Bütün bunların nasıl oluşturulacağını şimdi göreceğiz.

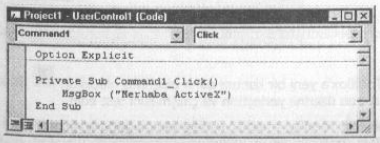
ActiveX projenizi tasarım esnasında çalıştıramazsınız. Bunları test edebilmek için önce projenizi bir OCX dosyası olarak derlemeniz ve yeni bir **Standart EXE** uygulaması başlatıp bu OCX dosyasını o projeye dahil ederek test etmeniz gerekecektir. Bu yüzdende kontrolü bir OCX olarak derlemeden önce, henüz tasarlarken rahatça deneme yanılma yapabilmek için standart bir EXE uygulaması başlatın ve **Project** menüsündeki **Add User Control** seçeneği ile kendi kontrolünüzü tasarlayın. Böylece hem kontrolü tasarlayabilir hemde projenizdeki form üzerine alıp test edebilirsiniz. Ayrıca çıkabilecek hata mesajlarını adım adım çalıştırarak ayıklayabilirsiniz. Projenizin son halinde ise tasarladığınız kontrolü bir ActiveX projesine ekleyerek OCX dosyası olarak derleyebilirsiniz. Exe uygulamasında tasarladığınız UserControlleri bir OCX dosyası olarak derlemeden önce Public özelliğini True yapmanız gerekir.

ÖRNEK:

Önce basit bir OCX dosyası oluşturalım. Örneğimiz için oluşturduğunuz UserControl'ün üzerine bir komut düğmesi yerleştirin.

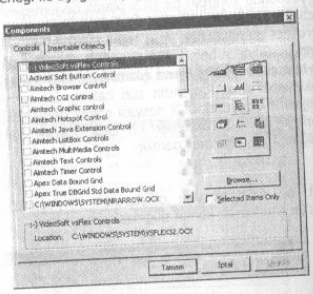


Ve Click olayına da aşağıdaki kodu yazın.

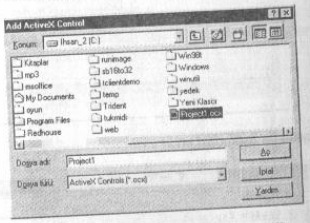


File menüsünden **Make Project1.ocx** seçeneğini kullanarak projemizi Project1.Ocx olarak derleyelim.

Daha sonra **File** menüsünden **New Project** seçeneği ile **Standart Exe** uygulamasını seçip yeni bir projeye başlayalım. Daha sonra **Project** menüsündeki **Component** seçeneği ile aşağıdaki pencereyi açın.



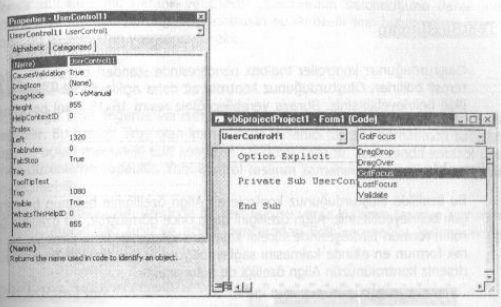
Penceredeki **Browse** düğmesini kullanarak oluşturduğumuz OCX dosyasını bulun ve bunu projenize ekleyin.



Artık ToolBox'a yeni bir kontrol daha eklenmiş olacaktır. Bu kontrolü seçerek formunuzun üzerine yerleştirin ve çalışmasını test edin.

Herhangi bir özellik, metod ve olay tanımı yapmadığımız halde kontrolümüzün aşağıdaki özelliklerinin bulunduğunu göreceksiniz. VB oluşturduğumuz kontrole

standart olarak bu özellikleri vermektedir. Buraya daha farklı özellikleri eklemekte bizim elimizde. Bunun nasıl olacağını da birazdan göreceğiz.



VB'nin kontrolümüze verdiği standart özellik, olay ve metodlar şunlardır:

Properties: Container, DragIcon, DragMode, Height, HelpContextID, CausesValidation, Index, Left, Name, Object, Parent, TabIndex, TabStop, Tag, ToolTipText, Top, Visible, WhatsThisHelpID, Width

Events: DragDrop, DragOver, GotFocus, LostFocus

Methods: Drag, Move, SetFocus, ShowWhatsThis, Zorder

Görüldüğü gibi hiç bir kod yazmadığımız halde oluşturduğunuz kontrol bir çok işlemi yerine getirebilecek kapasitededir. Bu kontrole yine kod yazmadan bir çok özelliği de verebilmekteyiz.

Properties

Şimdi tekrar ActiveX projemizi açalım ve bu kontrole ait özelliklere bir göz atalım.

Public

Bu özellik True ise oluşturduğunuz kontrolü diğer uygulamalarda kullanabilir, False ise sadece projenizdeki formlarda kullanabilirsiniz. Örneğin bir OCX dosya-

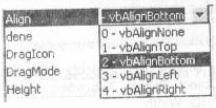
sındaki User Kontrollerin dahil edildikleri projeden kullanılabilmeleri için bu özelliğin True olması gerekir.

ToolBoxBitmap

Oluşturduğunuz kontroller toolbox penceresinde standart olarak simgesi ile temsil edilirler. Oluşturduğunuz kontrole ait daha açıklayıcı bir simgeyi bu özellikle belirleyebilirsiniz. Buraya verebileceğiniz resim 16x15 pixel boyutlarında olabilir.

Alignable

Bu özellikle oluşturduğunuz kontrole ait Align özelliğinin bulunup bulunmayacağını belirleyebilirsiniz. Align özelliğini daha önce görmüştük. Bu özellikle bir kontrolün formun bir köşesinde sürekli kalması sağlanabiliyordu. Örneğin sürekli olarak formun en altında kalmasını sağlayabiliyorduk. Bu özelliğe True değerini verirseniz kontrolünüzün Align özelliği de bulunacaktır.



CanGetFocus

Kontrolün klavye kontrolünü alıp alamayacağı bu özellikle belirlenir. Normalde True'dir ve kontrol Focus alabilir.

ControlContainer

Bazı kontrollerin diğer kontrolleri içinde barındırabildiklerini biliyorsunuz. Daha önce gördüğümüz Frame ve PictureBoxlar bu tip kontrollerdi ve içinde diğer kontrolleri barındırabiliyordu. Bu özelliğe True değerini verirseniz oluşturduğunuz kontrol diğer kontrolleri barındırabilir. Yani bir gruplandırma elemanı gibi davranır ve içine konan kontroller bu kontrole bağımlı olurlar.

DefaultCancel

Komut düğmelerinin Default ve Cancel özelliklerinin bulunduğunu daha önce görmüştük. Eğer oluşturduğunuz kontrolün bu iki özelliğinin bulunmasını istiyorsanız bu özelliği True yapmanız gerekir.

EditAtDesignTime

Bu özelliğe True değerini verirseniz kontrolünüz tasarım modunda Edit moduna geçebilir. Bu özelliği True olan kontrollerde kullanıcı tasarım zamanında sağ fare tuşu ile açılan menüdeki Edit komutunu seçerek tasarım zamanında kontrol üzerinde düzenleme yapabilir. Yani kontrol tasarım zamanında da çalışma durumuna geçirilebilir.

Örneğin kontrolünüzün üzerinde bir text kutusu varsa kullanıcı tasarım zamanında iken farenin sağ tuşu ile açılan pencereden Edit seçeneğini kullanarak bu kutuya bilgi girişi yapabilir.

InvisibleAtRunTime

Bazı kontrollerin sadece çalışma zamanında görülebildiğini biliyorsunuz. Örneğin Timer kontrolünü tasarım zamanı ekranda görebilirsiniz ancak program çalışırken Timerin yaptığı iş dolayısıyla ekranda göremezsiniz. Eğer tasarladığınız kontrolde böyle bir kontrolse bu özelliğe False değerini vererek çalışma zamanında görülmemesini sağlayabilirsiniz.

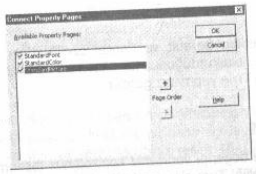
Kontrolünüze bu özelliği verdiğinizde toolboxtaki bir çok kontrolü de projenizde kullanamazsınız.

AccessKeys

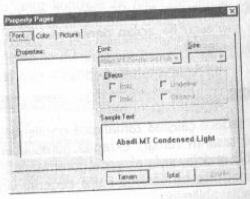
İsterseniz tasarladığınız kontrole bir kısayol tuşu da verebilirsiniz. Bu tuşa basıldığında kontrolün AccessKeyPress olayı meydana gelir. Örneğin Alt-x tuşlarına basıldığında kontrolün aktif olmasını istiyorsanız bu özelliğe "x" değerini verebilirsiniz.

PropertyPages

Bazı kontrollerin properties penceresinde birden fazla özelliği değiştirebileceğiniz pencerelerin bulunduğunu biliyorsunuz. Eğer sizin kontrolünüzde de böyle bir pencere bulunmasını istiyorsanız bu özellikle açılan aşağıdaki pencereden istediğiniz özellikleri properties penceresine ekleyebilirsiniz.



Bu işlemden sonra kontrolünüzde Custom isimli yeni bir özellik daha bulunacaktır. Bu özellik çift tıklandığında ise Font, Renk ve Resim özelliklerinin topluca değiştirilebildiği aşağıdaki pencere açılacaktır.



İsterseniz kendi Property Page'lerinizi tasarlayabilir veya hazır olanlardan bir tanesini de kullanabilirsiniz. **Project** menüsündeki **Add Property Page** seçeneği ile programınıza yeni Property Page'ler ekleyebilir veya yenilerini oluşturabilirsiniz. Oluşturduğunuz bu Page'ler yukarıdaki listede görülecektir.

Bu özelliği bir dizi gibi kullanarak her bir property sayfasının tek tek değiştirebilirsiniz. PropertyPages(0) ile ilk property sayfasına ulaşılabilir.

Ambient

Bu özellikle User Kontrolün içinde bulunacağı yerin bazı özellikleri öğrenilebilir. Örneğin tasarladığınız kontrol bir form içine konursa Ambient özellikleri ile o formun, bir Frame içine koyulursa o framenin özellikleri öğrenilebilir.

Ambient özelliğinin şu alt özellikleri vardır: BackColor, ForeColor, Font, DisplayAsDefault, DisplayName, LocaleID, MessageReflect, Palette, RightToLeft, ScaleUnits, ShowGrabHandles, ShowHatchings, SupportsMnemonics, TextAlign, UserMode, UIDead

Bunlardan BackColor, ForeColor, Font gibi standart özelliklerin ne olduğunu daha önce görmüştük.

Örneğin oluşturduğunuz kontrolün yazı rengi, zemin rengi ve font özelliklerinin içine konduğu yere uyum göstermesi için:

```
BackColor = Ambient.BackColor
ForeColor = Ambient.ForeColor
Font = Ambient.Font
```

atamaları yapılabilir. User kontrolünüzdeki bütün kontrollerin yukarıdaki özelliklerini Ambient özelliklerine atarsanız user kontrolünüz içine konduğu yere uyum gösterecektir. Bu işlemi AmbientChange olayında yaptığınızda sürekli olarak oluşacak değişimlere uyum göstermiş olursunuz.

Ambient.DisplayName bu özellikle User Kontrolünüzün ismini öğrenebilirsiniz. Bildiğiniz gibi kontrollerin isimleri Name özellikleri ile değiştirilebilmektedir.

Ambient.ScaleUnits bu özellikle user kontrolün içinde bulunduğu yerin Scale Mode özelliğine verilen değer öğrenilebilir. Örneğin kontrolün içinde bulunduğu formun ScaleMode özelliği Pixel verilmişse bu özellikte Pixel değerini alır. Eğer kontrolünüzde koordinat özellikleri bulunuyorsa bu özelliği kullanarak kontrolün içinde bulunduğu formla aynı ölçü modunu kullanmasını sağlayabilirsiniz.

Ambient.UserMode bu özellikle o anda içinde bulunulan kontrolün tasarım zamanında mı çalışma zamanında mı olduğu öğrenilebilir. Eğer bu özellik True değerini alıyorsa çalışma zamanı, False ise tasarım zamanıdır.

Ambient.UIDead bu özellikle de o anda kontrolün içinde bulunduğu yerin break modunda olup olmadığı öğrenilebilir. VB'de bir program çalışırken kullanıcı Ctrl-Break tuşlarına basarak işlemleri kırabilir ve daha sonra devam ettirebilir. Bu break durumunda kontroller kullanıcının hareketlerine cevap vermez. UIDead özelliğinin True olması bu modun aktif olduğu gösterir.

Ambient.RightToLeft bu özellikle sistemin yazıları soldan sağa mı yoksa sağdan sola mı yazdığı öğrenilebilir. Bu özellik true değerini alıyorsa yazılar sağdan sola doğru yazılıyordur. Yani Arapça gibi sağdan sola doğru yazılan bir Windows kullanılıyordur.

Ambient.TextAlign user kontrolün içinde bulunduğu yerdeki yazı yerleşimi bu özellikle öğrenilebilir. Bu özelliğin alabileceği değerler ve anlamları şunlardır:

0-General:Yazılar için sola, sayılar için sağa
1-Left: Sola dayalı
2-Center: Ortada
3-Right: Sağa dayalı
4-FillJustify: İki yana dayalı

Extender

UserControl'ün aşağıdaki özellikleri bununla öğrenilip değiştirilebilir.

Name, Visible, Parent, Cancel, Default, Container, DragIcon, DragMode, Enabled, Height, HelpContextID, Index, Left, TabIndex, TabStop, Tag, ToolTipText, Top, WhatThisHelpID, Width

Ayrıca user kontrole ait aşağıdaki metodlar da Extender ile birlikte kullanılır.

Drag, Move, SetFocus, ShowWhatsThis, Zorder

Örneğin User Kontrolün yüksekliğini değiştirmek için aşağıdaki gibi kullanılır.

```
Extender.Height=50
```

Events

Initialize()

UserControl form üzerine veya başka bir yere konulduğunda kısacası yeni bir tanesi türetildiğinde Initialize olayı meydana gelir. Yapılacak ilk değer atamaları ve kontroller bu olay kullanılarak kodlanır.

InitProperties()

User control oluşturulduğunda Initialize olayından sonra InitProperties olayı meydana gelir. Bu olay daha çok özelliklere ilk değerlerini atamak için kullanılır.

Terminate()

UserControl bellekten atıldığında bu olay meydana gelir. Initialize olayında ayrılmış kaynaklar varsa burada o kaynaklar serbest bırakılmalıdır. Örneğin Initialize olayında bir dosya açılmışsa bu olayda da kapatılmalıdır.

Kontrolün oluşturulan her kopyası için Initialize olayı ve her kopyanın kaldırılması durumunda da Terminate olayı meydana gelecektir.

AccessKeyPress(KeyAscii As Integer)

Kullanıcı AccessKeys özelliği ile belirlenen tuşlara bastığında bu olay meydana gelir. Eğer kontrolün DefaultCancel özelliği de True yapılmış ise Enter ve Esc tuşları da bu olayı meydana getirir.

Hide()

Kontrolün Visible özelliği False yapıldığında bu olay meydana gelir.

Show()

Kontrolün Visible özelliği True yapıldığında bu olay meydana gelir.

AmbientChanged(PropertyName As String)

Ambient özelliklerini yukarıda görmüştük. Bu özelliklerden biri değiştiğinde bu olay meydana gelir. Örneğin UserControl bir form üzerine yerleştirilmiş ise formun zemin rengi değiştiğinde bu olay meydana gelecektir. Böylece üzerinde bulunduğunuz yere uyum göstermek için gerekli atamaları burada yapabilirsiniz. Hangi özelliğin değiştirildiği ise PropertyName özelliği ile öğrenilebilir.

Örneğin user kontrolün içinde bulunduğu yerin zemin rengi değiştiğinde user kontrolün zemin renginin de aynı olmasını sağlamak için

```
Private Sub UserControl_AmbientChanged(PropertyName As String)
    If PropertyName = "BackColor" Then BackColor =
Ambient.BackColor
End Sub
```

Artık formun zemin rengi değiştiğinde user kontrolünde zemin rengi değişecektir.

EnterFocus(),ExitFocus()

Klavye kontrolü User Control içindeki bir nesneye geçtiğinde veya kaybettiğinde o nesnenin GotFocus ve LostFocus olaylarından önce UserControl'ün EnterFocus ve ExitFocus olayları meydana gelir.

ReadProperties(PropBag As PropertyBag)

Bir properties değeri tasarım zamanında farklı çalışma zamanında farklı olabilir. Örneğin Meslek yazılması gereken bir Text kutusunun Text özelliği tasarım zamanında text1 olsun. Program çalıştırıldığında bu text kutusunun Text özelliği kullanıcının yazacağı mesleklerle değişecektir. Program kapatılıp tekrar tasarım moduna döndüğünde ise Text kutusunun içeriğinde kullanıcının yazdığı meslek değil orijinal değer yani çalıştırılmadan önceki değer bulunmalıdır.

Bunun için ReadProperties ve WriteProperties olayları geliştirilmiştir. ReadProperties olayında propertieslerin orijinal değerlerini okuyacak WriteProperties olayında da yazacak kodları koymak gerekir. Ayrıca bu işlemler

program kaydedilip daha sonra açıldığında da verilen propertieslerin değerlerinin korunmasını sağlar.

Propertieslerin değerlerini öğrenme ve değiştirme işlemi PropBag parametresinin ReadProperty ve WriteProperty metodları ile yapılır.

```
Değer=PropBag.ReadProperty("PropertiesAdı", VarsayılanDeğeri)
```

şeklinde bir kodla bir propertiesin değeri bir global değişkene alınabilir.

WriteProperties(PropBag As PropertyBag)

Bu olayda da PropBag parametresinin WriteProperty metodu ile propertiesin daha önce ReadProperty özelliği ile saklanmış değeri propertiese verilir.

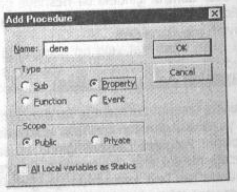
```
Call PropBag.WriteProperty("PropertiesAdı ", Değer, VarsayılanDeğer)
```

şeklinde bir kodla bir propertiesin daha önce hafızaya alınmış değeri propertiese tekrar verilir.

Kontrolün Propertieslerini Belirleme

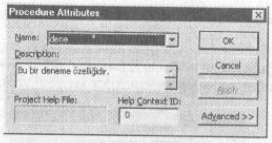
Evet User Kontrolün yukarıdaki özellikleri ile, oluşturduğunuz kontrolde hangi özelliklerin bulunabileceğini belirleyebilirsiniz. Ancak bunlar VB'nin sunduğu standart özelliklerdir. Kendi özelliklerinizi ve olaylarınızı da tanımlayabilirsiniz. Daha önce Class Module kısmında bir Propertiesin nasıl yazıldığını görmüştük. Aynı yöntemi kullanarak kendi kontrolünüz için yeni propertiesler de yazabilirsiniz.

Tools menüsündeki Add Procedure menü seçeneği ile açılan aşağıdaki pencereden yeni propertiesler oluşturabiliyorduk.



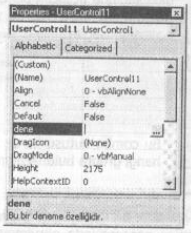
Bu işlem sonucunda Get ve Let isimli iki yeni prosedür oluşuyordu ve bunlardan biri özelliğe değer atanırken diğeri de değeri öğrenilirken çalışıyordu.

Bu şekilde oluşturduğumuz propertiesler, oluşturduğumuz kontrolün properties penceresinde de görülecektir. Bu propertiesin nasıl görüleceğini yine Tools menüsündeki Procedure Attributes seçeneği ile açılan aşağıdaki pencereden belirleyebiliyoruz.

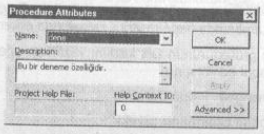


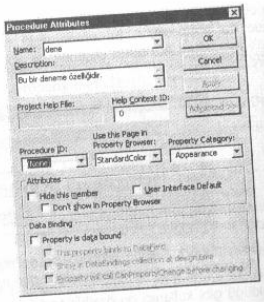
Penceredeki Description kutusuna propertiesin ne yaptığına ilişkin bir açıklama yazabilirsiniz. Bu metin properties penceresinde görülecek metini belirler.

Aşağıda görüldüğü gibi kullanıcı bu özelliği seçtiğinde buraya yazdığımız açıklama properties penceresinde görülecektir.



Bu penceredeki **Advanced>>** düğmesi ile daha gelişmiş özellikleri belirleyebileceğimiz aşağıdaki pencerede açılacaktır.

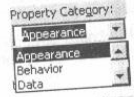




Use this Page in Property Browser:

Bu combo kutusu aracılığı ile özelliğin yanındaki ... düğmesi tıklandığında açılacak pencereyi belirleyebilirsiniz. İsterseniz Renk, Font veya Resim yükleme pencerelerinden birinin açılmasını sağlayabilirsiniz.

Property Category:



Bu combo kutusu ile de özelliğin properties penceresinde hangi grupta bulunacağını belirleyebilirsiniz.

Penceredeki ☐ Don't show in Property Browser seçeneğini işaretlerseniz yazdığınız properties, Properties penceresinde görülmeyecektir. Yani tasarım zamanında değiştirilemeyen bir özellik olacaktır.

Eğer bu propertiesin kullanıcı tarafından kullanılmaması gerekiyorsa ☐ Hide this member seçeneğini işaretleyebilirsiniz. Bu durumda kullanıcı bu özelliği ne çalışma zamanında nede tasarım zamanında öğrenip değiştiremez.

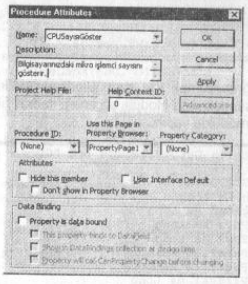
Bir property prosedürünün Let kısmını yazarken eğer kullanıcının yaptığı değer atamasını kabul ediyorsanız

```
PropertyChanged "Propertiesadı"
```

metodu ile bunu bildirmeniz gerekir.

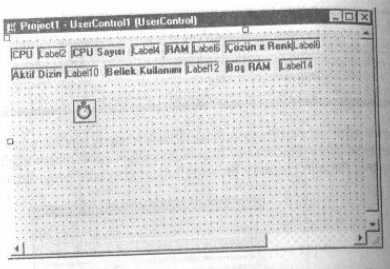

```
Public Property Let CPUSayısıGöster(ByVal vNewValue As Variant)
  Label3.Visible = vNewValue
  Label4.Visible = vNewValue
  diz
  PropertyChanged "CPUSayısıGöster"
End Property
```

Ayrıca eğer properties tasarladığınız bir property sayfasını kullanarak değiştirilebilecekse Tools-Procedure Attributes seçenekleri ile açılan aşağıdaki penceredeki Use this Page in Property Browser kutusundan bu sayfayı seçmeniz gerekir.



ÖRNEK OCX

Örnek olarak bilgisayarın mikro işlemcisini, bilgisayarda bulunan mikro işlemci sayısını, fiziksel bellek miktarını, ekran çözünürlüğünü ve renk sayısını, aktif dizini, belleğin kullanım oranını ve boş fiziksel bellek miktarını gösterecek bir OCX oluşturalım. Örneğimizde kullanıcı bunlardan istediklerini gösterip gizleyebilsin. Örneğimiz için aşağıdaki kontrolleri UserControlünüzün üzerine yerleştirin. Ve timer kontrolünün Interval özelliğini 100 yapın.



Bu bilgileri öğrenebilmek için bazı Windows API'lerini kullanacağız. Projenize Project menüsündeki Add Module seçeneği ile boş bir modül ekleyin ve aşağıdaki tanımları o modülde yapın.

```
Declare Function GetDeviceCaps Lib "GDI32" (ByVal hDC%, ByVal
nIndex%) As Integer
Type MEMORYSTATUS
    dwLength As Long
    dwMemoryLoad As Long
    dwTotalPhys As Long
    dwAvailPhys As Long
    dwTotalPageFile As Long
    dwAvailPageFile As Long
    dwTotalVirtual As Long
    dwAvailVirtual As Long
End Type
Declare Sub GlobalMemoryStatus Lib "kernel32" (lpBuffer As
MEMORYSTATUS)
Type SYSTEM_INFO
    dwOemID As Long
    dwPageSize As Long
    lpMinimumApplicationAddress As Long 'Any
    lpMaximumApplicationAddress As Long 'Any
    dwActiveProcessorMask As Long
    dwNumberOfProcessors As Long
    dwProcessorType As Long
    dwAllocationGranularity As Long
    dwReserved As Long
End Type
Declare Sub GetSystemInfo Lib "kernel32" (lpSystemInfo As
SYSTEM_INFO)
```



```
Declare Function GetDiskFreeSpace Lib "kernel32" (ByVal
lpRootPathname As String, lpSectorsPerCluster As Long,
lpBytesPerSector As Long, lpNumberOfFreeClusters As Long,
lpTtoalNumberOfClusters As Long) As Long
```

Projemizde göstereceğimiz bazı bilgiler bilgisayar çalışırken sürekli değişebileceği için onları timer olayına, bazıları ise sürekli sabit olacağı için onları da Initialize olayına yazarak sadece başlangıçta çalışmalarını sağlayacağız. Ayrıca kullanıcı istemediği özellikleri gizleyebileceği için labellerin yerleşim sırasını düzenleyecek birde *diz* isimli prosedürü yazacağız.

```
Private Sub UserControl_Initialize()
 Const PLANES = 14
 Const BITSPIXEL = 12
 Const colorres = 108
 Label8 = Screen.Width \ Screen.TwipsPerPixelX & " x " &
Screen.Height \ Screen.TwipsPerPixelY & "x " & GetDeviceCaps(hDC,
PLANES) * 2 ^ GetDeviceCaps(hDC, BITSPIXEL)
 Dim m As MEMORYSTATUS
 GlobalMemoryStatus m
 Label6 = Int(m.dwTotalPhys / 1024) & "kb"
 Label12 = "%" & m.dwMemoryLoad
 Label14 = Int(m.dwAvailPhys / 1024) & "kb"
 Dim s As SYSTEM_INFO
 GetSystemInfo s
 Label4 = s.dwNumberOfProcessors
 Label2 = s.dwProcessorType
 diz
End Sub

Private Sub Timer1_Timer()
 Dim s As SYSTEM_INFO
 GetSystemInfo s
 Label4 = s.dwNumberOfProcessors
 Dim m As MEMORYSTATUS
 GlobalMemoryStatus m
 Label12 = "%" & m.dwMemoryLoad
 Label10 = CurDir
 diz
End Sub

Public Sub diz()
 Label1.Top = 0
 Label2.Top = 0
 Label3.Top = 0
 Label4.Top = 0
 Label5.Top = 0
 Label6.Top = 0
 Label7.Top = 0
 Label8.Top = 0
 Label9.Top = 0
 Label10.Top = 0
```



```
 Label11.Top = 0
 Label12.Top = 0
 Label13.Top = 0
 Label14.Top = 0
 Dim l
 If Label1.Visible Then
  Label1.Left = l
  Label2.Left = Label1.Left + Label1.Width
  l = Label2.Left + Label2.Width
  l = l + 15
 End If
 If Label3.Visible Then
  Label3.Left = l
  Label4.Left = Label3.Left + Label3.Width
  l = Label4.Left + Label4.Width
  l = l + 15
 End If
 If Label5.Visible Then
  Label5.Left = l
  Label6.Left = Label5.Left + Label5.Width
  l = Label6.Left + Label6.Width
  l = l + 15
 End If
 If Label7.Visible Then
  Label7.Left = l
  Label8.Left = Label7.Left + Label7.Width
  l = Label8.Left + Label8.Width
  l = l + 15
 End If
 If Label9.Visible Then
  Label9.Left = l
  Label10.Left = Label9.Left + Label9.Width
  l = Label10.Left + Label10.Width
  l = l + 15
 End If
 If Label11.Visible Then
  Label11.Left = l
  Label12.Left = Label11.Left + Label11.Width
  l = Label12.Left + Label12.Width
  l = l + 15
 End If
 If Label13.Visible Then
  Label13.Left = l
  Label14.Left = Label13.Left + Label13.Width
  l = Label14.Left + Label14.Width
  l = l + 15
 End If
End Sub
```

Şimdi kontrolümüzün kullanacağı propertiesleri yazalım. Kontrolümüzde bulunan her bilgiyi gösterip gizleyecek ve timer kontrolünün intervalini belirleyebileceğimiz propertiesleri yazacağız.


```
Public Property Get CPUGöster() As Variant
 CPUGöster = Label1.Visible
End Property

Public Property Let CPUGöster(ByVal vNewValue As Variant)
  Label1.Visible = vNewValue
  Label2.Visible = vNewValue
  diz
  PropertyChanged "CPUGöster"
End Property

Public Property Get CPUSayısıGöster() As Variant
 CPUSayısıGöster = Label3.Visible
End Property

Public Property Let CPUSayısıGöster(ByVal vNewValue As Variant)
  Label3.Visible = vNewValue
  Label4.Visible = vNewValue
  diz
  PropertyChanged "CPUSayısıGöster"
End Property

Public Property Get RAMGöster() As Variant
 RAMGöster = Label5.Visible
End Property

Public Property Let RAMGöster(ByVal vNewValue As Variant)
  Label5.Visible = vNewValue
  Label6.Visible = vNewValue
  diz
  PropertyChanged "RAMGöster"
End Property

Public Property Get ÇözünxRenkGöster() As Variant
 ÇözünxRenkGöster = Label7.Visible
End Property

Public Property Let ÇözünxRenkGöster(ByVal vNewValue As Variant)
  Label7.Visible = vNewValue
  Label8.Visible = vNewValue
  diz
  PropertyChanged "ÇözünxRenkGöster"
End Property
Public Property Get AktifDizinGöster() As Variant
 AktifDizinGöster = Label9.Visible
End Property

Public Property Let AktifDizinGöster(ByVal vNewValue As Variant)
  Label9.Visible = vNewValue
  Label10.Visible = vNewValue
  diz
  PropertyChanged "AktifDizinGöster"
End Property
```

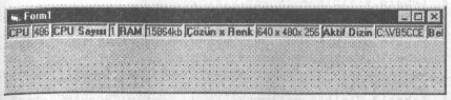
```
Public Property Get KulBellekGöster() As Variant
 KulBellekGöster = Label11.Visible
End Property

Public Property Let KulBellekGöster(ByVal vNewValue As Variant)
  Label11.Visible = vNewValue
  Label12.Visible = vNewValue
  diz
  PropertyChanged "KulBellekGöster"
End Property

Public Property Get BoşRAMGöster() As Variant
 BoşRAMGöster = Label13.Visible
End Property

Public Property Let BoşRAMGöster(ByVal vNewValue As Variant)
  Label13.Visible = vNewValue
  Label14.Visible = vNewValue
  diz
  PropertyChanged "BoşRAMGöster"
End Property

Public Property Get GüncellemeSüresi() As Variant
 GüncellemeSüresi = Timer1.Interval
End Property

Public Property Let GüncellemeSüresi(ByVal vNewValue As Variant)
 Timer1.Interval = vNewValue
 PropertyChanged "GüncellemeSüresi"
End Property

Public Property Get Hakkında() As Variant
 MsgBox ("SysInfoOcx32 V.1.0 İhsan KARAGÜLLE Mart 1997
Ahlat/BİTLİS")
End Property

Public Property Let Hakkında(ByVal vNewValue As Variant)
 '
End Property
```

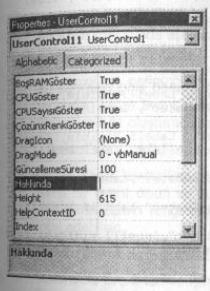
Kontrolümüze ait propertiesleri tanımladık. Ayrıca birde metod tanımlamış olduk. Projenizde kullandığınız Public sıfatlı alt program ve fonksiyonlar kontrolün birer metodu olarak görülür. Yukarıda tanımladığımız Diz prosedürü bu kontrolün bir metodu olacaktır. Eğer bu metodların kullanıcı tarafından çağrılması gerekmiyorsa bunları Public olarak değil Private olarak belirlemeniz gerekir.

Ayrıca kontrolümüzün align özelliğinin bulunması da yerinde bir karar olacaktır. Bunun içinde UserControl'ün Alignable özelliğini True yapalım.

Projemizi bu haliyle derleyerek (File menüsündeki Make OCX seçeneği ile) OCX dosyamızı oluşturun. Ve standart bir EXE uygulamasında Project-Components-

Browse seçenekleri ile OCX kontrolümüzü projenize ekleyin ve formun üzerine yerleştirin.





Kontrolümüz bu haliyle istediğimiz işlemleri yapabilmektedir. Kullanıcı istediği sistem bilgilerini properties penceresi ile gösterip gizleyebilir. Ancak kullanıcı bazı propertiesleri False yaparak gizleyip programı çalıştırdığında False yaptığı özelliklerin tekrar True olduğunu görecektir. Bunun nedeni ise WriteProperties ve ReadProperties olayları ile propertieslerin orijinal değerlerini saklayıp okuyacak kodun yazılmamış olduğundandır.

Özelliklerin Eski Değerlerini Hatırlamasını Sağlama

Yukarıda gördüğünüz gibi properties penceresi aracılığı il bir özeliği değiştirdikten sonra programı çalıştırıp sona erdirirseniz veya programı kaydedip tekrar açarsanız eski değerleri hatırlamadığını göreceksiniz. Eski değerleri hatırlayabilmesi için **ReadProperties ve WriteProperties** olaylarına kod yazmak gerekir.

Propertieslerin orijinal değerlerini saklayıp-okuyacak kodları da aşağıdaki gibi yazalım.

```
Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
 Label1.Visible = PropBag.ReadProperty("CPUGöster", True)
 Label2.Visible = Label1.Visible
 Label3.Visible = PropBag.ReadProperty("CPUSayısıGöster", True)
```



```
 Label4.Visible = Label3.Visible
 Label5.Visible = PropBag.ReadProperty("RAMGöster", True)
 Label6.Visible = Label5.Visible
 Label7.Visible = PropBag.ReadProperty("ÇözünxRenkGöster", True)
 Label8.Visible = Label7.Visible
 Label9.Visible = PropBag.ReadProperty("AktifDizinGöster", True)
 Label10.Visible = Label9.Visible
 Label11.Visible = PropBag.ReadProperty("KulBellekGöster", True)
 Label12.Visible = Label11.Visible
 Label13.Visible = PropBag.ReadProperty("BoşRAMGöster", True)
 Label14.Visible = Label13.Visible
 Timer1.Interval = PropBag.ReadProperty("GüncellemeSüresi", 100)
 diz
End Sub

Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
 Call PropBag.WriteProperty("CPUGöster", Label1.Visible, True)
 Call PropBag.WriteProperty("CPUSayısıGöster", Label3.Visible,
True)
 Call PropBag.WriteProperty("RAMGöster", Label5.Visible, True)
 Call PropBag.WriteProperty("ÇözünxRenkGöster",Label7.Visible,
True)
 Call PropBag.WriteProperty("AktifDizinGöster",Label9.Visible,
True)
 Call PropBag.WriteProperty("KulBellekGöster",Label11.Visible,
True)
 Call PropBag.WriteProperty("BoşRAMGöster",Label13.Visible, True)
 Call PropBag.WriteProperty("GüncellemeSüresi", Timer1.Interval,
100)
 diz
End Sub
```

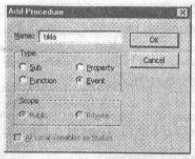
Artık tasarladığımız kontrol properties değerlerini çalışma ve tasarım zamanları arasında koruyabilecektir.

Kontrole Events Yazma

Bir kontrole nasıl properties ve metod tanımlanabileceğini gördük. Ancak bildiğiniz gibi kontrollerin birde Eventsleri vardır. Yani kullanıcının bazı hareketlerini algılayarak buna karşılık gelebilecek bir olay alt programını devreye sokması ve programcının bu olay alt programı ile programı yönlendirebilmesini sağlamalıdır.

VB'de oluşturduğumuz kontrollere Event'lerde tanımlayabilmekteyiz. Bu işi daha önce properties tanımlamak için kullandığımız aynı pencereden yapacağız.

Tools menüsündeki Add Procedure seçeneği ile açılan aşağıdaki pencereden Event seçeneğini işaretleyerek bir event oluşturabilmekteyiz.



Bu işlem sonucunda aşağıdaki gibi bir satır oluşacaktır.

Public Event tıkla()

Bu satırdaki giriş parametrelerinin ne olacağını belirlemek sizin elinizdedir. Bu tanımdan sonra Event'in ne zaman olacağını ise başka bir standart olaydayken RaiseEvent ile oluşturacağız.

RaiseEvent tıkla()

Örnek olarak kontrolümüze ait iki tane event tanımlayalım. Bunlardan biri değiş isimli bir event olsun ve kullanıcı kontrolümüzü çift tıklarsa kontrolü üzerindeki etiketlerin zemin rengi rasgele değişsin. Ayrıca programcı isterse bu değişimi iptal edebilsin.

Public Event değiş(iptal As Boolean)

yukarıdaki Event tanımı işimizi görecektir. Programcı isterse İptal parametresine True değerini atayarak eventin çalışmasını iptal etsin.

Şimdi bu eventin ne zaman oluşacağını belirleyelim. Kullanıcı kontrolü çift tıkladığında bu işlemin oluşmasını istediğimiz için UserControl'ün DblClick olayını kullanabiliriz.

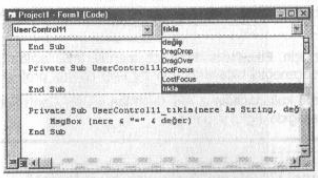
```
Private Sub UserControl_DblClick()
 Dim x, i
 RaiseEvent değiş(x)
 On Local Error Resume Next
 If Not x Then 'iptal edilmedi ise
  For i = 0 To Controls.Count - 1
   Controls(i).BackColor = QBColor(Rnd * 15)
  Next
 End If
End Sub
```

Yukarıda görüldüğü gibi kontrol çift tıklandığında **RaiseEvent** ile **değiş** olayını meydana getiriyoruz. Eğer kullanıcının yazdığı kod x parametresine **true** atamamışsa kontrol üzerindeki bütün zemin renklerini değiştiriyoruz.

İkinci olayımız ise tıkla olsun. Ve kullanıcı kontrol üzerinde bir değeri tıkladığında o değerin ne olduğunu ve kaç olduğunu iki parametre ile bildirsin.

```
Public Event tıkla(nere As String, değer As Variant)
Private Sub Label1_Click()
 RaiseEvent tıkla("CPU", Label2.Caption)
End Sub
Private Sub Label10_Click()
 RaiseEvent tıkla("Aktif Dizin", Label10.Caption)
End Sub
Private Sub Label11_Click()
 RaiseEvent tıkla("Bellek Kullanımı", Label12.Caption)
End Sub
Private Sub Label12_Click()
 RaiseEvent tıkla("Bellek Kullanımı", Label12.Caption)
End Sub
Private Sub Label13_Click()
 RaiseEvent tıkla("Boş RAM", Label14.Caption)
End Sub
Private Sub Label14_Click()
 RaiseEvent tıkla("Boş RAM", Label14.Caption)
End Sub
Private Sub Label2_Click()
 RaiseEvent tıkla("CPU", Label2.Caption)
End Sub
Private Sub Label3_Click()
 RaiseEvent tıkla("CPU Sayısı", Label4.Caption)
End Sub
Private Sub Label4_Click()
 RaiseEvent tıkla("CPU Sayısı", Label4.Caption)
End Sub
Private Sub Label5_Click()
 RaiseEvent tıkla("RAM", Label6.Caption)
End Sub
Private Sub Label6_Click()
 RaiseEvent tıkla("RAM", Label6.Caption)
End Sub
Private Sub Label7_Click()
 RaiseEvent tıkla("Çözün x Renk", Label8.Caption)
End Sub
Private Sub Label8_Click()
 RaiseEvent tıkla("Çözün x Renk", Label8.Caption)
End Sub
Private Sub Label9_Click()
 RaiseEvent tıkla("Aktif Dizin", Label10.Caption)
End Sub
```

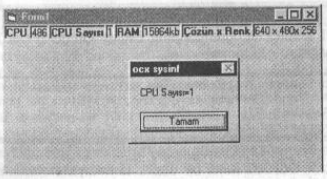
Artık kontrolümüze ait iki tanede event var. Kontrolümüzü tekrar derleyip bir programda kullanırsanız bu olayları kod penceresinde görebilirsiniz.



Aşağıdaki kodlarla yazdığımız olayları test edebilirsiniz.

```
Private Sub UserControl1_değiş( iptal As Boolean)
 If MsgBox("Zemin renkleri değişsinmi",vbyesno,"Değiş")=vbno Then
  iptal = True
 End If
End Sub

Private Sub UserControl1_tıkla(nere As String, değer As Variant)
 MsgBox (nere & "=" & değer)
End Sub
```

Kontroldeki bir etiket tıklandığında tıkla olayı meydana gelecek ve yukarıdaki kodla da tıklanan yerin değeri aşağıdaki gibi bir mesaj kutusu ile bildirilecektir.



ÖRNEK OCX 2

Örnek olarak sistemde bulunan sürücülerle ilgili bilgi verecek bir OCX tasarlayalım. Harfi verilen sürücüye ait şu bilgileri bu OCX'in properties'leri aracılığı ile öğrenebilelim: Sürücü boyutu, boş alanı, Volume ismini, seri numarasını, dosya sistemini, sıkıştırma olup olmadığını. Bütün bu bilgileri öğrenebilmek için iki tane API kullanacağız. GetVolumeInformation ve GetDiskFreeSpaceEx API'leri istedi-

ğimiz bu bilgileri bize verecektir. Bu iki apinin kullanımını API'ler bölümünde bulabilirsiniz.

Örneğimiz için **File-New Project** menüleri ile açılan pencereden **ActiveX Control** seçeneğini tıklayarak yeni bir user control projesi başlatın.

Örneğimizde kullanacağımız API tanımlarını User Controlün General Declaration kısmında aşağıdaki gibi yapalım.

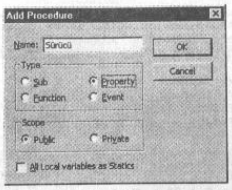
```
Option Explicit
Private Declare Function GetVolumeInformation _
 Lib "kernel32" Alias "GetVolumeInformationA" _
   (ByVal lpRootPathName As String,
 ByVal lpVolumeNameBuffer As String, _
 ByVal nVolumeNameSize As Long, _
 lpVolumeSerialNumber As Long, _
 lpMaximumComponentLength As Long, _
 lpFileSystemFlags As Long, _
 ByVal lpFileSystemNameBuffer As String, _
 ByVal nFileSystemNameSize As Long) As Long
Private Declare Function GetDiskFreeSpaceEx Lib "kernel32" _
 Alias "GetDiskFreeSpaceExA" (ByVal lpRootPathName As String,_
 lpFreeBytesAvailableToCaller As uzunsayi,_
 lpTotalNumberOfBytes As uzunsayi,_
 lpTotalNumberOfFreeBytes As uzunsayi) As Long
Private Type uzunsayi
 l As Long 'Düşük kısım
 h As Long 'Yüksek kısım
End Type
```

Bu iki API aracılığı ile öğreneceğimiz bilgiler için gerekli değişkenleri de aynı bölümde aşağıdaki gibi tanımlayalım.

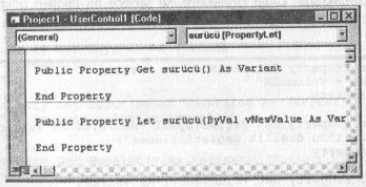
```
Dim sürücüharfi, VolumeName As String, VolumeSN As Long,_
 maxuz As Long, FsName As String, MaxFNLen As Long,_
 flags As Long
Dim topkapasite, bAlan
```

Örneğimizde **Sürücü** isimli bir özellik tanımlayalım. Kullanıcı hakkında bilgi almak istediği sürücünün ana dizinini (c:\, d:\ gibi) bu özelliğe girsin ve bu sürücü hakkındaki bilgiler diğer özellikler aracılığı ile listelensin.

Bu işlem için **Sürücü** isimli bir özellik tanımlayalım. **Tools-Add Procedure** menüleri ile aşağıdaki pencereyi açın.



Penceredeki **Name** kutusuna özelliğin adı olan **Sürücü** ismini yazın ve penceredeki **Property** seçeneğini işaretleyerek bunun bir properties olduğunu belirtin. Böylece kod penceresine iki alt program eklenecektir.



Daha önce anlattığımız gibi bunlardan **Let** ile başlayanı o özelliğe bir değer atandığında, **Get** ile başlayanı ise o özelliğin değeri öğrenilmek istendiğinde çalışır.

Kullanıcı **Sürücü** özelliğine bir sürücü atadığında o sürücüye ait bilgileri göstermek istediğimiz için bu özelliğin **Let** kısmında sürücüyle ilgili bilgileri bulacak kodu yazmamız gerekir. **Get** kısmına ise şu an gösterilen sürücüyü bildirecek kodu yazmamız yeterli olacaktır.

```
Public Property Get Sürücü() As Variant
 Sürücü = sürücüharfi
End Property

Public Property Let Sürücü(ByVal vNewValue As Variant)
 sürücüharfi = vNewValue 'Left(vNewValue, 1)
 Dim r As Long
 Dim pos As Integer
 VolumeName = Space(14)
 FsName = Space(32)
```

```
 r = GetVolumeInformation(sürücüharfi, VolumeName,
Len(VolumeName), VolumeSN&, maxuz, flags, FsName, Len(FsName))
 If r = 0 Then MsgBox ("Geçersiz Sürücü: Sürücüyü C:\ şeklinde
giriniz")
 pos = InStr(VolumeName, Chr(0))
 If pos Then VolumeName = Left(VolumeName, pos - 1)
 pos = InStr(FsName, Chr(0))
 If pos Then FsName = Left(FsName, pos - 1)

 Dim afb As uzunsayi, tb As uzunsayi, fb As uzunsayi
 Call GetDiskFreeSpaceEx(sürücüharfi, afb, tb, fb)
 topkapasite = tb.h * 2 ^ 32 + tb.l
 bAlan = fb.h * 2 ^ 32 + fb.l
End Property
```

Kontrolümüz için gerekli kodun neredeyse tamamı Sürücü özelliğinin Let kısmında yapılmaktadır. Diğer özelliklere ise sadece bu bilgileri gösterecek kodu yazmamız yeterlidir. Diğer özellikleri de aynı şekilde tanımlayarak aşağıdaki kodları yazın.

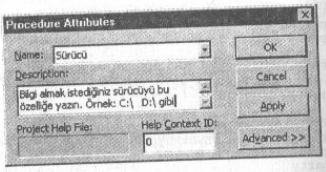
```
Public Property Get SeriNo() As Variant
 SeriNo = VolumeSN
End Property

Public Property Let SeriNo(ByVal vNewValue As Variant)
 MsgBox ("Bu özellik değiştirilemez")
End Property

Public Property Get VolümİsmI() As Variant
 VolümİsmI = VolumeName
End Property

Public Property Let VolümİsmI(ByVal vNewValue As Variant)
 MsgBox ("Bu özellik değiştirilemez")
End Property

Public Property Get DosyaAdıUzunluğu() As Variant
 DosyaAdıUzunluğu = maxuz
End Property

Public Property Let DosyaAdıUzunluğu(ByVal vNewValue As Variant)
 MsgBox ("Bu özellik değiştirilemez")
End Property

Public Property Get DosyaSistemi() As Variant
 DosyaSistemi = FsName
End Property

Public Property Let DosyaSistemi(ByVal vNewValue As Variant)
 MsgBox ("Bu özellik değiştirilemez")
End Property
```



```
Public Property Get Sıkıştırma() As Variant
 Sıkıştırma = "Yok"
 If flags And &H8000 Then Sıkıştırma = "DblSpace veya DrvSpace"
 If flags And &H10 Then Print Sıkıştırma = "Dosya tabanlı"
End Property

Public Property Let Sıkıştırma(ByVal vNewValue As Variant)
 MsgBox ("Bu özellik değiştirilemez")
End Property

Public Property Get BüyükKüçükHarfAyrımı() As Variant
 If flags And 1 Then BüyükKüçükHarfAyrımı = "Var" Else
BüyükKüçükHarfAyrımı = "Yok"
End Property

Public Property Let BüyükKüçükHarfAyrımı(ByVal vNewValue As
Variant)
 MsgBox ("Bu özellik değiştirilemez")
End Property


Public Property Get Boyut() As Variant
 Boyut = topkapasite
End Property

Public Property Let Boyut(ByVal vNewValue As Variant)
 MsgBox ("Bu özellik değiştirilemez")
End Property

Public Property Get BoşAlan() As Variant
 BoşAlan = bAlan
End Property

Public Property Let BoşAlan(ByVal vNewValue As Variant)
 MsgBox ("Bu özellik değiştirilemez")
End Property


Public Property Get Hakkında() As Variant
 Dim msg
 msg = "HdInfo Harddisk Bilgisi Versiyon 1.0 Mayıs 99 İhsan
Karagülle"
 MsgBox (msg)
 Hakkında = msg
End Property

Public Property Let Hakkında(ByVal vNewValue As Variant)

End Property
```

- Böylece gerekli propertiesleri tanımlamış olduk. İsterseniz bütün propertieslere birer açıklama ekleyerek bu bilgilerin Properties penceresinde görülmesini sağlayabilirsiniz. Kod penceresinde **Sürücü** özelliğine ait koda

gelin ve **Tools** menüsünde **Procedure Attributes** komutunu seçin. Böylece bu özelliğe bir açıklama girmemiz için aşağıdaki pencere açılır.



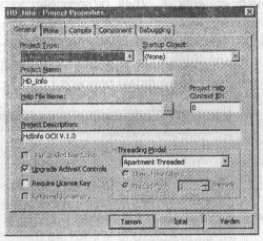
Penceredeki **Decription** kutusuna özelliğe ait açıklamayı yazın. Diğer özellikler için de aynı yöntemle açıklamalar yazabilirsiniz.

- Şimdi kontrolümüzü ToolBox'da temsil edecek bir resim seçelim. Bu işlem için **ToolBoxBitmap** özelliğini çift tıklayın ve açılan pencereden uygun bir bitmap seçin. Common\graphics\bitmaps\offctlbr\small\color dizinindeki Save.bmp dosyasını kullanabilirsiniz. Bu küçük bir disket resmidir.
- Kontrolümüzün bir ekran görüntüsü olmayacak, çünkü gerekli bilgilerin tamamı propertiesler aracılığı ile bildirilecektir. Bu yüzden **ToolBoxBitmap** özelliğine verdiğimiz resmi **Picture** özelliğine de vererek aynı görüntünün form üzerinde de görülmesini sağlayalım.
- Kontrolümüzün ekran görüntüsü olmayacağı için boyutlarının da sabitlenmesi gerekir. Bu işlem için de **Resize** olayına kod yazarak kontrolün boyutları değiştirilmek istendiğinde hep orjinal boyutunda olmasını sağlamamız gerekiyor.

```
Private Sub UserControl_Resize()
 'Buraya Picture özelliğine verdiğiniz resimin boyutlarını girin
 Width = 330
 Height = 330
End Sub
```

- Kontrolümüzün çalışma zamanında ekranda gözükmesi gerekmez. Çünkü gerekli bilgilerin tamamı propertiesler aracılığı ile bildirilecektir. **InvisibleAtRuntimes** özelliğini **False** yaparsanız kontrolümüz çalışma zamanında ekranda gözükmeyecektir.
- Son olarak UserControl'ümüze bir isim verelim. Bunun için Properties penceresindeki **Name** özelliğine bir isim verelim. Örneğimizdeki kontole isim olarak HDInfo ismini verebiliriz. Böylece kullanıcı bu kontrollerden yerleştirdikçe HDInfo1, HDInfo2 gibi isimler otomatik olarak verilecektir.

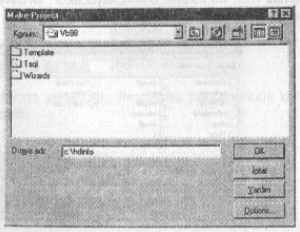
- Şimdi de yaptığımız OCX dosyası için Components penceresinde gösterilecek bir isim verelim. Bu işlem için **Project-Properties** menüleri ile açılan pencerenin **Project_Name** ve **Project Description** kutularına uygun birer açıklama yazın.



Yukarıdaki pencerenin **Description** penceresine yazdığımız açıklama Components penceresinde gösterilecek açıklamadır.

Artık User Control'ümüz hazır. Şimdi File-Save Project menüleri ile projemizi kaydedelim. İsim olarak HDInfo OCX verebilirsiniz.

Şimdi de **File-Make** seçenekleri ile açılan aşağıdaki pencereye oluşturmak istediğimiz OCX dosyasının ismini verelim. Örnek olarak C:\HDInfo ismini verelim.



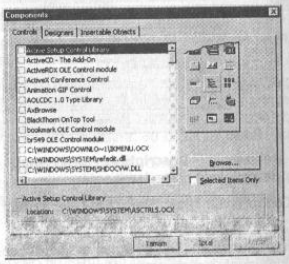
Böylece programımız derlenecek ve C:\ dizinine HDINFO.OCX ismiyle kaydedilecektir.

Bir program içinde kullanma

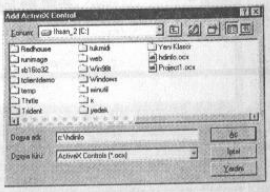
Artık OCX dosyamız hazır. Bunu istediğimiz programa ekleyerek orada kullanabiliriz.

Bu OCX dosyasını kullanacak örnek bir program yapalım. Örneğimiz için File-New Project menüleriyle açılan pencereden Standart Exe seçeneğini tıklayarak normal bir uygulama projesi başlatalım.

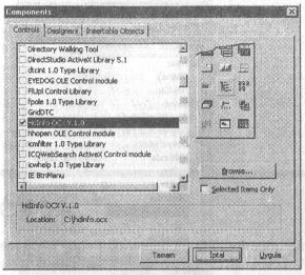
Hazırladığımız OCX dosyasını toolboxa eklememiz gerekiyor. **Project-Components** menüleri ile aşağıdaki pencereyi açın.



Oluşturduğumuz OCX bu listede bulunmayacaktır. **Browse** düğmesine basın ve açılan pencereden HDInfo.ocx dosyasını bulun.

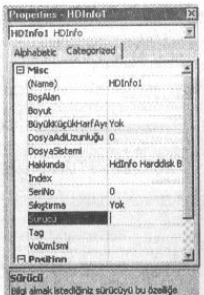


Böylece hazırladığımız OCX dosyası ona verdiğimiz Description açıklamasıyla birlikte aşağıdaki listede yerini alacaktır.

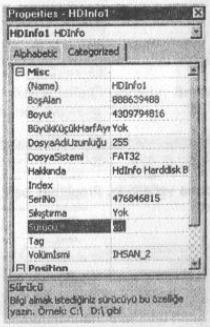


Pencereden HDInfo OCX V.1.0 seçeneğini işaretleyin ve pencereyi kapatın. Artık ToolBox penceresinde bize ait kontrol görülecektir.

Kontrolümüzden forma yerleştirin. Properties penceresinde kontrolümüze ait özellikleri göreceksiniz.



Sürücü özelliğine bir sürücü ismi yazarsanız (C:\ gibi) o sürücüye ait bilgiler properties penceresindeki özellikler aracılığıyla listelenir.



Diğer kontrollerde olduğu gibi bizi kontrolümüzün de özelliklerini program aracılığı ile de öğrenebiliriz.


```
Private Sub Form_Load()
 Dim s
 s = InputBox("Sürücü:", "Sürücü girişi", "C:\")
 HDInfo1.Sürücü = s
 Show
 Print "Kapasite:"; HDInfo1.Boyut / 1024 / 1024; " MB"
 Print "Boş alan:"; HDInfo1.BoşAlan / 1024 / 1024; " MB"
 Print "Volüme ismi:"; HDInfo1.VolümİsmI
 Print "Seri No:"; Hex(HDInfo1.SeriNo)
 Print "Sıkıştırma:"; HDInfo1.Sıkıştırma
 Print "Fat:"; HDInfo1.DosyaSistemi
End Sub
```

ÖRNEK OCX 3

Örnek olarak girilen sayıyı yazıya çevirecek bir OCX yazalım. Sayı özelliğine bir sayı girildiğinde bunu yazı ile ifade etsin. Örneğin 261320051 girildiğinde bunu yazıya çevirsin ve karşılık olarak ikiyüzaltmışbirmilyon üçyüzyirmibin ellibir değerini versin.

Örneğimiz için bir ActiveX Control projesi başlatın ve aşağıdaki kodları yazın.

```
Option Explicit
'TURKSAYI.OCX İhsan Karagülle, Haziran 1999

Dim yazi, sayi
Function yaziya_cevir(s) As String
 Dim sayi As String
 sayi = s
 Dim max_basamak_say
 max_basamak_say = 18 '18 basamaklı yani katrilyon, daha da
artırılabilir
 ReDim birler(10), onlar(10), binler(max_basamak_say / 3), bas(3)
 birler = Array("", "bir", "iki", "üç", "dört", "beş", "altı",
"yedi", "sekiz", "dokuz")
 onlar = Array("", "on", "yirmi", "otuz", "kırk", "elli",
"altmış", "yetmiş", "seksen", "doksan")
 'Eğer 18 basamaktan daha fazla kullanacaksanız katrilyondan önce
o basamakları da ekleyin
 binler = Array("katrilyon", "trilyon", "milyar", "milyon",
"bin", "")
 Dim i, uz, sonuc, arasonuc
 uz = Len(sayi)

 'sayının kullanılmayan basamaklarını sıfırla doldur
 sayi = String(max_basamak_say - uz, "0") + sayi

 For i = 0 To max_basamak_say / 3 - 1
  'sayıyı üçerli basamaklar halinde ele al
```

```
    bas(1) = Mid(sayi, (i * 3) + 1, 1) 'üçlü basamaktaki birinci
sayı yani yüzler basamağı
    bas(2) = Mid(sayi, (i * 3) + 2, 1) 'üçlü basamaktaki ikinci
sayı yani onlar basamağı
    bas(3) = Mid(sayi, (i * 3) + 3, 1) 'üçlü basamaktaki üçüncü
sayı yani birler basamağı
    If bas(1) = 0 Then
     arasonuc = "" 'yüzler basamağı boş
    Else
     If bas(1) = 1 Then
      arasonuc = "yüz" 'yüzler basamağında 1 varsa 1 yüz olmaz
sadece yüz
     Else
      arasonuc = birler(bas(1)) + "yüz" 'yüzler basamağındaki sayı
ve yüz ikiyüz gibi
     End If
    End If
    'yüzler+onlar+birler basamağını birleştir
    arasonuc = arasonuc + onlar(bas(2)) + birler(bas(3))
    'basamak değeri oluşmadıysa yani 000 şeklindeyse binler
basamağını ekle
    If arasonuc <> "" Then arasonuc = arasonuc + binler(i)
    'birbin olmaz
    If (i > 1) And (arasonuc = "birbin") Then arasonuc = "bin"
    sonuc = sonuc + arasonuc + " "
  Next
  If sonuc = "" Then sonuc = "sıfır"
  yaziya_cevir = sonuc
 End Function

 Public Property Get sayı() As Variant
  sayı = sayi
 End Property

 Public Property Let sayı(ByVal vNewValue As Variant)
  yazi = yaziya_cevir(vNewValue)
  sayi = vNewValue
 End Property

 Public Property Get yazı() As Variant
  yazı = yazi
 End Property

 Public Property Let yazı(ByVal vNewValue As Variant)
  yazi = yazi
 End Property

 Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
  yazi = PropBag.ReadProperty("Yazı", yazi)
  sayi = PropBag.ReadProperty("Sayı", sayi)
 End Sub

 Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
```



```
 Call PropBag.WriteProperty("Sayı", sayi)
 Call PropBag.WriteProperty("Yazı", yazi)
End Sub
```

Yukarıdaki kodu yazdıktan sonra UserControl'un properties penceresine geçin ve **Name** özelliğine Turksayi yazın. Ayrıca kontrolün çalışma zamanı görüntüsü olmayacağı için **InVisibleAtRuntime** özelliğini de **True** yapın.

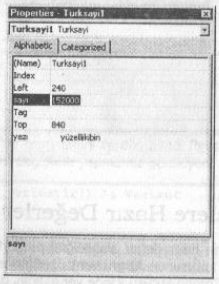
Project-Properties menüleri ile açılan pencerenin **Project Name** kutusuna TurkSayıÇevir yazın.

File-Make menüleri ile açılan pencereye \TURKSAYI.OCX yazarak projeyi ana dizine turksayi.ocx ismiyle derlenmesini sağlayın ve projenizi kaydedin.

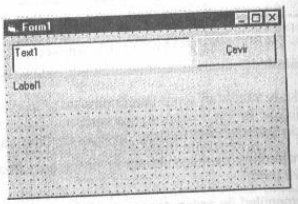
Program İçinde Kullanma

Hazırladığımız bu OCX kontrolünü bir program içinde kullanalım. Yeni bir EXE projesi başlatın ve **Project-Components** menüleri ile açılan penceredeki **Browse** düğmesine basarak \TURKSAYI.OCX dosyasını bulun ve projenize ekleyin.

Hazırladığımız kontrole ait simge ToolBox penceresine gelmiş olması lazım. Bu kontrolden bir tane form üzerine yerleştirin. Properties penceresinde buna ait özellikler listelenecektir.



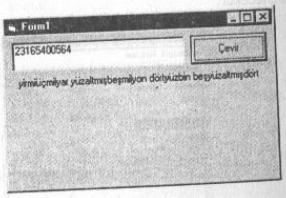
Şimdi formumuza bir Text kutusu, bir Label ve bir komut düğmesi yerleştirerek hazırladığımız kontrolü test edelim.



Text kutusuna yazılan sayıyı hazırladığımız kontrol aracılığı ile yazıya çevirip Label içinde gösterecek kod aşağıdaki gibi komut düğmesine yazalım.

```
Private Sub Command1_Click()
 Turksayi1.sayı = Text1
 Label1 = Turksayi1.yazı
End Sub
```

Programı çalıştırırsanız Text1 kutusu içine yazdığınız sayı Çevir düğmesi aracılığı ile yazıya çevrilecek ve Label1 içinde gösterilecektir.



Propertieslere Hazır Değerler Verme

Bazı propertieslerin hazır değer alabildiğini properties penceresinde görebilirsiniz. Örneğin Text kutusunun **Alignment** özelliğini aşağı doğru açarsanız üç tane hazır değerin bir Combo kutusu içinde yer aldığını görürsünüz.

Peki siz de kendi kontrolünüzün bazı özelliklerine bu tür hazır değerler verebilir misiniz? Evet. Eğer o özelliğin içeriği sayısal bir değer içeriyorsa yani 0, 1, 2 gibi sayısal bir değer içerebiliyorsa bunu yapabilirsiniz. Aynı şekilde bunlara birer sabit atanarak hem sayı hem de metin şeklinde kullanılması sağlanabilir. (0-Left Justify, 1- Right Justifiy gibi)

Bu işlemi yapabilmek için önce **Enum** tipinden bir tip tanımlamanız ve olası değerleri burada tanımlamanız gerekir. Örneğin bir özelliğe 0-Sola, 1-Sağa, 2-Ortaya değerlerini vermek istiyorsak aşağıdaki gibi bir Enum tip içerisinde bu değerleri tanımlamamız gerekir.

```
Public Enum Yerlesim
 Sola=0
 Sağa=1
 Ortaya=2
End Enum
```

Tanımladığımız bu tipe Yerlesim ismini verdik. Şimdi **Property Let** ve **Property Get** olaylarında küçük bir değişiklik yapmamız gerekiyor.

```
Public Property Get yerlestir() As Variant

End Property

Public Property Let yerlestir(ByVal vNewValue As Variant)

End Property
```

Normalde bu iki olaydaki giriş parametreler **Variant** olarak tanımlanmıştır. Bunları değiştirerek tanımladığımız **Enum** tipini ismini vermemiz gerekir. (Örnekte Yerlesim ismini vermiştik)

```
Public Property Get yerlestir() As Yerlesim

End Property

Public Property Let yerlestir(ByVal vNewValue As Yerlesim)

End Property
```

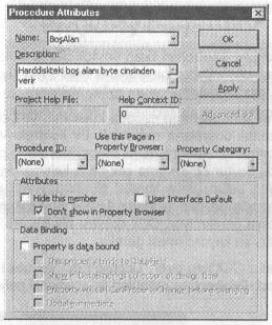
Bu işlemlerden sonra özellik için gerekli kodları yazıp derledikten sonra bu kontrolü bir programda kullandığınızda, özellik penceresinde **Yerlestir** özelliğinin yanında bir açılan kutu bulunacak ve bu kutuda verdiğimiz değerler yer alacaktır.

Özellikleri RunTime, DesignTime olarak tanımlama

Bazı özellikler vardır ki sadece tasarım anında değiştirilebilir, program çalışırken değiştirilemez**(DesignTime),** bazı özellikler ise tasarım zamanında değiştirilemez ve properties penceresinde yer almaz**(P.unTime).** User Control'ünüz için tanımladığınız propertieslere bu özelliklerden vermek isteyebilirsiniz.

RunTime Özelliği Verme

Bir özelliğin sadece program çalışırken değiştirilebilmesini istiyorsanız kod penceresinde bu özelliğin yazılı olduğu kısma gelin ve **Tools** menüsünde **Procedure Attributes** komutunu seçin. Aşağıdaki pencere açılacaktır. Bu pencerdeki **Advanced** düğmesine basarak diğer seçeneklerin de gösterilmesini sağlayın.



Penceredeki **Don't show in Property Browser** seçeneğini işaretlerseniz bu özellik **Properties** penceresinde listelenmeyecek ve dolayısıyla da tasarım anında değiştirilemeyecektir.

DesignTime Özelliği Verme

Eğer bir properties'in sadece tasarım anında değiştirilebilmesini, program çalışırken değiştirilememesini istiyorsanız **Ambient.UserMode** ile o anki durumu öğrenmelisiniz. Eğer program çalışma modunda ise **Ambient.UserMode** özelliğinden geriye **true** değeri dönecektir. Bunu **Property Let** olayına yazacağınız kodla kontrol ederek çalışma moduna girildiğinde özelliğin değerinin değiştirilememesini sağlayabilir ve bu işlem için kullanılan 382 nolu hatayı aktif hale getirebilirsiniz.

Örneğin **Biçim** isimli bir özelliğimiz olduğunu kabul edelim. Bunun **Propery Let** kodunu aşağıdaki gibi değiştirerek **DesignTime** özellik olmasını sağlayabiliriz.

```
Property Let Biçim() As Variant
 If Ambient.UserMode Then 'Çalışma modunda ise hata ver
  Err.Raise Number:=382 Description:= "Çalışma zamanı değişmez"
 End If
 ..............
End Property
```


Lisans Anahtarı Ekleme

Eğer hazırladığınız kontrolleri ticari amaçlı olarak kullanacaksanız, yani bu kontrolleri satmayı düşünüyorsanız bunlara birer lisans key ekleterek izinsiz kopyalanması durumunda çalışmamasını sağlayabilirsiniz.

Visual Basic'in kontroller için eskiden kullandığı VBX dosyalarında bu tür bir koruma LIC uzantılı dosyalarla sağlanırdı. Bir kontrolü VB içinde kullanmak istediğinizde, o kontrol LIC uzantılı kendi lisans dosyasının olup olmadığına bakar yoksa çalışmazdı. Bu tür bir koruma çok basit kalıyordu ve kullanıcı o lisans dosyasını bir yerden bulup (veya kendisi oluşturup) VBX dosyasını kullanabiliyordu. OCX dosyalarında ise bunun yerine Lisans Key sistemi geliştirildi. Bu durumda ayrıca bir dosya gerekmiyor. OCX dosyası bilgisayara kaydedildiğinde bu key ile birlikte registry'ye kaydedilmesi gerekir.

Oluşturduğunuz OCX'lere birer "License Key" vermek için OCX projenizi VB ile açın ve **Project-Properties** menüleri ile aşağıdaki pencereye gelin.

Penceredeki **Require License Key** seçeneğini tıklayın. Bunun aktif hale gelmesi için de **File-Make** menüleri ile OCX dosyanızı yeniden derleyin. Böylece OCX dosyanız normal yollarla başka bir bilgisayara kopyalandığında kullanılamayacaktır.

Peki bunun, programı satın almış kişiler tarafından kullanılabilmesi için nasıl bir yol izlemek gerekecektir. Bu işlemi **Package & Deployment Wizard** programı yerine getirir. Eğer programınızın kurulum programını bu programa yaptırırsanız, kurulduğu yere gerekli lisans kaydını yapacaktır. Böylece ancak kurulum programı ile kurulmuş yerlerde OCX dosyanız çalışacak, kopyalama yöntemiyle başka bir bilgisayardan alındığında ise çalışmayacaktır.

Eğer yaptığınız OCX dosyasını satmayı düşünmüyorsanız (freeware veya shareware türünden olacaksa) lisans anahtarı eklemeniz gerekmez. Çünkü bu programınızın yaygınlaşmasını önleyecektir.

Property Page

VB'de, propertiesleri Properties penceresinden tek tek değiştirebileceğiniz gibi hazırlayacağınız pencerelerle topluca değiştirilebilmesini de sağlayabilirsiniz. Bu görsel bir kolaylık sağlayacağı gibi kullanıcıya daha kullanışlı bir arabirim de sunacaktır.

Yukarıda bir OCX dosyası nasıl oluşturabileceğimizi ve kendi kontrollerimizi nasıl yapabileceğimizi gördük. Hazırladığımız kontrole ait özellikleri kullanıcı properties penceresinden değiştirebilmekteydi. Ancak istersek bu özelliklerin bir pencere aracılığı ile değiştirilebilmesini de sağlayabiliriz. Property Page ile bu iş yapılabilmektedir.

Properties

Changed

Properties sayfasında bir özellik değiştiğinde bunun uygulanabilmesi için Changed özelliğine True değerini aktarmak gerekir.

SelectedControls

Kullanıcı form üzerine bir veya daha fazla aynı user kontrolü yerleştirip bunlara ait özellikleri ayrı ayrı veya topluca değiştirmek isteyebileceği için hangi kontrolün seçildiği o kontrolün ismi ile değil SelectedControls özelliği ile öğrenilir.

SelectedControls.Count ile seçili olan kontrollerin sayısını

SelectedControls.(i).PropertiesAdı ile de seçili olan i nolu kontrolün ilgili properties değeri öğrenilip değiştirilebilir.

```
Private Sub PropertyPage_SelectionChanged()
  dim i
 For i=0 to SelectedControls.Count-1
    x = SelectedControls(i).propertiesadı
   Next
End Sub
```

şeklinde bir kod ile seçili kontrollerin ilgili propertiesleri öğrenilip değiştirilebilir.

Events

ApplyChanges( )

Kullanıcı property sayfasındaki *Uygula* veya *Tamam* düğmesini tıklarsa ve sayfada bir özellik değişmişse bu olay meydana gelir. Sayfadaki değişimlerin etkili olması içinde Changed özelliğine True atanarak bunun bildirilmiş olması gereklidir.

```
Private Sub PropertyPage_ApplyChanges()
 SelectedControls(0).PropertiesAdı= Değer
End Sub
```

şeklinde bir kodla yapılan değişiklikleri kontrole uygulayabilirsiniz.

SelectionChanged( )

Property penceresinde birden fazla property sayfası bulunabilir. Kullanıcı bu sayfalar arasında geçiş yaparken ve property penceresi açılırken ilk sayfa için bu olay meydana gelir. Dolayısıyla ilk sayfa açılırken veya yeni sayfaya geçerken o sayfadaki özelliklerin gösterilmesi için bu olay kullanılır.

```
Private Sub PropertyPage_SelectionChanged()
 Text1 = SelectedControls(0).PropertiesAdı
End Sub
```

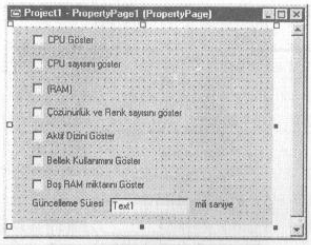
şeklinde bir kod ile ilgili propertieslerin değerleri ilgili kontrollerde gösterilebilir.

EditProperty(PropertyName As String)

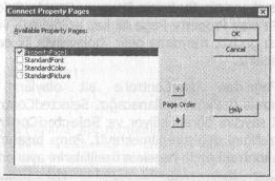
Kullanıcı properties penceresini hangi özellikten aktif hale getirmişse bu olaydaki PropertyName parametresi ile o özelliğin ismi öğrenilebilir. Bu olay, sayfadaki uygun kontrole focus vermek için faydalı olabilir.

ÖRNEK:

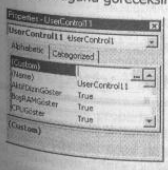
Yukarıda hazırladığımız (birinci örnekteki) User kontrolümüze bir property page ekleyerek kullanıcının özellikleri toplu olarak değiştirmesini sağlayabiliriz. O örneğimizi açın ve **Project** menüsündeki **Add Property Page** seçeneği ile projemize boş bir property page ekleyin ve aşağıdaki kontrolleri oluşturun.



Kontrolümüzün bu property sayfasını kullanması içinde oluşturduğumuz UserControl1'i seçin ve PropertyPages özelliğini çift tıklayarak bu oluşturduğumuz sayfayı aşağıdaki pencereden işaretleyin.

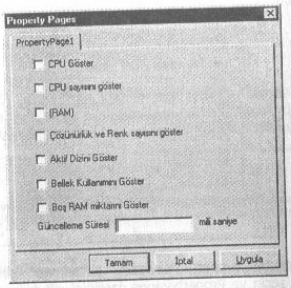


Projemizi bu haliyle derlerseniz kontrolümüzde (Custom) isimli bir özelliğin daha bulunduğunu göreceksiniz.



Bu özelliği çift tıkladığınızda ise oluşturduğumuz aşağıdaki pencerenin çıktığını göreceksiniz.

Henüz hiç bir kod yazmadığımız için burada yapacağımız işlemler herhangi bir özelliğin değişmesine sebep olmayacaktır.



Tekrar projemize dönelim ve Property Page için gerekli kodları yazalım. Öncelikle property sayfası açıldığında şu anki özelliklerin değerlerini pencerede göstermemiz gerekecektir. Bu iş için Property Page'in SelectionChanged olayını kullanacağız. Bu olay Property Page ilk kez açıldığında ve birden fazla sayfa varsa sayfalar arası geçişte meydana gelir. Dolayısıyla ilk değerleri göstermek için bu olayı kullanabiliriz.

Property Page'den UserControl'e ait olayları kullanabilmek için ise SelectedControls özelliğini kullanacağız. SelectedControls.Count özelliği ile seçili olan kontrol sayısını öğrenebiliyor ve **SelectedControls(i)** ile de seçili i nolu kontrolün özelliğini değiştirebilmekteyiz. Form tasarımını düşünürseniz kullanıcı birden fazla kontrolü seçip hepsinin özelliklerini aynı anda değiştirebilmektedir.

```
Private Sub PropertyPage_SelectionChanged()
 Check1.Value = Abs(SelectedControls(0).CPUGöster)
 Check2.Value = Abs(SelectedControls(0).CPUSayısıGöster)
 Check3.Value = Abs(SelectedControls(0).RAMGöster)
 Check4.Value = Abs(SelectedControls(0).ÇözünxRenkGöster)
 Check5.Value = Abs(SelectedControls(0).AktifDizinGöster)
 Check6.Value = Abs(SelectedControls(0).KulBellekGöster)
 Check7.Value = Abs(SelectedControls(0).BoşRAMGöster)
 Text1 = SelectedControls(0).GüncellemeSüresi
End Sub
```

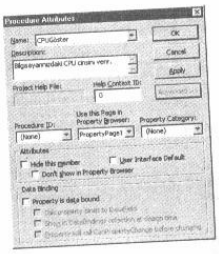
Bu işlem property sayfası açılırken özelliklerin doğru olarak gösterilmesini sağlayacaktır. Kullanıcı bu özellikleri değiştirdiğinde bunları kontrole uygulamak için ise ApplyChanges olayını kullanacağız. Bu olay, kullanıcı yaptığı değişiklikleri Tamam düğmesi ile iptal etmemişse ve bir değişiklik yapılmışsa meydana gelmektedir.

Dolayısıyla özellikler değiştiğinde Changed=True yaparak değişim yapıldığını bildirmemiz gerekiyor.

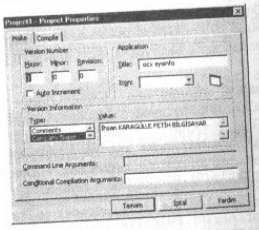
```
Private Sub Check1_Click()
 Changed = True
End Sub

Private Sub Check2_Click()
 Changed = True
End Sub

Private Sub Check3_Click()
 Changed = True
End Sub

Private Sub Check4_Click()
 Changed = True
End Sub

Private Sub Check5_Click()
 Changed = True
End Sub

Private Sub Check6_Click()
 Changed = True
End Sub

Private Sub Check7_Click()
 Changed = True
End Sub

Private Sub PropertyPage_ApplyChanges()
 SelectedControls(0).CPUGöster = Check1.Value
 SelectedControls(0).CPUSayısıGöster = Check2.Value
 SelectedControls(0).RAMGöster = Check3.Value
 SelectedControls(0).ÇözünxRenkGöster = Check4.Value
 SelectedControls(0).AktifDizinGöster = Check5.Value
 SelectedControls(0).KulBellekGöster = Check6.Value
 SelectedControls(0).BoşRAMGöster = Check7.Value
 SelectedControls(0).GüncellemeSüresi = Text1
End Sub
```

Property Page'i User Control içinde Kullanma

Kullanıcının properties penceresindeki her bir özellikle property sayfasını kullanabilmesini de sağlayabiliriz. Bunun için UserControl için yazdığımız her bir Property için Tools menüsündeki Procedure Attributes seçeneği ile açılan aşağıdaki pencereden **Use this Page in Property Browser** listesinden PropertyPage1 sayfasını seçmeniz gerekir. Böylece kullanıcı o özelliklerin yanındaki ... düğmesini kullanarak Property sayfasına ulaşabilecektir.



CPUGöster | True | ... Artık OCX dosyamız kendinden beklenen birçok modern özelliği destekliyor. Son olarak bunu derlemeden önce versiyon bilgilerinizi de belirleyerek ticari bir sürüm haline getirebilirsiniz.



Sonuç olarak projemizde bulunacak bütün kodlar aşağıdaki gibi olacaktır.

```
'ÖRNEK: UserControl,Property Page, Properties, Event, Methods
'Bu kısım Modüle içine yazılacak
Option Explicit
Declare Function GetDeviceCaps Lib "GDI32" (ByVal hDC%, ByVal
nIndex%) As Integer
Type MEMORYSTATUS
 dwLength As Long
 dwMemoryLoad As Long
 dwTotalPhys As Long
 dwAvailPhys As Long
 dwTotalPageFile As Long
 dwAvailPageFile As Long
```



```
 dwTotalVirtual As Long
 dwAvailVirtual As Long
End Type

Declare Sub GlobalMemoryStatus Lib "kernel32" (lpBuffer As
MEMORYSTATUS)
Type SYSTEM_INFO
 dwOemID As Long
 dwPageSize As Long
 lpMinimumApplicationAddress As Long
 lpMaximumApplicationAddress As Long
 dwActiveProcessorMask As Long
 dwNumberOfProcessors As Long
 dwProcessorType As Long
 dwAllocationGranularity As Long
 dwReserved As Long
End Type
Declare Sub GetSystemInfo Lib "kernel32" (lpSystemInfo As
SYSTEM_INFO)
Declare Function GetDiskFreeSpace Lib "kernel32" (ByVal
lpRootPathname As String, lpSectorsPerCluster As Long,
lpBytesPerSector As Long, lpNumberOfFreeClusters As Long,
lpTtoalNumberOfClusters As Long) As Long

'Bu kısım UserControl'e yazılacak

Public Event tıkla(nere As String, değer As Variant)
Public Event değiş(iptal As Boolean)


Private Sub Label1_Click()
 RaiseEvent tıkla("CPU", Label2.Caption)
End Sub

Private Sub Label10_Click()
 RaiseEvent tıkla("Aktif Dizin", Label10.Caption)
End Sub

Private Sub Label11_Click()
 RaiseEvent tıkla("Bellek Kullanımı", Label12.Caption)
End Sub

Private Sub Label12_Click()
 RaiseEvent tıkla("Bellek Kullanımı", Label12.Caption)
End Sub

Private Sub Label13_Click()
 RaiseEvent tıkla("Boş RAM", Label14.Caption)
End Sub

Private Sub Label14_Click()
 RaiseEvent tıkla("Boş RAM", Label14.Caption)
End Sub
```



```
Private Sub Label2_Click()
 RaiseEvent tıkla("CPU", Label2.Caption)
End Sub

Private Sub Label3_Click()
 RaiseEvent tıkla("CPU Sayısı", Label4.Caption)
End Sub

Private Sub Label4_Click()
 RaiseEvent tıkla("CPU Sayısı", Label4.Caption)
End Sub

Private Sub Label5_Click()
 RaiseEvent tıkla("RAM", Label6.Caption)
End Sub

Private Sub Label6_Click()
 RaiseEvent tıkla("RAM", Label6.Caption)
End Sub

Private Sub Label7_Click()
 RaiseEvent tıkla("Çözün x Renk", Label8.Caption)
End Sub

Private Sub Label8_Click()
 RaiseEvent tıkla("Çözün x Renk", Label8.Caption)
End Sub

Private Sub Label9_Click()
 RaiseEvent tıkla("Aktif Dizin", Label10.Caption)
End Sub

Private Sub Timer1_Timer()
 Dim s As SYSTEM_INFO
 GetSystemInfo s
 Label4 = s.dwNumberOfProcessors
 Dim m As MEMORYSTATUS
 GlobalMemoryStatus m
 Label12 = "%" & m.dwMemoryLoad
 Label10 = CurDir
 diz
End Sub

Private Sub UserControl_DblClick()
 Dim x As Boolean, i
 RaiseEvent değiş(x)
 On Local Error Resume Next
 If Not x Then 'iptal edilmedi ise
  For i = 0 To Controls.Count - 1
   Controls(i).BackColor = QBColor(Rnd * 15)
  Next
 End If
End Sub
```



```
Private Sub UserControl_Initialize()
 Const PLANES = 14
 Const BITSPIXEL = 12
 Const colorres = 108
 Label8 = Screen.Width \ Screen.TwipsPerPixelX & " x " &
Screen.Height \ Screen.TwipsPerPixelY & "x " & GetDeviceCaps(hDC,
PLANES) * 2 ^ GetDeviceCaps(hDC, BITSPIXEL)
 Dim m As MEMORYSTATUS
 GlobalMemoryStatus m
 Label6 = Int(m.dwTotalPhys / 1024) & "kb"
 Label12 = "%" & m.dwMemoryLoad
 Label14 = Int(m.dwAvailPhys / 1024) & "kb"
 Dim s As SYSTEM_INFO
 GetSystemInfo s
 Label4 = s.dwNumberOfProcessors
 Label2 = s.dwProcessorType
 diz
End Sub

Public Sub diz()
 Label1.Top = 0
 Label2.Top = 0
 Label3.Top = 0
 Label4.Top = 0
 Label5.Top = 0
 Label6.Top = 0
 Label7.Top = 0
 Label8.Top = 0
 Label9.Top = 0
 Label10.Top = 0
 Label11.Top = 0
 Label12.Top = 0
 Label13.Top = 0
 Label14.Top = 0
 Dim l
 If Label1.Visible Then
  Label1.Left = l
  Label2.Left = Label1.Left + Label1.Width
  l = Label2.Left + Label2.Width
  l = l + 15
 End If
 If Label3.Visible Then
  Label3.Left = l
  Label4.Left = Label3.Left + Label3.Width
  l = Label4.Left + Label4.Width
  l = l + 15
 End If
 If Label5.Visible Then
  Label5.Left = l
  Label6.Left = Label5.Left + Label5.Width
  l = Label6.Left + Label6.Width
  l = l + 15
 End If
```

```
If Label7.Visible Then
 Label7.Left = l
 Label8.Left = Label7.Left + Label7.Width
 l = Label8.Left + Label8.Width
 l = l + 15
End If
If Label9.Visible Then
 Label9.Left = l
 Label10.Left = Label9.Left + Label9.Width
 l = Label10.Left + Label10.Width
 l = l + 15
End If
If Label11.Visible Then
 Label11.Left = l
 Label12.Left = Label11.Left + Label11.Width
 l = Label12.Left + Label12.Width
 l = l + 15
End If
If Label13.Visible Then
 Label13.Left = l
 Label14.Left = Label13.Left + Label13.Width
 l = Label14.Left + Label14.Width
 l = l + 15
End If
End Sub

Public Property Get CPUGöster() As Variant
 CPUGöster = Label1.Visible
End Property

Public Property Let CPUGöster(ByVal vNewValue As Variant)
  Label1.Visible = vNewValue
  Label2.Visible = vNewValue
  diz
  PropertyChanged "CPUGöster"
End Property

Public Property Get CPUSayısıGöster() As Variant
 CPUSayısıGöster = Label3.Visible
End Property

Public Property Let CPUSayısıGöster(ByVal vNewValue As Variant)
  Label3.Visible = vNewValue
  Label4.Visible = vNewValue
  diz
  PropertyChanged "CPUSayısıGöster"
End Property

Public Property Get RAMGöster() As Variant
 RAMGöster = Label5.Visible
End Property
```



```
Public Property Let RAMGöster(ByVal vNewValue As Variant)
  Label5.Visible = vNewValue
  Label6.Visible = vNewValue
  diz
  PropertyChanged "RAMGöster"
End Property

Public Property Get ÇözünxRenkGöster() As Variant
 ÇözünxRenkGöster = Label7.Visible
End Property

Public Property Let ÇözünxRenkGöster(ByVal vNewValue As Variant)
  Label7.Visible = vNewValue
  Label8.Visible = vNewValue
  diz
  PropertyChanged "ÇözünxRenkGöster"
End Property

Public Property Get AktifDizinGöster() As Variant
 AktifDizinGöster = Label9.Visible
End Property

Public Property Let AktifDizinGöster(ByVal vNewValue As Variant)
  Label9.Visible = vNewValue
  Label10.Visible = vNewValue
  diz
  PropertyChanged "AktifDizinGöster"
End Property

Public Property Get KulBellekGöster() As Variant
 KulBellekGöster = Label11.Visible
End Property

Public Property Let KulBellekGöster(ByVal vNewValue As Variant)
  Label11.Visible = vNewValue
  Label12.Visible = vNewValue
  diz
  PropertyChanged "KulBellekGöster"
End Property

Public Property Get BoşRAMGöster() As Variant
 BoşRAMGöster = Label13.Visible
End Property

Public Property Let BoşRAMGöster(ByVal vNewValue As Variant)
  Label13.Visible = vNewValue
  Label14.Visible = vNewValue
  diz
  PropertyChanged "BoşRAMGöster"
End Property

Public Property Get GüncellemeSüresi() As Variant
 GüncellemeSüresi = Timer1.Interval
```



```
End Property

Public Property Let GüncellemeSüresi(ByVal vNewValue As Variant)
 Timer1.Interval = vNewValue
 PropertyChanged "GüncellemeSüresi"
End Property

Public Property Get Hakkında() As Variant
 MsgBox ("SysInfoOcx32 V.1.0 İhsan KARAGÜLLE Mart 1997
Ahlat/BİTLİS")
End Property

Public Property Let Hakkında(ByVal vNewValue As Variant)
 'Koda gerek yok
End Property

Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
 Label1.Visible = PropBag.ReadProperty("CPUGöster", True)
 Label2.Visible = Label1.Visible
 Label3.Visible = PropBag.ReadProperty("CPUSayısıGöster", True)
 Label4.Visible = Label3.Visible
 Label5.Visible = PropBag.ReadProperty("RAMGöster", True)
 Label6.Visible = Label5.Visible
 Label7.Visible = PropBag.ReadProperty("ÇözünxRenkGöster", True)
 Label8.Visible = Label7.Visible
 Label9.Visible = PropBag.ReadProperty("AktifDizinGöster", True)
 Label10.Visible = Label9.Visible
 Label11.Visible = PropBag.ReadProperty("KulBellekGöster", True)
 Label12.Visible = Label11.Visible
 Label13.Visible = PropBag.ReadProperty("BoşRAMGöster", True)
 Label14.Visible = Label13.Visible
 Timer1.Interval = PropBag.ReadProperty("GüncellemeSüresi", 100)
 diz
End Sub

Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
 Call PropBag.WriteProperty("CPUGöster", Label1.Visible, True)
 Call PropBag.WriteProperty("CPUSayısıGöster", Label3.Visible,
True)
 Call PropBag.WriteProperty("RAMGöster", Label5.Visible, True)
 Call PropBag.WriteProperty("ÇözünxRenkGöster", Label7.Visible,
True)
 Call PropBag.WriteProperty("AktifDizinGöster", Label9.Visible,
True)
 Call PropBag.WriteProperty("KulBellekGöster", Label11.Visible,
True)
 Call PropBag.WriteProperty("BoşRAMGöster", Label13.Visible,
True)
 Call PropBag.WriteProperty("GüncellemeSüresi", Timer1.Interval,
100)
 diz
End Sub
```

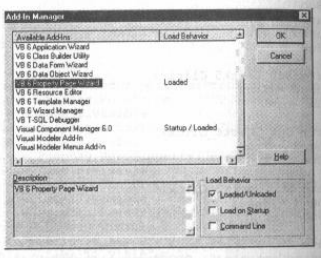


```
'Bu kısım Property Page1'e yazılacak
Option Explicit
Private Sub Check1_Click()
 Changed = True
End Sub
Private Sub Check2_Click()
 Changed = True
End Sub
Private Sub Check3_Click()
 Changed = True
End Sub
Private Sub Check4_Click()
 Changed = True
End Sub

Private Sub Check5_Click()
 Changed = True
End Sub

Private Sub Check6_Click()
 Changed = True
End Sub

Private Sub Check7_Click()
 Changed = True
End Sub

Private Sub PropertyPage_ApplyChanges()
 SelectedControls(0).CPUGöster = Check1.Value
 SelectedControls(0).CPUSayısıGöster = Check2.Value
 SelectedControls(0).RAMGöster = Check3.Value
 SelectedControls(0).ÇözünxRenkGöster = Check4.Value
 SelectedControls(0).AktifDizinGöster = Check5.Value
 SelectedControls(0).KulBellekGöster = Check6.Value
 SelectedControls(0).BoşRAMGöster = Check7.Value
 SelectedControls(0).GüncellemeSüresi = Text1
End Sub

Private Sub PropertyPage_SelectionChanged()
 Check1.Value = Abs(SelectedControls(0).CPUGöster)
 Check2.Value = Abs(SelectedControls(0).CPUSayısıGöster)
 Check3.Value = Abs(SelectedControls(0).RAMGöster)
 Check4.Value = Abs(SelectedControls(0).ÇözünxRenkGöster)
 Check5.Value = Abs(SelectedControls(0).AktifDizinGöster)
 Check6.Value = Abs(SelectedControls(0).KulBellekGöster)
 Check7.Value = Abs(SelectedControls(0).BoşRAMGöster)
 Text1 = SelectedControls(0).GüncellemeSüresi
End Sub

Private Sub Text1_Change()
 Changed = True
End Sub
```

Wizard Kullanarak Property Page Oluşturma

VB içinde bulunan Wizard property page için gerekli işlemleri kolayca yapmanızı sağlayacaktır. UserControl'ü tasarladığınız projenizi açın ve **Add-Ins** menüsünden **Property Page Wizard** komutunu seçin.

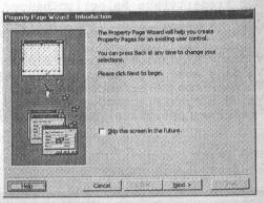
Eğer **Add-Ins** menüsünde bu seçeneği göremiyorsanız aynı menüdeki **Add-In Manager** komutunu kullanarak aşağıdaki pencereyi açın.



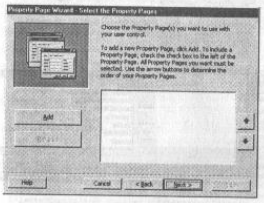
Bu penceredeki **Vb 6 Property Page Wizard** seçeneğini çift tıklayın. Yanında **Loaded** yazacaktır. Eğer her seferinde yüklenmesini istiyorsanız penceredeki **Load on Startup** seçeneğini de işaretleyin.

Bu işlemden sonra **Add-Ins** menüsünde **Property Page Wizard** seçeneği bulunacaktır. Bu menüyü seçerek Wizardı aktif hale getirin.

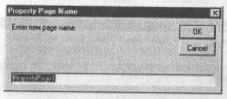
Wizardın ilk ekranı sadece bilgi verecektir.



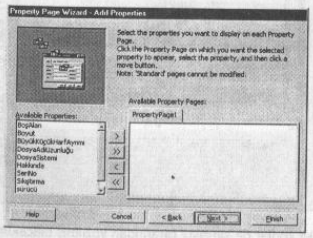
Next düğmesi ile sonraki adıma geçin. Bu adımda projenizde bulunan Property Page'lerin listesi yer alır.



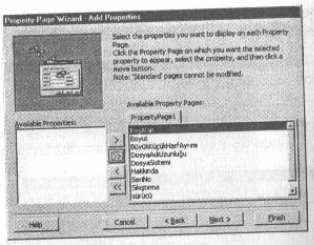
Eğer henüz bir property page tasarladıysanız listeniz boş olacaktır. **Add** düğmesine basın ve açılan aşağıdaki pencereye Property Page için bir isim girin.



Next düğmesi ile sonraki adıma geçtiğinizde Wizard projenizde bulunan User Control'e ait özellikleri tanıyacak ve aşağıdaki gibi listeleyecektir.

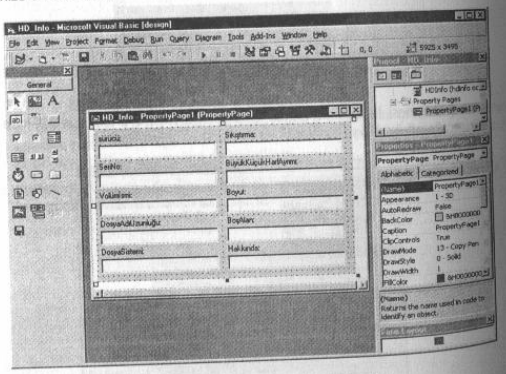


Property Page içinde bulunmasını istediğiniz özellikleri listeden seçin ve > düğmesi ile sağ taraftaki listeye aktarın. Eğer hepsinin bulunmasını istiyorsanız >> düğmesine basın.

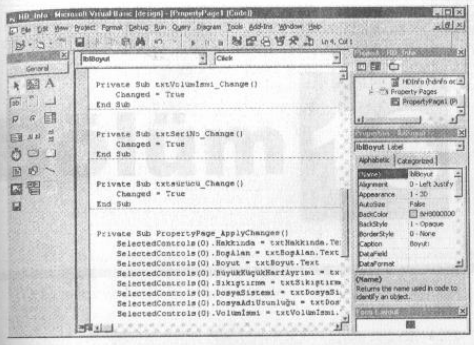


Bu işlemden sonra **Finish** düğmesine basarak Wizard'ı sona erdirin.

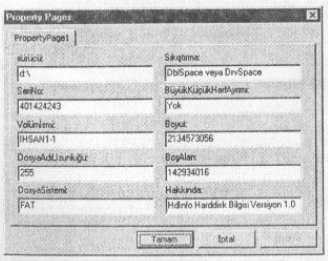
Bu işlemden sonra gerekli Property Page kontrolleriyle birlikte hazırlanarak projenize eklenecektir.



Üstelik gerekli kodlar da sihirbaz tarafından yazılmış olacaktır.



User Control'ünüzü yeniden derlediğinizde kontrolünüzde **Custom** isimli yeni bir özellik daha bulunacak ve bu özelliğin çift tıklanmasıyla hazırladığınız property page açılacaktır.



Bölüm 13

Veri Tabanı

Visual Basic ile Veritabanı Tasarımı

Visual Basic 6.0 ile beraber gelen veritabanı tasarım ortamı kullanıcıya büyük kolaylıklar sağlamaktadır. Gerek tasarım ve gerekse çalışma zamanında visual basic ortamından çıkmadan veritabanı dosyası tasarlamak, tablolar oluşturmak dolayısıyla form tasarımını kolay yapmak ve rapor oluşturmak oldukça zevkli bir hale getirilmiştir.

Burada etkin bir şekilde veritabanı dosyası ve buna ait olacak form tasarımı, raporlar, sorgular oluşturmak için gerekli olacak adımları metodları tüm detaylarına kadar vermeye çalışacağız.

Visual Database Bileşenleri

Geliştirdiğiniz projelerinizin içerisine visual database bileşenlerini dahil ederek etkileşimli veritabanı projeleri geliştirebilirsiniz. Microsoft SQL Server aracılığıyla veritabanlarını sorgulayabilir ve gerekli düzenlemeleri yapabilirsiniz.

Query Designer ile SQL komutlarını kullanabilir ve tablolardan gerekli sorgu sonuçlarını elde edebilirsiniz.

Database Designer programı ile de veritabanı dosyalarını oluşturabilirsiniz.

Visual Database bileşenleri ile ODBC uyumlu veritabanlarına bağlantı sağlayabilirsiniz.

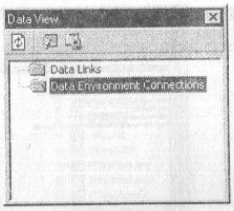
Database diyagramları kullanarak SQL veritabanlarını oluşturabilir ve gerektiğinde değiştirebilirsiniz. Sorguları tasarlayabilir, çalıştırabilir ve kaydedebilirsiniz.

Visual database elemanları aşağıdaki bileşenlerden meydana gelirler.

- DataView Window
- Data Environment Designer
- Data Report Designer
- SQL Editor

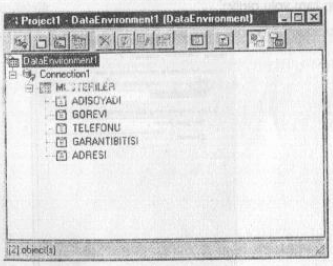
Data View

Data View penceresi her projenin bağlandığı veritabanı arasında görsel bir arabirim sağlar. Data View penceresini göstermek için **View-Data View Window** menülerini kullanabilirsiniz.

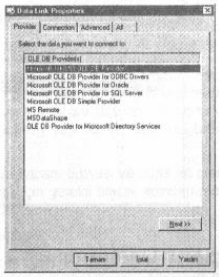


Data Environment Designer

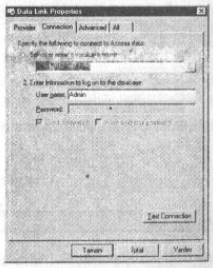
Data environment designer tasarım zamanı ile çalışma zamanı arasında etkileşimli bir arabirim sunar.



Veritabanı tablolarına bağlantı işleminin ilk merhalesi **Data Environment** penceresindeki **Connection** tıklanarak başlanır. Ve karşımıza aşağıdaki pencere çıkar:



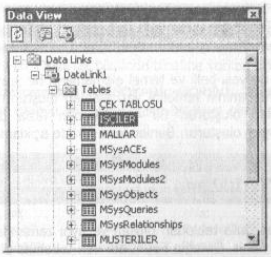
Penceredeki ilk seçenek olan **Microsoft Jet 3.51 OLE DB Provider** seçildikten sonra **Next>>** düğmesi ile sonraki adıma geçilir. Burada istenilen bir veritabanı dosyasının tam yolu girilir:



Veritabanı dosyasıyla bağlantı sağlandıktan sonra test işlemine geçilebilir.

Test Connection düğmesine basılarak bağlantının başarılı olup olmadığı kontrol edilebilir. Ve **Tamam** düğmesiyle bağlantı işlemi gerçekleştirilir.

Bu işlemden sonra bağlantının sağlandığı veritabanına ait tablolar **Data View** penceresinde yer alır.



Data View penceresinde yer alan tablolardan istenilenler fareyle sürüklenmek suretiyle **Data Environment** penceresine bırakılır. Buraya bırakılan her tablo fare yardımıyla form üzerine alınabilir. Yani otomatik olarak veri alanları için text kutuları oluşturulabilir.

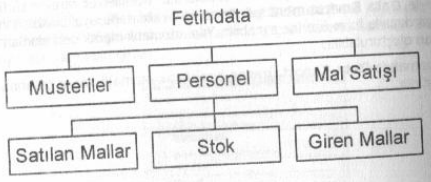
Aynı zamanda **Data Report** üzerine bırakılarak otomatik rapor tasarımı da gerçekleştirilir.

Veritabanını oluşturan elemanlar

Bir veritabanı dosyası belli ve temel elemanların bir araya getirilmesiyle meydana gelir. Veritabanının temelini veri alanları oluşturur. Veri alanları bir araya gelerek tabloları oluşturur. Bir yada birden fazla tablo bir araya gelerek veritabanı dosyası oluşturur. Bunları kısa şekilde açıklamaya çalışalım:

Database(Veritabanı)

Bir yada birden fazla tablodan oluşan ve aynı zamanda birbirleriyle ilişkili olan tablo grubuna denir. Örneğin FetihData veri tabanına ait tablolar aşağıdaki gibidir:



Table(Tablo)

Bir grup veri alanını içeren yapıya tablo denir. Örneğin Müşteriler ile ilgili bilgilerin yer aldığı tablo gibi.

MUSTERILER : Tablo

ADISOYADI	GOREVI	TELEFONU	ADRESI	GBTARIHI
AHMET PALA	İNŞAAT MÜH.	5443673	İST.	18/03/1999
M.SÜMEYYE PALA	yok	4125307	Ahlat	18/03/1999
Mehmet Pala	üni.öğ.	5443673	Beykent Üni./is	20/03/1999
SELMAN PALA	Ögrenci	5443673	istanbul	20/03/1999

Record(Kayıt)

Bir tabloda yer alan ve bir kişiye ait bilgilerin tümüne verilen addır.

Örneğin Müşteriler tablosundaki ADISOYADI, GOREVI, TELEFONU, ADRESI, GBTARIHI başlıklarının altında yer alan bilgiler bir kayıt oluşturur. Aşağıdaki seçili satır gibi.

MUSTERILER : Tablo

ADISOYADI	GOREVI	TELEFONU	ADRESI	GBTARIHI
AHMET PALA	İNŞAAT MÜH.	5443673	İST.	18/03/1999
M.SÜMEYYE PALA	yok	4125307	Ahlat	18/03/1999
Mehmet Pala	üni.öğ.	5443673	Beykent Üni./is	20/03/1999
SELMAN PALA	Ögrenci	5443673	istanbul	20/03/1999

Field(Veri alanı)

Bir tablodaki her bir kaydı oluşturan en temel yapıya veri alanı denir. Örneğin Müşteriler tablosundaki ADISOYADI, GOREVI, TELEFONU, ADRESI, GBTARIHI gibi her biri bir alan oluşturur.

MUSTERILER : Tablo

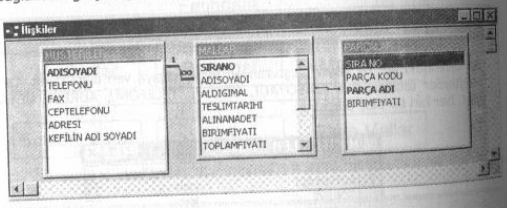
Alan Adı	Veri Türü
ADISOYADI	Metin
GOREVI	Metin
TELEFONU	Metin
ADRESI	Metin
GBTARIHI	Tarih/Saat

Index(Anahtar kayıt)

Her bir kaydı temsil eden ve kayıtların sıralanmasında önemli bir rol oynayan anahtar kayıt alanına denir. Aşağıdaki seçili alan bir anahtar alandır.

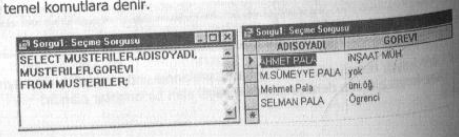
Birden fazla tablo etkileşimli olarak çalışabilmesi için aralarında bir ilişkinin olması gerekmektedir. Böyle bir ilişkinin sağlanması için anahtar alanlar arasında bir bağlantı sağlanır.

Bağlantının gerçekleşmesi için alan tipleri aynı türden olmalıdır.



Query(Sorgu)

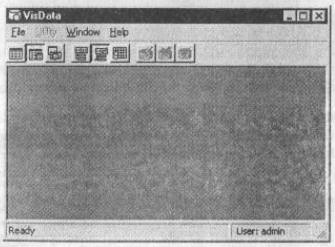
Bir yada birden fazla tablodan belli şartlara bağlı olarak istenilen kayıtları listeleyen temel komutlara denir.



Visual Data Manager ile Veritabanı Tasarlama

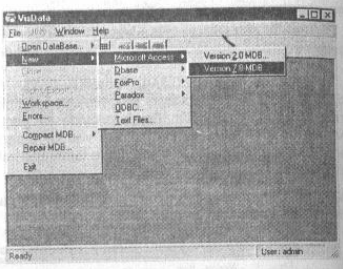
Visdata programıyla Access(2.0,7.0), Dbase(III,IV,V), Foxpro(3,2.6,2.5,2), Paradox(5.0,4.x,3.x), ODBC veritabanı programlarının destekleyeceği veritabanı dosyaları oluşturmak mümkündür. Gerek yukarıda adı geçen veritabanı programlarına ait bir dosya ve gerekse direkt visdata programı vasıtasıyla bu formatlarda oluşturulan bir dosya Visual Basic tarafından rahatlıkla kullanılabilmektedir.

Programa girmek için Visual basic ortamındaki **Add-ins/Visual Data Manager** menü seçeneklerini kullanırız:

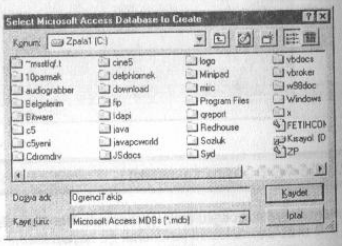


Visdata ile veritabanı tasarımı

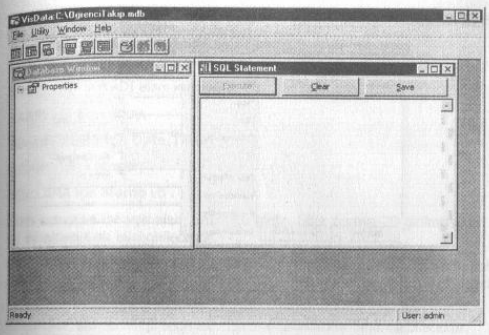
İlk veritabanı dosyamızı oluşturmak için sırasıyla **File-New-Microsoft Access-version 7.0 mdb** seçeneklerini kullanarak işe başlayalım:



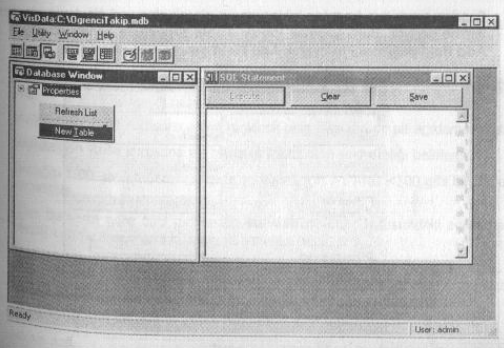
Bundan sonra karşımıza aşağıdaki diyalog penceresi çıkar. Access te olduğu gibi burada da öncelikle çalışılacak olan veritabanı dosyası kaydedilerek işe başlanır. Burada veritabanı dosyamıza OgrenciTakip ismini vererek kayıt işlemini tamamlıyoruz:



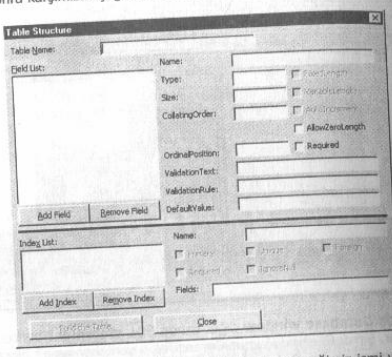
Kayıt işleminden sonra visdata'yı aşağıdaki şekliyle karşımızda buluyoruz:



Bir yanda **Database window** diğer yanda **SQL statement** pencereleri yer almaktadır. Şimdi Oluşturduğumuz OgrenciTakip veritabanına ait olacak olan tabloları oluşturmak için **Database window** penceresinde iken sağ tıklayalım. Çıkan menüden **New Table** seçeneğini seçelim.

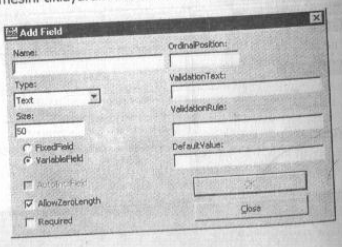


Bundan sonra karşımıza aşağıdaki veritabanı tasarım penceresi çıkacaktır:



Buradaki **Table Name** yazılan kutuya tablomuza vereceğimiz ismi yazalım. Örneğin Hocalar gibi.

Bundan sonra sıra **hocalar** tablosuna ait olacak olan veri alanlarını eklemek işlemine geçiyor. Bunu yapmak için de **Table Structure** penceresinde yer alan **Add field** düğmesini tıklayarak **Add Field** penceresine ulaşalım:



Penceredeki;

Name kutusuna girilecek olan veri alanı adı girilir. Örneğin ADISOYADI, BOLUMU, MEMLEKETI, MAASI, DOGUMTARIHI gibi.

Type kutusunda da girilecek veri alanı türü seçilir.

Örneğin ADISOYADI alanı için **Text**,

MAASI için **Currency**,

DOGUMTARIHI için **Date/Time**,

SIRANO için **Integer**,

ACIKLAMA için **Memo** vb .

Size kutusuna ise veri alanı uzunluğu girilir. Eğer buraya 20 girilmişse en fazla 20 karakterlik bir bilgi girilecektir.

FixedField seçeneği seçili ise alana girilecek olan her kaydın uzunluğu sabit olacaktır. Örneğin ALI için de 20 karakter ayrılacak, MUHAMMED ALI, içinde 20 karakter ayrılacaktır.

VariableField seçeneği seçili ise alanların uzunlukları farklı olabilecektir.

AutoIncField seçeneği seçili ise ilgili alan otomatik olarak değerini birer artıracaktır. Bu sadece sayı alanları için aktif hale gelir.

AllowZeroLength seçeneği seçili ise ilgili alana hiçbir şey girilmese de işlem devam edebilecektir.

Required seçeneği seçili ise ilgili alana bilgi girişi yapılmadan bir sonraki kayda geçiş mümkün olmayacaktır. Özellikle ADI SOYADI gibi alanlara bu özelliği vermek işleri kolaylaştıracaktır.

OrdinalPosition kutusuna veri alanı sırası girilir. İlk alan için 1,ikinci alan için 2 vb.

Validation Text kutusuna alana girilecek olan bilgi türüne ait açıklama yazılır.

Validation Rule kutusuna ise alana girilecek olan sınır aralığı belirlenir.

Örneğin 1-00 arası bir sayı girmeye zorlamak için >1 AND <100 gibi bir ifade girilmelidir.

Eğer bu şarta göre bilgi girilmezse **Validation Text** kutusundaki açıklama bir mesaj kutusu aracılığıyla kullanıcıya iletilecektir.

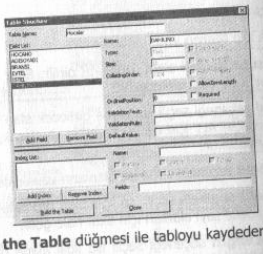
Default Value kutusuna ise girilecek alana ait varsayılan değer girilecektir. Örneğin Öğrencilerin hepsi "YYU" de okuyor olsun. O zaman "OKULU" alanına bu değeri vermek ilgili haneye tekrar bu bilgiyi girmeye gerek kalmayacaktır.

Add Field penceresine aşağıdaki gibi bilgi girişi yapıyoruz:

Alan Adı	Type	Size	AutoIncrField	Required

HOCANO	Long		Evet	
HOCAADISOYADI	Text	20		Evet
BRANSI	Text	20		
EVTEL	Text	10		
ISTEL	Text	10		
DAHILINO	Text	10		

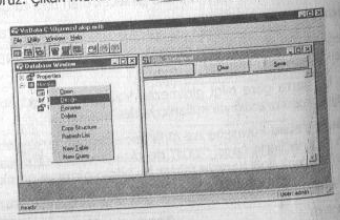
Alan tanımlaması bittikten sonra **Close** düğmesi ile **Table Structure** penceresine ulaşıyoruz:



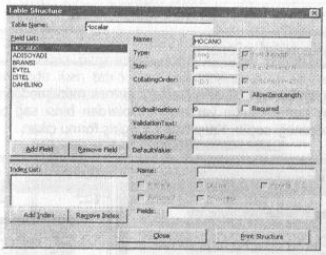
Penceredeki **Build the Table** düğmesi ile tabloyu kaydederek VisData ortamına dönüyoruz.

Tablo yapısını değiştirmek

Tablo yapısını değiştirmek, yeni alanlar eklemek ve ya varolan alanlardan istenileni silmek için **Database window** penceresindeki Hocalar tablosu üzerinde iken sağ tıklıyoruz. Çıkan menüden **Design** seçeneğini seçiyoruz.



Bundan sonra karşımıza **Table Structure** çıkar:

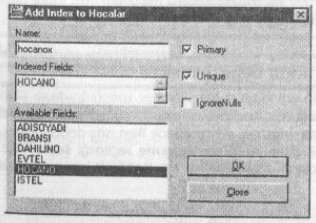


Burada veritabanı tablosu üzerinde istenilen her değişiklik yapılabilir.

Tabloda İndex tanımlamak

İndex tanımlamak özellikle birden fazla tabloda çalışıldığı zaman tablolar arasında ilişki olayında önemli rol oynamaktadır.

İndex tanımlamak için yukarıda anlatıldığı gibi **Table Structure** penceresine ulaşılır. Bu penceredeki **Add İndex** düğmesine basılarak aşağıdaki pencere görüntülenir:

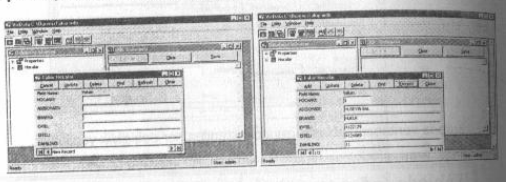


Burada **İndex** yapılacak olan veri alanı seçilerek **Indexsed Fields** kutusuna aktarılır. **Name** kutusuna ise **index** yapılacak olan alana bir isim verilir. Biz bu-

rada **HOCANO** adlı alana **hocanox** adını verip **OK** düğmesi ile işlemi tamamladık.

Tabloya bilgi girmek

Oluşturulan tabloya ait alanlara bilgi girmek mümkündür. Bunun için **Database window** penceresinde yer alan tablolardan birisi sağ tıklanır çıkan menüden **Open** seçeneği seçilir. Karşımıza bilgi giriş formu çıkar:



Penceredeki;

Add düğmesiyle yeni kayıt eklenir.

Update düğmesiyle alanlara girilen bilgiler güncelleştirilir.

Delete düğmesiyle aktif kayıt silinir.

Find düğmesiyle istenilen bir kayıt bulunur.

Refresh düğmesiyle tablo alanları yeniden görüntülenir.

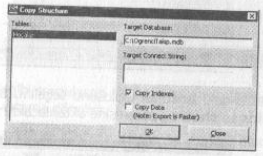
Close düğmesiyle bilgi giriş formu kapatılır.

Tablo adını değiştirmek

Oluşturduğunuz tabloların adını değiştirmek isteyebilirsiniz. Bunun için **Database window** penceresinde iken ismi değiştirilecek olan tablonun ismi sağ tıklanarak çıkan menüden **Rename** seçeneği seçilecektir. İlgili tablonun ismi otomatik olarak seçilecektir ve siz istediğiniz ismi girebileceksiniz.

Bir tablonun kopyasını oluşturmak

Bir tablonun kopyasını oluşturmak isteyebilirsiniz. Belki sadece bazı alanlarını değiştirip ikinci bir tablo oluşturmak isteyebilirsiniz. Bunun için **Database window** penceresinde iken tablo ismi sağ tıklanacak çıkan menüden **copy structure** seçeneği seçilecektir. Bunu yaptığımızda karşımıza aşağıdaki pencere çıkar:



Burada hangi tablonun kopyası alınacaksa o tablo **tables** kutusundan seçilir. Ve **Ok** düğmesine basılarak **tablo adı** giriş kutusuna ulaşılır. Burada yeni oluşturulacak olan tabloya bir isim girilir ve işlem tamamlanır:

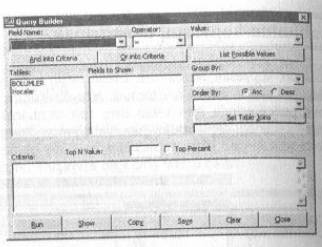
Bir tablo oluşturmak yada silmek

Yeni bir tablo oluşturmak için **Database window** penceresinde fareyi sağ tıklamak ve çıkan menüden **new table** seçeneğini seçmek, bir tablo silmek için de ilgili tablo seçildikten sonra aynı menüden **Delete** seçeneğini seçmek olacaktır.

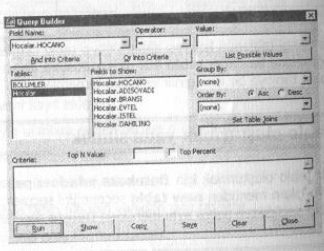
Query Builder(Sorgu oluşturmak)

Visdata programında oluşturduğunuz veri tablolarına ait sorgu oluşturabilirsiniz. Sorgular sadece belirlenen belli kriterler doğrultusunda çalışan süzgeçlerdir.

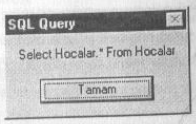
Utility-Query Builder menü seçenekleriyle **Query Builder** penceresine ulaşalım:



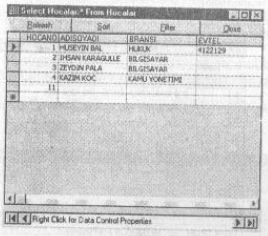
Tables kutusundaki **hocalar** tablosunu seçelim. Bu seçildiğinde buna ait veri alanları **Fields to shows** penceresine aktarılacaktır:



Bundan sonra **Show** düğmesine bastığımızda karşımıza aşağıdaki mesaj penceresi çıkacaktır:



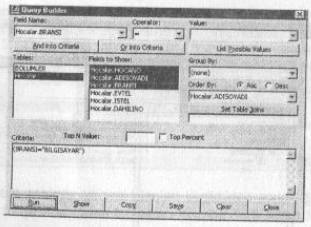
Bu mesaj kutuda yer alan tüm alanların sorgu penceresinde gözükeceğini göstermektedir. Bundan sonra **Run** düğmesine bastığımız da karşımıza sorgu sonucu çıkacaktır:



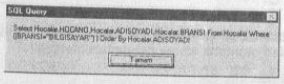
Yukarıda görüldüğü gibi sorgulanan tüm alanlar gösterilmektedir. **Close** düğmesiyle tekrar normal ortama dönelim.

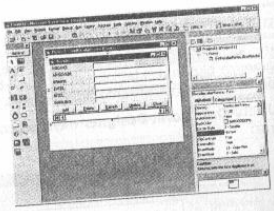
Şimdi Branşı "Bilgisayar" ve alfabetik olarak sıralanacak olan kayıtları sorgulayalım.

Bunun için **Fields to show** hanesinden ilk üç alanı, **Order By** kutusundan ADISOYADI alanını seçip, **criteria** kutusuna ise (BRANSI= "BILGISAYAR") ifadesini girelim:



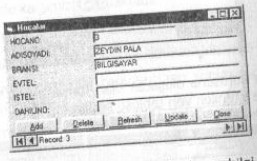
Bundan sonra **Show** düğmesine bastığımızda oluşturduğumuz kriterlere karşılık gelen SQL ifadeleri aşağıdaki mesaj penceresiyle gösterilecektir:





Form oluşturulduğunda veritabanı ile ilgili bazı işlemleri yapmak için hazır komutlar da form üzerinde yerlerini alacaktır:

Form oluşturulduktan sonra vb ortamında F5 tuşu ile programımızı çalıştırabiliriz:



Artık normal bir program gibi yukarıdaki veri alanlarına bilgi girişi yapabiliriz.

Add düğmesiyle veritabanı dosyasına yeni kayıtlar ekleyebilir,
Delete düğmesiyle aktif kaydı silebilir,
Refresh düğmesiyle alan içeriklerini güncelleyebilir,
Update düğmesiyle veritabanı dosyasını güncelleyebilir,
Kayıt gezinti düğmeleriyle kayıtlar arasında dolaşabilir ve **close** düğmesiyle programdan çıkabiliriz.

Düğmelerin başlıklarını tasarım modunda iken istediğiniz şekilde değiştirebilirsiniz.

Görüldüğü gibi bir veritabanı programını tasarlamak hiç de zor bir iş değildir.

İlk aşamada bir veritabanı dosyası tasarlanacaktır. Bu dosya Vb6 nin desteklediği veritabanı dosyalarından herhangi birisi olabileceği gibi yukarıda anlatılan **VisData** programıyla da kolay bir şekilde tasarlanabilir.

Daha sonra buna ait tablolar, sorgular oluşturulacaktır. Ve son olarak bu tablolar vb tasarım ortamına aktarılacaktır.

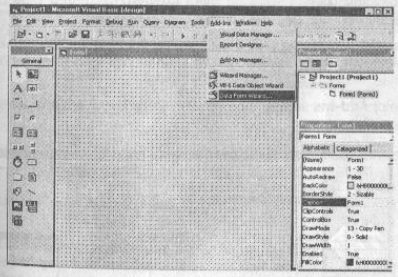
Veritabanı ile ilgili sihirbazlar

Veritabanı ile ilgili sihirbazlar program için gereken ara adımları hallederek programcının işini kolaylaştırmaktadırlar.

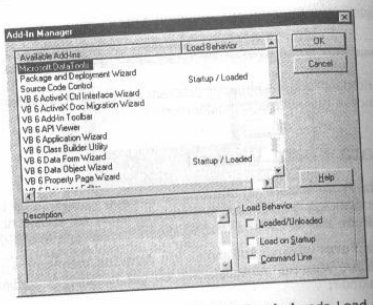
Data Form Wizard

Data From wizard vb arabirimi ile veritabanı arasında bir bağlantı kurarak bir veritabanı için gerekli olan tüm arabirimi kolay bir şekilde hazırlar. Bu wizardl'a tek form, ana-alt form ve flexgrid içeren formlar hazırlanabilir.

Data Form wizardı kulllanmak için **Add-İns** menüsünü kullanabiliriz. Burada genellikle wizardlar ve kullanıcının en çok kullanacağı yardımcı programlar yer almaktadır:



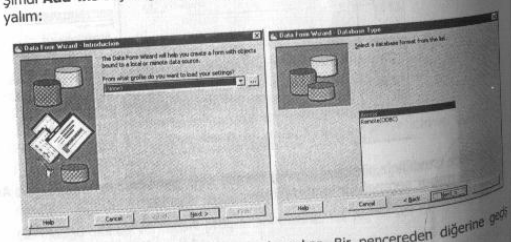
Eğer istediğiniz sihirbazı(wizard) burada bulamıyorsanız aynı menüdeki **Add-in manager** seçeneğiyle aşağıdaki pencereye ulaşınız:



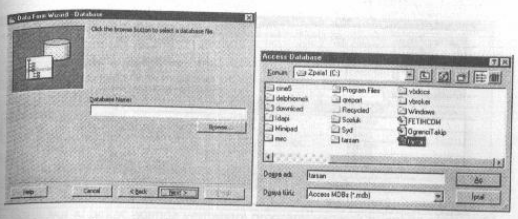
İstediğiniz bileşeni seçtikten sonra **Loaded/Unloaded** yada Load **on Startup** seçeneklerinden birini seçebilirsiniz.

Birincisi yeni bir projeye başladığınızda **Add-ins** menüsüne eklediğiniz bileşeni tekrar kaldırırken ikincisi her vb oturumunda işaretli olan bileşenleri yeniden yükler.

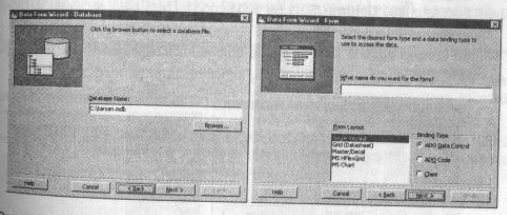
Şimdi **Add-ins** seçeneğinde yer alan **Data Form Wizard** seçeneğiyle işe başlayalım:



Karşımıza sırasıyla yukarıdaki pencereler çıkar. Bir pencereden diğerine geçiş **Next>>** düğmesiyle yapılmaktadır:



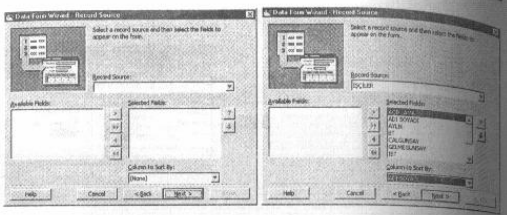
İstenilen veritabanı dosyası bulunarak **Database name** kutusuna aktarılır. Ve sonraki adımda oluşturulacak olan form tasarım tipi seçilir:



Pencerenin **Form Layout** listesindeki

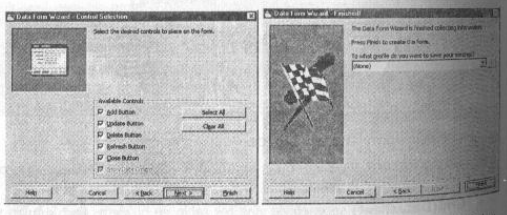
- **Single Record** seçeneğiyle bir tek tabloya bağlantı sağlanarak form oluşturulur.
- **Grid(Datasheet)** seçeneğiyle yine tek tablo ve bu defa text kutuları yerine grid(ızgara) bileşeni kullanılır. Bununla bir çok kayıt ayna anda görülebilir ve üzerinde işlem yapılabilir.
- **Master/Detail** seçeneğiyle **master/detail(ana/alt)** formlar oluşturmak ve birbirleriyle ilişkili iki tabloyu kullanmak için seçilir.
- **MS HflexGrid** seçeneğiyle seçenekle MsFlexGrid bileşeni kullanılarak form oluşturulur.
- **Ms Chart** seçeneğiyle Mschart(grafik bileşeni) kullanılarak form oluşturulur.

Biz birinci seçenekle işleme devam ediyoruz:

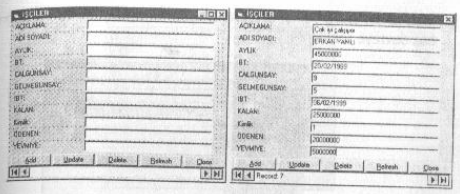


Record source kutusunda bağlantı yapılacak tablo seçilir. Buna ait veri alanları **available fields** listesinde yer alır. İstenilen alanlar sağ taraftaki kutuya alınır. Eğer istenirse **Column Sort by** kutusunda bir veri alanı seçilerek alanların buna göre sıralanması sağlanır.

Bir sonraki adımda form üzerinde bulundurulacak düğmeler seçilir ve işlem tamamlanır:



Sihirbazın işi bittikten sonra veritabanı içeren form tasarımız aşağıdaki gibi olur:



Görülüyor ki birkaç adımda veritabanı programı yapmak mümkün olmaktadır.

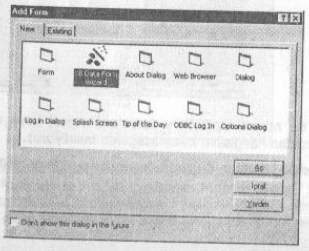
Sihirbaz kullanılarak (ana-alt/master-detail) veritabanı formunun hazırlanması

VB6 nın veritabanı form tasarımlarına yönelik olan sihirbazları kullanılarak birkaç adımda hiç program kodu yazmadan veritabanı programı oluşturmak mümkündür.

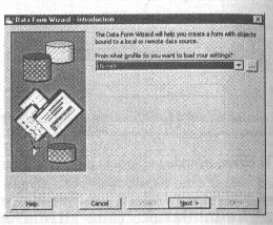
Böyle bir proje için ilk adımda veritabanı dosyası ve dolayısıyla buna ait tablolar oluşturulur. Bundan önceki kısımda anlatılan visdata programıyla bu işlemin nasıl yapılacağı detaylı olarak ele alınmıştır.

İkinci adımda visual Basic ortamına normal olarak giriş yapılır.

Üçüncü adımda **Project -Add Form** menü seçenekleriyle aşağıdaki diyalog penceresine ulaşılır:

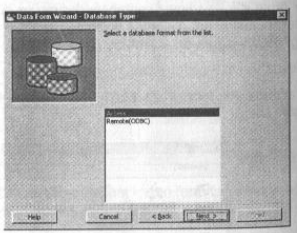


Burada **VB Data Form Wizard** simgesi seçildikten sonra **Aç** düğmesiyle sihirbaz devreye sokulur:



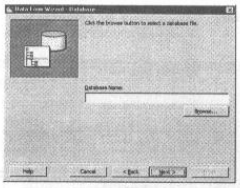
Data Form Wizard olarak adlandırılan bu pencerede ilk etapta yapılacak bir şey yok. Fakat bu sihirbazın son adımında istenirse takip edilen tadımlar kaydedilebilir. Eğer bu yapılırsa işte bu ilk adımda o dosyaya erişilebilir. Yanı hazır bir şablona kolay bir şekilde erişim imkanı sağlar.

Penceredeki **Next>>** düğmesiyle sonraki adıma geçelim:



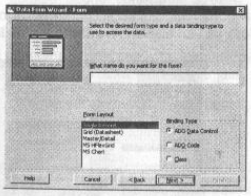
Burada ise normal veritabanı dosyaları(Access) veya uzak(Remote) veritabanı dosyalarından hangisinin kullanacağımızı belirliyoruz.

Penceredeki **Next>>** düğmesiyle sonraki adıma geçelim:

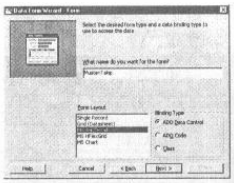


Burada projede kullanacağımız ve daha önceden hazırladığımız veritabanı dosyasını devreye sokuyoruz. Bu işlemi yapmak için **Browse** düğmesiyle aşağıdaki pencereye ulaşacağız:

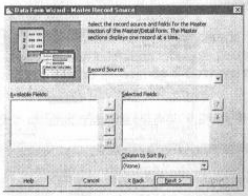
Burada veritabanı dosyamızın yer aldığı konum belirlendikten sonra **Aç** düğmesiyle geri dönelim. Penceredeki **Next>>** düğmesiyle sonraki adıma geçelim:



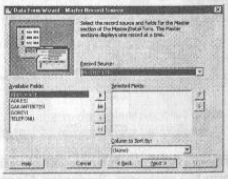
Burada vb ortamındaki veritabanı formumuza bir isim girip ve **Form Layout** listesinde ise bir form tipi seçiyoruz. Bunlardan **Master/Detail** seçeneği birden fazla ve aynı zamanda birer alanları **index** olarak belirlenmiş veritabanı tabloları için kullanılırken diğer seçenekler bir tek tablo içeren veritabanı dosyaları için kullanılır.



Bunları belirledikten sonra penceredeki **Next>>** düğmesiyle sonraki adıma geçelim:

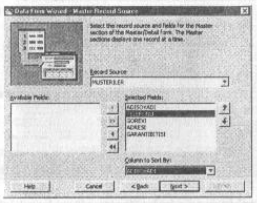


Burada **Record Source** kutusunda önceki adımda konumunu belirlediğimiz veritabanı tablolarından hangisini seçeceğimizi belirliyoruz. Bunu seçerken aynı zamanda bu tabloya ait veri alanları **Available Fields** listesinde yer alır:

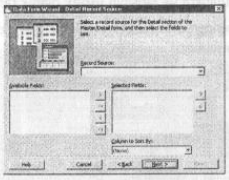


Penceredeki listelerin arasında yer alan (>,<,>>,<<) düğmeleriyle veritabanı tablosuna ait alanları karşı kutuya aktarıp kaldırabiliyoruz. **Selected Fields** kutusunda yer alanlar form üzerinde gözükür diğerleri gözükmez. Burada hepsini yada istediğiniz alanları alıp almamakta serbestsiniz. Alanların form üzerindeki

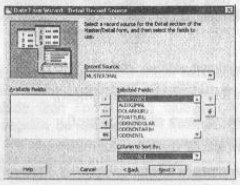
yerleşimlerini de yine (^,v) düğmeleriyle yapabiliriz. Ayrıca **Column to sort By** kutusunda seçtiğiniz alana göre veriler sıralanır.



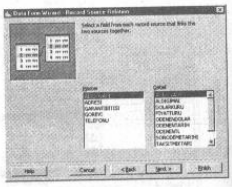
Gerekli ayarlamaları yaptıktan sonra penceredeki **Next>>** düğmesiyle sonraki adıma geçelim:



Yukarıdaki bir önceki adımda **master(ana)** tabloya ait ayarlamalar yapılmıştı burada ise aynı şekilde **Detail(alt)** tabloya ait ayarlamalar yapılır:



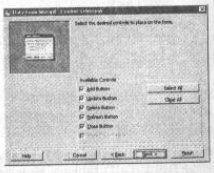
Gerekli ayarlamaları yaptıktan sonra penceredeki **Next>>** düğmesiyle sonraki adıma geçelim:



Burada ise iki tablo arasında bağlantının kurulması için her iki tablodan birer alanın seçilmesi gerekiyor. Bu işlem gelişigüzel olmaz.

Tablolar tasarlanırken buna dikkat etmek gerekir. Eşlenecek alanların aynı veri tipine ve aynı uzunlukta ve birisinin **index** olarak tanımlanması gerekmektedir.

Bundan sonra penceredeki **Next>>** düğmesiyle sonraki adıma geçelim:



Burada ise form üzerinde yer alacak ve aynı zamanda veritabanı tablolarını yönetecek olan kontroller seçilir.

Bunlar sırasıyla;

Add Button(Yeni kayıt ekleme düğmesi),

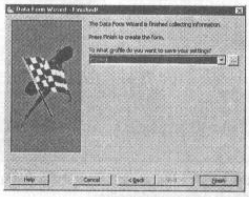
Update Button(Kayıt güncelleme düğmesi),

Delete Button(Aktif kaydı silme düğmesi),

Refresh Button(İçeriği yenileme düğmesi),

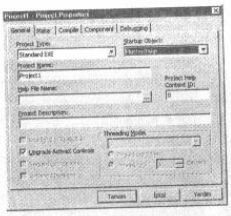
Close Button(Veritabanı formunu kapatma düğmesi) dir.

Gerekli ayarlamaları yaptıktan sonra penceredeki **Next>>** düğmesiyle sonraki adıma geçelim:



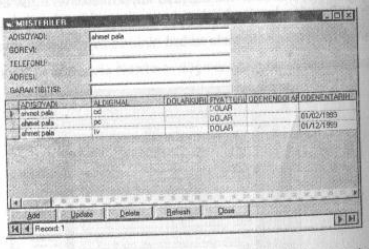
Veritabanı formu için bu son adımdır. Burada eğer istenirse **...** düğmesine basılarak yapılan ayarlamalar bir dosyaya kaydedilebilir. Ve tekrar kullanılabilir. Ve **Finish** düğmesine basılarak işlem tamamlanır. Vb ortamında formumuz aşağıdaki gibi gözükür:

Programı F5 tuşuyla çalıştırdığımızda karşımıza normal olarak form1 gelir. Bunu kaldırmak ve direkt olarak veritabanı formumuzun gelmesi için **project explorer** penceresinde iken Form1'i sağ tıklayıp çıkan menüden **Remove Form1** seçeneğini seçelim Bu işlem form1'i kaldıracaktır. Ve database formumuzun birincil form olarak çalışması için **Project-Project1 properties** seçeneğini seçelim. Karşımıza aşağıdaki pencere çıkar:



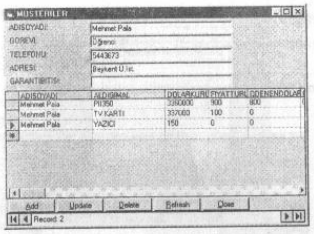
Buradaki **Startup object** kutusunda veritabanı formumuzun adını seçelim **Tamam** Düğmesiyle vb ortamına dönelim:

Ver artık F5 tuşuyla programımızı çalıştırabiliriz:



Görüldüğü gibi veritabanı programımız hiçbir koda gerek duyulmadan çalışmaktadır. Veritabanı dosyasının birinci tablosu müşteri bilgileri için, ikinci tablosu ise müşterilerin yaptığı alışveriş için ayarlanmıştır. Kayıt eklemek ve düzenlemek için gerekli düğmeler ve ayrıca kayıtlar arasında dolaşmak için kayıt gezinti kontrolü hazır olarak karşımıza çıkmaktadır.

Birinci tabloda müşterilere ait bilgileri sadece bir kere girmiş olacaksınız ikinci tabloda ise müşterilerin belli zamanlarda yaptıkları alışverişi kolay bir şekilde takip edebileceksiniz. Her iki tabloda da ADISOYADI alanları birbirleriyle ilişkilendirildikleri için ikinci tabloda tekrar isim girmeye gerek kalmamaktadır:



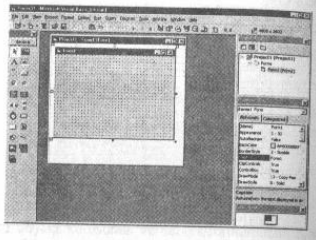
Sihirbaz kullanmadan veritabanı programı oluşturmak

Tasarım zamanında veritabanı dosyalarına bağlantı kurmak ve veritabanı programlarını oluşturmak için ActiveX Data Object (ADO) bileşeni kullanılır. Bu bileşen vasıtasıyla gerekli veritabanı bağlantıları yapılarak işleme devam edilir.

Bu bileşen ile aşağıdaki işlemler yapılır:

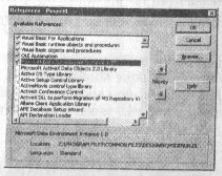
- Vb ortamındaki projeye ActiveX Data Object (ADO) bileşeni eklenerek işe başlanır.
- Veritabanı bağlantıları yapılır.
- Tablo ve SQL tabanlı Command nesneleri oluşturulur.
- Bağlantı ve kayıt setleri için gerekli program kodları yazılır.
- Command nesnesi üzerinde yer alan veritabanına ait tablo alanları form üzerine sürüklenerek bırakılır.
- Aynı şekilde raporlar da düzenlenir.

File-New Project menüleriyle yeni bir proje başlatın:



Project-More ActiveX Designers-Data Environment seçenekleriyle **Data Environment** bileşenini projeye ekleyelim:

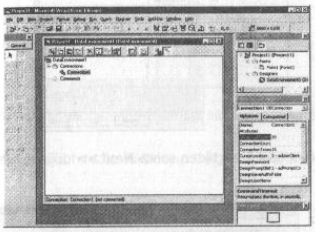
Eğer burada böyle bir seçenek yoksa o zaman **project-References** menüleriyle aşağıdaki pencereye ulaşın:



Burada **Data Environment** seçeneğini seçtikten sonra **OK** düğmesine basın.

Bu işlem yukarıda verilen **More ActiveX Designers** menüsünün altına **Data Environment** bileşenini ekleyecektir.

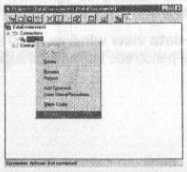
Projeye **Data Environment** bileşeni eklendikten sonra vb ortamı aşağıdaki gibi görünecektir:



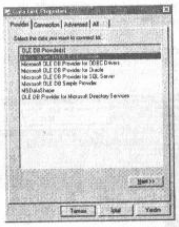
Veritabanı bağlantılarının yapılması

Veritabanları ile çalışmak için mutlaka bir bağlantının olması şartı vardır. Böyle bir bağlantı **Data environmet** penceresiyle karşımıza **Connection1** varsayılan adıyla çıkmaktadır. Bu bağlantıya, çalışılacak olan veritabanı dosyasının adı verilirse işlem daha çok anlaşılır hale gelecektir. Bunun için **Connection1** seçili iken **properties** penceresindeki **Name** özelliğine veritabanımızın ismini veriyoruz. Örneğin FETIHCOM gibi..

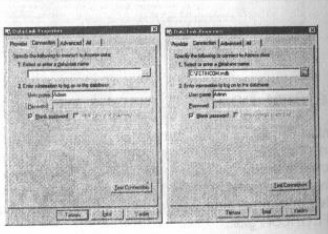
Daha sonra aynı bağlantı üzerinde iken sağ tıklayalım:



Çıkan menüden **properties** seçeneğini seçerek aşağıdaki pencereye ulaşalım:

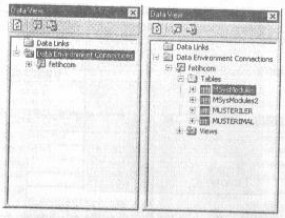


Buradaki ilk seçeneği seçtikten sonra **Next>>** düğmesiyle sonraki adıma geçelim:



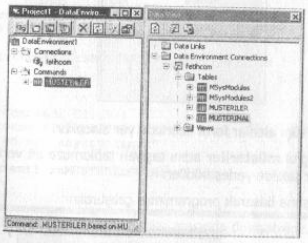
Buradaki **...** düğmesi aracılığıyla bağlantı kuracağımız veritabanı dosyasını belirleyeiim. Burada sadece Access'e ait veritabanı dosyalarını erişilebilir. **Tamam** düğmesiyle vb ortamına geri dönelim.

Bundan sonra **view-data view window** menü seçenekleriyle **Data View** penceresine ulaşalım. Bu pencerede bağlantı kurduğumuz veritabanı dosyamizin adı yer alacaktır:



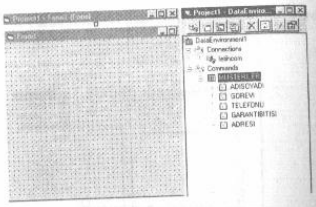
Data View penceresinde veritabanı dosyamıza ait kaç tane tablo varsa bunlar listelenir.

Şimdi müşteriler tablosunu fare ile sürükleyip **Data Environment** penceresine bırakalım:

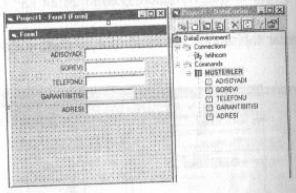


Sürüklenen tablolar artık **Data Environment** penceresinde de yer almaktadırlar.

Şimdi **Data evironment** penceresini ve formumuzu ekranda yan yana gözükecek şekilde ayarlayalım:



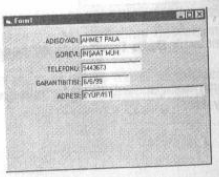
Bundan sonra **Müşteriler** tablosunu fare ile sürükleyip form üzerine bırakalım:



Tabloya ait tüm alanlar form üzerinde yer alacaktır.

Görüldüğü gibi **müsteriler** adını taşıyan tablomuza ait veri alanları form üzerinde kolay bir şekilde yerleştirildiler.

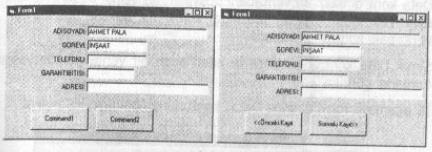
Şimdi f5 tuşuna basarak programımızı çalıştıralım:



Programımız çalışıyor fakat eksik olan bir şeyler vardır. Bunlar gezinti düğmeleri ve kayıt işlemleri ile ilgili diğer düğmelerdir.

Programa gezinti düğmeleri eklemek

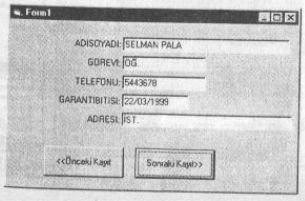
Programdaki kayıtlar arasında dolaşmak için gezinti düğmeleri gerekecektir. Bunun için form üzerine iki tane **command** düğmesi alalım.



Sırasıyla Command1 ve Command2 düğmelerinin click olayına aşağıdaki program kodlarını yazalım:

```
Private Sub Command1_Click()
If DataEnvironment1.rsMUSTERILER.BOF Then
   MsgBox "ilk kayıtta bulunmaktasınız"
Else
      DataEnvironment1.rsMUSTERILER.MovePrevious
End If
End Sub

Private Sub Command2_Click()
If DataEnvironment1.rsMUSTERILER.BOF Then
 MsgBox "Zaten son kayıttasınız"
Else
 DataEnvironment1.rsMUSTERILER.MoveNext
End If
End Sub
```

Program kodunu yazdıktan sonra kayıtlar arasında dolaşabiliriz.

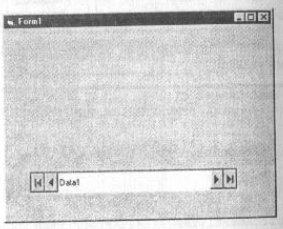


Data Kontrol ile Çalışmak

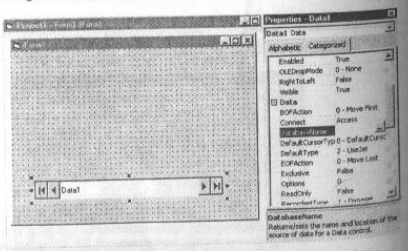
Data kontrol elemanı veritabanı dosyaları ile Visual Basic ortamını birleştiren ve aynı zamanda kayıtlar arasında kolay bir şekilde gezinmeyi sağlayan bir veritabanı kontrol elemanıdır.

Şimdi Data kontrol elemanı ile adım adım bir veritabanı dosyasına bağlantı işlemlerini görelim.

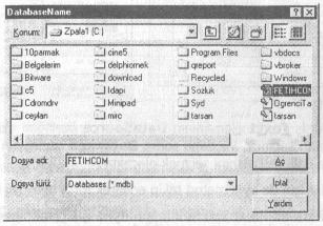
İlk aşamada form üzerine Data1 kontrol elemanını yerleştirelim:



İkinci aşamada daha önce oluşturduğumuz bir veritabanı dosyasıyla **data kontrol** arasında gerekli bağlantıyı sağlamaya çalışalım.

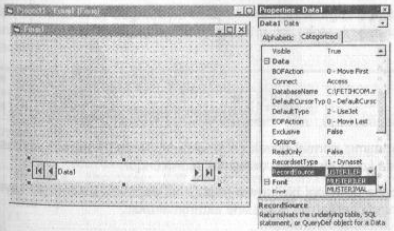


Bunun için **data1** seçili iken **properties** penceresinde **DatabaseName** özelliğinin karşısında yer alan ... düğmesini tıklayarak diskte daha önce kaydedilmiş olan veritabanı dosyasını seçelim:

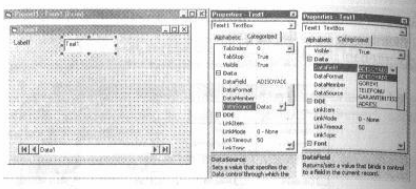


Aç düğmesini tıkladığımızda istenilen bağlantı gerçekleştirilmiş olacaktır.

Bir veritabanı dosyasında birden fazla tablo yer alabilir. Hangi tablo ile çalışacağımızı da **RecordSource** özelliği ile belirleyeceğiz. Bu özellik tıklandığında aşağıya açılan kutu aracılığıyla veritabanı dosyasında yer alan tabloların listesi görülecektir. Burada istenilen bir tanesi seçilebilir.



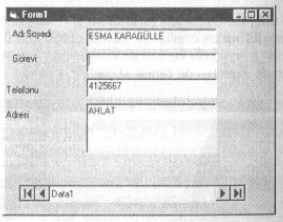
Üçüncü aşamada form üzerinde veri alanları için gerekli text kutuları yerleştirmek ve bağlantıyı sağlamaktır:



Yapılacak ilk iş **Text1** elemanının **DataSource** özelliğini tıklamak ve açılan kutudan **data1'**i seçmek. Ve daha sonra **DataField** özelliğinin karşısındaki kutuyu tıklamak ve ilgili veri alanını seçmek olacaktır.

Diğer alanlar için de aynı metod takip edilerek form tasarımı tamamlanır.

Veri alanlarına ait text kutularının yerleşim işlemi bittikten sonra F5 tuşu ile programımızı çalıştıralım:

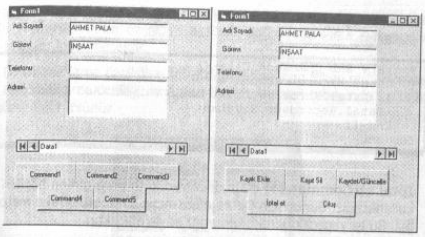


Programımız normal olarak çalışmaktadır. **Data1** kontrolü vasıtasıyla kayıtlar arasında kolay bir şekilde dolaşmak mümkündür.

Bu arada herhangi bir kayıtta yaptığımız değişiklik bir sonraki kayıtta geçtiğimizde anında veritabanı dosyasına aktarılmaktadır.

Kayıt İşlemleri

Programımızın yukarıdaki haliyle ancak kayıtlar arasında dolaşabiliyorduk. Şimdi aşağıdaki tasarımı kullanarak veritabanı üzerinde gerekli düzenlemeleri yapabileceğiz:



Form tasarımında görüldüğü gibi düğmeler kullanılarak gerekli düzenlemeler yapılabilecektir.

Kayıt ekle(Command) düğmesiyle veritabanı dosyasına yeni kayıt ekleyeceğiz. Bunu için Data1 elemanının **AddNew** metodunu kullanacağız. Bu işi yapacak gerekli kodumuz şöyle olacaktır:

```
Private Sub Command1_Click()
 Data1.Recordset.AddNew
 Text1.SetFocus
End Sub
```

Kayıt Sil(Command2) düğmesiyle aktif olan kaydı sileceğiz. Bunun için Data1 elemanının **Delete** metodunu kullanacağız. Bu işi yapacak gerekli kodumuz şöyle olacaktır:

```
Private Sub Command2_Click()
Dim mesaj As String
   mesaj = "Kaydı silmek istediğinizden emin misiniz "
    If MsgBox(mesaj, vbQuestion + vbYesNo + vbDefaultButton2) =
vbYes Then
        Data1.Recordset.Delete'sil
        Data1.Recordset.MoveNext'sonrakini göster
        If Data1.Recordset.EOF Then 'eğer sonraki kayıt yoksa
         Data1.Recordset.MoveLast 'son kaydı göster
        End If
   End If
End Sub
```

Kaydet/Güncelle(Command3) düğmesini veri kutularına girilen bilgileri veritabanına kaydetmek(güncellemek) için kullanacağız. Bunun için Data1 ele-

manının **Update** metodunu kullanacağız. Bu işi yapacak gerekli kodumuz şöyle olacaktır:

```
Private Sub Command3_Click()
   Data1.Recordset.Update
   If Data1.Recordset.EditMode <> dbEditAdd Then
    Data1.Recordset.MoveLast
   End If
End Sub
```

İptal et(Command4) düğmesini veri kutularına girilen bilgileri iptal etmek için kullanacağız. Bunun için Data1 elemanının **CancelUpdate** metodunu kullanacağız. Bu işi yapacak gerekli kodumuz şöyle olacaktır:

```
Private Sub Command4_Click()
    Data1.UpdateControls
    If Data1.Recordset.EditMode = dbEditAdd Then
    Data1.Recordset.CancelUpdate
    End If
End Sub
```

Çıkış(Command5) düğmesiyle programdan çıkacağız. Bu işi yapacak gerekli kodumuz şöyle olacaktır:

```
Private Sub Command5_Click()
 Unload Me
End Sub
```

Properties

DatabaseName

Data kontrolu için kullanılacak veri tabanı dosyasını belirler.

```
Data1.DatabaseName = "MUSTERILER.MDB"
```

RecordSource

Bir veri tabanı dosyasında birden fazla tablo bulunabilir. Bu tablolardan hangisinin kullanılacağı **RecordSource** özelliği ile belirlenir.

DatabaseName özelliği ile bir veri tabanını belirledikten sonra, properties penceresinde **RecordSource** özelliğini aşağıya doğru açarsanız kullanılabilir durumdaki tablolar listelenecektir.

Örneğin DatabaseName özelliğine VB dizininde bulunan Biblio.mdb dosyasını girerseniz RecordSource özelliği ile açılan listede bu veri tabanında bulunan tabloların listesini görürsünüz.

Bu listeden bir tablo ismi seçtiğinizde Data kontrolü o tabloyu kullanacaktır.

Options

Bu özellik data bileşeninin özelliklerini ayarlar.

```
Data1.Options=dbForwardOnly
```

Aşağıdaki değerlerden birini alır. Bu değerler çalışma zamanında kullanıldığı zaman **Refresh** metodu kullanılması gerekir.

DbDenyWrite	1	Çoklu kullanım ortamında diğer kullanıcılar kayıtsetindeki kayıtları değiştiremezler.
DbDenyRead	2	Çoklu kullanım ortamında diğer kullanıcılar kayıt setindeki kayıtları okuyamazlar.
DbReadOnly	4	Kayıtlar değiştirilemez.
DbAppendOnly	8	Kayıt ekleyebilirsiniz. Fakat varolan kayıtlar okunamaz.
DbInconsistent	16	Tüm kayıtlar güncellenebilir.
DbConsistent	32	Birleşim şartını çiğnemeyen alanları günceller.

DbSQLPassThrough	64	Data bileşeninin SQL ile kullanıldığı zaman SQL serlere gibi sql deyimlerini gönderir.
DbSeeChanges	512	Girdiğiniz verileri birileri değiştirdiği zaman hata mesajı üretir.

Bu değerlerden aynı anda birden fazlası kullanılabilir. Aralarına "+" işlemi konarak işlem gerçekleştirilir.

```
Data1.Options = dbAppendOnly + dbInconsistent
```

ReadOnly

True ve false değerlerinden birini alan bu özellik veritabanı dosyasının sadece okunur olması durumunu belirler.

```
Data1.RedOnly=true
```

EOFAction

Data kontrolü son kaydı gösterdiği halde ileri düğmesine basıldığında ne tür bir işlem yapılacağı bu özellikle belirlenir. Bu özelliğin değeri normalde 0'dır ve son kayıttan sonra ileri düğmesine basıldığında herhangi bir işlem yapılmaz. Bu özelliğe 1 değeri verildiğinde, son kayıtta ileri düğmesi pasif yapılır. 2 değeri verildiğinde ise son kayıttayken ileri düğmesine basıldığında yeni bir kaydın eklenmesini sağlar.

EOFAction özelliğine aşağıdaki değerlerden biri verilerek, son kayıttan sonra sonraki düğmesine basılırsa nasıl davranacağı belirlenir:

vbEOFActionMove Last	0	Son kayıtta kalınır.
vbEOFActionEOF	1	Sonraki düğmesi pasif yapılırç
vbEOFActionAddNew	2	Yeni bir kayıt eklenir.

BOFAction

Data kontrolü ilk kaydı gösterdiği halde önceki düğmesine basıldığında ne tür bir işlem yapılacağı bu özellikle belirlenir. Bu özelliğin değeri normalde 0'dır ve ilk kayıttayken önceki düğmesine basıldığında herhangi bir işlem yapılmaz. Bu özelliğe 1 değeri verildiğinde, ilk kayıttayken önceki düğmesi pasif yapılır.

BOFAction aşağıdaki değerleri alır:

VbBOFActionMove First	0	İlk kayıtta kalınır

VbBOFActionBOF	1	Önceki düğmesi pasif yapılır

EditMode

Bu özellikle bir kaydın düzenlenmekte olup olmadığı öğrenilebilir. Eğer kullanıcı bir kayıt üzerinde değişiklik yapmakta ise bu özelliğin değeri **DbEditInProgress** olur. Eğer yeni bir kayıt eklenmiş ve bu kayıt üzerinde giriş yapılıyorsa bu özelliğin değeri **dbEditAdd** olur. Eğer herhangi bir değiştirme işlemi yapılmıyorsa bu özelliğin değeri **dbEditNone** olur.

```
If Data1.EditMode=dbEditInProgress Then Caption="Kayıt üzerinde
değişiklik yapılıyor"
If Data1.EditMode=dbEditAdd Then Caption="Yeni bir kayıt
ekleniyor"
```

RecordsetType

Recordset nesnesinin tipini belirler. Aşağıdaki değerlerden birini alır:

VbRSTypeTable	0	Tipi **table-type** olan **Recordset**
VbRSTypeDynaset	1	Tipi **dynaset-type** olan **Recordset**
VbRSTypeSnapshot	2	Tipi **snapshot-type** olan **Recordset**

Table Type: Tablo bu modda açıldığında, tablo üzerindeki bütün bilgilere erişilebilir, değişiklik yapılabilir ve yeni kayıtlar eklenebilir.

Snapshot Type: Bu modda, tablonun bir kopyası üzerinde çalışılır ve tablo üzerinde değişiklik yapılamaz. Eğer tablodaki bilgiler üzerinde arama, listeleme, rapor hazırlama gibi işlemler yapılacaksa (yani değişiklik yapılmayacaksa) tabloyu bu modda açmak hız açısından avantaj sağlar.

Dynaset Type: Bu modda, tablo bir sorgu sonucunda bir veya bir kaç tablonun birleşiminden oluşabilir. Snapshot'tan farklı olarak update de yapılabilir.

DefaultType

Data kontrol tarafından kullanılacak olan veritabanı motoru tipini belirler.

```
Data1. DefaultType = dbUseJet
```

Aşağıdaki değerlerden birini alır.

DbUseODBC	1	Veri erişimi için ODBC kullanılır
DbUseJet	2	Microsoft Jet database veri motoru kullanılır.

Events

Validate ([index As Integer,] action As Integer, save As Integer)

Yeni bir kayda geçilirken bu olay meydana gelir. Kullanıcı yeni bir kayda geçerken aktif kayıt veri tabanına kaydedilecektir. İstenirse bu olaya kod yazılarak bazı alanlardaki bilgiler kontrol edilebilir ve uygun olmayan veriler girilmişse bunlar tespit edilip **Action** özelliğine 0 verilerek geçiş iptal edilebilir.

Örneğin Data1 kontrolü ile bağlanmış Text1 kontrolü olsun ve bu kontrol içinde öğrencinin sınavdan aldığı notun gösterildiğini kabul edelim. Eğer kullanıcı bu alana 100'den büyük bir not girerse bir başka alana geçmesine izin vermeyelim.

```
Private Sub Data1_Validate(Action As Integer, Save As Integer)
 If Text1 > "100" Then
  MsgBox ("Not 100 den büyük olamaz")
  Action = 0 'işlemi iptal et
 End If
End Sub
```

İstenirse Action özelliğine farklı değerler verilerek başka işlemlerin yapılması da sağlanabilir. **Action** özelliğine verilebilecek değerler ve bu değerlere göre yapılacak işlemler şunlardır:

vbDataActionCancel	0	İşlemi iptal et
vbDataActionMoveFirst	1	İlk kayda git
vbDataActionMovePrevious	2	Önceki kayda git
vbDataActionMoveNext	3	Sonraki kayda git
vbDataActionMoveLast	4	Son kayda git
vbDataActionAddNew	5	Yeni bir kayıt ekle
vbDataActionUpdate	6	Güncelle

Reposition()

Yeni bir kayıt aktif hale geldiğinde bu olay meydana gelir. Validate olayı ise bir kayıttan diğerine geçerken, henüz geçiş gerçekleşmeden meydana geliyordu. Bu olay ise geçiş gerçekleştikten sonra meydana gelir. Yeni kayıt aktif hale geldiğinde yapılması istenen işlemler varsa bu olaya kod yazılabilir.

Örneğin iki tablomuz olsun ve bunlardan birinde müşteri bilgileri diğerinde ise satış bilgileri bulunsun. Bir müşteriye ait kayıt ekrana geldiğinde o müşteriye satılmış olan malların listesini diğer tabloda görmek istersek bu olaya kod yazabiliriz.

```
Private Sub Data1_Reposition()
 Data2.RecordSource = "Select * from SatisBilgileri where Adi = "
& Data1.Recordset("Adi")
 Data2.Refresh
End Sub
```

Error (dataerr As Integer, response As Integer)

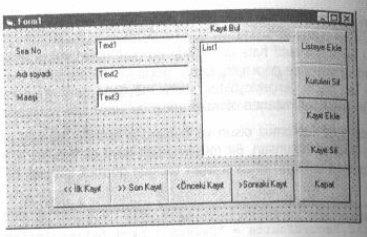
Bir hata oluştuğunda bu olay meydana gelir. **dataerr** parametresi ile hatanın numarası öğrenilebilir. Hatayı işledikten sonra **response** parametresine 0 değeri vererek işleme devam etmesini veya 1 değerini vererek hata mesajını göstermesini sağlayabilirsiniz.

Data kontrol kullanmadan veritabanı dosyaları ile çalışmak

Data kontrol bileşeni kullanmadan da veritabanı dosyası ve dolayısıyla istenilen bir tablo ile bağlantıyı sağlayarak kayıtlar arasında dolaşabileceğiz.

Örnek olarak tablodaki tüm kayıtlara ait isimleri form üzerindeki bir list1 elemanına ekleyeceğiz. List1 deki bir isimin tıklanmasıyla ilgili isimle kaydı bulacağız. Ayrıca yeni kayıt ekleyebileceğiz ve aktif kaydı silebileceğiz.

Örneğimiz için aşağıdaki form tasarımını kullanacağız:



Burada **Datakod** adını taşıyan veritabanı dosyası içindeki **PERSONEL** tablosu ile bağlantı sağlayacağız.

Personel tablosunda SIRANI, ADISOYADI ve MAASI adında üç tane alan bulunmaktadır.

Form üzerindeki **tex1** bileşenini **SIRANO** için, **Text2** bileşenini **ADISOYADI** için ve **Text3** bileşenini de **MAASI** için kullanacağız.

Ayrıca;

İlk kayıt(Command1) düğmesi ile ilk kayda gideceğiz.

Son kayıt(Command2) düğmesiyle son kayda gideceğiz.

Önceki Kayıt(Command4) düğmesiyle adım adım bir önceki kayda ulaşırken,

Sonraki Kayıt(Command3) düğmesiyle adım adım bir sonraki kayda ulaşacağız.

Listeye ekle(command5) düğmesiyle PERSONEL tablosunda yer alan tüm kayıtları List elemanına ekleyeceğiz.

Kutuları sil(Command9) düğmesiyle text kutularının içeriğini sileceğiz.

Kayıt Ekle(command8) düğmesiyle kutulara girilen bilgileri kaydedeceğiz.

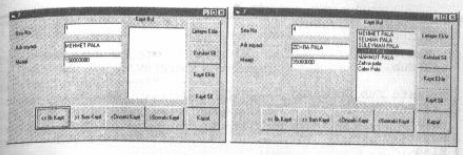
Kayıt sil(Command10) düğmesiyle aktif kaydı sileceğiz.

Kapat(Command6) düğmesiyle programı sonlandıracağız.

List_Click() olayıyla listede seçili olan isme ait kaydı bulacağız.

Ayrıca tablodaki tüm kayıtları silmek için **Kayıt sil(Command10)** düğmesini sağ tıklayacağız.

F5 tuşu ile programımızı çalıştırarak kayıtlar arasında dolaşalım:



Programımıza ait kaynak kodumuz aşağıdaki gibidir:

```
Option Explicit
Dim DataBaseAdi As String
Dim TabloAdi As String
Dim ws As Workspace
Dim db As Database
Dim KayitSeti As Recordset

Public Sub KayitOku()
On Error Resume Next
    Text1 = KayitSeti.Fields(0)
    Text2 = KayitSeti.Fields(1)
    Text3 = KayitSeti.Fields(2)
End Sub

Private Sub Command1_Click()
 KayitSeti.MoveFirst
 KayitOku
End Sub

Private Sub Command10_Click()
  Dim cevap
  'KayitSeti.MoveFirst
  cevap = MsgBox("Kaydı silecek mısınız?", vbYesNo + vbQuestion,
"sil")
  If cevap = vbYes Then
  KayitSeti.Delete
  KayitSeti.MoveNext
  KayitOku
  End If
End Sub

Private Sub Command2_Click()
KayitSeti.MoveLast
KayitOku
End Sub

Private Sub Command3_Click()
If Text1 >= Caption Then
MsgBox "son kayıttasınız"
Else
```

```
  KayitSeti.MoveNext
  KayitOku
  End If
End Sub

Private Sub Command4_Click()
 If Not Text1 <= 1 Then
KayitSeti.MovePrevious
KayitOku
Else
MsgBox "son kayıttasınız"
Exit Sub
End If
End Sub

Private Sub Command5_Click()
List1.Clear

            If KayitSeti.RecordCount > 0 Then
            KayitSeti.MoveFirst
            Do Until KayitSeti.EOF
            List1.AddItem KayitSeti![ADISOYADI]
            KayitSeti.MoveNext
                Loop
                End If
End Sub

Private Sub Command7_Click()

End Sub

Private Sub Command6_Click()
End
End Sub

Private Sub Command8_Click()

KayitSeti.AddNew
KayitSeti![SIRANO] = Caption + 1
KayitSeti![ADISOYADI] = Text2
KayitSeti![MAASI] = Text3
KayitSeti.Update
Caption = Caption + 1
End Sub

Private Sub Command9_Click()
Text1 = 0
Text2 = ""
Text3 = 0
End Sub

Private Sub Form_Activate()
Command1_Click
```



```
End Sub

Private Sub Form_Load()
  DataBaseAdi = "D:\VB98\Datakod.mdb"
  TabloAdi = "Personel"
  Set ws = DBEngine.CreateWorkspace("dbTemp", "admin", "")
  Set db = ws.OpenDatabase(DataBaseAdi)
  Set KayitSeti = db.OpenRecordset(TabloAdi, dbOpenTable)
Caption = KayitSeti.RecordCount
End Sub

Private Sub Form_Unload(Cancel As Integer)
KayitSeti.Close
db.Close
Set KayitSeti = Nothing
Set db = Nothing
End Sub

Private Sub List1_Click()
Dim aranan
KayitSeti.Index = "PrimaryKey"
'list1 deki ilk elemanin sırası 0 olduğu için
'+1 artırarak seçili değere ulaşıyoruz.
aranan = List1.ListIndex + 1

KayitSeti.Seek "=", aranan
KayitOku
End Sub

Private Sub Command10_MouseDown(Button As Integer, Shift As
Integer, X As Single, Y As Single)
If Button = vbRightButton Then
Dim cevap
  'KayitSeti.MoveFirst
  cevap = MsgBox("Tüm kayıtlar silinecek emin misiniz?", vbYesNo
+ vbQuestion, "sil")
  If cevap = vbYes Then
  Do Until KayitSeti.EOF
  KayitSeti.Delete
  KayitSeti.MoveNext
  Loop
  End If
End If
End Sub
```

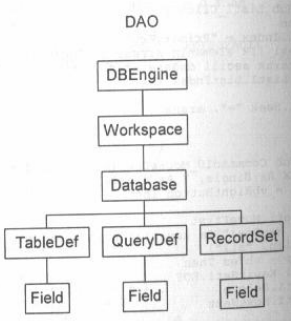

Data Access Object(DAO) ile programlama

Veritabanı programlamanın temeli DAO oluşturmak ve kullanmaktır. DAO Access bacic ve visual basic dillerinde çokça kullanılmaktadır.

DAO **Microsoft Jet database engine**'i kullanarak **workspace, Database, TableDef, Field, Recordset** ve **Querydef** gibi veri erişim bileşenleriyle bağlantı sağlar.

DAO bir hiyerarşi yapısına sahiptir. Bu yapının en üst kısmında **Microsoft Jet database engine** yer alır.

Bu ilişkiyi aşağıdaki gibi temsil edebiliriz:



- Bir Database uygulamasında **Microsoft Jet database engine** bir yada daha fazla nesne içerebilir.
- Her bir **Workspace** bir yada daha fazla **Database** koleksiyonu içerebilir.
- Her bir **Database** bir yada daha fazla **TableDef** nesnesi içerebilir.
- Her bir **TableDef** bir yada daha fazla **Field(alan)** içerebilir.

- DAO Microsoft Access tarafından oluşturulan **.MDB** formatındaki dosyaları kullanır.

DBEngine

DBEngine DAO nun nesne modellemesinde kullandığı bir bileşendir.

DBEngine DAO hiyerarşisinde diğer tüm nesneleri içerir ve aynı zamanda diğer nesneleri kontrol eder.

Properties

DefaultType

Bir sonraki Workspace nesnesi oluşturulduğunda onun tipini ayarlar.

```
DBEngine.DefaultType = dbUseODBC
```

Aşağıdaki değerlerden birini alır:

dbUseJet
Microsoft Jet database engine'e bağlantı sağlayan bir bir workspace nesnesi oluşturur.

dbUseODBC

ODBC veri kaynağına bağlantı sağlayan bir workspace oluşturur.

DefaultUser

Oluşturulacak workspace nesnesinin varsayılan kullanıcı adını belirler.

DefaultPassword

Workspace nesnesi oluşturulduğunda varsayılan şifreyi belirler.

```
DBEngine.DefaultUser = "AHLAT"
DBEngine.DefaultPassword = "FETIHCOM"
```

IniPath

Jet database engine için Windows registreyinde kullanılan yol bilgilerini belirler.

```
DBEngine.IniPath ="HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\" & _
"Jet\3.5\ISAM Formats\dBase 5.0"
```

LoginTimeout

ODBC veri kaynağına bağlantı sağlanmaya çalışıldığında bir hata oluşmadan önce geçen süreyi belirler.

```
DBEngine.LoginTimeout = 180
```

Metodları

BeginTrans, CommitTrans, Rollback

Veri tabanı üzerinde değişiklik yaparken çıkabilecek aksilikler düşünülerek önlem alınmalıdır. Örneğin bir grup veri değiştirilmeye başlandıktan sonra bütün veriler başarılı bir şekilde değiştirilemeyebilir. Bu durumda gruptaki verilerden bazıları eski değerleri bazıları ise yeni değerleri içerecektir. Bu tür durumlardan korunabilmek için veri tabanında değişikliğe başlamadan önce **BeginTrans** metodu kullanılır. Böylece yapılan değişiklikler **CommitTrans** metodu verilinceye kadar etkili olmayacaktır. Eğer işlem başarılı bir şekilde tamamlanırsa **CommitTrans** kullanılarak değişiklikler aktif hale getirilir. Eğer işlem başarılamazsa, bir aksilik çıkarsa veya iptal edilirse **Rollback** metodu çağrılarak değişiklikler iptal edilir ve **BeginTrans** metodundan önceki duruma dönülmüş olur.

Örneğin bir müşterinin banka hesabından başka bir hesaba havale yaptığını düşünelim. Yapılan havale miktarı müşterinin hesabından düşülecek ve karşı tarafın hesabına yazılacaktır. Ancak müşterinin hesabından düşme yapıldıktan sonra karşı tarafın hesabına yazarken bir problem çıkarsa hesapta uyuşmazlık olacaktır. Bu gibi durumlarda değişikliğe başlamadan önce **BeginTrans** metodu kullanılır. İşlemler başarılı bir şekilde biterse **CommitTrans**, bir aksaklık çıkarsa **Rollback** metodu kullanılır.

Ancak bazı veritabanları (paradox gibi) bu yöntemi desteklememektedir. Bu gibi durumlarda komutlarda bir hata oluşmaz ancak metodlar da görevini yerine getirmez. Eğer kullandığınız veri tabanı dosyasının bu metodu destekleyip desteklemediğini bilmiyorsanız **Transactions** özelliği ile bunu öğrenebilirsiniz. Eğer bu özellik **True** ise işlem destekleniyordur.

CreateDatabase

Yeni bir veritabanı dosyası oluşturmak için bu metod kullanılır.

```
Dim Yeni_DB As DATABASE
Set Yeni_DB = wrkDefault.CreateDatabase("fetihCom.mdb",
dbLangGeneral, dbEncrypt)
```

CreateWorkspace

Yeni bir workspace nesnesi oluşturur.

```
Dim Yeni_workODBC As Workspace
Set Yeni_workODBC = CreateWorkspace("ODBCWorkspace", "admin", "",
dbUseODBC) Workspaces.Append wrkODBC
```

OpenDatabase

Bir veritabanını açar.

```
Dim workJet As Workspace
Dim fetihCom_DB As Database
Set Fetihcom_DB = workJet.OpenDatabase("FETIHCOM.mdb", True)
```

RegisterDatabase

Bir ODBC veri kaynağının bilgilerini windows registreyine aktarır.

```
DBEngine.RegisterDatabase "FetihCom", "SQL Server", True,
strAttributes
```

RepairDatabase

Microsoft Jet Database'i onarmaya çalışır.

```
DBEngine.RepairDatabase "Fetihcom.mdb"
```

WorkSpace

Workspace nesnesi Visual Basic uygulamalarının veritabanı ile nasıl haberleşeceğini tanımlar.

Bir workspace nesnesi DBEngine.Workspaces(0) biçiminde tanımlanır.

Metodları

Close

Workspace nesnesini kapatır.

```
Dim rstMusteriler As Recordset
RstMusteriler.close
```

CreateDatabase

Yeni bir database nesnesini oluşturur.

```
Dim Varsayilan_Wrk As Workspace
Dim Yeni_DB As DATABASE

Set wrkDefault = DBEngine.Workspaces(0)
Set Yeni_DB = Varsayilan_Wrk.CreateDatabase("Ahlat.mdb",
dbLangGeneral, dbEncrypt)
```

CreateGroup

Yeni bir `Group` nesnesini oluşturur.

```
Dim Varsayilan_Wrk As Workspace
Dim YeniGrup As Group

Set Varsayilan_Wrk = DBEngine.Workspaces(0)
Set YeniGrup = Varsayilan_Wrk.CreateGroup("Hesap_Adi", "0418")
```

CreateUser

Yeni bir `User` nesnesi oluşturur.

```
Dim Yeni_Kullanici As User
Dim Varsayilan_Wrk As Workspace
Set Varsayilan_Wrk = DBEngine.Workspaces(0)

Set Yeni_Kullanici = Varsayilan_Wrk.CreateUser("Mehmet PALA",
"0418", "Sifre1")
```

OpenConnection

ODBC veri kaynağı için bir bağlantı açar.

OpenDatabase

Aktif workspace ile veritabanı açar.

```
Dim workJet As Workspace
Dim Fetih_DB As Database

Set workJet = CreateWorkspace("", "admin", "", dbUseJet)
Set Fetih_DB = workJet.OpenDatabase("FetihCom.mdb", True)
```

Database

Database nesnesi açık olan veritabanı dosyalarına erişimi sağlar.

Metodları

Close

Aktif Database nesnesini kapatır.

CreateProperty

Yeni kullanıcı tanımlı bir **property** nesnesi oluşturur.

CreateTableDef

Yeni bir **TableDef** nesnesini oluşturur.

Execute

Belirlenen SQL ifadesini çalıştırır.

NewPassword

Database nesnesine yeni bir şifre atanır.

OpenRecordset

Yeni bir Recordset nesnesini oluşturarak onu Recordset koleksiyonuna ekler.

```
Set Fetih_DB = OpenDatabase("FetihCom.mdb")
Set rsMusteriler= Fetih_DB.OpenRecordset("Musteriler",
dbOpenDynaset)
```

TableDef

TableDef nesnesi veri tabanında bulunan tablolara erişmek ve onları kullanmak için kullanılır.

Özellikleri

RecordCount

Tablodaki kayıt sayısını belirler.

Replicable

Nesnenin kopyasının alınıp alınmayacağını belirler.

ReplicaFilter

Tam kopyalama işleminden hangi kayıtların alınacağını belirler.

SourceTableName

Bağlantısı sağlanmış tablonun adını belirler.

Updatable

DAO nesnesinin güncellenebilir durumunu belirler.

ValidationRule

Bir tablodaki alana girilecek olan veri için gerekli şartı belirler.

ValidationText

Şartın gerçekleşmemesi halinde görüntülenecek olan mesajı belirler.

Metodları

CreateField

TableDef için yeni bir Field nesnesi oluşturarak onu field koleksiyonuna ekler.

CreateIndex

TableDef için yeni bir index nesnesi oluşturarak onu index koleksiyonuna ekler.

CreateProperty

TableDef için yeni bir property nesnesi oluşturur.

OpenRecordset

TableDef için yeni bir Recordset nesnesi oluşturarak onu Recordset koleksiyonuna ekler.

RecordSet

Recordset nesnesi ana tabloda yer alan kayıtları temsil eder. Recordset nesnesi üç ayrı tipe sahiptir:

Table

Dynaset

Snapshot

Bu üç tip birbirinden farklı özelliklere sahiptir.

- Table

Bu tip yerel veritabanındaki aktif tabloya erişebilir. Burada tablo değişkeni başka bir veritabandaki tabloya erişemez.

Veri sıralamak, index oluşturmak ve **seek** metodunu kullanmak gibi işlemler ancak **table** tipli bir **recordset** ile gerçekleşir.

- Dynaset

Dynaset tipine sahip bir **recordset** nesnesi hem yerel , hem sorgu sonucu ve hem de başka bir tabloya erişebilir. Birden fazla tablo aynı anda güncellenebilir.

- Snapshot

Snapshot tipine sahip bir recordset nesnesi dynaset'e benzer fakat sabit veri içerir. Bu tipe sahip bir Recodset nesnesi güncellenemez. Bir sorgu sonucunda elde edilen tüm verilerin kopyasını içerir.

Eğer verileri sıralamak ve index oluşturmak istiyorsanız **table** tipini kullanın.

Eğer sorgu sonucunda elde ettiğiniz verileri güncellemeyecekseniz **dynaset** tipini kullanın.

Eğer güncelleme yapmaksızın en yüksek hızı elde etmek istiyorsanız o zaman **snapshot** tipini kullanın.

Recordset değişkeni oluşturmak

OpenRecordset metodu recordset değişkeni oluşturmak için kullanılan en temel metotlardandır.

OpenRecordset metodu **Database**,**TableDef** ve **QueryDef** için vardır.

Aşağıdaki gibi tanımlanır:

Set değişken=Veritabanı_Adi.**OpenRecordSet**(Kaynak,tip,seçenekler)

Diğer nesneler için kullanım şekli ise şöyledir:

Set değişken=Nesne_Adi.OpenRecordSet(tip [,seçenekler])

Burada **Nesne**_Adi olarak verilen ifade **Database**, **TableDef**, **QueryDef** ve **RecordSet** nesnelerine karşılık gelmektedir.

Kaynak parametresi TableDef veya QueryDef nesnelerini temsil etmektedir.

Type parametresi ise Recordsetin tipini belirler.

DB_OPEN_TABLE

DB_OPEN_DYNASET

DB_OPEN_SNAPSHOT

Seçenekler parametresi ise bir yada daha fazla sabiti temsil eder. Bununla çok kullanıcılı veri erişimi gerçekleştirilir.

```
Dim Bizim_DB as Database
Dim BizimSet as Recordset
Set Bizim_DB=DbEngine.Workspaces(0).Database(0)
set BizimSet=Bizim_DB.OpenRecordSet("Musteriler", DB_OPEN_TABLE )
```



```
Query için Recordset değişkeni tanımlamak
Dim Bizim_DB as Database
Dim BizimSet as Recordset
Set Bizim_DB=DbEngine.Workspaces(0).Database(0)
set BizimSet= Bizim_DB.OpenRecordSet("Musteri Listesi")
```

Eğer hazır bir sorgu yoksa OpenRecordSet metoduyla tablo sorgulanabilir:

```
Dim Bizim_DB as Database
Dim BizimSet as Recordset
Dim SQL_ifade As String
Set Bizim_DB=DbEngine.Workspaces(0).Database(0)
SQL_ifade="SELECT * From Musteriler ORDER by [Sira No]"
Set BizimSet= Bizim_DB.OpenRecordSet(SQL_ifade)
```

Properties

BOF, EOF

Bu iki özellik veri tabanının başına veya sonuna ulaşıldığında True değerini alır. Son kayıttayken sonraki kayda geçilmek istendiğinde EOF özelliği, ilk kayıttayken önceki kayda geçilmek istendiğinde de BOF özelliği True olur. Eğer bu özelliklerden biri True ise kayıt sınırlarının dışına çıkılmıştır. Özellikle MoveNext metodu ile bir veri tabanında ilerlerken EOF özelliği ile sona ulaşılıp ulaşılmadığı, MovePrevious metodu ile bir veri tabanında geriye doğru ilerlerken BOF özelliği ile başa ulaşılıp ulaşılmadığı kontrol edilmelidir.

```
Dim wrks As Workspace
Dim veritabani As Database
Dim tablo As Recordset

Set wrks = DBEngine.Workspaces(0)
Set veritabani = OpenDatabase("firma.mdb")
Set tablo = veritabani.OpenRecordset("Satış")
With tablo
'Bütün tabloyu tarayarak borcu ellimilyondan fazla olanları
yazdır
  Do Until .EOF 'Sona gelinceye kadar
   If !borc >50000000 Then
    Printer.print !Adı ; !Soyadı , !Borc
   End If
  .MoveNext 'sonraki
  Loop
End With
```

Bookmark

Veri tabanındaki kayıt konumunu belirlemek ve gerektiğinde bu noktaya dönebilmek için Bookmark özelliği kullanılır. Veri tabanı üzerinde aktif kaydı değiştirecek işlemler yapılacaksa önce Bookmark özelliği ile aktif kaydın konumu bir değişkene atanır ve işlemler bittikten sonra başlangıçtaki kayda ulaşabilmek için değişkene alınan yer bilgisi tekrar Bookmark özelliğine atanarak önceki konuma geri dönülebilir.

```
Dim wrks As Workspace
Dim veritabani As Database
Dim tablo As Recordset
dim yer as variant

Set wrks = DBEngine.Workspaces(0)
Set veritabani = OpenDatabase("firma.mdb")
Set tablo = veritabani.OpenRecordset("Satış")

With tablo
 yer= .bookmark 'su anki kaydın yerini sakla
'Bütün tabloyu tarayan bir kod
 Do Until .EOF 'Sona gelinceye kadar
  'herhangi bir kod
 .MoveNext 'sonraki
 Loop
 .bookmark=yer 'eski yere geri dön
End With
```

Bookmarkable

Kayıt setinin **Bookmark** işlemini destekleyip desteklemediğini belirler.

```
If KayitSeti.Bookmarkable = False Then
 msgbox("Kayıt seti Bookmark işlemini desteklemiyor")
End if
```

LastModified

Recordset nesnesinin en son güncelleme yerini belirler.

Bu yer bilgisi BookMark özelliğine atanarak son değişiklik yapılan kayıt aktif hale getirilebilir.

CacheSize

ODBC veri kaynağından alınacak ve yerel olarak saklanmış kayıt sayısını belirler.

CacheStart

ODBC veri kaynağından yerel olarak saklanmış kayıt setindeki ilk kaydın yerini dönderir.

Connection

Mevcut Recodset nesnesine sahip olan Connection nesnesini atarlar.

DateCreated

Mevcut nesnenin oluşturulduğu tarih ve saatı dönderir.

LastUpdated

Mevcut nesnenin en son güncellendiği tarih ve saatı dönderir.

EditMode

Aktif kaydın bilgi giriş durumunu belirler.

Filter

Kayıtları filtrelemek için gerekli şartı ayarlar.

Index

Recordset nesnesi için aktif index i ayarlar.

LockEdits

Bilgi giriş için kilit durumunu belirler.

Name

DAO nesnesi için kullanıcının tanımladığı adı belirler.

NoMatch

Seek ve Find metodlarından sonra bir kaydın bulunup bulunmadığını belirler.

Aşağıdaki gibi bir kodla aranan kritere uyan bütün kayıtlara ulaşılabilir.

```
 DataBaseAdi = "D:\VB98\Datakod.mdb"
  TabloAdi = "Personel"
  Set ws = DBEngine.CreateWorkspace("dbTemp", "admin", "")
  Set db = ws.OpenDatabase(DataBaseAdi)
  Set KayitSeti = db.OpenRecordset(TabloAdi, dbOpenTable)
'aktif index tanımlanıyor.
KayitSeti.index= "SIRANO"
'ilk kayıt aranıyor
KayitSeti.Seek "=",1
Do until KayitSeti.NoMatch
 KayitSeti.edit
 KayitSeti("SIRANO")=2
 KayitSeti.update
 KayitSeti.seek "=",1
Loop
KayitSeti.close
```

PercentPosition

Recordset nesnesi içinde aktif kaydın yaklaşık olarak pozisyonunu % olarak belirler.

RecordCount

Aktif Recordset nesnesi içindeki kayıt sayısını belirler.

```
Form1.Caption = "Kayıt Sayısı: " & KayitSeti.RecordCount
```

RecordStatus

Aktif kaydın güncellme durumunu belirler.

Restartable

Aktif nesnenin ReQuery metodunu destekleyip desteklemediğini belirler.

Sort

Aktif Recordset nesnesi içindeki kayıtları sıralar.


```
'Kayıtlar ADISOYADI alanına göre sıralanır.
Kayitseti.sort= "[ADISOYADI]"
'Kayıtlar önce ADISOYADI sonra MESLEGI alanına göre sıralanır.
Kayitseti.sort= "[ADISOYADI], [MESLEGI]"
```

StillExecuting

Bir işlemin bitip bitmediğini belirler.

Updatable

Aktif nesnenin güncellenip güncellenmeyeceğini belirler.

ValidationRule

Bir veri alanına bilgi giriş şartını belirler. Aşağıdaki gibi bir kodla 1 ile 100 arası bir değer girilebileceği belirlenebilir.

```
VizeNotu.ValidationRule = "BETWEEN 1 AND 100"
```

ValidationText

İlgili veri alanına yapılan şartlı giriş için gerekli mesajı görüntüler.

```
VizeNotu.ValidationText="Vize notunuz 1-100 arasında olmalıdır"
```

Methods

AddNew

Kayıt setine(recordset) yeni bir kayıt eklemek için kullanılır.

```
Option Explicit
Dim DataBaseAdi As String
Dim TabloAdi As String
Dim ws As Workspace
Dim db As Database
Dim KayitSeti As Recordset
KayitSeti.AddNew
```

Cancel

Tüm senkron işlemleri iptal eder.

CancelUpdate

Askıdaki güncellemeleri iptal eder.

Clone

Mevcut recordset nesnesinin bir kopyasını oluşturur.

Close

Aktif kayıt setini kapatır.

```
Private Sub Form_Unload(Cancel As Integer)
KayitSeti.Close
db.Close
Set KayitSeti = Nothing
Set db = Nothing
End Sub
```

Delete

Aktif kaydı siler.

```
Private Sub Command10_Click()
  Dim cevap
  'KayitSeti.MoveFirst
  cevap = MsgBox("Kaydı silecek misiniz?", vbYesNo + vbQuestion,
"sil")
  If cevap = vbYes Then
   KayitSeti.Delete
   KayitSeti.MoveNext
  End If
End Sub
```

Edit

Aktif kaydı vb kodu ile düzenleme moduna alır. Bir alanın değeri değiştirilmeden önce bu metodla düzenleme moduna alınır.

```
KayitSeti.Edit
```


FindFirst

Belirlenen şarta uyan ilk kaydı bulur.

```
Dim aranan as string
Aranan= "ADISOYADI= 'MEHMET PALA' "
KayitSeti.Findfirst aranan
```

FindLast

Belirlenen şarta uyan son kaydı bulur.

```
KayitSeti.FindLast aranan
```

FindNext

Belirlenen şarta uyan sonraki kaydı bulur.

```
KayitSeti.FindNext aranan
```

FindPrevious

Belirlenen şarta uyan önceki kaydı bulur.

```
KayitSeti.FindPrevious aranan
```

Move

Aktif kaydın pozisyonunu Recordset nesnesinde hareket ettirir.

MoveFirst

Recordset nesnesi içindeki ilk kayda hareket eder.

```
KayitSeti.MoveFirst
```

MoveLast

Recordset nesnesi içindeki son kayda hareket eder.

```
KayitSeti.MoveLast
```


MoveNext

Recordset nesnesi içindeki sonraki kayda hareket eder.

```
KayitSeti.MoveNext
```

MovePrevious

Recordset nesnesi içindeki önceki kayda hareket eder.

```
KayitSeti.MovePrevious
```

Requery

Aktif veritabanı bilgileriyle tabloyu yeniden sorgular.

Seek

İndex ile beraber kullanılan şart kriterine uyan kaydı bulur.

```
Private Sub List1_Click()
Dim aranan
KayitSeti.Index = "PrimaryKey"
aranan = "10"
KayitSeti.Seek "=", aranan
End Sub
```

Update

AddNew ve Edit metodlarıyla belirlenen değişiklikleri recordset nesnesine kaydeder.

```
Private Sub Command8_Click()
'Kayıt ekle
KayitSeti.AddNew
KayitSeti![SIRANO] = Caption + 1
KayitSeti![ADISOYADI] = Text2
KayitSeti![MAASI] = Text3
KayitSeti.Update
Caption = Caption + 1
End Sub
```


Field

Veritabanlarını meydana getiren en temel birimlerden biridir. Her tablo değişik özelliklerdeki alanlardan (field) oluşur.

Properties

AllowZeroLength

Veri alanının sıfır uzunluklu olup olmayacağını belirler.

```
Dim Fetih_DB As Database
Dim tdfMusteriler As TableDef
Dim Bizim_Alan As Field

Set Fetih_DB = OpenDatabase("Fetihcom.mdb")
Set tdfMusteriler = Fetih_DB.TableDefs("Musteriler")
Set Bizim_Alan = tdfMusteriler.CreateField("Memleketi",
dbText, 24)
Bizim_Alan.AllowZeroLength = True
tdfMusteriler.Fields.Append Bizim_Alan
```

DataUpdatable

Nesnenin güncellenebilme durumunu belirler.

DefaultValue

Alan için varsayılan değer belirlenir. Bu özellikle belirlenen değer her kayıtta otomatik olarak alana yazılır. Kullanıcı isterse bu değeri değiştirir.

```
tdfMusteriler.Fields!Memleketi.DefaultValue = "VAN"
```

FieldSize

Kullanılacak olan alanın büyüklüğünü belirler.

Name

DAO nesnesi için kullanıcı tanımlı adı belirler.

OrdinalPosition

Fields koleksiyonunda alanın sıra numarasını belirler.

OriginalValue

En son toplu güncelleme yapıldığında alana ait orijinal değeri dönderir.

Required

Bir alana bilgi girişinin zorunlu olup olmadığını belirler.

Size

Alanın uzunluğunu belirler.

```
Dim Fetih_DB As Database
Dim tdfMusteriler As TableDef
Dim Bizim_Alan As Field
Set Bizim_Alan = tdfMusteriler.CreateField("Memleketi")
Bizim_Alan.Type = dbText
Bizim_Alan.Size = 20
tdfMusteriler.Fields.Append Bizim_Alan
```

SourceField

Aktif filed nesnesi için veri kaynağında yer alan alanı belirler.

SourceTable

Aktif filed nesnesi için veri kaynağında yer alan tablo adını belirler.

Type

Nesnenin işlem tipini belirler.

ValidateOnSet

Alanın value özelliği ayarlandığında nesneye ait değerin durumunu belirler.

ValidationRule

Alana girilecek olan bilginin giriş aralığı bu özellikle belirlenebilir. Örneğin sınav notunun girilmesi gereken bir alana 0-100 arası bir not girilmesi şartı bu özellikle koyulabilir.

```
Bizim_Alan.ValidationRule = "BETWEEN 1 AND 100"
Bizim_Alan.ValidationText = "Vize notunuz 1-100 arasında
olmalıdır"
Fetih_DB.TableDefs!Musteriler.Fields.Append Bizim_Alan
```

ValidationText

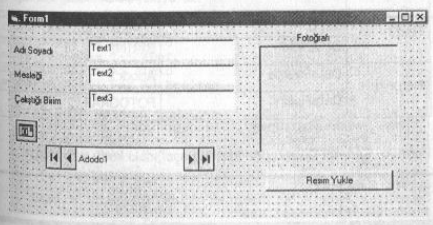
Belirlenen şartlar haricinde bir bilgi girişi yapıldığında gösterilecek hata mesajını belirlemek için bu özellik kullanılır.

Value

Alanda saklı olan değeri dönderir.

Bir veri alanına resim eklemek

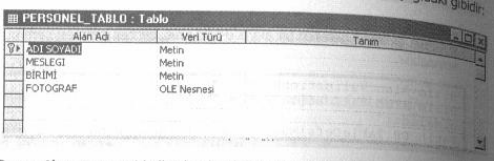
Bir verialanına çalışma zamanında resim eklemek için örnek olarak aşağıdaki form tasarımını kullanacağız.



Form tasarımı için Text1, Text2, Text3, Comondialog1,Adodc1, picture1, command1 ve labeller kullanacağız.

Adodc1 bileşenini **personel** adındaki veritabanı dosyamızla bağlantı sağlamak için kullanacağız.

Programda kullanacağımız veritabanına ait tablomuzun yapısı aşağıdaki gibidir:



Properties penceresi kullanılarak Adodc1 bileşeninin **ConnectionString** özelliği

Provider=Microsoft.Jet.OLEDB.3.51;Persist Security Info=False;Data Source=C:\Program Files\Microsoft Visual Studio\VB98\Personel.mdb

yapılıyor.

Recordsource özelliği de **PERSONEL_TABLO** olarak ayarlanıyor.

Ayrıca **properties** penceresinde aşağıdaki ayarlamaları da yapıyoruz:

Text1	DataSource	Adodc1
	DataField	ADI SOYADI
Text2	DataSource	Adodc1
	DataField	MESLEGI
Text3	DataSource	Adodc1
	DataField	BİRİMİ
Picture1	DataSource	Adodc1
	DataField	FOTOGRAF

Formun General-Declaration kısmına aşağıdaki kodu yazıyoruz:

```
Option Explicit
Const Per_PATH = "C:\Program Files\Microsoft Visual
Studio\VB98\Personel.MDB"
```

Form_load olayına aşağıdaki kodu yazalım:

```
Private Sub Form_Load()
With CommonDialog1
```



```
.Filter = "Bitmap files (*.bmp)|*.bmp|JPEG files (*.jpg,
*.jpeg)|*.jpg;*.jpeg|GIF Files (*.gif)|*.gif|All Files (*.*)|*.*"
.DefaultExt = "Bmp"
.Flags = &H1000
End With
Resim
End Sub
```

Resim yüklemek için kullanacağımız Command1 in click olayı na aşağıdaki kodu yazalım:

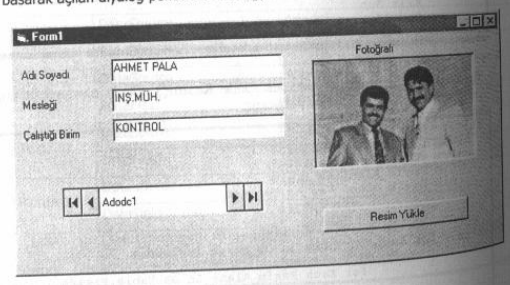
```
Private Sub Command1_Click()
On Error GoTo iptal
CommonDialog1.ShowOpen
Picture1 = LoadPicture(CommonDialog1.FileName)
On Error GoTo 0
        Exit Sub
iptal:
    If Err.Number <> cdlCancel Then
        MsgBox Err.Description, vbExclamation
        Else
        Exit Sub
        End If
End Sub
```

Ayrıca veri alanına resim eklemek içinde **Resim()** adında oluşturduğumuz alt program için aşağıdaki kodu yazıyoruz:

```
Private Sub Resim()
    Dim DB_Dosya As Database
    Dim DB_Table As TableDef
    Dim Resim_Alani As Field
    Dim Resim_alani_Var As Boolean

    On Error GoTo Hata
        Set DB_Dosya = Workspaces(0).OpenDatabase(Per_PATH)
        For Each DB_Table In DB_Dosya.TableDefs
            If DB_Table.Name = "PERSONEL_TABLO" Then
                For Each Resim_Alani In DB_Table.Fields
                    If Resim_Alani.Name = "Fotografi" Then
                        Resim_alani_Var = True
                        Exit For
                    End If
                Next
                Exit For
            End If
        Next

        If Resim_alani_Var = False Then
```

```
        If DB_Table.Updatable Then
            Set Resim_Alani = New Field
            With Resim_Alani
                .Name = "Fotografi"
                .Type = dbLongBinary
            End With
            DB_Table.Fields.Append Resim_Alani
        Else

        End If
    End If
  On Error GoTo 0
Exit Sub

Hata:
    MsgBox Err.Description, vbExclamation
    End
End Sub
```

F5 tuşuyla programımızı çalıştıralım. **Resim Yükle(Command1)** düğmesine basarak açılan diyalog penceresi aracılığıyla aktif kayda ait resimi yüklüyoruz:



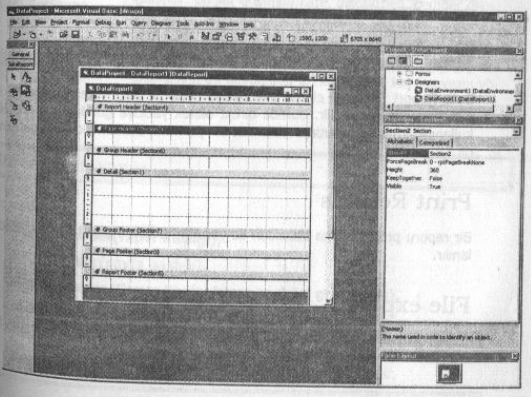
Raporlarla Çalışmak

Rapor oluşturmak bir çok ticari uygulamanın en önemli özelliklerinden biridir. İyi bir programı ön plana çıkaran özelliklerin başında şüphesiz iyi bir raporlama tekniği yatmaktadır. Diğer vb versiyonlarından farklı olarak raporlar vb programlarıyla entegre edilebilecek hale getirilmiştir.

Buradaki raporlama özellikle Access teki raporlama tekniğine oldukça fazla benzetilmiştir.

Bir veritabanı kaynağına bağlantı kurularak değişik ve birbirleriyle ilişkili tablolara ait rapor oluşturmak oldukça kolay bir hale getirilmiştir.

Aşağıda raporun tasarım görüntüsü yer almaktadır:



Data Report Designer(DRD) özellikleri

Data Report Designer birkaç önemli özelliğe sahiptir.

Drag and Drop(Sürükle ve bırak teknolojisi)

Data Report Designer, Data Environment designer üzerindeki veri alanlarının sürüklenip bırakılmasıyla otomatik olarak text kutularını oluşturmaktadır. Ve aynı zamanda sürüklenen alan yada alanlara ait **DataMember** ve **DataField** özellikleri de ayarlanmaktadır.

Toolbox kontrolleri

Data Report Designer bileşeni kendisine ait olan elemanları kullanır. Bir Data Report Designer elemanının projeye eklenmesiyle buna ait olan ve **DataReport** adını taşıyan kontrol elemanları toolbox ta yerlerini almaktadırlar. Bunlar Label, Shape, Image, TextBox, ve Line bileşenleridir. Ayrıca Function bileşeni, otomatik olarak Sum(toplama), Average(ortalama), Minimum(en küçük sayı), Maximum(En büyük sayı) işlemlerini desteklemektedir.

Print Preview

Data Report Designer elemanının **Show** metodu kullanılarak veriler ön izleme penceresinde görülebilir. Burada direkt olarak yazıcıya gönderilir. Bu özelliğinin kullanılabilmesi için mutlaka sistemde bir yazıcının tanıtılmış olması gerekmektedir.

Print Reports

Bir raporu programlama teknikleri ile yazdırmak için **PrintReport** metodu kullanılır.

File export

ExportReport metodu kullanılarak veriler HTML yada TEXT formatında kaydedilebilir.

Data Report Designer(DRD) Kontrolleri

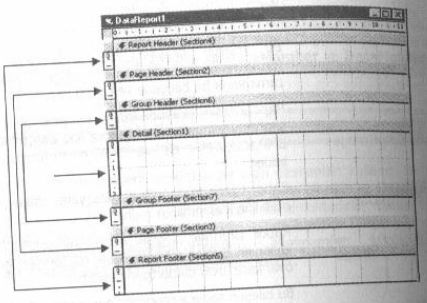
Şu anda rapor tasarımında kullanılabilecek 6 tane eleman bulunmaktadır. Bunun haricindeki diğer kontrol elemanları rapor tasarımında kullanılamamaktadırlar.

RptLabel	Rapor başlıkları ve sabit ifadeler için kullanılan bir elemandır. Tasarım zamanında düzenlenebilir.
RptTextBox	Bu bileşen çalışma zamanında ve sadece text formatındaki veri alanlarının içeriğini görüntüler. Tasarım zamanında bu bileşene veri girişi yapılamaz.
RptImage	Rapora resim eklemek için bu bileşeni kullanırız. Bu bileşen aynı zamanda standart image kontrolüne de bezer. BMP,ICO,WMF, GIF ve JPEG resim formatlarını da destekler.
RptLine	Rapor üzerinde değişik çizgiler(yatay, dikey, çapraz) çizmek için kullanırız.
RptShape	Rapor üzerinde değişik geometrik şekiller çizmek için kullanılır. Bunlar arasında kare, dikdörtgen, çember,elips, oval kare, oval dikdörtgen sayılabilir.
RptFunction	Bu bileşen sadece raporun Footer bölümünde yer alabilir. Bununla rapor üzerinde yer alan verilerin toplamını, ortalamasını, en büyüğünü, en küçüğünü, kayıt sayısını vb. hesaplayabileceğiz.

Bir raporun yapısı

Bir rapor ;

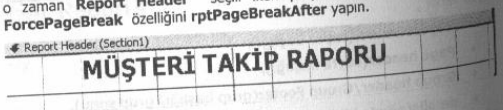
- **Report header(rapor başlığı),**
- **Page header(sayfa başlığı),**
- **Group header/Group Footer(grup başlığı/ grup sonu),**
- **details(gövde bölümü),**
- **Page Footer(sayfa sonu),**
- **Report footer(Rapor Sonu)** bölümlerinden meydana gelir.



Report Header

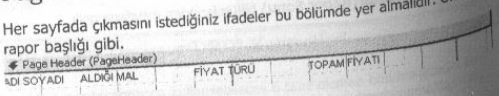
Her raporun başlığında çıkacak olan ifadeler burada yazılır. Örneğin rapor başlığı, yazar adı ve veritabanı dosyasının adı gibi.

Eğer oluşturduğunuz raporda rapor başlığının ilk sayfada olmasını istiyorsanız o zaman **Report Header** seçili iken properties penceresinde buna ait **ForcePageBreak** özelliğini **rptPageBreakAfter** yapın.



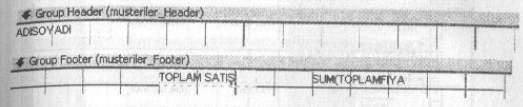
Page Header

Her sayfada çıkmasını istediğiniz ifadeler bu bölümde yer almalıdır. Örneğin rapor başlığı gibi.



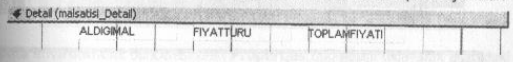
Group Header/Footer

Burası Data Report un tekrarlanan bölümlerini içerir. Her bir grup başı, grup sonu ile eşittir. Bu çift birlikte rapora eklenir yada kaldırılır.



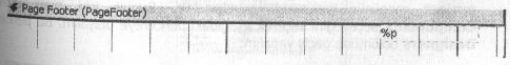
Details

Bir raporun temel gövdesini burası oluşturur. Raporun tekrarlanan en temel bileşeni de burasıdır. Veri alanları sürüklenip bu alana bırakılarak rapor oluşturulur.



Page Footer

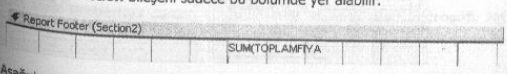
Rapora ait her sayfanın alt kısmında çıkacak olan bilgiler burada yer alır. Örneğin sayfa numarası gibi.



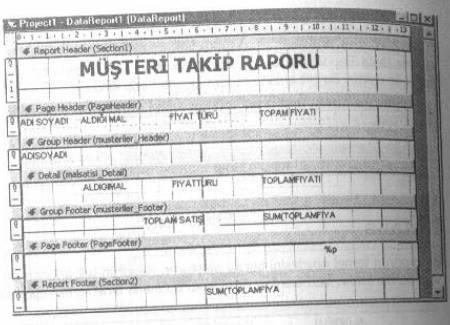
Report Footer

Raporun sununda yer alması gereken bilgiler burada yer alır. Örneğin rapor özet bilgileri ve adres gibi. Report Footer son sayfanın sayfa başlığı ve sayfa sonu arasında yer alır.

Ayrıca **rptTFunction** bileşeni sadece bu bölümde yer alabilir.

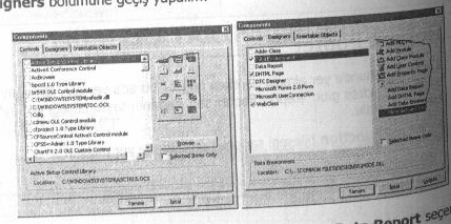


Aşağıda bir raporun tasarım görüntüsü yer almaktadır:

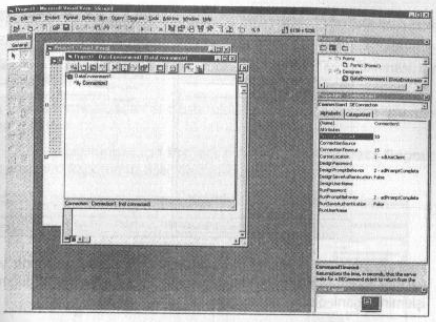


Bir rapor oluşturmak

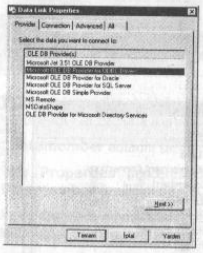
Yeni bir proje başlatalım. Project menüsünden **Add Data environment** seçeneğini tıklayalım. Bu seçenekle veritabanı tasarımcısı projeye eklenecektir. Eğer Project menüsünde Add Data environment yer almıyorsa o zaman önce bunu **project** menüsüne eklemek gerekecektir. Bunun için aynı menüden **Components** seçeneğini seçerek aşağıdaki pencereye ulaşalım. Bu pencerenin **Designers** bölümüne geçiş yapalım:



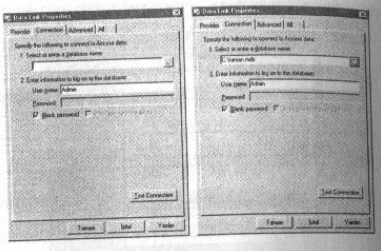
Designers listesinde yer alan **Data environment** ve **Data Report** seçeneklerini seçelim Ve **Tamam** düğmesiyle çalışma ortamına geri dönelim.



Data environment penceresindeki **Properties** düğmesini tıklayarak aşağıdaki pencereye ulaşalım:

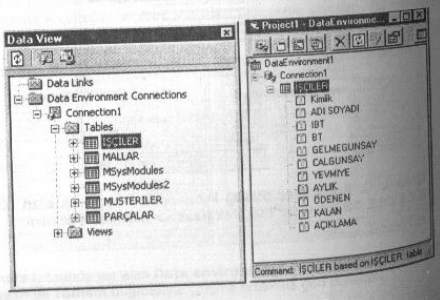


Data Link Properties diyalog kutusundaki ilk seçenek **olan Microsoft Jet 3.51 OLE DB Provider'ı** seçerek **Next>>** düğmesine basalım:



Penceredeki ... düğmesine basarak bağlantı kurmak istediğimiz veritabanı dosyasını seçelim. Bağlantı işleminden sonra **Test Connection** düğmesine basarak işlemin başarılı olup olmadığını anlayabiliriz:

Bundan sonra Veritabanımıza ait olan tablolar **Data View** penceresinde yer alacaktır.

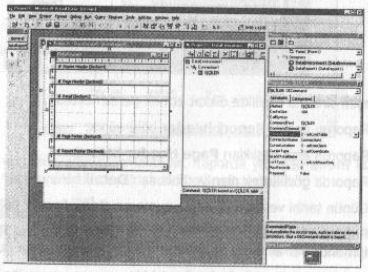


Bu tablolardan hangisi ile çalışacaksak onu sürükleyip **Dataenvironment** penceresine bırakıyoruz.

Buraya kadar ki kısım veritabanı dosyasıyla bağlantının sağlanması için gerekli olan adımlardır.

Data Environment tasarımcısı oluşturulduktan sonra artık kolaylıkla raporumuzu oluşturabiliriz.

Yeni bir rapor oluşturmak için **Project** menüsünden **Add Data Report** seçeneğini tıklayalım. Raporumuz aşağıdaki gibi gözükecektir:

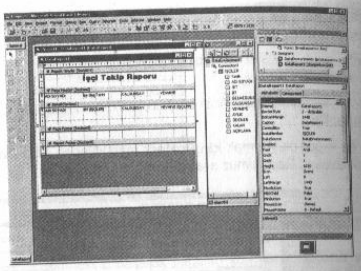


Raporda veritabanı dosyasında yer alan tabloların içeriklerini göstermek için Data Report un iki önemli özelliği işin başında iken ayarlanmalıdır.

Bunlar **DataSource** ve **Datamember** adlarını taşırlar.

Data Report seçili iken **Properties** penceresinde **Datasource** özelliğini **DataEnvironment1**, **Datamember** özelliğini de İŞÇİLER yapalım.

Bundan sonra yapılacak şey **Dataenvironment** penceresindeki tabloya ait veri alanlarını fareyle sürükleyip rapor üzerine bırakmaktır:



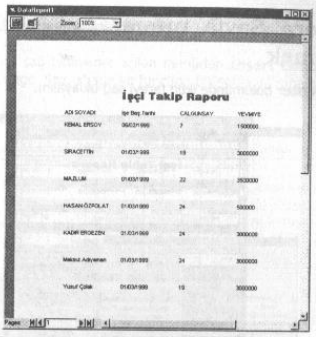
Yerleşim sırasına özellikte dikkat etmek gerekmektedir.

- Raporun başlığı **Report header** bölümünde
- Raporun alan başlıkları **Page Header** bölümünde
- Raporda gözükecek olan veri alanları **Detail** bölümünde
- Günün tarihi ve raporun sayfa numarası **Page footer** bölümünde
- Rapora ait alanların hesaplamalarıyla ilgili işlemler ise **Report Footer** bölümünde yer almalıdır.

Rapor üzerine ilgili veri alanları bırakıldıktan sonra projemize ait olan form1 üzerine Command1 düğmesi alalım ve Click olayına aşağıdaki kodu yazalım:

```
Private Sub Command1_Click()
DataReport1.Show
End Sub
```

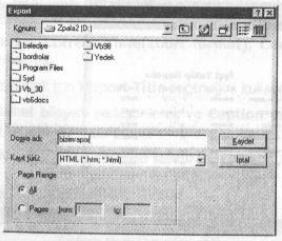
Ve programımızı çalıştıralım:



Raporumuz normal olarak çalışmaktadır. Raporda yer alan **zoom** kutusuyla rapor belli bir % lik oranında büyük yada küçük gösterilebilir.

Raporu yazdırmak için düğmesini kullanırız.

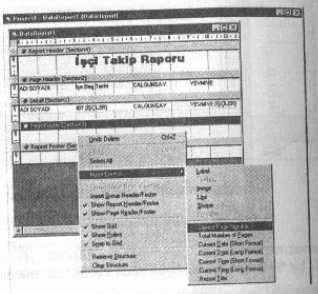
Raporumuzun verilerini düğmesi aracılığıyla HTML formatında kaydedebiliriz:



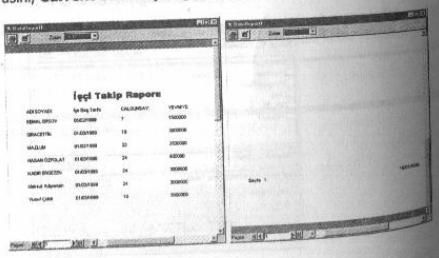
Pencerede de görüldüğü gibi raporun tamamı yada istenilen aralıktaki sayfaları HTML, yada TEXT formatında kaydedilebilir.

Rapora Sayfa numarası ve Günün tarihini eklemek

Page footer bölümünde iken fareyi sağ tıklayalım:

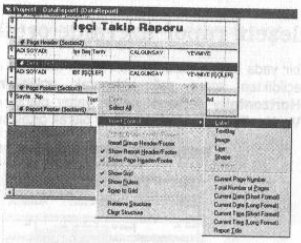


Açılan menüden **Insert Control/Current Page Number** seçeneğiyle sayfa numarasını, **Current Date** seçeneğiyle de günün tarihini ekleyelim:



Rapora bileşen eklemek

Bunun gibi raporu sağ tıklayarak açılan menüden **İnsert Control** seçeneğiyle Label,Textbox, image, line, shape ve function bileşenlerini rahatlıkla rapor üzerine alabiliriz.



Sayfa numaralarını ilgili sayfada bastırmak için

- **Current page number** seçeneğini,
- Sayfa adedini bulmak için **total page number** seçeneğini,
- Günün tarihini bastırmak için **Current Date(short format), Current Date(long format)** seçeneklerini,
- Saatı bastırmak için **Current Time(short format), Current Time(long format)** seçeneklerini,
- Rapora başlık eklemek için **Report Title** seçeneğini kullanırız.

Rapor üzerine bir **label** bileşeni yerleştirilerek ve **Caption** özelliğine aşağıdaki ifadeler verilerek te bir çok işlem gerçekleştirilebilir:

Label1.Caption	Gerçekleşen İşlem
%p	Aktif sayfanın numarası
%P	Toplam sayfa sayısı
%d	Günün tarihi(kısa format)-19/3/99
%D	Günün tarihi(uzun format)-19 mart 1999 Cuma
%t	Sistem tarihi(kısa format)
%T	Sistem tarihi(uzun format)
%i	Rapor başlığı

Yukarıdaki ifadeler birleştirilerek te kullanılabilir:

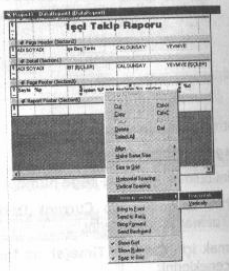
Örneğin "x adet sayfanın y. Sayfası" biçiminde göstermek için

Toplam %P adet sayfanın %p. Sayfası

Şeklinde bir ifade kullanılabilir.

Bir bileşeni rapor üzerinde ortalamak

İstenilen bir yada birden fazla bileşeni rapor üzerinde ortalamak için ilgili bileşenler seçildikten sonra sağ tıklanarak açılan **menüden Center in section/Horizontally** seçilerek nesne yatay olarak ortalanırken **Center in section/Vertically** seçeneğiyle de dikey olarak ortalanır:



Bir raporun alanlarına ait hesaplamaların yapılması

Bir raporda hesap bölümü **Report footer** kısmıdır. Buraya **rptFunction** bileşeni alınarak bir çok işlem yapılabilir.

Toolbox penceresinden **rptfunction** elemanını rapor üzerine alalım. Bu bileşenine ait o-rada dikkat edilmesi gereken önemli noktalardan birincisi lan **Function Type, Datamember** ve **Datafield** özelliklerinin ayarlanmasıdır.

Function Type özelliği aşağıdaki değerleri alır.

0-rptFuncSum

DataField özelliğine atanan veri alanına ait toplamları bulur.

1-rptFuncAve

DataField özelliğine atanan veri alanına ait ortalamayı bulur.

2- rptFuncMin

En küçük değeri bulur.

3-rptFuncMax

En büyük değeri bulur.

4- rptFuncRCnt

Bir bölümdeki satır sayısını bulur.

5- rptFuncVCnt

Boş olmayan(sıfır olmayan) alan sayısını bulur.

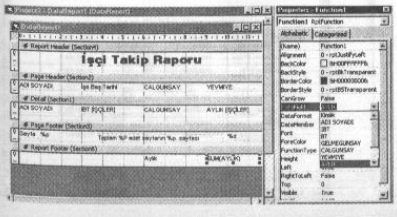
6-rptFuncSDEV

Standart sapmayı bulur.

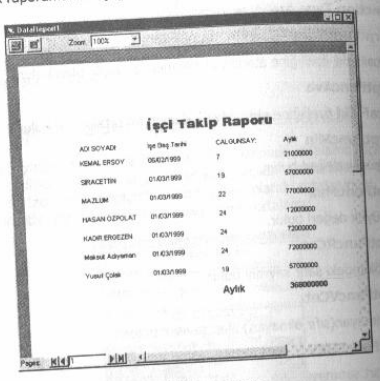
7-rptFuncSERR

Standart hatayı hesaplar.

Aşağıda Function1 bileşeninin **Data Member** özelliği **"İŞÇİLER" DataField** özelliği **aylık** olarak belirlenmiştir.

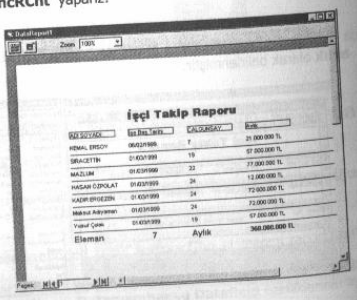


Toplam aylık raporumuzda aşağıdaki gibi gözükecektir:



Function1 bileşeninin içeri **SUM(AYLIK)** olacaktır.

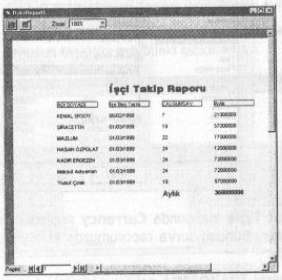
Tablodaki kişi sayısını bulmak için **Function** bileşeninin **Function Type** özelliğini **4- rptFuncRCnt** yaparız:



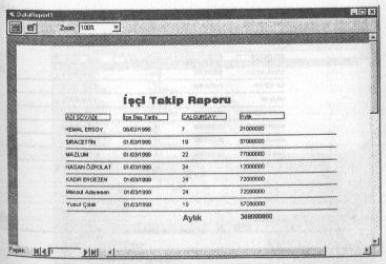
Veri alanlarını biçimlendirmek

Form üzerindeki veri alanlarına ait bileşenleri kendi isteğinize göre biçimlendirebilirsiniz.

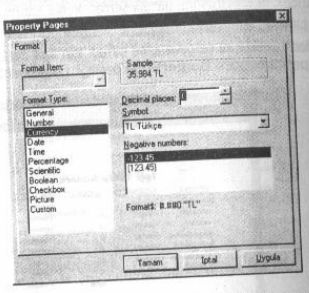
İstenilen alan başlıklarını çerçeve içine almak için ilgili başlıklar seçildikten sonra properties penceresindeki **BorderStyle** özelliği **1-rptbssolid** yapılmalıdır.



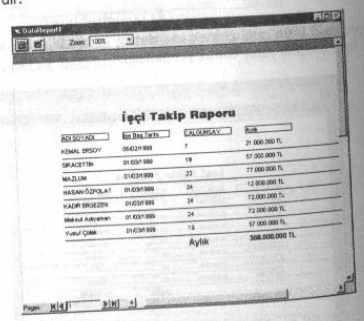
Veri alanları arasına çizgi çizmek için **detail** kısmında yer alan veri alanlarının altında **line** bileşeni yatay olarak yer almalıdır.



Sayısal içerikli alanların rakamlarını üçer basamaklı ayırmak için **DataFormat** özelliği kullanılmalıdır. Bu özellik seçildiğinde karşımıza aşağıdaki pencere çıkar:



Burada **format Type** kutusunda **Currency** seçilerek **decimal places** kutusu da sıfırlanmalıdır. Bundan sonra raporumuzda ki sayısal alanlar aşağıdaki gibi gözükeceklerdir.



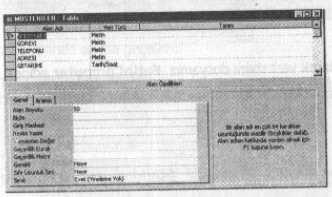
Master/Detail(Ana-Alt) rapor oluşturmak

Yukarıda verdiğimiz örnek bir tek tablo içeren veritabanı dosyasına ait raporu oluşturuyordu.

Şimdi iki tabloyu içeren ve aynı zamanda tabloları kendi arasında ilişkili olan veritabanı dosyasına ait rapor oluşturacağız.

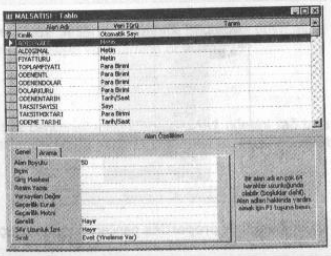
Rapora geçmeden evvel raporlarımızı Access veritabanını kullanarak tasarladık.

MUSTERILER adını taşıyan ve aynı zamanda **ana(master)** tablo olarak işlem görecek olan tablomuzun tasarımı aşağıdaki gibidir:



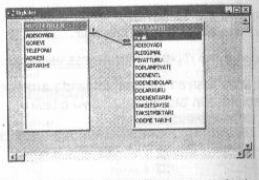
Tablonun birinci sıradaki veri alanı olan **ADISOYADI** alanı anahtar alan olarak belirtilmiş ve **sıralı** özelliği de **evet(yineleme yok)** değeri seçilmiştir.

Diğer tablomuz ise **MALSATISI** adını taşıyor. Ve **alt(detail)** tablo olarak işlem görecektir.



Bu tablodaki **ADISOYADI** hanesi seçilerek **sıralı özelliğine evet(yineleme var)** değeri verilmiştir.

Ve tablolar aşağıdaki gibi **ADISOYADI** alanları **bire-çok** olarak ilişkilendirilmişlerdir.



Ve son olarak veritabanı dosyamız **FetihComputer** adında kaydedilmiştir.

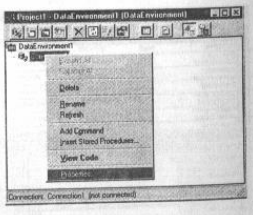
Buraya kadar temel bilgileri verdikten sonra artık rapor tasarımına geçebiliriz.

Adım 1

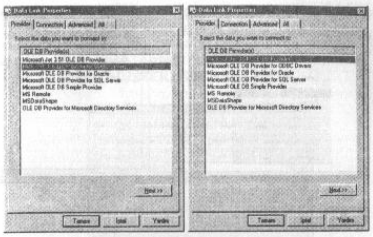
File/New project menüleriyle **Standart exe** seçeneğiyle yeni bir proje başlatın.

Project menüsünden **Add Data Environment** seçeneğiyle projeye **Data environment** bileşenini ekleyelim.

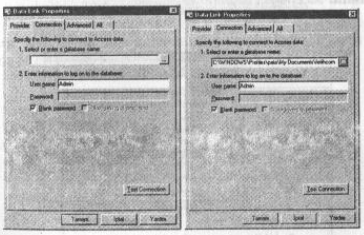
Data environment penceresindeki **Connection1** seçeneğini sağ tıklayarak çıkan menüden **Properties** seçeneğini seçelim.



Karşımıza çıkan **Data link properties** penceresindeki ilk seçeneği seçelim:



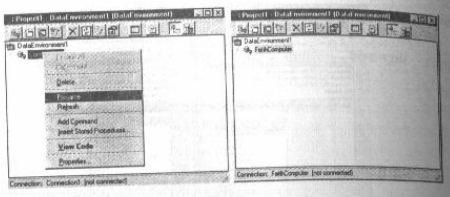
Ve **Next** düğmesiyle sonraki adıma geçelim:



düğmesiyle **Fetihcomputer** adını taşıyan veritabanı dosyamızla bağlantı kuralım. Ve **tamam** düğmesiyle vb ortamına dönelim.

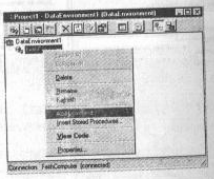
Adım 2

Data Environment penceresindeki **Connection1** öğesini sağ tıklayalım. Çıkan menüden **rename** seçeneğiyle yeniden adlandıralım. **Fetihcomputer** adını verelim.

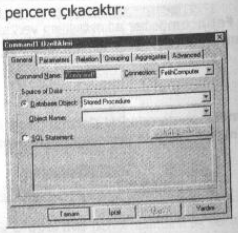


Adım 3

Şimdi **FetihComputer** seçeneğini sağ tıklayalım. Çıkan menüden **Add Command** seçeneğini seçelim:

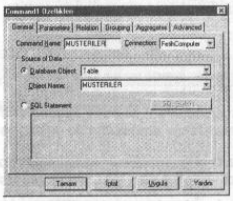


Ve karşımıza aşağıdaki pencere çıkacaktır:

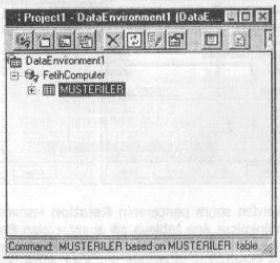


Bu pencerede aşağıdaki ayarlamaları yapalım:

Command Name kutusuna **ana(master)** tablomuzun adı olan **MUSTERILER** ifadesini girelim. **Database object** açılan kutusundan **Table** seçelim. Ve **Object name** kutusunda **MUSTERILER** adını seçelim.



Ve **Tamam** düğmesiyle vb ortamına dönelim. Bu adımla **Data environmet** penceresine MUSTERILER tablosu eklenecektir:

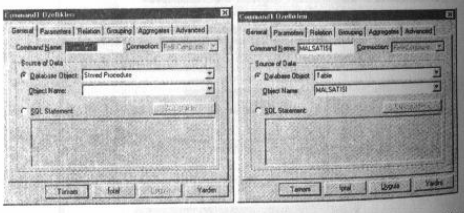


Adım 5

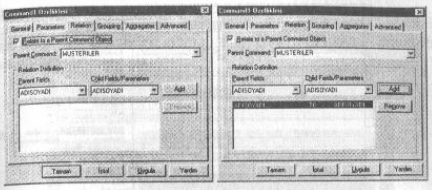
Şimdi **MUSTERILER** tablosunu sağ tıklayarak çıkan menüden **Add Child Command** seçeneğini seçelim.

Karşımıza çıkacak olan aşağıdaki pencerenin **Command Name** kutusuna **alt(detail)** tablomuzun adı olan **MALSATISI** ifadesini girelim.

Database object açılan kurusundan **Table** seçeneğini seçelim. Ve **Object name** kutusunda **MALSATISI** adını seçelim.



Bu ayarlamalardan sonra pencerenin **Relation** kısmına geçelim. Bura eğer iki tablo ilişkilendirilmişse Ana tabloya ait anahtar alan **Parent Fields** kutusunda, alt forma ait anahtar alan ise **Child fields/Parameters** kutusunda yer alacaktır. Bu alanları birbirleriyle bağlamak için **Add** düğmesine basmamız gerekecektir.

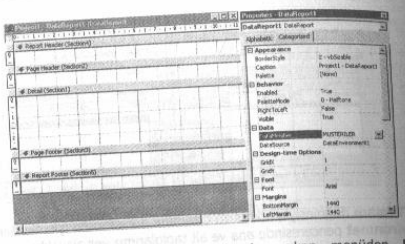


Ve **Tamam** düğmesiyle tekrar vb ortamına dönelim. Bu işlemlerden sonra **Data Environmet** penceresinde ana ve alt tablolarımız yer alacaktır:

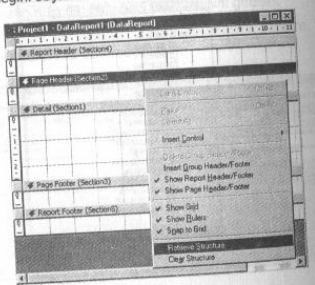
Adım 6

Şimdi **Project** menüsündeki **Add DataReport** seçeneğini seçerek **Data Report**'u projeye ekleyelim.

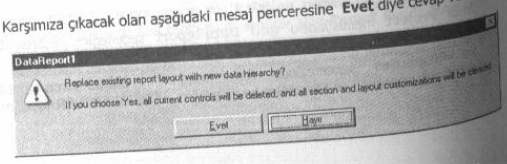
Ve **Properties** penceresinde Datareport1'in **DataSource** özelliğini **DataEnvironment1** ve **DataMember** özelliğini de **MUSTERILER** yapalım:



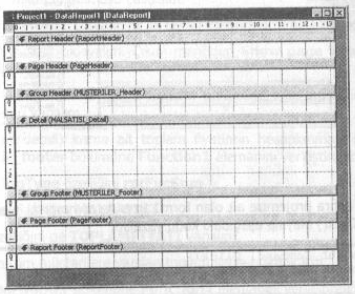
Bundan sonra **DataReport1** sağ tıklayalım çıkan menüden **Retrieve Structure** seçeneğini seçelim.



Karşımıza çıkacak olan aşağıdaki mesaj penceresine **Evet** diye cevap verelim.



Bu işlem **Data Environmet** penceresindeki iki tablo arasında sağlanmış olan **master/detail** yapısını aynen rapora yansıtacaktır. Ve raporun bölümleri buna göre ayarlanacaktır:

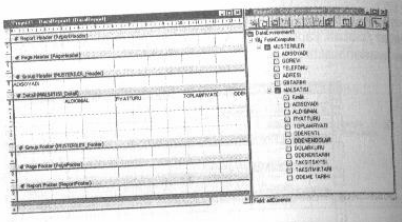


Adım 7

Şimdi yapacağımız işlem Data Environment penceresindeki MUSTERILER tablosundaki ADISOYADI alanı fareyle taşıyıp raporun **MUSTERILER_header** bölümüne bırakalım.

Bunu yaptığımızda veri alanı ile birlikte bir de etiket bırakılacaktır. Bunu silebilirsiniz.

Aynı şekilde Data Environment penceresindeki MALSATISI tablosundaki ALDIGIMAL, TOPLAMFIYATI, FIYATTURU VE ODENENDOLAR alanlarını fareyle taşıyıp raporun **MALSATISI_detail** bölümüne bırakalım.

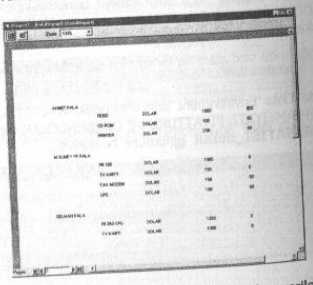


Adım 8

Bundan sonra projemize ait olan form1 üzerine command1 elemanını yerleştirelim. Ve Click() olayına aşağıdaki kodu yazalım:

```
Private Sub Command1_Click()
DataReport1.Show
End Sub
```

Son olarak F5 tuşuna basarak programımızı çalıştıralım. Ve command1 düğmesini tıklayarak raporumuzun ön izleme penceresine ulaşalım:



Görüldüğü gibi ana/alt bir raporu oluşturmak ve istenilen verileri görüntülemek hiç de zor bir iş değildir.

Adım 9

Şimdi raporun başlığını, alan başlıklarını, bir kişiye yapılan toplam satış miktarını ve genel satış miktarını bulmaya çalışalım.

Bunun için programımızı sonlandırarak tekrar tasarım ortamına dönelim.

Öncellikle raporu sağ tıklayalım. Çıkan menüden **Show page header/footer** seçeneğini seçelim.

Rapordaki alan başlıkları için **paga header** bölümüne label yerleştirelim ve gerekli ayarlamaları yapalım.

Daha sonra alt(detail) kısma ait toplam fiyatların hesaplanması için raporun **MUSTERILER_footer** bölümüne **Function1** elemanını yerleştirelim.

Bunun **FunctionType** özelliğini **rptFucSum**

Datamember özelliğini **MALSATISI**

DataField özelliğini de **TOPLAMFIYATI** yapalım.

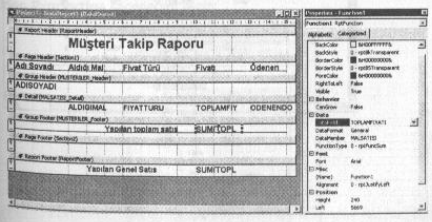
Bundan sonra tüm müşterilere yapılan toplam satışı bulmak için

Raporun **Reportfooter** bölümüne **Function2** elemanını yerleştirelim.

Bunun **FunctionType** özelliğini **rptFucSum**

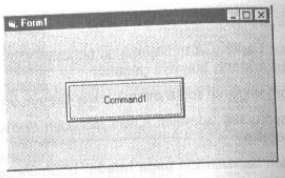
Datamember özelliğini **MALSATISI**

DataField özelliğini de **TOPLAMFIYATI** yapalım.

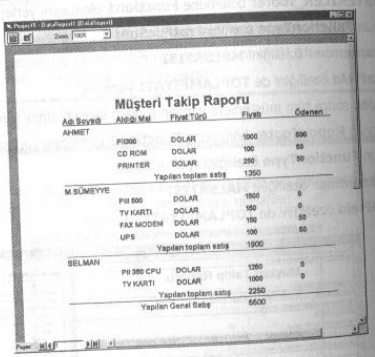


Ayrıca **Page header** ve **MUSTERILER_footer** bölümlerinde yer alan alanların altını çizmek için **line** bileşenini kullanalım.

Programımızı tekrar çalıştırarak rapor ön izleme bölümüne geçelim:



Command1 düğmesini tıkladığımız da raporumuzun çalışma zamanı görüntüsü karşımıza çıkacaktır:



SQL Builder ile Veri Sorgulamak

SQL builder kullanılarak bir yada daha fazla tabloyu sorgulamak, sorgulanan sonuçları bir tablo olarak saklamak, bir tabloyu güncellemek, silmek ve kayıt eklemek mümkündür.

SQL builderin kullanılması için öncellikle DataEnvironment bileşeniyle bir veritabanı dosyasına bağlantının sağlanmış olması gerekiyor.

Şimdi yapılması gereken işlemleri adım adım sıralayalım:

Adım 1

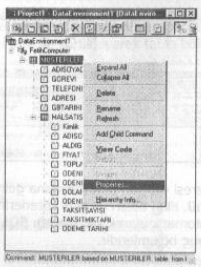
Yeni bir proje başlatın. Ve projeye Dataenvironment bileşenini ekleyin.

Adım 2

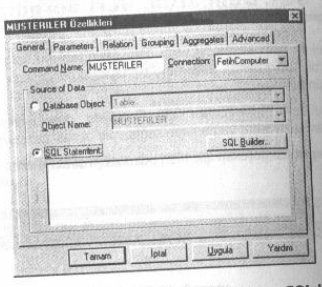
Data Environmet bileşenini bir veritabanı dosyasına bağlayın. Data View penceresinde yer alan tablo yada tabloları DataEnvironment penceresine alın.

Adım 3

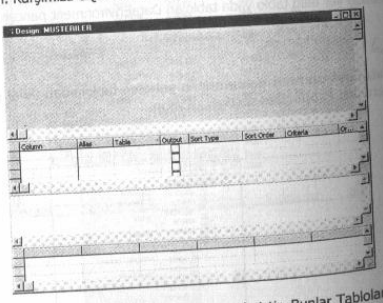
DataEnvironment penceresinde yer alan tablolardan birini sağ tıklayın. Çıkan menüden **Properties** seçeneğini seçin.



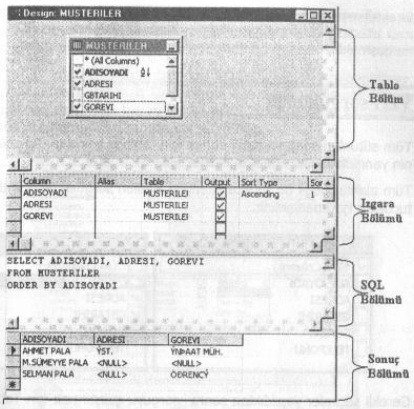
Bu işlemden sonra karşımıza aşağıdaki pencere çıkar.



Bu pencerenin **SQL statement** seçeneğini seçikten sonra **SQL builder** düğmesine basalım. Karşımıza SQL tasarım penceresi çıkacaktır:

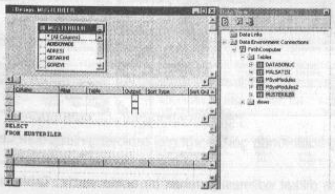


SQL builder penceresi 4 bölümden meydana gelmiştir. Bunlar Tabloların yer alacağı **tablo bölümü**, alan adları ve şart kriterlerinin yer alacağı **Izgara bölümü**, SQL ifadelerinin otomatik olarak yer alacağı **SQL bölümü** ve Sorgu sonuçlarının gösterileceği **sonuç** bölümleridir.



Tablo bölümüne tablo eklemek/kaldırmak

Tablo bölümüne tablo eklemek için **Data View** penceresinde yer alan veri tablolarından istediklerimizi fare ile sürükleyerek tablo bölümüne bırakacağız:

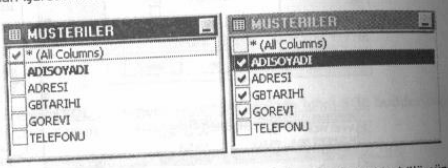


Bıraktığımız tablo bir pencere içinde yer alacaktır. Eğer buradaki bir tablo silinmek istenirse yapılması gereken şey tabloyu aktif hale getirmek ve **delete** tuşuna basmaktır.

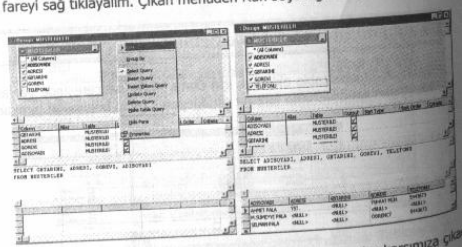
Veri sorgulamak

Tüm sütunları sorguya dahil etmek için tabloda yer alan (All columns) seçeneğinin yanındaki kutucuğu işaretlemek yeterli olacaktır.

Tüm alanlar değil de sadece istenilen alanları sorgulamak için ilgili alanların kutucukları işaretlenmelidir.



Gerekli seçimler yapıldıktan sonra sorguyu çalıştırmak için Tablo bölümünde iken fareyi sağ tıklayalım. Çıkan menüden Run seçeneğini seçelim.

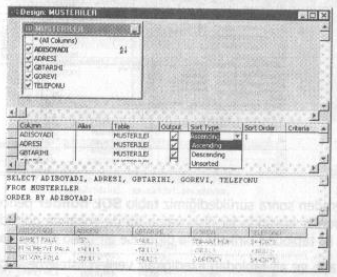


Sonuç bölümünde görüleceği gibi tabloya girilmiş olan veriler karşımıza çıkacaktır.

Burada dikkat edilmesi gereken bir nokta vardır. Tablodaki alanlar işaretlendikleri sırada sonuç bölümünde karşımıza çıkarlar.

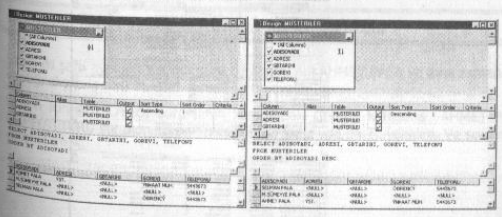
Verileri a-z yada z-a olarak sıralamak

Tablodaki veriler istenilen sütun üzerinde alfabetik olarak sıralanabilirler. Örneğin yukarıdaki tabloda yer alan kişi adlarını a-z olarak sıralamak istersek yapmamız gereken şey Izgara bölümündeki ADISOYADI sütunu karşısındaki **Sort type** kutusunda **Ascending** seçeneğini seçmektir.



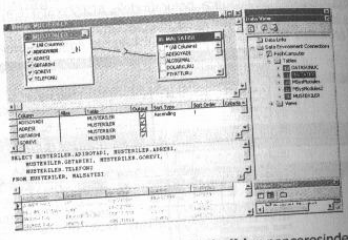
Eğer verileri z-a olarak sıralamak istersek o zaman aynı kutudan **descending** seçeneğini seçmemiz gerekecektir.

Her işlemden sonra tekrar sorguyu çalıştırmamız gerekir.



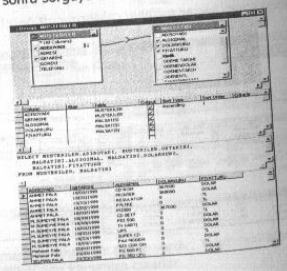
Birden fazla tablo ile çalışmak

Şimdi **Data view** penceresinde yer alan ikinci bir tabloyu da Tablo bölümüne alalım:



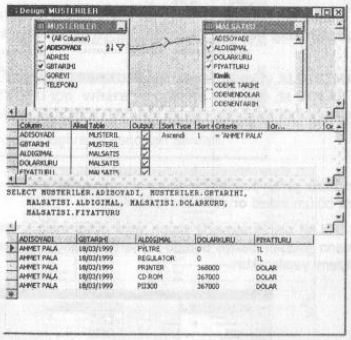
Bu işlemden sonra sürüklediğimiz tablo **SQL builder** penceresinde yer alacaktır. Eğer bunlar arasında bir ilişki varsa bu da yukarıda görüldüğü gibi gösterilecektir. Bu iki tablodan birincisi ana diğer ise alt tablo olarak daha önce belirtilmişti.

Şimdi ana ve alt tablodan istediğimiz verileri listelemeye çalışalım. MUSTERILER tablosunda ADISOYADI alanını ve aynı zamanda a-z olarak sıralanacak şekilde ayarlıyoruz. MALSATISI tablosunda ise ALDIGIMAL, DOLARKURU ve FIYATTURU alanlarını seçtikten sonra sorguyu tekrar çalıştıralım.



Görüldüğü gibi iki yada daha fazla tablo alınarak sorgulanabilir. Yukarıdaki sorgulamadan bir müşterinin kaç kere alış veriş yaptığını ve neler aldığını bulmak mümkündür.

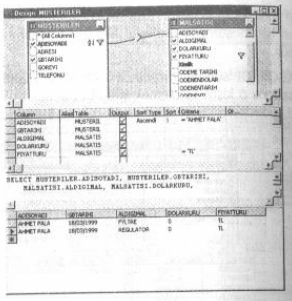
Şimdi ADISOYADI sütunu karşısındaki **Criteria** alanına AHMET PALA yazalım. Bununla sadece o kişiyi listelemek istediğimizi belirtmiş oluyoruz. Ve sonuç aşağıdaki gibi sadece AHMET PALA `ya ait bilgiler listelenecektir:



İki yada daha fazla şartı aynı anda sağlama

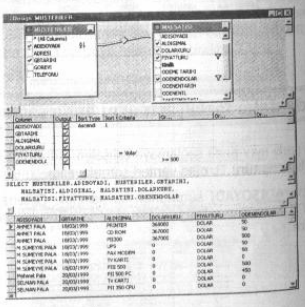
Şimdi iki şartı aynı anda sağlamaya çalışalım. Adı AHMET PALA ve aynı zamanda yaptığı alışveriş türü TL olan verileri sorgulayalım.

Bunun için ADISOYADI sütununun karşısındaki Criteria kutusuna AHMET PALA,FIYATTURU sütunun karşısındaki criteria kutusuna da TL yazalım. Ve sorguyu çalıştıralım.



Birden fazla şartın sadece bir tanesinin gerçekleşmesiyle veri sorgulamak için **Izgara** bölümündeki **or** sütunları kullanılır.

Şimdi DOLAR olarak alışveriş yapan veya aldığı mala karşılık 500 dolar ödeme yapanların listesini alalım. Burada iki şarttan hangisi sağlanırsa sağlansın listeleme işlemi yapılacaktır:



Verileri gruplandırarak sorgulamak

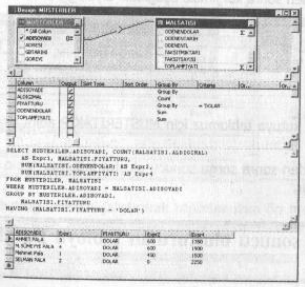
Verileri gruplandırma olayı genel olarak yapılan işlem hakkında kişiye önemli bir fikir verir.

Örneğin bir adam birden fazla alış veriş yapmıştır. Bu adamın yaptığı alış veriş sayısı, toplam olarak ne kadar mal aldığını ve ne kadar kaldığını sorgulayabiliriz.

Bunun için ADISOYADI sütununun karşısındaki herhangi bir sütunu sağ tıklayarak çıkan menüden **Group by** seçeneğini seçelim. Bu işlem **Group by** adında bir sütun ekleyecektir.

Bu sütunda sırasıyla **ADISOYADI** alanı için **Group By**,**ALDIGIMAL** alanı için **count**, **FIYATTURU** için **where** ve **criteria** kutusuna da **DOLAR** yazalım. **ODENENDOLAR** ve **ODENENTOPLAM** alanları için **Sum** fonksiyonunu seçelim.

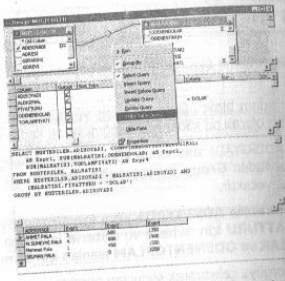
Son olarak sorguyu çalıştırarak sonuçları görelim.



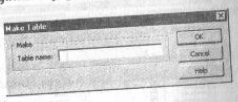
Sorgu sonuçlarını bir tablo olarak saklamak

SQL builder penceresiyle oluşturduğunuz sorguları bir tablo olarak saklayabilirsiniz. Ve dilediğinizde bunları raporunuz da kullanabilirsiniz.

Bunun için tabloyu sağ tıkladıktan sonra çıkan **menüden Make table query** seçeneğini seçiyoruz.



Bu işlemden sonra karşımıza aşağıdaki diyalog penceresi çıkar:



Buradaki kutuya tablomuz için MUSTERITAKIP adını girelim. Ve **OK** düğmesine basalım.

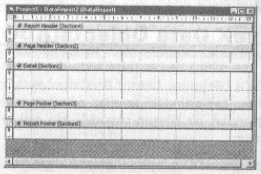
Bu işlemden sonra sorgu sonucu oluşturduğumuz tablo **Data view** penceresinde yer alır.

Sorgu sonucu oluşturulan tabloyu DataReport ile yazdırmak

Sorgu sonucu oluşturduğumuz sorgu tablolarımızı datareport ile kolay bir şekilde yazdırabiliriz. Yeter ki oyunu kuralına göre oynayabilelim!

Yukarda oluşturduğumuz MUSTERITAKIP adlı tabloyu Data view penceresinden sürükleyerek DataEnvironment penceresine bırakalım.

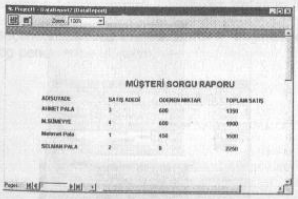
Bundan sonra yapacağımız şey projemize **Datareport** bileşenini eklemek ve **DataReport'un DataSource** özelliğini **DataEnvironment1**, **DataMember** özelliğini de **MUSTERITAKIP** yapmaktır.



Ve yapacağımız işlemlerde biri de Datareport' u sağ tıklamak ve çıkan menüden **Retrieve Structure** seçeneğini seçmek olacaktır.

Son olarak Dataenvironment penceresindeki **MUSTERITAKIP** tablosuna ait veri alanlarını raporun ilgili yerine sürükleyip bırakmak ve rapor tasarımını tamamlamaktır:

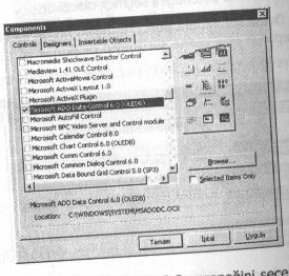
Şimdi f5 tuşu ile programımızı çalıştırarak raporumuzun ön izleme penceresine geçelim:



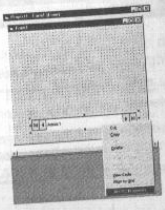
8.2 ActiveX Data Objects(ADO) Bileşeni

ADO bileşeni data bileşenine benzer özellikler taşımaktadır. Şimdi Bir ADO bileşeni ile veritabanı tablosundaki kayıtlar arasında nasıl dolaşacağımızı görelim. ADO bileşeni standart bir bileşen olmadığı için evvela onu toolbox penceresine eklemek gerekiyor.

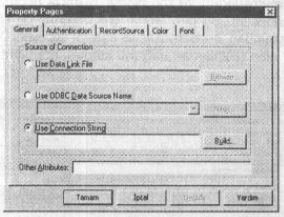
Toolbox penceresine ADO bileşenini eklemek için Toolbox penceresini sağ tıkadıktan sonra çıkan menüden **Components** seçeneğini seçerek Aşağıdaki pencereye ulaşacağız:



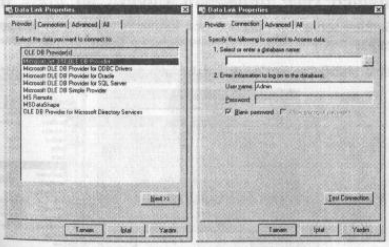
Burada **Microsoft ADO Data Controls 6.0** seçeneğini seçerek **Tamam** düğmesiyle VB ortamına dönüyoruz. ADO elemanını form üzerine alarak sağ tıklayalım:



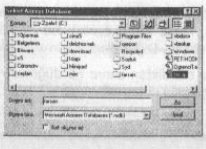
Çıkan menüden **ADODC properties** seçeneğini seçerek aşağıdaki pencereye ulaşalım:



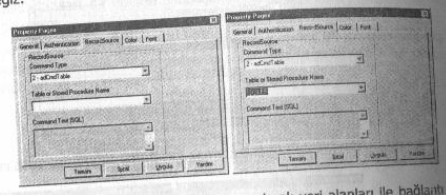
Veritabanı dosyasıyla bağlantı kurmak için **Build** düğmesine basalım.



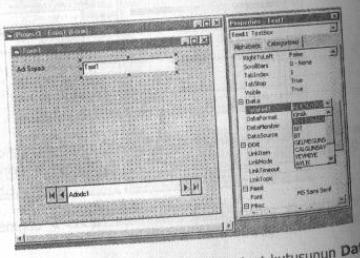
Penceredeki ilk seçeneği seçtikten sonra **Next>>** düğmesine basalım. Ve **...** düğmesiyle diyalog penceresine ulaşalım.



Karşımıza çıkacak olan diyalog penceresinden veritabanı dosyamızı seçelim. **Test connection** düğmesiyle veritabanı ile bağlantının başarılı olup olmadığını anlamak mümkündür. Ve son olarak **Tamam** düğmesiyle **Property page** penceresine dönelim. Bu pencerenin **RecordSource** kısmına geçerek **Command type** kutusundan **2-adcmdtable** öğesini seçelim. Dolayısıyla veritabanı dosyasında yer alan tablolardan bağlantı kurmak istediğimizi bir alttaki kutudan seçebileceğiz:

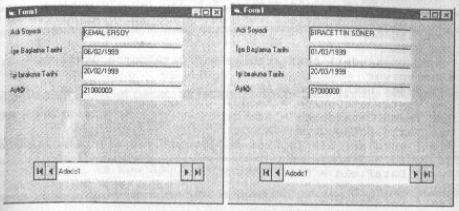


Bundan sonra form üzerine text kutularını alarak veri alanları ile bağlantı kuracağız:



Yukarıda görüldüğü gibi form üzerine alınan her text kutusunun **DataSource** özelliği **Adodc1**, **Data Field** özelliği ise istenilen bir veri alanı ile bağlantısının sağlanması gerekiyor. Diğer alanlar için aynı işlem devam ediyor.

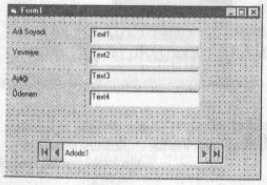
Form tasarımını bitirdikten sonra F5 düğmesiyle programımızı çalıştırarak kayıtlar arasında rahatlıkla dolaşabiliyoruz:



Kayıtlar arasında gezinirken herhangi bir alanda yapılan değişiklik anında kaydediliyor.

Program kodu ile Adodc1 bileşeninin veri bağlantılarını düzenlemek

Form üzerine adodc1, dört tane de text kutusu alarak aşağıdaki tasarımı gerçekleştirelim:



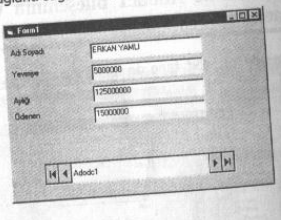
Properties penceresinde herhangi bir ayarlama yapmadan Form_load() olayına aşağıdaki kodu yazalım:

```
Private Sub Form_Load()
With Adodc1
'veritabanı dosyası ile bağlantı kuruluyor.
.ConnectionString = "Provider=Microsoft.Jet.OLEDB.3.51;Persist
Security Info=False;Data Source=C:\tarsan.mdb"
'tablodaki alanlar sorgulanıyor.
```



```
.RecordSource = "select [ADI SOYADI],YEVMIYE,AYLIK, ÖDENEN,KALAN
from İŞÇİLER"
End With
'Form üzerindeki text kutularının DataSource özelliği
ayarlanıyor.
Set Text1.DataSource = Adodc1
Set Text2.DataSource = Adodc1
Set Text3.DataSource = Adodc1
Set Text4.DataSource = Adodc1
'Text kutuları ile İşçiler tablosunda yer alan alanlar arasında
bağlantı sağlanıyor.
Text1.DataField = "ADI SOYADI"
Text2.DataField = "YEVMIYE"
Text3.DataField = "AYLIK"
Text4.DataField = "ÖDENEN"
End Sub
```

Programımızı F5 tuşuyla çalıştıralım. Görüldüğü gibi program kodu yazılarak ta Adodc bileşeniyle bağlantı sağlanabiliyor:



Properties

BOF,EOF

Bir kayıtların başına gelinip gelinmediğini anlamak için BOF, kayıtların sonuna gelinip gelinmediğini anlamak için de EOF özellikleri kontrol edilir.

ConnectionString

Veri kaynağı ile bağlantının sağlanması için gerekli bilgiyi içerir. Bu özellik birden fazla parametre aynı anda kabul eder.


```
Provider=Microsoft.Jet.OLEDB.3.51;Persist Security
Info=False;Data Source=C:\tarsan.mdb
```

Recordset

Kullanılacak RecordSet nesnesini belirler. **Recordset** özelliği **set** özelliği ile beraber kullanılır.

Örneğin

```
Dim rsmusteri As New ADODC.Recordset
Set ADODC1.Recordset = rsMusteri
```

RecordSource

Bağlanılacak olan veritabanına ait tabloyu yada sorguyu belirler.

MaxRecords

Maximum kayıt sayısını verir.

Events

WillMove(ByVal adReason As ADODB.EventReasonEnum, adStatus As ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)

Bu olay **Recordset** içinde mevcut kaydın değişmesinden hemen önce meydana gelir.

Olayda yer alan adReason bileşeni olayın nedenini belirler. Ve adRsnMoveFirst, adRsnMoveLast, adRsnMoveNext, dRsnMovePrevious, adRsnMove ve adRsnRequery değerlerinden birini alır.

MoveComplete(ByVal adReason As ADODB.EventReasonEnum, ByVal pError As ADODB.Error, adStatus As ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)

RecordSet içinde mevcut kayıt pozisyonunun değişmesiyle **MoveComplete** olayı meydana gelir.

WillChangeField(ByVal cFields As Long, Fields As Variant, adStatus As ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)

Bu olay Recordset içinde bir yada daha fazla hücre içeriği değişmeden hemen önce meydana gelir.

Burada **cFields** veri alan sayını belir. **Fields** bir variant dizi olup veri alanlarını içerir.

RecordsetChangeComplete(ByVal adReason As ADODB.EventReasonEnum, ByVal pError As ADODB.Error, adStatus As ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)

Bir yada daha fazla kayıt değiştiğinde meydana gelir.

WillChangeRecord(ByVal adReason As ADODB.EventReasonEnum, ByVal cRecords As Long, adStatus As ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)

Recordset içinde bir yada daha fazla satır değişmeden bu olay meydana gelir.

WillChangeRecordset(ByVal adReason As ADODB.EventReasonEnum, adStatus As ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)

RecordSet içinde herhangi bir değişiklik meydana gelmeden önce meydana gelir.

RecordsetChangeComplete(ByVal adReason As ADODB.EventReasonEnum, ByVal pError As ADODB.Error, adStatus As ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)

Recordset değiştikten sonra bu olay meydana gelir.

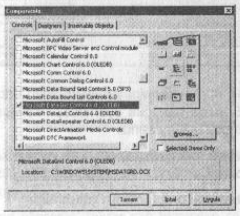
EndOfRecordset(fMoreData As Boolean, adStatus As ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)

Recordsetin en sonuna bir girişimde bulunulduğu zaman bu olay meydana gelir.

DataGrid

Veritabanı programlarında en çok kullanacağımız bileşenlerden biridir. Bunun çok kullanılmasının sebeplerinden biri de birden fazla kaydı aynı anda göstermesi birden çok girişin aynı anda yapılabilmesidir.

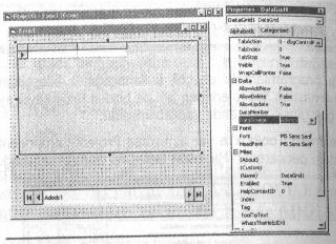
Eğer bu eleman Toolbox'ta yer almıyorsa Project-Components menüleri ile açılan aşağıdaki pencereden Microsoft DataGrid Control seçeneğini işaretleyin.



DataGrid Bileşenini ADO bileşenine bağlamak

Yukarıda bir ADO bileşeninin nasıl veri kaynağına bağlanacağını açıkladık. Şimdi bir datagrid bileşenini nasıl ADO bileşenine bağlayacağımızı gösterelim.

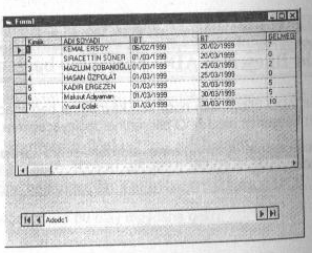
DataGrid bileşeninin form üzerine aldıktan sonra **properties** penceresinde **DataSource** özelliğini ADODC1 Yapalım.



Eğer form üzerine ADO bileşenini yerleştirmeden önce DataGrid bileşenini yerleştirip ve **DataSource** özelliğini ayarlamaya kalkışırsanız o zaman bağlantıyı sağlayacak bir sunucu bulamazsınız.

İlk önce ADO bileşeni form üzerine alınacak ve veri kaynağıyla bağlantısı sağlanacaktır. Daha sonra DataGrid alınarak gerekli düzenlemeler yapılacaktır.

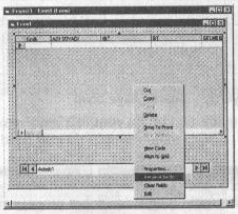
Şimdi F5 tuşuna basarak programımızı çalıştıralım.



Görüldüğü gibi veritabanı dosyası içindeki **İşçiler** tablosuna ait tüm veriler DataGrid bileşeni içinde yer aldılar. Kayıtlar arasında yön tuşları ile rahat bir şekilde dolaşmak mümkündür. Fakat henüz yeni kayıt girişleri yapamıyoruz. Yeni kayıt girişlerine imkan sağlamak, istenilen sütunları göstermek ve DataGrid içindeki verileri biçimlemek için gerekli ayarlamaları yapmak gerekir. Bunları adım adım açıklayacağız.

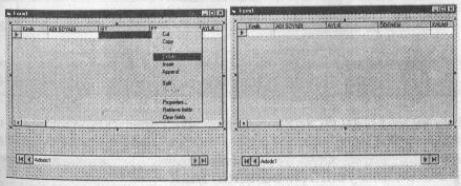
Sadece istenilen Sütunları göstermek

Şimdi DataGrid bileşeni sağ tıklayalım çıkan menüden **Retrieve Fields** seçeneğini seçelim:



Bunu yaptığımızda tabloya ait tüm alan başlıkları DataGrid bileşeninin sütunlarında yer alacaktır. Tabii ki biz bunların tamamını değil sadece istediklerimizi görüntülemek istiyoruz.

Bunun için DataGrid bileşenini tekrar sağ tıklayalım çıkan menüden **Edit** seçeneğini seçelim. Ve artık silmek istediğimiz sütun başlığını seçtikten sonra fareyi sağ tıklayarak çıkan menüden **Delete** seçeneğiyle seçili sütunu silelim.



Edit modunda iken herhangi bir sütunun sağ tıklanmasıyla açılan menüdeki seçenekleri açıklamaya çalışalım:

Delete

Seçili olan sütunları siler.

Insert

Seçili olan sütunden bir önce yeni bir sütun ekler.

Append

En sonda yer alan sütunun yanına yeni bir sütun ekler.

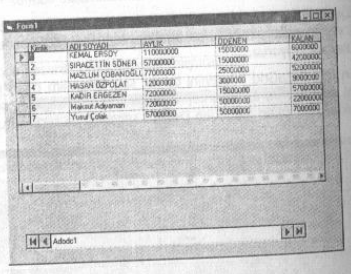
Split

Tabloyu seçili yerden itibaren ikiye böler.

Clear Fileds

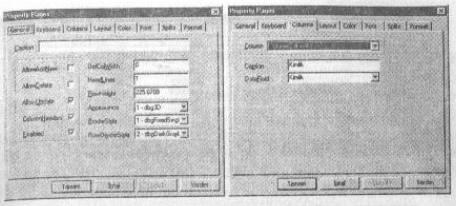
Sütun başlıklarını siler.

Şimdi sütun düzenleme işleminden sonra programımıza bir göz atalım:

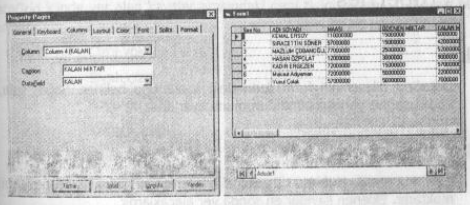


Sütunları yeniden adlandırmak

Database tablosunda tanımladığınız alan başlıklarını DataGrid bileşeninde kullanmak istemeyebilirsiniz. Farklı bir isim kullanmak isteyebilirsiniz. Bunun için DataGrid bileşeni sağ tıklanacak çıkan menüden **properties** seçeneği seçilecektir. Karşımıza çıkacak olan aşağıdaki pencerenin **Columns** bölümüne geçiş yapacağız:



Burada yapılması gereken işlem **Column** kutusunda sütun başlıklarını seçmek Ve **caption** kutusunda buna yeni bir isim vermektir:

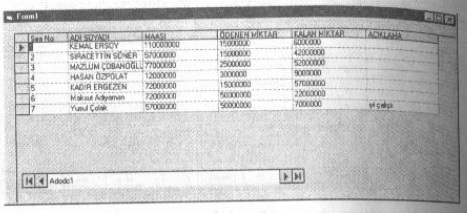


Görüldüğü gibi istediğiniz sütunun başlığını değiştirme hakkına sahipsiniz.

DataGrid bileşeninin form üzerinde ayarlamak

Formu büyüttüğünüz oranında yatay yada dikey olarak DataGrid bileşeninde büyümesini istiyorsanız aşağıdaki program kodunu Formun Resize() olayına yazın:

```
Private Sub Form_Resize()
'Form uzadığında Dikey olarak Datagrid1 uzat
DataGrid1.Height = Me.ScaleHeight
'Form genişlendiğinde yatay olarak Datagrid1 genişlet
DataGrid1.Width = Me.ScaleWidth
End Sub
```

Properties

Align

DataGrid bileşeninin form üzerindeki konumunu belirler. Sağa, sola, yukarıya yada aşağıya yanaşık olma durumunu ayarlar.

AllowAddNew

Eğer bu özellik true ise DataGrid bileşenine yeni kayıt eklemek mümkün olmaktadır. Eklenen kayıtlar en sonda yer alacaklardır.

AllowArrows

Bu özellik true ise DataGrid hücreleri arasında yön tuşları ile dolaşılabilir.

AllowDelete

Bu özellik true ise istenilen bir kayıt (satır) delete tuşu ile silinebilir.

AllowUpdate

Bu özellik true ise yapılan değişiklikler otomatik olarak kaydedilecektir.

```
DataGrid1.AllowUpdate=true
```


ColumnHeaders

Alan isimlerinin yazılı olduğu sütun başlıklarını gizler (false) yada gösterir(true).

HeadLines

Sütun başlıkları için ayrılacak satır sayısını belirler. Varsayılan değeri 1 dir ve sütun başlıkları (alan isimleri) bir satırlık bölgede yazılır. Bu özelliğe 0 değeri verilirse alan başlıkları ortadan kalkar. Eğer uzun alan isimleri söz konusu ise bu değer artırılarak alan isimlerinin bir kaç satıra yazılması sağlanabilir.

RowDividerStyle

DataGrid içinde yer alan kayıtlar arasındaki çizgi tipini belirler. 0-5 arası bir değer alır.

0	Çizgi yok
1	Siyah çizgi
2	Koyu gri çizgi
3	Kabarık çizgi
4	Basık çizgi
5	ForeColor özelliği ile belirlenmiş renkteki çizgi.

TabAction ,WrapCellPointer

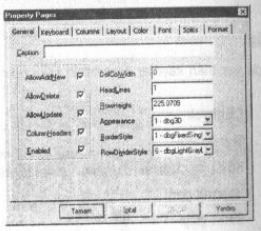
Tab tuşunun kontroller arasında gezmek için kullanıldığını biliyorsunuz. Ancak Grid söz konusu olunca Tab tuşu ile hücreler arasında da gezilmek istenebilir. **tabAction** özelliği Tab tuşunun hangi amaçla kullanılacağını belirler.

dbgControlNavigation	0	Tab tuşu kontroller arasında gezmek için kullanılır. Kullanıcı hücreler arasında gezmek için yön tuşlarını kullanabilir.
dbgColumnNavigation	1	Tab tuşu hem hücreler hem de kontroller arasında gezmek için kullanılır. Grid içinde Tab tuşuna basılarak sonraki kolona geçilebilir. Ancak son kolonda ise alttaki satıra geçmek yerine sonraki kontrole geçer.
dbgGridNavigation	2	2 değeri gibidir ancak farklı olarak **WrapCellPointer** özelliği True yapılarak Tab tuşunun tamamen hücreler arasında

gezinmek için kullanılmasını sağlar. Bir satırdaki son hücreye gelindiğinde otomatik olarak alttaki hücreye geçer.

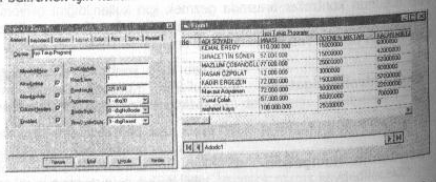
DataGrid bileşeninin diğer özellikleri

Bunun yanında DataGrid sağ tıklanarak çıkan menüden **Properties** seçeneği seçilere aşağıdaki pencereye ulaşılacaktır. Burada DataGrid bileşenine ait birçok özelliği kolay bir şekilde bulmak ve ayarlamak mümkündür:



- Pencerenin ilk bölümü olan **general** bölümünde aşağıdaki işlemler yapılır:

Caption: Datagrid bileşeninin başlık kısmını belirtmek için kullanılır:



AllowAddNew Seçeneği işaretli ise DataGrid bileşenine yeni kayıtlar eklenebilir. İşaretli değilse eklenemez.

AllowDelete Seçeneği işaretli ise istenilen kayıt(satır) seçilerek delete tuşuyla silinebilir.

AllowUpdate Seçeneği işaretli ise yapılan değişiklikler anında kaydedilir.

ColumnHeaders Seçeneği işaretli ise sütun başlıkları gözükür değilse gözükmez.

Enabled Seçeneği işaretli ise Datagrid bileşeni aktif olur yani üzerinde işlem yapılabilir değilse yapılmaz.

DefColWidth 0 Kutusunda sütunların genişlikleri belirlenir.

HeadLines 1 Sütun başlıkları için yatay satır sayısını belirler.

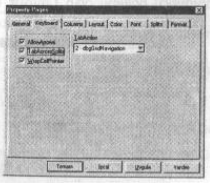
RowHeight 225.0709 Satır yükseklikleri burada belirlenir.

Appearance 1 - dbg3D Datagrid bileşeninin görünüşü ayarlanır.

BorderStyle 1 - dbgFixedSingl Datagrid bileşeninin çerçeve tipi belirlenir. İlk seçenek çerçeveli hali ikinci seçenek ise çerçevesiz halidir.

RowDividerStyle 6 - dbgLightGrayL Satır çizgilerinin biçimini ayarlar.

- Pencerenin ikinci bölümü olan **Keyboard** bölümünde aşağıdaki işlemler yapılır:



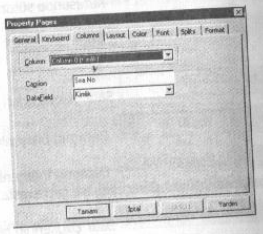
AllowArrows Seçeneği işaretli ise yön tuşlarıyla kayıtlar arasında dolaşılabilir.

TabAcrossSplits Seçeneği işaretli ise DataGrid içinde bölünen kısım içinde tab tuşunun kullanılmasını sağlar.

WrapCellPointer Seçeneği işaretli ise hücreler arasında dolaşılırken son sütuna geldiğinde imleç otomatik olarak bir sonraki satırın ilk hücresine konumlandırılacaktır.

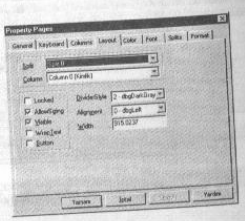
TabAction 2 - dbgGridNavigation Bu kutu aracılığıyla Tab tuşunun görevi ayarlanır.

- Pencerenin üçüncü bölümü olan **Columns** bölümünde aşağıdaki işlemler yapılır:



Penceredeki **column** kutusunda veri alanları ve hangi sütunde yer aldıkları belirtilmiştir. Caption kutusuna seçili olan veri alanına ait yeni bir sütun ismi girilir.

- Pencerenin dördüncü bölümü olan **Layout** bölümünde aşağıdaki işlemler yapılır:



Bu bölümde genel olarak **column** kutusunda seçili olan alan üzerinde bazı ayarlamalar yapılır.

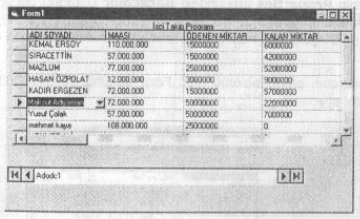
Locked Seçeneği seçili ise o hücreye bilgi girişi yapılamaz.

AllowSizing Seçeneği seçili ise o sütun fare yardımıyla istenilen genişliğe getirilebilir.

Visible Seçeneği seçili ise sütun görünür, değilse görünmez.

WrapText Seçeneği seçili ise bir hücreye yatay olarak sığmayan ifadeler alt satırdan itibaren devam eder.

Button Seçeneği seçili ise hücrenin sağ tarafına aşağı doğru açılan bir düğme eklenir.

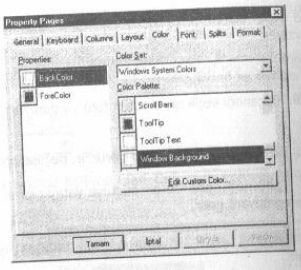


DividerStyle 2 - dbgDarkGray Kutusunda sütun çizgilerinin biçimi ayarlanır.

Alignment 0 - dbgLeft Bir hücrenin içerik durumunu belirler. Sağa, sola yada ortaya alınabilir.

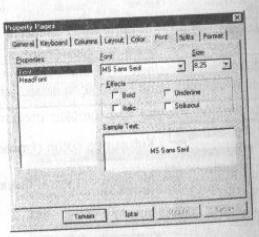
Width 915.0237 Kutusunda ilgili sütunun genişliği belirlenir.

- Pencerenin beşinci bölümü olan **Color** bölümünde DataGrid bileşeninin yazı ve zemin renkleri ayarlanır:

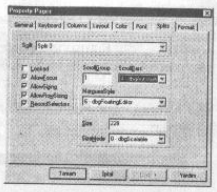


Pencerenin **Properties** listesindeki **Backcolor** Datagrid bileşeninin zemin rengini ayarlarken **Forecolor** ise bileşenin yazı rengini belirler. Ayarlama yapıldıktan sonra **Uygula** düğmesine basılarak yapılan ayarlamaların etkisi anında görülebilir.

- Pencerenin altıncı bölümü olan **Font** bölümünde DataGrid bileşeninin font biçimi ayarlanır.

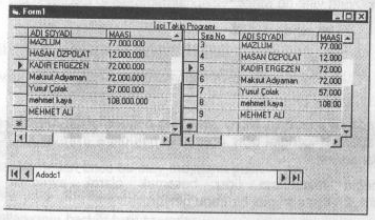


- Pencerenin yedinci bölümü olan **splits** bölümünde aşağıdaki ayarlamalar yapılır:



Burada genel olarak eğer tablo bölünmüşse onunla ilgili ayarlamalar yapılır. **Split** kutusundaki seçili **split**(bölüm) üzerinde aşağıdaki ayarlamalar etkili olur:

Locked Seçeneği seçili ise ilgili bölüme(split) bilgi girişi yapılamaz.



AllowFocus Seçeneği seçili ise ilgili bölüme(**split**) geçiş yapılabilir.

AllowSizing Seçeneği seçili ise ilgili bölüm(**split**) yeniden boyutlandırılabilir.

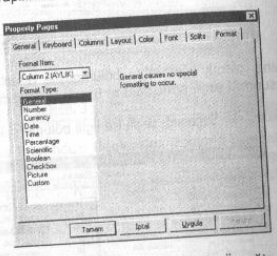
AllowRowSizing Seçeneği seçili ise ilgili bölümün(**split**) satır büyüklüğü ayarlanabilir.

RecordSelectors Seçeneği seçili ise ilgili bölüme(**split**) ait kayıtlar seçilebilir.

ScrollBars
4 - dbgAutomatic Kutusunda DataGrid bileşenine eklenecek olan yatay, dikey, otomatik veya hem yatay hem de dikey kaydırma çubuklarını belirler.

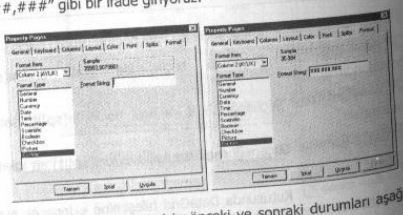
MarqueeStyle
6 - dbgFloatingEditor Kutusunda hücreler ile ilgi ayarlamalar yapılır. Hücrelerin satır satır seçilmesi isteniyorsa bu kutudan **dbgHighlightRow** seçilmelidir.

- Pencerenin son bölümü olan **Format** bölümünde DataGrid bileşenine ait sütun biçimleri yapılır:

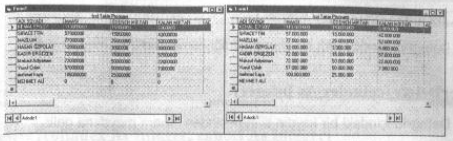


Burada bir hücrenin biçimi kolaylıkla değiştirilebilir. Örneğin sayısal içerikli olan bir sütunun içeriğini üçer üçer basamak olara ayırmak için **Custom** seçeneğini kullanabiliriz.

Örneğin rakamların üçer basamak olarak ayrılması için bu kutuya "###,###,###" gibi bir ifade giriyoruz:



Buna karşılık DataGrid bileşenimizin önceki ve sonraki durumları aşağıdaki gibi olacaktır:



Events

AfterColEdit(ByVal ColIndex As Integer)

Herhangi bir hücreye bilgi girişi tamamlandıktan sonra bu olay meydana gelir.

AfterColUpdate(ByVal ColIndex As Integer)

Datagrid içerisinde bir verinin taşınmasıyla meydana gelir.

BeforeColEdit(ByVal colindex As Integer, ByVal keyascii As Integer, cancel As Integer)

Bir kolon düzenlemeden önce bu olay meydana gelir.

Colindex:Bilgi girişi yapılan sütunu belirler.

Keyascii:Kullanıcının bastığı tuşu tanımlar.

ColEdit(ByVal colindex As Integer)

Herhangi bir hücreye bir karakterin girilmesiyle meydana gelir.

Error(ByVal dataerror As Integer, response As Integer)

Bir hata olduğu zaman meydana gelir.

OnAddNew()

Yeni bir kayıt girişiminde bulunulduğu zaman meydana gelir.

SelChange(Cancel As Integer)

Datagrid içinde farklı bir alan seçildiği zaman bu olay meydana gelir.

AfterDelete (colindex As Integer)

Datagrid içindeki bir kaydın silinmesinden sonra meydana gelir.

```
Private Sub DataGrid1_AfterDelete()
 MsgBox "kayıt silindi"
End Sub
```

AfterInsert ()

Datagrid bileşenine yeni bir kayıt eklendiği zaman meydana gelir.

```
Private Sub  DataGrid1_AfterInsert ()
 If DataGrid1.Columns(1).Value <> "" Then
  Data2.Recordset.AddNew
  Data2.Recordset.Fields("SIRANO") = DataGrid1.Columns(1).Value
  Data2.Recordset.Update
 End If
End Sub
```

AfterUpdate ()

Datagrid bileşeninde yapılan değişiklik veritabanı dosyasına yazıldıktan sonra meydana gelir.

```
Private Sub DataGrid1_AfterUpdate ()
    Label1.Caption = "En son olarak " & Format$(Now, "Long Date" &
"tarihinde değişiklik yapıldı")
End Sub
```

BeforeColUpdate (colindex As Integer, oldvalue As Variant, cancel As Integer)

Datagrid bileşenine bilgi girişinden sonra fakat bilginin güncellemesinden önce meydana gelir. **ColIndex** parametresi ile değişiklik yapılmak istenen kolon numarasını, **OldValue** parametresi ile de değiştirilmeden önceki değerini öğrenebilirsiniz. Değişiklik iptal edilmek isteniyorsa **Cancel** parametresine **True** değeri atanması yeterlidir.

```
Private Sub DataGrid1.BeforeColUpdate (ColIndex As Long, OldValue
As Variant, Cancel As Integer)
```



```
 If ColIndex = 1 Then
  If DataGrid1.Columns(1).Value < Now Then
   Cancel = True
   MsgBox "Bu günün tarihinden sonraki bir tarih girmelisiniz."
  End If
 End If
End Sub
```

ColResize (colindex As Integer, cancel As Integer)

Datagrid içinde bir sütun yeniden boyutlandırıldığı zaman meydana gelir.

HeadClick (colindex As Integer)

Datagrid içindeki sütun başlıklarından biri tıklandığı zaman meydana gelir.

```
Private Sub DataGrid1_HeadClick (ColIndex As Integer)
   Data1.RecordSource = "Select * From MUSTERILER Order By " & _
   DataGrid1.Columns(ColIndex).DataField
   Data1.Refresh
End Sub
```

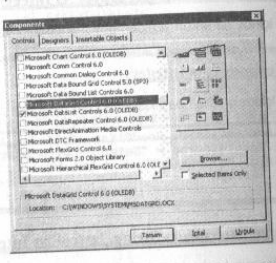
RowColChange (lastrow As String, lastcol As Integer])

Mevcut hücrenin başka bir hücre olmasıyla meydana gelir. Hangi hücreden gelindiği **LastRow** ve **LastCol** parametreleri ile öğrenilebilir.

DataList / DataCombo

Bu bileşenleri veritabanı dosyasında yer alan ve bir alana ait kayıtların tamamını gösterilebilir. Her ikisi de verileri liste formatında gösterir.

Bu kontrolleri projede kullanabilmek için: Project-Components menü seçeneğini tıklayarak aşağıdaki pencereye ulaşacağız:



Burada **Microsoft DataList Controls 6.0** bileşenini seçtikten sonra **Tamam** düğmesiyle VB ortamına dönüyoruz. Bu işlemle iki bileşen de Toolbox a eklenir. Çünkü her ikisi de aynı ocx dosyasındadırlar.

Properties

RowSource, ListField

Her iki kontrolde hem listeleme hem de veri tabanındaki bilgileri değiştirme işlevini görebilmektedir. Listeleme işleminin hangi tablonun hangi alanından olacağını bu iki özellik belirlenir. **RowSource** özelliğine bir veri tabanı ile bağlantısı sağlanmış **Adodc** kontrolünün ismi verilir. Böylece o veri tabanındaki tablo liste tarafından kullanılabilir hale gelecektir. Tablodaki hangi alanın listeleneceği ise **ListField** özelliği ile belirlenir.

DataSource, DataField

Liste kontrollerinin bir diğer özelliği de bir alandaki bilgiyi değiştirebilmesidir. Eğer liste üzerinden seçilen bir elemanın herhangi bir alana yazılmasını isterseniz **DataSource** ve **DataField** özellikleri ile tabloyu ve alanı belirlemeniz gerekir.

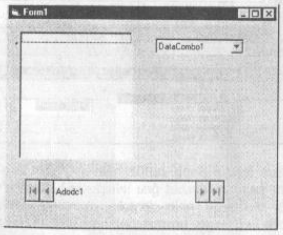
SelectedItem

Listelerde seçili olan elemanın yeri bu özellikle öğrenilebilir. Bu özellikten döner değer BookMark türündendir. Dolayısıyla herhangi bir değişkene alındıktan sonra Bookmark özelliğine atanarak eski yere dönülebilir.

BoundColumn

Tabloya girilecek olan veri alanını tanımlar.

ÖRNEK: Şimdi form üzerine Adodc1 Datalist1 ve DataCombo1 bileşenlerini yerleştirelim:

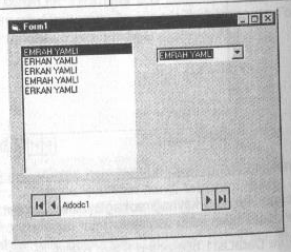


Adodc1 elemanını bir veri kaynağına bağlayalım. Datalist1 ve Datacombo1 bileşenlerini de Adodc1 bileşenine bağlayalım.

DataCombo1 ve DataList1 bileşenleri 5 ayrı özellik ile veri setlerine bağlanırlar.

İki özellik bunların güncellenmesi içindir:

Bileşen Adı	Özelliği	Özelliğin aldığı değer
Adodc1	RecordSource	ÇALIŞANLAR
	CommandType	2-adcmdTable
	ConnectionString	Provider=Microsoft.Jet.OLEDB.3.51;Persist Security Info=False; Data Source=C:\tarsan.mdb
DataList1	DataSource	Adodc1
	RowSource	Adodc1
	ListField	ADI SOYADI
	BoundColumn	ADI SOYADI
	DataField	ADI SOYADI
DataCombo1	DataSource	Adodc1
	RowSource	Adodc1
	ListField	ADI SOYADI
	BoundColumn	ADI SOYADI
	DataField	ADI SOYADI



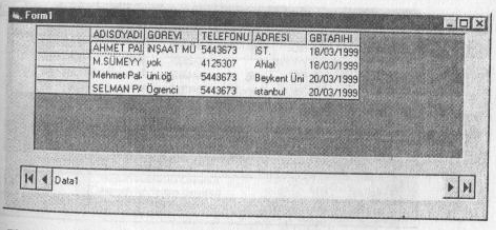
MsFlexGrid Bileşeni

Verileri tablo biçiminde gösteren bir bileşendir. Bu bileşenle veriler sıralanabilir, sütunları yeniden boyutlandırılabilir ve aynı zamanda resimlerde gösterilebilir. Eğer **Data** bileşenine bağlı olarak kullanılırsa veriler yalnız okunur bir şekilde kontrol edilebilir.

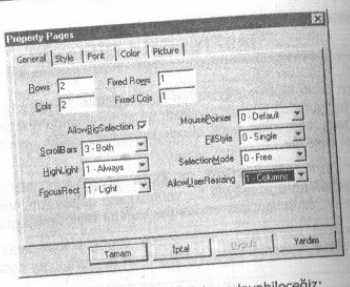
Form üzerine **MsflexGrid** ve **Data** bileşenlerini alalım. **Data** bileşenini **DatabaseName** özelliğiyle bir veritabanına bağlayalım. Ve **RecordSource** özelliğini de Veritabanı dosyasında yer alan bir tablo ile eşleştirelim.

MsflexGrid bileşeninin **DataSource** özelliğini de data1 yapalım. Böylece **MsflexGrid** bileşenini Data1'e Data1'i de bir veritabanı dosyasına bağlamış olduk.

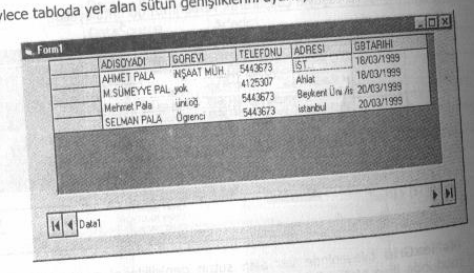
Programımızı F5 tuşuyla çalıştırarak verilerimizi görüntüleyelim.



MsFlexGrid bileşeninde yer alan sütun genişliklerini ayarlamak için tasarım modunda iken MsFlexGrid bileşenini sağ tıklayalım çıkan menüden **properties** seçeneğini seçelim. Karşımıza çıkacak olan aşağıdaki diyalog penceresinin **allowUserResizing** kutusundan 1-**columns** seçeneğini seçelim.



Böylece tabloda yer alan sütun genişliklerini ayarlayabileceğiz:

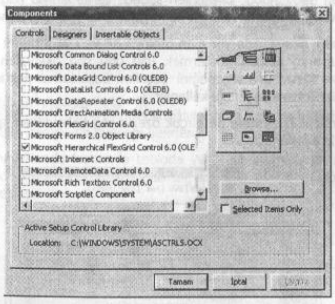


MSHFlexGrid

Verileri düz tablo biçiminde gösteren bir bileşendir. Bu bileşenle veriler sıralanabilir, sütunları yeniden boyutlandırılabilir ve aynı zamanda resimler de gösterilebilir.

MsflexGrid Data bileşeni ile birlikte kullanılırken MSHFlexGrid **ADODC** bileşenine bağlı olarak kullanılmaktadır. Ve dolayısıyla veriler yalnız okunur bir şekilde kontrol edilebilir.

Yukarıda olduğu gibi aynı şekilde form üzerine **MSHFlexGrid** ve **Adodc** bileşenlerini yerleştirelim. **Adodc** bileşenini sağ tıklayarak çıkan **properties** menüsünü seçerek veritabanı bağlantısını sağlayalım. Bu bileşen normal olarak toolbox penceresinde yer almaz. Bunu eklemek için toolbox penceresini sağ tıklayalım. Çıkan menüden **components** seçeneğini seçerek Components penceresine ulaşalım.



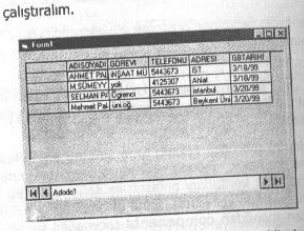
Penceredeki ☑ Microsoft Hierarchical FlexGrid Control 6.0 (OLE elemanını seçerek **Tamam** düğmesine basalım.

Bileşeni toolbox penceresine aldıktan sonra artık form üzerine alabiliriz. Bunun yanında veritabanı ile bağlantı sağlamak için Adodc bileşenini de form üzerine alalım.

Adodc bileşenini sağ tıklayalım çıkan properties menüsünü kullanarak daha önce yaptığımız gibi bir veritabanı dosyasıyla bağlantı sağlayalım.

Properties penceresinde **MSHFlexGrid** bileşenine ait **DataSource** özelliğini Adodc yapalım.

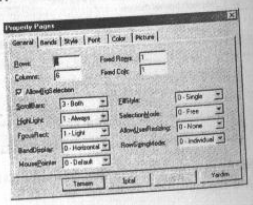
Ve programımızı çalıştıralım.



MSFlexGrid bileşeninde olduğu gibi verileri bir tablo şeklinde gösterebiliriz. Ve kullanılan data, Adodc bileşenleri sadece veritabanıyla bağlantı sağlanması açısından kullanılmışlardır. Bunlar ile kayıtlar arasında dolaşamayız.

Veri tabanı dosyasında yer alan tabloya ait veri başlıklarını **MSFlexGrid** bileşeninde tasarım anında göstermek İçin onu sağ tıklayalım ve çıkan menüden **Retrieve Structure** seçeneğini seçelim. Başlıkları silmek için de aynı menüden **Clear Structure** seçeneği kullanılır.

MSFlexGrid bileşenine ait birçok özellik sağ tıklandıktan sonraki properties seçeneğiyle ayarlanabilir.



ScrollBars: 3 - Both Seçeneğiyle bileşene yatay, dikey kaydırma çubukları eklenir.

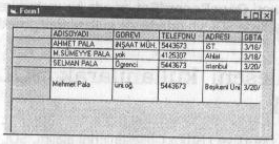
HighLight: 1 - Always Seçeneğiyle sütun başlıkları tıklandığında ilgili sütuna ait verilerin seçilip seçilmeyeceğini belirler.

Buradaki **1-Always** seçeneğiyle veriler seçilirken **0-Never** seçeneğiyle seçim yapılmaz.

FocusRect: 1 - Light Seçeneğiyle hücrelerin seçim biçimi ayarlanır.

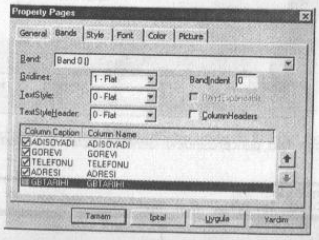
SelectionMode: 0 - Free Seçeneğiyle hücrelerin seçilme şekli ayarlanabilir.

AllowUserResizing: 0 - None Seçeneğiyle kullanıcının satır,sütun genişliklerini ayarlama yetkisinin olup olmayacağı belirlenir.



RowSizingMode: 0 - Individual Seçeneğiyle de satırların birbirinden bağımsız olarak genişletilip genişletilmeyeceği belirlenir.

Aynı pencerenin **Bands** kısmında tabloda yer alan sütun başlıklarının sıralanış şekli belirleneceği gibi tamamının yada bir kısmının gösterilip gösterilmeyeceği işlemi de gerçekleştirilir.

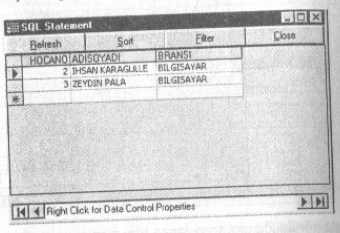


Bölüm 8

Drag & Drop

Sürükle-Bırak

Ve **Run** düğmesiyle sorguyu çalıştırarak sonucu görelim:



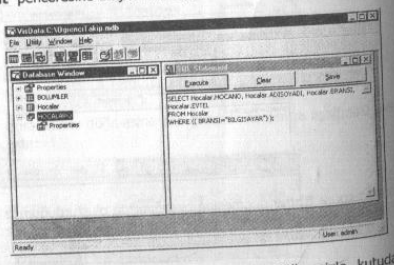
Görüldüğü gibi seçili olan ilk üç alan, branşı "bilgisayar" olanlar ve a-z alfabetik olarak sıralanmış şekilde karşımıza çıkmaktadır.

Pencerede yer alan **Refresh** düğmesi tabloyu günceller.

Sort düğmesi numarası girilen sütuna göre sıralama yapar.

Filter düğmesi ise yukarıdaki gibi belli şartların sonucunu görüntüler. **Close** düğmesi ise pencereyi kapatır.

Program **save** düğmesiyle kaydedilip normal ortama dönüldüğünde **SQL statement** penceresine oluşturulan SQL kodlarını aktarır:

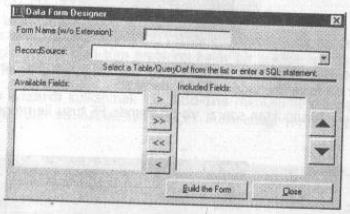


SQL Statement penceresinde yer alan **Execute** düğmesiyle kutuda yer alan SQL deyimleri otomatik olarak çalıştırılabilmektedir.

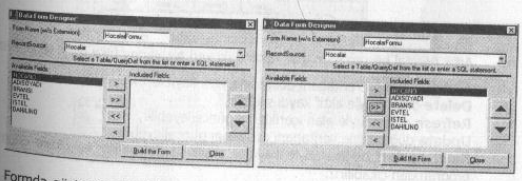
VB ortamına otomatik veri formu aktarmak

Visdata programıyla oluşturulan tablolar vb ortamına otomatik olarak veri formu aktarılabiliyor. Bu işlem veritabanı programlarını oldukça basite indirgeyecektir.

Bunun için **Utility-Data Form Designer** menü seçenekleriyle aşağıdaki pencereye ulaşıyoruz:



Form Name kutusuna oluşacak olan forma ait bir isim girilir. **RecordSource** kutusunda ise istenilen veri tablosu seçilir:



Formda gösterilecek tabloya ait veri alanları seçildikten sonra **Build the form** düğmesiyle form oluşturma işlemi başlatılır. Kısa bir süre sonra formumuz oluşturulur: