

## Visual Basic 6.0

1963 yılında Darmouth College’de John G. Kemeny ve Thomas E. Kurtz tarafından Basic dili geliştirilmiştir. Daha sonralarda Microsoft tarafından PC’lerde kullanılmak üzere uyarlanmıştır. Microsoft Qbasic ve Microsoft-Dos Qbasic’de dahil olmak üzere çeşitli sürümleri bulunmaktadır. Microsoft ileriki yıllarda Basic dilini geliştirerek Windows ortamına uyarlamış ve geliştirilen bu yeni dile Visual Basic adını vermiştir. Microsoft en son Visual Basic’in 6.0 sürümünü çıkarmıştır. Visual Basic, devamlı geliştiği bu süre sonunda; yüksek hızlı uygulamalar, OLE Serverlar, ActiveX kontrolleri ve daha birçok projeyi geliştirebilecek hale gelmiştir. Visual Basic yapısal bir programlama dili olan Basic dilinden türetilmiş olmasına rağmen olaya bağlı bir programlama dilidir.

Yapısal yada yordamsal uygulamalarda, uygulama kodun hangi kısımlarının çalışacağını ve hangi sırada çalışacağını denetler. Uygulama kodun ilk satırı ile başlar ve gerektiğinde yordamları çağırarak uygulama boyunca önceden tanımlanmış bir yolu izler.

Olaya bağlı bir uygulamanın çalışması, önceden belirlenmiş bir yolu izlemez. Farklı kod bölümleri olaylara bağlı olarak çalışır. Olaylar, kullanıcın eylemlerinden, sistem yada diğer uygulamalardan gelen iletilerden tetiklenir. Olaya bağlı programlamanın en gerekli bölümü bir uygulamada oluşabilecek olası tüm olaylara yanıt veren kodlar yazmaktır.

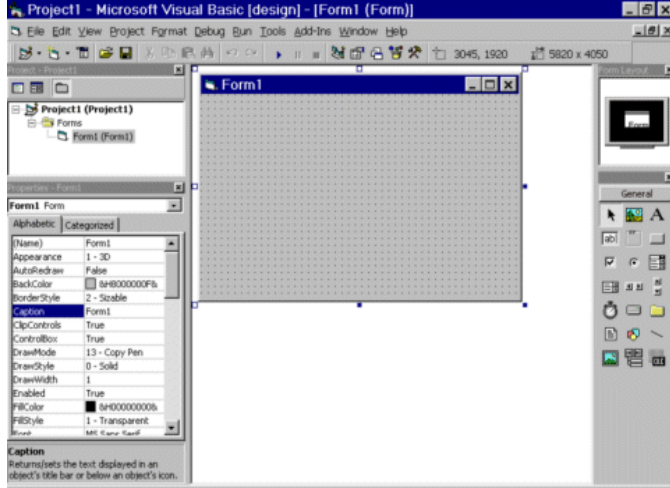


Visual Basic’i çalıştırdığımızda karşımıza yandaki dialog penceresi gelir. Bu dialog penceresinde 3 adet sekme bulunur. **New** sekmesinde oluşturulmak istenilen yeni proje için alternatifler bulunmaktadır. Genellikle **Standart. EXE** seçeneği seçilerek yeni bir projeye başlanır. Eğer istenirse diğer seçeneklerde kullanılarak ActiveX denetimleri, Dll dosyaları, DHTML sayfalar oluşturulabilir.

**Existing** sekmesi ile daha önceden oluşturulmuş projeler sürücü ve klasör seçimi yapılarak açılabilir. **Recent** sekmesi ise üzerinde çalışmış olduğumuz projelerin bir listesini verir ve bunlar arasından istediğimizi seçerek çalıştırabiliriz.

## Visual Basic Çalışma Ortamı

Visual Basic’de bir proje başlattığımızda aşağıdaki gibi bir görüntü ile karşılaşırız. Bu görüntüyü elde edebilmeniz için açılışta "Standart EXE" seçeneğini kullanmalıyız.



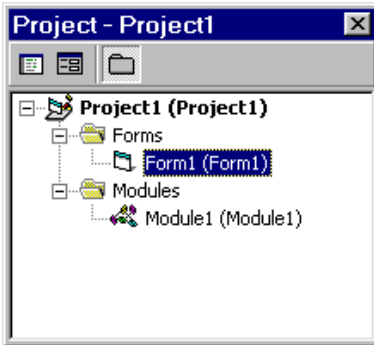
Proje geliştirme ekranında aşağıdaki araçlar bulunur.

- Menü Çubuğu
- Araç Çubuğu
- Project Explorer
- Properties penceresi
- Form Layout penceresi
- Araç Kutusu
- Form Designer

Menü çubuğu Visual Basic penceresinin üst tarafında duran metin satırıdır. Diğer Windows uygulamalarında bulunan menü çubukları ile hemen hemen aynıdır. **File** Menüsünde projeyi açma kaydetme gibi işlemler, **Edit** Menüsünde standart edit işlemleri, **View** Menüsünde programın mevcut olan fakat ekranda açık olmayan pencerelerini açma işlemleri, **Project** Menüsünde form ekleme-kaldırma gibi proje ile ilgili işlemler, **Format** Menüsünde forma eklenen nesnelerin düzenleme işlemleri, **Debug** Menüsünde program çalışırken programı kontrol etmeye yarayan işlemler bulunur. **Run** Menüsü aracılığı ile programı çalıştırabilir veya durdurabiliriz. **Tolls** Menüsünde Visual Basic'i özelleştirebileceğimiz ve Projeye menü ekleyebileceğimiz seçenekler bulunur. **Add-Ins** Menüsü ise raporlar ve database oluşturma seçeneklerini bulunudur.



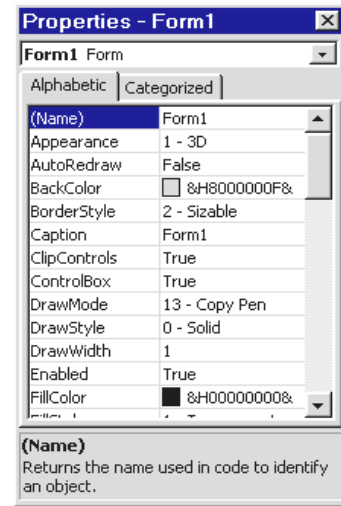
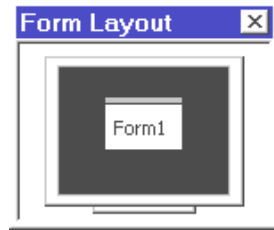
Menü çubuğunun hemen altında yukarıda görülen araç çubuğu bulunur.



Project Explorer penceresi projenizde bulunan elemanlara erişmenizi sağlar. Bu pencerede formalar, sınıflar ve modüller listelenir. Bu pencerenin araç çubuğunda 3 adet buton bulunur. Project Explorer penceresi içerisinden bir form seçip View Object butonuna tıklayarak formu görüntüleyebilirsiniz. View Code butonu Code Editöre ulaşmanızı sağlar. Toggle Folders butonu ise tüm form ve modülleri kategoriler halinde görebilmemizi sağlar.

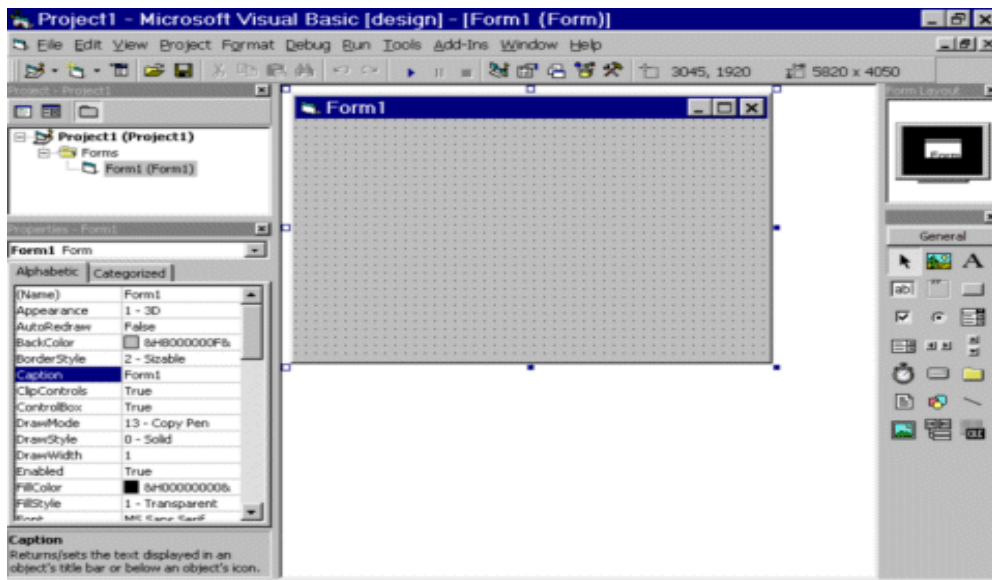
Project Explorer penceresinde bir öğeye sağ düğme ile tıkladığımızda bir çok işlev sunan bir menü açılır.

Visual Basic’de bütün nesnelerin kendilerine has özellikleri bulunur. Properties penceresi kullanılarak nesnelere ait özellikler değiştirilebilir. Bir nesne seçildikten sonra Properties penceresinde seçili olan nesneye ait özellikler yer alır. Visual Basic’de formlarda birer nesnedir. Yanda görülen pencerede Form1’e ait özellikler listelenmektedir. Properties penceresinin altında aktif olan özelliğe ait bir açıklama görülebilir.

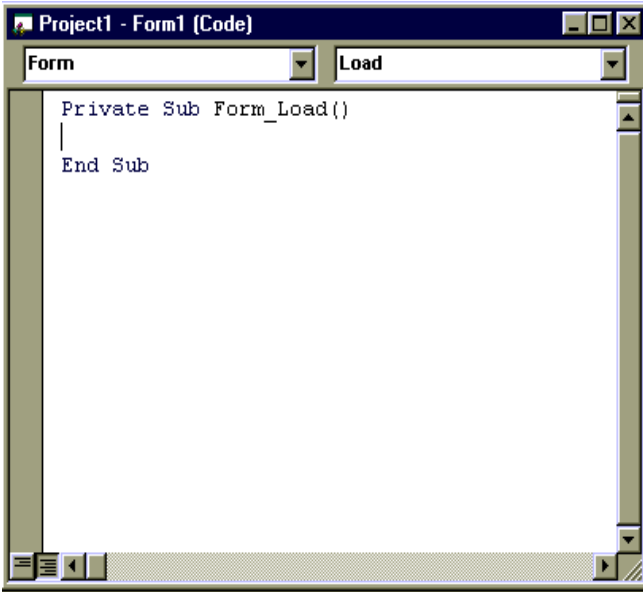


Form Layout penceresi ile formun çalışma esnasında ekranda nasıl görüleceğini belirleriz.

Toolbox uygulama arabirimimizi oluşturmak için gereksinim duyacağımız denetimleri içerir. Toolbox’da bulunan bütün simgeler birer denetimi temsil ederler. Visual Basic, mouse ikonunu bir denetimin üzerine getirdiğimizde bu denetimlerin adını bize verir. Toolbox’a yeni denetimler ekleyebilir yada varolan denetimleri çıkartabiliriz. Toolbox’da bulunan bir denetimi kullanmak istersek mouse ikonu ile denetim üzerine bir kez tıklarız ve ardından formumuza bu denetimi çizeriz. Bu şekilde istediğimiz denetimi formumuza eklemiş oluruz.



Yukarıdaki resimde ekranının ortasında Form Tasarımcısı görülür. Form görünümünün, form üzerindeki denetimlerin düzenlendiği yerdir.



adet açılır liste kutusu bulunmaktadır. Sol taraftaki açılan metin kutusunda (Object), form içerisinde bulunan nesnelerin bir listesi bulunur, sağ taraftakinde (Procedure) ise seçili nesneye ait olaylar bulunur, aşağıdaki bölüm ise kodları yazacağımız kısımdır.

## Programlamaya Giriş

Program, bilgisayar üzerinde bir takım yararlı işlerin gerçekleşmesini sağlayan bir grup yönergedir. Program, kişilerin adres, telefon gibi bilgilerini tutmamızı sağlayan küçük bir uygulama olabileceği gibi Microsoft Excel, Microsoft Word gibi çok gelişmiş bir uygulamada olabilir.

Programlamanın ilk adımı programımızın tam ve kesin olarak neler yapması gerektiğini belirlemektir. Bu işlem oldukça basit gibi görünmektedir. Herşeyin yolunda gitmesi için en başta herşeyin iyice düşünülüp tasarlanması gereklidir. Burada yapılan planlama olayına **algoritma** adı verilir. Önceden yapılan bu çalışma ileride programı geliştirirken kaybedilecek zamanı azaltır.

Program için bir algoritma oluşturulduktan sonra kullanıcı arabirimini tasarlama işini gerçekleştirmeliyiz. Kullanıcı arabirimi Visual Basic denetimleri kullanılarak hazırlanır Kullanıcı arabirimi, uygulamayı kullananların gördüğü bütün menüleri, iletişim kutularını, nesneleri ve resimleri kapsar. Kullanıcı arabirimi ne kadar anlaşılır, sade, görsel ve Windows standartlarına uygun olursa programın kullanımı da bir o kadar kolay ve esnek olur.. Kullanıcı arabirimi tasarlandıktan sonra denetimlerin özelliklerini belirler ve verilerin nasıl işleneceği ile ilgili kodları yazarız.

Visual Basic'de proje geliştirmeye başlarken yukarıda bahsedilen işlemleri gerçekleştiririz. Bu işlemlerin neler olduğuna karar verebilmemiz için kendimize bir takım sorular sormalıyız. Bu sorular programın nasıl çalışacağına karar vermemize ve kullanıcı arabirimini tasarlamamıza yardımcı olur.

Yazdığım programın hedefi nedir ?

Programı kim kullanacak ?

Çalıştırıldığında program görüntüsü nasıl olacak ?

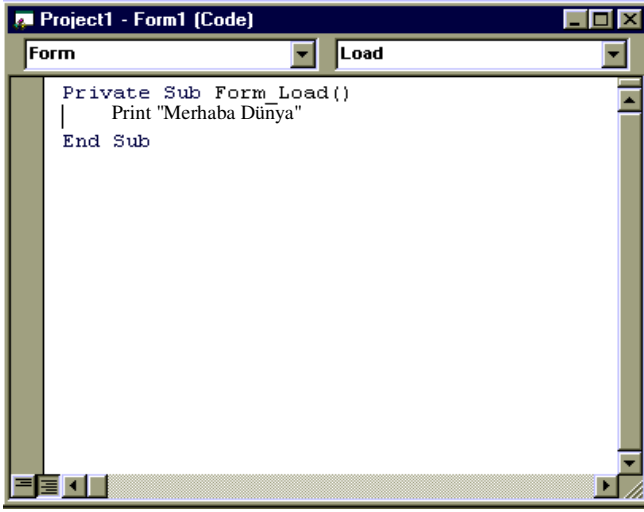
Programa, kullanıcı hangi verileri, nasıl girecek ?

Program bu verileri nasıl işleyecek ?

Program hangi bilgileri üretecek ?

### ***Merhaba Dünya Programı***

Hemen hemen programlamaya ilk başlayan herkesin öğrendiği ilk program “Merhaba Dünya” programıdır. Visual Basic’de bu programı yazabilmek için ilk önce yeni bir proje başlatmamız gerekir. File-New Project komutunu verdikten sonra gelen pencereden “Standart.EXE” seçeneğini seçip “OK”



butonuna basarız. Karşımıza Visual Basic proje geliştirme ortamı gelir. Visual Basic’de her projede en az bir adet Form olmalıdır. Şimdi çalıştığı zaman ekranda “Merhaba Dünya” yazan bir proje hazırlayacağız. Bunu yapabilmek için Code Editör’e bir satırlık bir kod yazmamız yeterli olacaktır. Code Editöre ulaşabilmek için Form üzerine çift tıklarız ve Code Editör karşımıza gelir. Visual Basic Code Editörde bizim için otomatik olarak yandaki gibi Form’a ait

Load prosedürünü açar. Code Editörde **Private Sub Form\_Load()** ve **End Sub** bildirimleri arasına **Print "Merhaba Dünya"** yazıp araç çubuğunda start düğmesine bastığımız anda programımız çalışacak ve ekrana Merhaba Dünya yazan bir pencere gelecektir. Böylece ilk programımızı yazmış olduk.

### ***Projeleri Çalıştırmak***

Visual Basic’de, hazırladığımız projeleri çalıştırmak için Run-Start komutunu veririz veya Araç çubuğunda bulunan Start butonuna tıklarız. Projeleri çalıştırmak için klavyeden F5 tuşunada basabiliriz. Çalışmakta olan projeyi durdurup tekrar tasarım moduna dönmek istersek Run-End komutunu veririz veya Araç Çubuğunda bulunan End butonuna tıklarız.

### ***Projeleri Kaydetmek***

Visual Basic’de hazırladığımız projeleri bilgisayara kaydetmek için File-Save Project komutunu veririz. Gelen dialog penceresinden projemizi kaydetmek istediğimiz klasörü seçip bir isim belirleriz. Visual Basic, forma ait kod ve denetimleri bir dosyaya, proje bileşenlerini ise farklı bir

dosyaya kaydeder. Visual Basic form dosyalarına “**frm**” uzantısı, proje dosyalarına ise “**vbp**” uzantısını verir.

### **Projeleri Derlemek**

Hazırladığımız projeleri Visual Basic olmadan çalışabilir hale getirebilmek için derlememiz gerekir. Derleme işlemini başlatabilmek için File-Make Project.EXE komutunu vermemiz gerekir. Bu komutu verdikten sonra karşımıza gelen dialog penceresine oluşturmak istediğimiz Exe dosyasının adını ve oluşturulmasını istediğimiz klasörü belirtmemiz gerekir ve OK butonuna tıklarız.

### **Visual Basic Denetimlerini Kullanmak**

Visual Basic’de kullanıcı arabirimini oluştururken Toolbox’dan faydalanırız. Form üzerinde bulunmasını istediğimiz denetimleri Toolbox üzerinden seçerek form üzerine çizebiliriz. Bu şekilde kullanıcı arabirimini tasarlamış oluruz. Toolbox üzerinde bulunan simgeler üzerine imleci getirdiğimizde bize o simgenin ne olduğunu açıklayan mesaj gelecektir. Toolbox üzerinde istediğimiz denetime ait simge üzerine tıkladıktan sonra form üzerinde mouse’un sol tuşunu basılı tutarak denetimimizi çizebiliriz.

Visual Basic’de form üzerinde bulunan bir denetimin boyutlarını değiştirmek için, denetim üzerine tıkladıktan sonra etrafında oluşan noktacıklar üzerine mouse’umuzu getirir ve sol tuş basılı tutarak yeni boyutları belirleyebiliriz. Bir nesnenin üzerine mouse imlecini getirip sol tuş basılı iken hareket ettirere taşıyabiliriz. Herhangi bir nesneyi seçili hale getirip klavyeden “**Delete**” tuşuna basarak form üzerinden silebiliriz.

**Not :** Birden fazla denetimi seçmek için, “**Shift**” tuşu basılı iken sırası ile nesnelere tıklayarak çok sayıda denetimi seçili hale getirebiliriz.

### **Özellikler (Properties)**

Visual Basic’de bütün nesnelere karakteristiklerini belirleyen özellikleri bulunur. Özellikler nesnenin tanımını, görünümünü ve hatta tutumunu değiştirebilmemizi sağlar. Bu özellikler tasarım aşamasında değiştirilebileceği gibi çalışma esnasında da değiştirilebilir. Bir nesneye ait özelliği değiştirmek için **Properties Penceresinin** ekranda olması gerekir. Eğer Properties Penceresi ekranda değilse View menüsü aracılığı ile ekrana getirilebilir.

Form üzerinde bir nesne seçildikten sonra **Properties Penceresine** o nesneye ait özelliklerin listesi gelir. Buradan istenilen özellik seçilerek istediğimiz değeri atayabiliriz. Bir başka nesnenin özelliklerini değiştirmek için ise form üzerinde diğer nesneyi seçili hale getirmemiz gerekir. Bu işlemi yaptığımız anda Properties Penceresinde listelenen özellikler değişir. Bir başka yol ise Properties Penceresinin üst tarafında bulunan açılan liste kutusundan farklı bir nesne adı seçmektir. Birden fazla nesne seçilerek, o nesnelere ait ortak özellikler değiştirilebilir.

Çalışma anında bir nesnenin özelliklerini değiştirmek için ise Code Editörde kod yazmamız gerekir. Bir nesneye ait kod yazarken önce nesne adı yazılır ardından bir “.” işareti konulur ve özellik adı yazılır. Aşağıdaki ifadede Command1 nesnesinin Caption özelliğini Durdur olarak değiştiriyoruz.

NesneAdı.Özellik=Değer

Command1.Caption = “ Durdur ”

### **Olaylar (Event)**

Visual Basic olay güdümlü bir programla dilidir. Visual Basic’de her nesne için önceden belirlenmiş bir takım olaylar vardır. Windows’a yeni geçiş yapan bir DOS programcısıysanız veya Visual Basic’e henüz yeni başlamış bir programcıysanız öğrenmeniz gereken en önemli konulardan biri olay güdümlü programlardır. Dos ortamında yazılan projeler, uygulama başladıktan sonra bütün olaylardan sorumludur. Visual Basic bu detayları bizim için düşünmüştür. Örneğin bir butona tıkladığımızda mesaj yazdıran bir program yazmak istiyorsunuz; DOS tabanlı bir programlama dilinde, mouse’un konumunu, mouse’un tıklanıp tıklanmadığını takip etmeniz gerekir. Visual Basic’de ise aynı işi yapmak için form üzerinde bulunan Command Button nesnesini **click** olayına bir satır kod yazmamız yeterlidir. Visual Basic bizim için mouse’u izler ve kullanıcı butona tıkladığı anda yazdığımız kodu devreye sokar.

Bir olay prosedürü yazmak için Code Editörde nesneyi ve olayı seçmemiz yeterlidir. Visual Basic bizim için Code Editörde bir prosedür yaratacaktır. Bizim ise tek yapmamız gereken bu olay gerçekleştiğinde meydana gelmesini istediğimiz işlemin kodunu yazmaktır.

### **Veri Tipleri**

Visual Basic’de verilerin geçici olarak depolandıkları yere **değişken** adı verilir. Değişkenlerde program çalışırken kullanıcının girdiği bir bilgi, bir hesaplamanın sonucu, sözcük, sayı, tarih veya özellikleri saklayabiliriz. Değişkenlerin kolay hatırlanır isimlere sahip olmaları gerekir. Visual Basic’de bir değişken kullanılmadan önce tanıtılması gerekir. Değişken tanımlanırken **Dim** bildirisi kullanır ve değişkenin tipi belirtilir.

Değişkenlerle çalışmayı düşünüyorsak projemizin **Declarations** bölümüne **Option Explicit** söz dizimini girmemiz ileride karşılaşılabileceğimiz sorunlara bir çözüm oluşturur. Bu söz dizimini kullandığımızda kod içerisinde tanıtılmamış bir değişken kullanırsak Visual Basic bizi uyaracaktır. Tools-Options komutunu vererek gelen pencerede Edit sekmesinden Require Variable Declaration onay kutusunu işaretlersek, Visual Basic başladığımız her projeye Option Explicit ifadesini otomatik olarak yerleştirecektir.

#### **String**

Metin türü bilgileri saklamak için kullanılacak veri türüdür. 0 ile 65,535 arasında karakter alabilir. Hafızada içerisinde bulunan karakter kadar yer kaplar. \$ işareti ile tanımlanabilir

Dim Ad As String	Dim Ad\$
Ad="Uğur ŞAHİN"	Ad="Uğur ŞAHİN"

String türü değişkenlere sabit bir uzunlukta yer ayırmak istersek aşağıdaki şekilde bir tanımlama yapmalıyız.

Dim Ad As String \*12

### **Integer**

Tam sayıları depolamak için kullanılan bir veri türüdür. Hafızada 2 Byte yer kaplarlar. -32.768 ile +32.767 arasındaki değerleri saklayabilir. % işareti ile tanımlanabilir

Dim Toplam As Integer	Dim Toplam%
Toplam=100+50	Toplam=100+50

### **Long**

Integer türü değişkenlerde tutamayacağımız büyüklükteki verileri saklar. Alabileceği değerler -2.147.483.648 ile +2.147.483.647 arasındaki tam sayılardır. Hafızada 4 Byte yer kaplar. & işareti ile tanımlanabilir.

Dim Sonuc As Long	Dim Sonuc&
Sonuc=460.000	Sonuc=460.000

### **Single**

Tam sayı olmayan küsürlü sayılar için kullanabileceğimiz bir veri tipidir. -3,402823E38 ile +3,402823E38 arasındaki değerlerini alabilirler. Hafızada 4 Byte yer kaplar. 7 haneye kadar hassastır, daha sonrası yuvarlatılır. ! işareti ile tanımlanabilir.

Dim Sonuc As Single	Dim Sonuc!
Sonuc=66,99	Sonuc=66,99

### **Double**

Visual Basic'de kullanılabilecek en büyük sayısal veri tipidir. -1,797693134862232D308 ile +1,797693134862232D308 arasındaki değerleri alabilirler. Hafızada 8 Byte yer kaplar. 16 haneye kadar hassastır, daha sonrası yuvarlatılır. # işareti ile tanımlanabilir.

Dim Pi As Double	Dim Pi#
Pi=3,1415926535	Pi=3,1415926535

### **Currency**

Sayısal tipdeki veriler için tanımlanmış özel bir veri tipidir. Özellikle parasal hesaplamalarda kullanılır. Hafızada 8 Byte yer kaplarlar. 14 hanelik sayılara kadar veri saklayabilir ve ayrıca virgülden sonra da 4 hanelik değer alabilir. Alabileceği maximum değerler  $-9.223.371.203.685.477,5808$  ile  $+9.223.371.203.685.477,5807$  arasındadır. @ işareti ilede tanımlanabilir.

Dim Kredi As Currency

Dim Kredi@

Kredi=100.760.030,50

Kredi=100.760.030,50

### **Variant**

Bu tipte tanımlanmış bir değişkene herhangi bir tip'te veri yüklenebilir. Her tür veriyi saklayabilir. Bu tür değişkenler sayılarda 16 Byte, dizilimlerde karakter sayısı +22 Byte kadar yer kaplarlar. Tarih, saat gibi verilerde bu tip değişkenlerde saklanırlar. Veri tipi belirtilmeden kullanılabilirler.

Dim X As Variant

Dim X

X=289,33

X=289,33

Dim DogumGunu As Variant

DogumGunu =#21-4-1974#

### **Boolean**

Mantıksal veri tipleri için kullanılır. İki seçenektan birisini alabilir. Bunlar True veya False değerleridir. Bellekte 2 byte yer işgal ederler.

Dim Cevap As Boolean

Cevap=True

### **Byte**

0 ile 255 arasında değer alabilen bir veri tipidir.

Dim Not As Byte

Not=78

## **Değişken Tanımlama**

Visual Basic'de değişken tanımlanırken dikkat edilmesi gereken kurallar vardır. Tanımlanacak değişkenlerin ilk karakteri mutlaka bir harf ile başlamalıdır. Geri kalan karakterler; harflerden, rakamlardan, alt çizgi karakterinden oluşabilir. Değişken isimlerinde noktalama işaretlerini, matematiksel ve mantıksal ve karşılaştırma operatörleri kullanamayız. Visual Basic'de kullanılan anahtar sözcükler, nesne adları, özellikler değişken adı olarak kullanılamaz. Değişken isimleri 255 karakter uzunluğunda olabilir. Değişken tanımlarken Visual Basic'te Dim bildiri deyimini

kullanabiliriz. Değişkenin tanımlanırken saklayabileceği veri türünün belirtilmesi hafızada ayrılacak miktarının belirli olmasını sağlar. Eger değişkenlerin tipini belirtmeden bir kullanım yaparsak bu değişkenlerin Variant tipinde olduğu kabul edilir. Bu da hafızada gereksiz yer kaybına sebep olur.

**Örnek**

```
Private Sub Form_Load()
```

```
    Dim Ad As String
```

```
    Dim Maas As Currency
```

```
    Dim D_tarihi As Date
```

```
    Dim Adres As String
```

```
    Dim Sira As Integer
```

```
End Sub
```

Burada görmüş olduğunuz gibi 5 adet değişken tanımlanmaktadır. Ad String tipinde, Maas Currency tipinde, D\_tarihi Date tipinde, Adres String tipinde, Sira ise Integer tipinde bir değişkendir.

**Örnek**

```
Private Sub Form_Load()
```

```
    Dim Ad As String , Soyad As String
```

```
    Dim Maas As Currency
```

```
End Sub
```

**Const**

Bunlar program içinde değeri değiştirilemeyen sabitlerdir. Public ve Private tipinde sabitler tanımlanabilir. Public sabitlere tüm modüller içinden ulaşılabilir. Private türündeki sabitler ise sadece tanımlandıkları modül içerisinde geçerlidirler.

```
Const Sehir="İstanbul"
```

```
Const Ulke="Turkey"
```

```
Const Posta_Kodu=34650
```

```
Const Tel_Kod=212
```

**Static**

Bir prosedür içerisinde oluşturulan değişken prosedür çalıştığı sürece değerini korur. Prosedür sona erdiğinde ise değişken yok olur. Sürekli değerini koruyabilen yerel değişkenler oluşturabilmek için Static deyimi kullanılır.

```
Private Sub Command1_Click()
```

```
    Static i As Integer
```

```
    i = i + 1
```

```
    MsgBox i & ". Denemeniz."
```

```
End Sub
```

**Type - End Type Yapısı**

Type yapısını kullanarak, programcı farklı veri tiplerini kullanarak kendi veri yapısını oluşturabilir. Bu C dilindeki Struct yapısına benzetilebilir. Bu yeni veri tipine record adı verilir. Herhangi bir modülün General Declarations kısmında aşağıdaki gibi bir tanımlama yapabiliriz.

Type Öğrenci

Ad As String \*10

Soyad As String \*12

Not As Byte

Kredi As Integer

End Type

Öğrenci veri tipi toplam hafızada 25 Byte yer kaplamaktadır. Bu veri tipini kullanmak için Öğrenci tipinde değişkenler tanımlamak gerekmektedir.

Private Sub Form\_Load()

Dim A As Öğrenci

Dim B As Öğrenci

'Bu değişkenlere bilgi atamak aşağıdaki şekilde gibidir.

A.Ad="Ali"

A.Soyad ="Armer"

End Sub

**Tip Dönüşümleri**

Visual Basic'de zaman zaman herhangi bir veri tipinde saklanan değeri farklı bir veri tipine dönüştürme ihtiyacı duyarız. Bu işlemi yapan fonksiyonlara tip dönüşüm fonksiyonları adı verilir.

Private Sub Command1\_Click()

Text1.Text = 6

Text2.Text = 10

Label1.Caption = Text1.Text + Text2.Text

End Sub

Bu kodu çalıştırdığımızda Label1 içerisinde **610** değeri yazacaktır. Visual Basic her iki metin kutusu içerisinde bulunan değerlerin birer metin olduğunu varsayarak iki metinide birleştirme işlemi yaptı. Eğer bu değerlerin toplanmasını istiyorsak tip dönüşümlerini kullanarak string tipindeki verileri integere çevirmemiz gerekirdi.

```

Private Sub Command1_Click()
    Text1.Text = 6
    Text2.Text = 10
    Label1.Caption = CInt(Text1.Text) + CInt(Text2.Text)
End Sub

```

Yukarıdaki kodda Text1 ve Text2 içeriği önce **CInt** adlı fonksiyonla tamsayıya çevrildi ve ardından toplandı. Label1'in içeriği de 16 olarak değişti.

Aşağıda Visual Basic'de kullanılan tip dönüşüm fonksiyonları verilmiştir.

Fonksiyon	Geri Dönen Deger	Yaptığı İşlem
CBool(Değer)	Boolean	Matemetiksel ifadeyi Boolean türüne dönüştürür.
CByte(Değer)	Byte	Matemetiksel ifadeyi Byte türüne dönüştürür.
CCur(Değer)	Currency	Matemetiksel ifadeyi Currency türüne dönüştürür.
CDate(Değer)	Date	Matemetiksel ifadeyi Date türüne dönüştürür.
CDbl(Değer)	Double	Matemetiksel ifadeyi Double türüne dönüştürür.
CDec(Değer)	Decimal	Matemetiksel ifadeyi Decimal sayıya dönüştürür.
CInt(Değer)	Integer	Matemetiksel ifadeyi tam sayıya dönüştürür.
CLng(Değer)	Long	Matemetiksel ifadeyi Long türüne dönüştürür.
CSng(Değer)	Single	Matemetiksel ifadeyi Single türüne dönüştürür.
CVar(Değer)	Variant	Matemetiksel ifadeyi Variant türüne dönüştürür.
CStr(Değer)	String	Matemetiksel ifadeyi String türüne dönüştürür.

Aşağıda tip dönüşümleri ile ilgili çeşitli örnekler verilmiştir.

### Örnek1

$A=10, B=5, C=10, D=0$

$Sonuc = CBool(A < B)$        $'Sonuc = False$

$Sonuc = CBool(A > B)$        $'Sonuc = True$

$Sonuc = CBool(A = C)$        $'Sonuc = True$

### Örnek2

$A=10, B=5, C=0$

$Sonuc = CByte(A < B)$        $'Sonuc = 0$

$Sonuc = CByte(A > B)$        $'Sonuc = 255$

$Sonuc = CByte(A = C)$        $'Sonuc = 255$

### Örnek3

$A=1, B=2, C=36000, D=36001$

$Sonuc = CDate(A)$        $'Sonuc = 12/31/1899$

$Sonuc = CDate(B)$        $'Sonuc = 1/1/1900$

*Sonuc = CDate(C) 'Sonuc =7/24/98*

*Sonuc = CDate(D) 'Sonuc =7/25/98*

#### Örnek4

*A=2.4 , B=2.5 , C=2.6 , D=3.5*

*Sonuc = CInt(A) 'Sonuc =2*

*Sonuc = CInt(B) 'Sonuc =3*

*Sonuc = CInt(C) 'Sonuc =3*

*Sonuc = CInt(D) 'Sonuc =4*

## Oparatörler

Visual Basic'de matematiksel işlemlerimizi yaptırabilmemiz için aşağıdaki oparatörleri kullanabiliriz.

+	Toplama
-	Çıkarma
*	Çarpma
/	Bölme
\	Tamsayı Bölem
^	Üs yani Kuvvet alma
Mod	Kalanlı Bölme
&	Dizilim Birleştirme

#### Örnek

*Dim Sonuc As Double 'Ondalık sayı tutabilecek bir değişken tanımlanıyor..*

*Dim Sayı1 As Integer, Sayı2 As Integer '2 Tane TamSayı değişken tanımlıyoruz..*

*Sayı1 = 9*

*Sayı2 = 2 ' Değişkenlere değer atıyoruz..*

*Sonuc = Sayı1 + Sayı2 ' Sonuc 11*

*Sonuc = Sayı1 - Sayı2 ' Sonuc 7*

*Sonuc = Sayı1 \* Sayı2 ' Sonuc 18*

*Sonuc = Sayı1 / Sayı2 ' Sonuc 4,5*

*Sonuc = Sayı1 \ Sayı2 ' Sonuc 4 . Bu işlem çıkan değerın sadece tamsayı kısmını alır*

*Sonuc = Sayı1 ^ Sayı2 ' Sonuc 81*

*Sonuc = Sayı1 Mod Sayı2 ' Sonuc 1. Bölme işlemi yapıldı ve sonuc olarak kalanı verdi.*

#### Örnek

*Dim Ad As String, Soyad As String, AdSoyad As String*

*Ad= "Uğur"*

*Soyad= "Şahin"*

*AdSoyad = Ad & Soyad 'AdSoyad değişkenin değeri "UğurŞahin"*

*AdSoyad = Ad & " " & Soyad 'AdSoyad değişkenin değeri "Uğur Şahin"*



**Örnek***Dim Sonuc As Integer*

*Sonuc = ( 3 < 5 )* 'Sonuc = -1  
*Sonuc = ( 7 > 9 )* 'Sonuc = 0  
*Sonuc = ( 10 <= SQR(100) )* 'Sonuc = -1  
*Sonuc = ( 2 = 6 )* 'Sonuc = 0  
*Sonuc = ( 4 = 4 )* 'Sonuc = -1  
*Sonuc = ( 2 <> 6 )* 'Sonuc = -1  
*Sonuc = ( 7 <> 7 )* 'Sonuc = 0  
*Sonuc = ("Kürşat" <> "Kürşat Volkan" )* 'Sonuc = 0

**AND Operatörü**

AND operatörü tüm şartların doğru olduğu anlarda -1 değerini döndürür, diğer durumlarda ise 0 değerini döndürür.

*sayı1 = 4**sayı2 = 5**Sonuc = (sayı1 = 4) AND (sayı2 = 5)* 'Sonuc = -1*Sonuc = (sayı1 = 4) AND (sayı2 = 8)* 'Sonuc = 0**OR Operatörü**

OR operatörü, tüm şartların yanlış olduğu anlarda 0 değerini döndürür, diğer durumlarda ise -1 değerini döndürür.

*sayı1 = 4**sayı2 = 5**Sonuc = (sayı1 = 2) OR (sayı2 = 5)* 'Sonuc = -1*Sonuc = (sayı1 = 4) OR (sayı2 = 5)* 'Sonuc = -1*Sonuc = (sayı1 = 2) OR (sayı2 = 4)* 'Sonuc = 0**XOR Operatörü**

XOR operatörü, şartlardan sadece birinin doğru olduğu anlarda -1 değerini döndürür, diğer durumlarda ise 0 değerini döndürür.

*sayı1 = 4**sayı2 = 5**Sonuc = (sayı1 = 2) XOR (sayı2 = 5)* 'Sonuc = -1*Sonuc = (sayı1 = 4) XOR (sayı2 = 5)* 'Sonuc = 0

**NOT Operatörü**

NOT operatörü çıkan sonucu tam tersine çevirir. Sonuç -1 çıkarsa 0'a, 0 çıkarsa -1'e çevirir.

sayı1 = 4

sayı2 = 5

Sonuc = sayı1=sayı2 'Sonuc = 0

Sonuc = NOT(sayı1=sayı2) 'Sonuc = 0

**EQV Operatörü :**

EQV operatörü iki ifadenin karşılaştırılması için kullanılır. İki koşulunda doğru veya yanlış olması durumunda -1 sonucunu üretir.

sayı1 = 4

sayı2 = 5

Sonuc = (sayı1=3) EQV (sayı2=8) 'Sonuc = -1

Sonuc = (sayı1=4) EQV (sayı2=1) 'Sonuc = 0

**Karar Yapıları**

Programlama dillerinde bilgileri işlemek için kullanılan en yararlı araçlardan biride koşullu bir ifadedir. Koşullu ifade bir özellik, değişken veya veride bulunan değere göre işlem yapmamızı sağlar. Visual Basic'de koşullu ifadeler kullanırken faydalanacağımız en önemli araçlar **If ...Then** ve **Select Case** karar yapılarıdır.

**If...Then Karar Yapısı**

Bir If...Then karar yapısı programdaki bir koşulu denetlememizi ve çıkan sonuca göre hareket etmemizi sağlar. En basit biçimiyle "**If Koşul Then Bildiri**" gibi kullanılabilir.

**Örnek**

*Dim Sinav1 As Byte*

*Dim Sinav2 As Byte*

*Dim Ortalama As Byte*

*Sinav1 = 80*

*Sinav1 = 60*

*Ortalama = ( Sinav1 + Sinav2 ) / 2* 'Ortalama 70

*If Ortalama >= 50 Then*

*Label1.Caption = "Kazandınız!.."*

*Else*

*Label1.Caption = "Kaybettiniz!.."*

*End If*

Yukarıdaki ifadede Ortalama 50 ve üzerinde ise **Label1.Caption = “Kazandınız!..”** işlemini gerçekleştirecek, aksi takdirde **Label1.Caption = “Kaybettiniz!..”** işlemi gerçekleştirecektir.

If...Then karar yapısında birden fazla koşulu kontrol ederken Else If söz dizimini kullanırız.

**Örnek**

*Dim Sinav1 As Byte*

*Dim Sinav2 As Byte*

*Dim Ortalama As Byte*

*Ortalama = ( Sinav1 + Sinav2 ) / 2*

*If Ortalama >= 85 Then*

*Label1.Caption = “Takdir Aldınız!..”*

*Else If Ortalama >= 75*

*Label1.Caption = “Teşekkür Aldınız!..”*

*Else If Ortalama >= 50*

*Label1.Caption = “Sınıfınızı Geçtiniz!..”*

*Else*

*Label1.Caption = “Sınıfta Kaldınız!..”*

*End If*

### **Select Case Karar Yapısı**

If...Then karar yapısının gelişmişidir. Sadece bir değişkenin durumunu kontrol eder. Kontrol yapılacak değişken birden fazla değer alabiliyorsa If...Then yapısına oranla daha kullanışlıdır. Kullanım şekli aşağıdaki gibidir.

**Örnek**

*Dim Ay*

*Ay = InputBox (“Bir ay giriniz”)*

*Select Case Ay*

*Case 1*

*MsgBox “Bu ay 31 gün”*

*Case 2*

*Dim Artık as Integer*

*Artık = 2001 Mod 4*

*If Artık=0 Then*

*MsgBox “Bu ay 29 gün”*

*Else*

*MsgBox “Bu ay 28 gün”*

*Endif*

*Case 3*

*MsgBox "Bu ay 31 gün"*

*Case 4*

*MsgBox "Bu ay 30 gün"*

*Case 5*

*MsgBox "Bu ay 31 gün"*

*Case 6*

*MsgBox "Bu ay 30 gün"*

*Case 7*

*MsgBox "Bu ay 31 gün"*

*Case 8*

*MsgBox "Bu ay 30 gün"*

*Case 9*

*MsgBox "Bu ay 31 gün"*

*Case 10*

*MsgBox "Bu ay 30 gün"*

*Case 11*

*MsgBox "Bu ay 31 gün"*

*Case 12*

*MsgBox "Bu ay 30 gün"*

*Case Else*

*MsgBox " Bir yılda 12 ay vardır."*

*End Select*

### **Örnek**

*Select Case No*

*Case 1304*

*Name="Murat Tuna"*

*Case 1305*

*Name="Ayse Sinem"*

*Case 1306*

*Name="Uğur Şahin"*

*Case 1307*

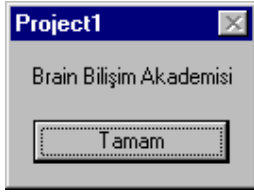
*Name="Osman Bilgin"*

*End Select*

### **Msgbox Fonksiyonu**

Msgbox fonksiyonu kullanıcıya bir dialog kutusu içerisinde bilgi vermek amacı ile kullanılır. Örneğin bir hata sonrasında kullanıcıya hatanın sebebini açıklayan bir bilgi verirken. Kullanım şekli aşağıdaki gibidir;

## Msgbox "Brain Bilişim Akademisi"



Yukarıdaki kodu kullanarak ekrana yandaki gibi içerisinde bir dialog penceresi getirebiliriz. İlk önce fonksiyon adını yazarız, ardından tırnak işaretleri içerisinde Brain Bilişim Akademisi yazarız. Tırnak işaretlerini kullanmamızın sebebi bir metnin gösterilmesini istiyor olmamızdır. Eğer bir metin değilse, bir nesne veya bir değişken içerisinde var olan değeri göstermek istersek tırnak işaretlerini kullanmamıza gerek kalmaz.

## Msgbox Text1.Text

Yukarıdaki ifade ise Text1 nesnesinin Text özelliğini dialog kutusu içerisinde görmemizi sağlar.

Msgbox fonksiyonu kullanıcıya seçenekler sunan bir dialog penceresi gibide kullanılabilir. Örneğin programdan çıkmak istendiğinde kullanıcı uyarılsın. Bu tip dialog kutuları kullanıcının verdiği cevaba göre bir sonuç döndürür. Formunuzun unload yordamına aşağıdaki kodu girin.

```
Private Sub Form_Unload(Cancel As Integer)
```

```
    Dim Cevap As Integer
```

```
    Cevap = MsgBox("Programdan çıkmak istediğinizde emin misiniz?", 20, "Brain")
```

```
    If Cevap = 6 Then
```

```
        Cancel = True
```

```
    End If
```

```
End Sub
```



Kullanıcı formu kapatmak istediğinde dialog penceresi ile karşılaşacak. Evet tıklanırsa form kapanacak, hayır tıklanırsa kapatma işleminden vazgeçilecektir. MsgBox fonksiyonu bu şekilde kullanıldığı zaman kullanıcının verdiği cevaba göre geriye bir değer gönderecektir. Daha sonra If karar yapısı ile Cevap değişkeninin içeriği kontrol edilmekte ve gerekli işlem yapılmaktadır.

```
MsgBox("Programdan çıkmak istediğinizde emin misiniz?", 20, "Brain")
```

Yukarıdaki yazımda "Programdan çıkmak istediğinizde emin misiniz?" bildirisi dialog kutusunda çıkacak bildiriye, 20 rakamı dialog kutusunda bulunacak komut butonlarını ve gösterilecek simgeyi ve "Brain" bildirisi de dialog kutusunun başlık çubuğunda yazmasını istediğimiz metni belirtir. MsgBox fonksiyonu geriye evet için 6 hayır için ise 7 değerini gönderir.

<b>Buton</b>	<b>Değer</b>	<b>VB Değeri</b>
Tamam	0	vbOK
Tamam, İptal	1	vbOKCancel
Durdur, Yeniden Dene, Yoksay	2	vbAbortRetryIgnore
Evet, Hayır, İptal	3	vbYesNoCancel
Evet, Hayır	4	vbYesNo
Yeniden Dene, İptal	5	vbRetryCancel
<b>Simge</b>	<b>Değer</b>	<b>VB Değeri</b>
Kritik	16	vbCritical
Soru	32	vbQuestion
Uyarı	48	vbExclamation
Bilgi	64	vbInformation
<b>Cevap</b>	<b>Değer</b>	<b>VB Değeri</b>
Tamam	1	vbOK
İptal	2	vbCancel
Durdur	3	vbAbort
Yeniden Dene	4	vbRetry
Yoksay	5	vbIgnore
Evet	6	vbYes
Hayır	7	vbNo

Msgbox fonksiyonunda çıkmasını istediğimiz butonlar ve simgeler için yukarıdaki değerler kullanılır. İstedığımız buton ve simgeye ait rakamı toplayarak yazarız. Cevap olarakda yine yukarıdaki değeri gönderilir. Bu değerler akılda pek kalıcı olmadığından dolayı VB değerlerini de kullanabiliriz.

```
MsgBox("Programdan çıkmak istediğinizde emin misiniz?", 20, "Brain")
```

```
MsgBox("Programdan çıkmak istediğinizde emin misiniz?", vbYesNo+ vbCritical, "Brain")
```

Her iki ifade arasında bir fark yoktur.

### ***Inputbox Fonksiyonu***

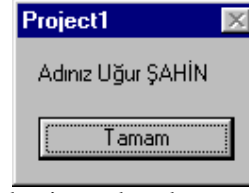
Inputbox fonksiyonu kullanıcıdan bilgi alınmak istendiği durumlarda kullanılır. Örneğin ;

```
Private Sub Form_Load()
    Dim Ad As String
    Ad = InputBox("Adınızı giriniz", "Bilgi Girişi")
    MsgBox "Adınız " & Ad
End Sub
```

Yukarıdaki kodu çalıştırdığımızda karşımıza aşağıdaki gibi bir bilgi giriş penceresi gelir.



Buraya girdiğimiz değer ise Ad



değişkenine aktarılır ve ardından bir

dialog kutusu ile bize bilgi verilir.

## Döngüler

Döngüler belirli işlemleri belirli sayıda veya herhangi bir şart sağlanana kadar tekrarlamak amacı ile kullanılırlar. Visual Basic'de kullanılan döngüler aşağıda verilmiştir.

For ..... Next Döngüsü

Do Until .....Loop Döngüsü

Do While .... Loop Döngüsü

Do ..... Loop Until Döngüsü

Do ..... Loop While Döngüsü

### *For Next*

Bu döngü belirli bir işlemi, belirli bir sayıda çalıştırmamıza olanak sağlar. Genel yazılım şekli aşağıdaki gibidir.

```
For Sayaç=BaşlangıçDeğeri To BitişDeğeri Step ArtımDeğeri
```

```
    Bildiri
```

```
Next Sayaç
```

Bu döngüde Sayac değişkenine BaşlangıçDeğeri verilir ve Sayaç BitişDeğerini geçene kadar çalıştırılır. Döngü her çalışmada Sayaç değeri ArtımDeğeri kadar artırılır. Eğer ArtımDeğeri belirtilmezse Sayaç her seferde 1 artırılır.

### **Örnek**

Liste kutusuna 1'den 10 kadar olan rakamları eklemek için aşağıdaki döngü kurulur.

```
Dim Sayac As Integer
```

```
For Sayac = 1 To 10
```

```
    List1.AddItem Sayac
```

```
Next Sayac
```

**Örnek**

Liste kutusuna 2'şer 2'şer 20'ye kadar olan rakamları eklemek için aşağıdaki döngü kurulur.

```
Dim Sayac As Integer
```

```
For Sayac = 2 To 20 Step 2
```

```
    List1.AddItem Sayac
```

```
Next Sayac
```

```
    For...Next döngüsünden çıkmak için Exit For bildirisi kullanılır
```

**Örnek**

```
Dim Sayac As Integer
```

```
For Sayac = 2 To 20 Step 2
```

```
    List1.AddItem Sayac
```

```
    If Sayac=10 Then
```

```
        Exit For
```

```
    End If
```

```
Next Sayac
```

**Do Until**

For...Next döngüsünün gelişmişidir. Bir koşul yanlış olduğu sürece devam eder. Genel yazım şekli aşağıdaki gibidir.

```
Do Until Koşul
```

```
    Bildiri
```

```
Loop
```

**Örnek**

Program çalıştığı anda kullanıcıya adını soran ve yanlış isim girildikçe soruyu tekrarlayan bir kod örneği. 3.denemeden sonra sonra Do..until döngüsünden çıkılıyor.

```
Private Sub Form_Load()
```

```
    Dim Ad As String
```

```
    Dim Tekrar As Integer
```

```
    Tekrar = 0
```

```
    Ad = InputBox("Adınızı Giriniz?")
```

```
    Do Until Ad = "Uğur"
```

```
        Tekrar = Tekrar + 1
```

```
    If Tekrar = 3 Then
```

```
        MsgBox "Üzgünüm. Deneme hakkınız dolmuştur."
```

```
        Exit Do
```

```
    End If
```

```
    MsgBox "Yanlış isim girdiniz. Lütfen tekrar deneyin."
```

```
Ad = InputBox("Adınızı Giriniz?")
Loop
End Sub
```

### ***Do While***

Do Until döngüsünün tersidir. Bir koşul doğru olduğu sürece devam eder. Genel yazım şekli aşağıdaki gibidir.

```
Do While Koşul
    Bildiri
```

```
Loop
```

### **Örnek**

Kullanıcıdan bir isim yazmasını isteyen bir kod örneği. Ad değişkeni “son” olmadığı sürece devam ediyor. Her defasında font boyutu 1 punto artırılıyor.

```
Private Sub Form_Load()
    Dim Ad As String
    Dim Boyut As Integer
    Boyut = 12
    Tekrar = 0
    Ad = InputBox("Lütfen adınızı yazınız. Çıkmak için son yazın")
    Do While Ad <> "son"
        Form1.FontSize = Boyut
        Print Ad
        Boyut = Boyut + 1
        If Boyut = 20 Then
            Exit Do
        End If
        Ad = InputBox("Lütfen adınızı yazınız. Çıkmak için son yazın")
    Loop
End Sub
```

### ***Do ..... Loop Until ve Do ..... Loop While***

Do Until ve Do While döngüleri koşul sağlandığı sürece devam ederler, yani belirtilen koşul sağlanmazsa döngü hiç çalışmayabilir. Eğer döngünün en az bir kez çalışması isteniyorsa Do..Loop Until veya Do...Loop While kullanılabilir. Bu döngülerin genel yazım şekli aşağıdaki gibidir.

```
Do
```

```
    Bildiri
```

```
Loop Until / While Koşul
```

Bu ifadede döngü içinde komutlar mutlaka bir defeye çalıştırılır, daha sonra Koşul kontrol edilir. Eğer Koşul doğru ise tekrar Do satırına dönülüp aradaki komutlar çalıştırılır.

## Formlar

Visual Basic'de pencerelere form adı verilir. Bütün Visual Basic projelerinde en az bir form olmalıdır. Formlar uygulama arabirimini oluşturabilmek için üzerlerine denetimler ekleyebileceğimiz nesnelere dir.

### Özellikler

Visual Basic'de formlara ait özellikler bulunur. Bu özellikler aracılığı ile formlarımızı istediğimiz gibi tasarlayabiliriz. Bu özelliklerin bir kısmı tasarım aşamasında değişebileceği gibi bir kısımda sadece çalışma modunda değiştirilebilirler. Aşağıda en sık kullanılan form özellikleri açıklanmıştır ;

#### **AutoRedraw**

Formun kendini yenilemesini sağlar. Özellikle form üzerine bir şeyler yazdırıyor veya form üzerine çizim yaptırıyorsak kullanırız.

#### **BackColor**

Bu özellik formun arka plan rengini belirlemizi sağlar.

#### **BorderStyle**

Formun kenarlarının nasıl görüleceğini belirler. Altı farklı değer alabilir.

0-None ; Form boyutları değiştirilemez ve form taşınmaz. Başlık çubuğu görülmez.

1-Fixed Single ; Form kenarlarından tutularak genişletilemez.

2-Sizable ; Varsayılan değerdir.

3-Fixed Dialog ; Form boyutları değiştirilemez. Sadece form taşınabilir ve kapatılabilir.

4-Fixed ToolWindow ; Fixed Dialog ile benzerdir. Başlık Çubuğundaki etiket daha küçük bir fontta gösterilir.

5-Sizable ToolWindow ; Sizable ile aynıdır ama Maximize ve Minimize düğmeleri görülmez.

#### **Caption**

Formun başlık çubuğunda görülecek etiketi belirler.

#### **ControlBox**

True veya False ayarını alır. Kontrol menüsünü gizler.

#### **Count**

Form üzerindeki menüler dahil kontrol sayısını verir.

#### **Controls(Index)**

Form üzerindeki nesnelere nin adını temsil eder.

*Private Sub Command1\_Click()*

*On Local Error Resume Next*

*Dim i*

*For i = 0 To Count - 1*

*Controls(i).BackColor = vbRed*

*Next*

*End Sub*

Yukarıdaki kod form üzerindeki nesnelerin BackColor özelliklerini kırmızıya ayarlar.

**CurrentX, CurrentY**

Form üzerine yapılan yazım ve çizimlerin nereden başlayacağını belirler.

*Private Sub Form\_Load()*

*Dim t, i*

*t = "Visual Basic 6.0"*

*For i = 1 To 10*

*FontSize = i \* 5*

*CurrentX = (ScaleWidth - TextWidth(t)) / 2*

*Print t*

*Next*

*End Sub*

**FillColor, FillStyle**

Circle ve Line metodu ile form üzerine çizilen çember ve kutuların rengini belirler.

**ForeColor**

Form üzerine yazılan yazının rengini belirler.

**Height**

Formun yüksekliği belirlenir. Ölçü birimi Twips'dir.

**Icon**

Formun başlık çubuğunda gösterilecek ikonunu ayarlar.

**KeyPreview**

Form aktifken basılan tuşlardan haberdar edilip edilmeyeceğini belirler.

**Left**

Formun ekranın sol kenarından ne kadar uzakta olacağını ayarlar. Formu ekrana ortalatmak için aşağıdaki kod kullanılabilir.

*Private Sub Form\_Load()*

*Left = (Screen.Width - Width) / 2*

*Top = (Screen.Height - Height) / 2*

*End Sub*

**MaxButton**

Formun başlık çubuğunda Maximize düğmesinin gösterilip gösterilmeyeceğini belirler.

**MinButton**

Formun başlık çubuğunda Minimize düğmesinin gösterilip gösterilmeyeceğini belirler.

***Moveable***

Formun taşınıp taşınamayacağını belirler.

***ScaleMode***

Form içerisinde kullanılacak ölçü birimini ayarlar.

***Name***

Formun en önemli özelliklerinden biridir. Proje içerisinde formun hangi ismi kullanacağını belir. Visual Basic bir nesneye gönderme yaparken bu name özelliğinden faydalanır. Genelde formlara verilen isimler frm harfleri ile başlar.

***ShowInTaskbar***

Formu görev çubuğunda gizlememizi sağlar.

***StartupPosition***

Form yüklendiğinde ekrandaki kordinatlarını belirler.

0: Form tasarlanırken bulunduğu pozisyonda açılır

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

2: Form ekranın ortasında açılır

3: Formun kordinatları Windows tarafından belirlenir

***Width***

Formun genişliğini belirler.

***WindowState***

Formun nasıl açılacağını ayarlar.

0-Normal ; Form normal durumda açılır

1-Minimized ; Form simge durumunda açılır

2-Maximized ; Form tam ekran durumda açılır

***Olaylar***

Visual Basic olay güdümlü bir programdır. Forma birkez tıkladığımızda, kapattığımızda ebatlarını genişlettiğinizde olaylar gelişir. Aşağıda en sık kullanılan form olayları açıklanmıştır ;

***Activate***

Formun ekranda aktif olması ile gelişir. Formun ilk yüklenmesinde Initialize ve Load olayından sonra gerçekleşir.

***Deactivate***

Activate olayın tam tersidir. Form aktiviteyi kaybettiğinde gerçekleşir.

***DragDrop***

Sürüklenen bir denetim form üzerine bırakıldığında gerçekleşir.

***Load***

Form henüz ekranda görülmeden yüklenmesi aşamasında, Initialize olayından hemen sonra gerçekleşir. En çok kullanılan olaydır.

**Resize**

Formun boyutları değiştirildiği zaman gerçekleşir. Ekranı kapla, simge durumuna küçült veya kullanıcı tarafından formun boyutu değiştirildiği anda meydana gelir.

**Unload**

Bu olay form kapatıldığında meydana gelir. Kullanıcıya formu kapatıp kapatmak istemediğini soran bir kod yazabilirsiniz. Cancel parametresine True değeri vererek kapatma işlemi iptal edebiliriz.

```
Private Sub Form_Unload(Cancel As Integer)
```

```
    Dim Cevap As Integer
```

```
    Cevap = MsgBox("Programdan çıkmak istediğinizde emin misiniz?", 20, "Brain")
```

```
    If Cevap = 5 Then
```

```
        Cancel = True
```

```
    End If
```

```
End Sub
```

**QueryUnload**

Form kapatıldığında meydana gelir. Unload olayından farkı formun nasıl kapatıldığını öğrenebilirsiniz. Kendi kodunuzla, kullanıcı veya windows tarafından ve kapatma işlemi iptal edebilirsiniz. Cancel ve UnloadMode parametreleri kullanılır. UnloadMode parametresi kontrol edilerek formun hangi yöntem ile kapatıldığı öğrenilebilir. Cancel parametresine True değeri vererek kapatma işlemi iptal edebiliriz. UnloadMode aşağıdaki değerleri alabilir;

0: Kontrol kutusunda kapat seçildi

1: Unload komutu kullanıldı

2: Windows'tan çıkılmaya çalışıldı

3: Task Manager aracılığı ile kapatılmaya çalışıldı

4:MDIChild bir form ise MDI form kapatılmaya çalışıldı

**Visual Basic Denetimlerine Giriş**

Form nesneleri Visual Basic uygulamalarının temelidir. Geliştiriceğimiz tüm uygulamalarda en az bir form olacaktır. Formları ise istediğimiz halde tasarlayabilmek için denetimleri kullanırız.

**CommandButton**

Kullanıcıdan tepkiler toplamak için form üzerine yerleştirilen denetimlerdir. Komut düğmelerinin en önemli iki özelliği Name ve Caption'dur. Name özelliği nesneyi projede kullanılan diğer kontrollerden ayırt edebilmemizi sağlar. Komut düğmelerine isim verirken genellikle cmd yazarak başlarız. Caption özelliği ekranda görülecek etiket belirler. Caption özelliği verirken & karakterini kullanarak istenilen harfin altı çizgili gösterilir. Komut düğmelerinde en çok kullanılan özelliklerden biride **Default** ve **Cancel** özellikleridir. Default özelliğine True değeri verildiğinde form üzerinde Enter tuşuna basılması ile komut düğmesinin click olayı meydana gelir. Cancel

özelliğine True değeri verilmesi ile ise form üzerinde herhangi bir denetim üzerinde ESC tuşuna basıldığında komut düğmesinin click olayı meydana gelir.

Komut düğmelerinde en çok kullanılan olay ise **Click** olayıdır. Kullanıcı komut düğmesi üzerine tıkladığı anda gerçekleşir.



### **TextBox**

Genellikle kullanıcıdan veri toplamak amacı ile kullanılır. Metin kutuların caption özelliği bulunmaz bunun yerine text özellikleri vardır. Metin kutusu içerisinde yazan değeri görmemizi sağlar. Aşağıda metin kutularının çok kullanılan **özellikleri** verilmiştir.

#### **Text**

Metin kutusunun içeriğini gösterir.

#### **MaxLenght**

Varsayılan değeri 0'dır. Metin kutusu içerisine en fazla kaç karakter yazılabileceğini ayarlar.

#### **PasswordChar**

Parola giriş kutularında kullanıcının girdiği karakterler metin kutusunda gösterilmez bunun yerin \* karakterleri gösterilir. Bu özellik ilede metin kutusuna girilen karakterler yerine gösterilmesi istenen karakter ayarlanır.

#### **MultiLine**

Metin kutusuna bir satırdan fazla bilgi girilebilmesini sağlar.

#### **ScrollBars**

MultiLine özelliği True yapılmış bir metin kutusuna kaydırma çubukları ekler.

#### **Locked**

Metin kutusu içeriğinde değişiklik yapılmamasını sağlar. Metin kutusunu kitlet.

#### **TabStop**

Tab tuşlarına basılarak Metin kutusuna ulaşıp ulaşılamayacağını ayarlar.

#### **TabIndex**

Tab tuşu ile denetimler arasında dolaşma sırasını belirler.

#### **Enabled**

Metin kutusunun aktif veya pasif olacağını ayarlar.

#### **Visible**

Metin kutusunun ekranda gösterilip gösterilmeyeceğini belirler.

Metin kutularında en sık aşağıdaki **olaylar** kullanılır.

#### **Change**

Metin kutusu içeriğinde bir değişiklik olduğunda gerçekleşir.

#### **GotFocus**

Metin kutusu etkinleştiği anda gerçekleşir.

**KeyDown**

Bir tuşa basıldığı anda gerçekleşir.

**KeyPress**

Bir tuş basılı olduğu sürece gerçekleşir.

**KeyUp**

Tuş bırakıldığı anda gerçekleşir.

**LostFocus**

İmleç metin kutusundan ayrıldığı anda gerçekleşir.

```
Private Sub Text1_LostFocus()
```

```
    MsgBox "Metin Kutusunda " & Text1.Text & " yazıyor."
```

```
End Sub
```

Metin kutularında kullanılan birkaç tane metod vardır. Bunlardan en önemlisi **SetFocus** metodudur. Bir metin kutusunu etkinleştirmek amacı ile kullanılır.

Text1.SetFocus

**Label**

Etiketlerde metin kutularına benzer, aralarında fark ise kullanıcı etiketlere değer giremez. Çok fazla özelliği yoktur. Caption özelliği kullanılarak içerisine yazılar yazılabilir.

**Alignment**

İçerisinde bulunan yazının hizalanmasını sağlar.

**AutoSize**

Etiket içerisinde bulunan değere göre boyutlarını otomatik olarak ayarlar

**BackColor**

Etiket form ile aynı rengi almasını sağlar.

**BorderStyle**

Etiket 3 boyutlu görülmesini sağlar.

**MouseMove**

Mouse işaretçisi üzerlerine geldiğinde gerçekleşir.

**UseMnemonic**

Etiketlerde Caption özelliğinde & karakteri kullanılmasını sağlar.

**Image**

Görüntü denetimleri resim gösterme işlemlerinde kullanılırlar. Resim denetime oranla daha az hafıza tüketir. Image'lar içerisinde \*.Bmp, \*.Ico, \*.Wmf, \*.Jpg, ve \*.Gif dosyaları görüntülenebilir.

**Picture**

Bu özelliği araçlığı ile resim eklenebilir.

**Stretch**

Bu özellik False ise resmin boyutlarını alır. Eğer özellik True olursa resim denetimin boyutlarına ayak uydurur.

**PictureBox**

Resim kutuları genellikle grafikleri (örneğin bmp, jpg, gif ) görüntülemek için kullanılır. Görüntü denetimleri ile çok benzerdirler. Aralarında çok az farklılık vardır. Eğer sadece bir resim göstermek isteniyorsa görüntü kutusu daha iyi bir seçimdir. Grafiği form üzerinde taşımayı düşünüyorsak resim kutuları daha esnektir. Resim kutuları içerisine çizim yapabiliriz.

**AutoSize**

Resim kutusuna bir resim yerleştirildiğinde normal boyutlarda gösterilir. Eğer resim, resim kutusundan büyük ise resim kırılır. Bu özelliğin değerini True yaparsak resim kutusu boyutlarını grafiğe uyacak şekilde ayarlamasını sağlar.

**Picture**

Resim kutusu içerisinde gösterilen resim bu özellik ile belirlenir. Hem tasarım modunda hemde çalışma modunda kullanılabilir.

**Image**

Bu özellik sadece çalışma modunda kullanılabilir. Resim kutusu içerisinde gösterilen resmi başka bir Resim kutusu içerisine kopyalarken kullanılabilir.

Picture2.Picture=Picture1.Image

**LoadPicture**

Bu metod çalışma kipinde resim kutusuna resim yüklemek için kullanılır.

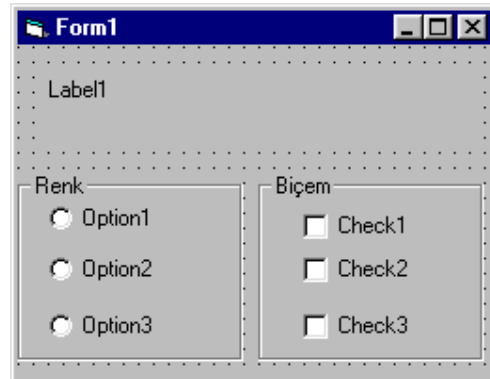
Picture1.Picture=LoadPicture(dosya adı) 'Bu kod dosya adı verilen resmi görüntüler

Picture1.Picture=LoadPicture( ) 'Bu kod ise resim kutusu içerisini boşaltır.

**Frame, OptionBox, CheckBox**

Frame'ler çerçeve olarak geçerler ve formların biraz daha profesyonel görülmelerini sağlarlar. Genellikle seçim kutuları ve onay kutuları ile birlikte kullanılırlar. Seçim kutuları ise kullanıcıya birkaç seçenek sunar. Kullanıcıda bu seçeneklerden sadece bir tanesini seçebilir. Onay kutuları da tıpkı seçim kutularına benzer fakat aynı anda birden fazla onay kutusu işaretlenebilir. Seçim ve onay kutularının en önemli özelliği Value'dur. Bu özellik True ve False değeri atayarak işaret koydurabilir veya işareti kaldırabiliriz.

File-New Project komutunu vererek yeni bir proje başlatalım. Aşağıdaki gibi bir form tasarlayalım. OptionBox ve CheckBox'ların Click olayına ise aşağıdaki kodları yazalım. Private Sub



```

Check1_Click()
    If Check1.Value = 1 Then
        Label1.FontBold = True
    Else
        Label1.FontBold = False
    End If
End Sub
Private Sub Check2_Click()
    If Check2.Value = 1 Then
        Label1.FontItalic = True
    Else
        Label1.FontItalic = False
    End If
End Sub
Private Sub Check3_Click()
    If Check3.Value = 1 Then
        Label1.FontUnderline = True
    Else
        Label1.FontItalic = False
    End If
End Sub
Private Sub Option1_Click()
    If Option1.Value = True Then
        Label1.ForeColor = vbRed
    End If
End Sub
Private Sub Option2_Click()
    If Option2.Value = True Then
        Label1.ForeColor = vbBlue
    End If
End Sub
Private Sub Option3_Click()
    If Option3.Value = True Then
        Label1.ForeColor = vbYellow
    End If
End Sub

```

Program çalıştırıldığında aşağıdaki gibi bir görünüm oluşur. Bu seçim kutuları ve onay kutuları kullanılarak istenilen özellikler verilebilir. Seçim Kutularında value değerleri için True veya False özelliği verebiliyoruz onaykutularında ise 0 ve 1 değerlerini girmemiz gerekir.





### Timer

Zamanlayıcı denetimidir. Çalışma kipinde görülmezler. Önemli iki özelliği vardır.

#### Enabled

Timer denetiminin aktif veya pasif olacağını belirler.

#### Interval

Timer denetiminin ne kadar aralıkla timer olayını üreteceğini belirleriz. Milisaniye cinsinden bir değer yazarız. 1 sn. için 1000 yazmalıyız.

#### Timer

Timer olayını kullanarak istediğimiz aralıklarla olaylar gerçekleştirebiliriz. Aşağıdaki kodu kullanarak bir saat yapabiliriz.

```
Private Sub Timer1_Timer()
    Text1.Text = Time
End Sub
```



### ScroolBars

Formumuza yatay ve dikey kaydırma çubukları eklememizi sağlarlar. Her iki denetimde aynı özellikleri taşırlar.

#### Max

Kaydırma çubuğunun alabileceği en büyük değeri belirler.

#### Min

Kaydırma çubuğunun alabileceği en küçük değeri belirler.

#### LargeChange

Kullanıcı kaydırma çubuğuna tıkladığında value değerinin en fazla ne kadar değişeceğini belirler.

#### SmallChange

Kullanıcı kaydırma çubuğuna tıkladığında value değerinin en az ne kadar değişeceğini belirler.

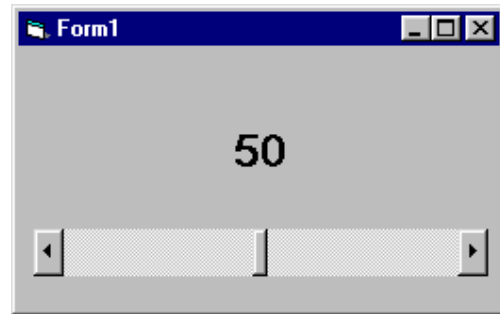
#### Value

Kaydırma çubuğunun değerini belirler.

#### Change

Kullanıcı kaydırma çubuğunun değerini değiştirdiğinde gerçekleşir.

Aşağıdaki şekilde form üzerine bir kaydırma çubuğu ve birde etiket yerleştirilmiştir. Kaydırma çubuğunun Max değeri 100 Min değeri ise 0 olarak ayarlanmıştır. Kullanıcının kaydırma çubuğunu sürüklediğinde etiket içerisinde değerini görmek için aşağıdaki kodu girmeliyiz.



```
Private Sub HScroll1_Change()  
    Label1.Caption = HScroll1.Value  
End Sub
```



### **Listbox**

Liste kutuları kullanıcılara bir liste sunmanın ideal bir yoludur. Kullanıcılar liste kusuntunda bulunan verileri inceleyebilir ve istediklerini seçebilirler. Liste kutularının bazı özellik ve metotları şunlardır.

#### **List**

Bu özellik ile listedeki herhangi bir öğenin değeri öğrenilebilir. Listedeki ilk elamanın list özelliği 0'dır. Örneğin listede bulunan 3. elemanın değerini öğrenmek için şunu yazmalıyız;

```
MsgBox List1.List(2)
```

#### **ListIndex**

Listede seçili olan elemanın indeks numarasını verir.

```
MsgBox List1.List(List1.ListIndex)
```

Yukarıdaki ifade yerine liste kutusunun text özelliğinde kullanılabilir.

```
List1.Text
```

Yukarıdaki her iki ifade de aynı işi gerçekleştirir.

#### **ListCount**

Listede bulunan eleman sayısını verir. Eğer ListCount 0 ise liste kutusu boştur.

#### **MultiSelect**

Bu özellik True olarak ayarlanmış ise listeden aynı anda birden fazla eleman seçebiliriz. Listelerden birden fazla seçim yapmak için **Shift** tuşu kullanılır.

#### **Sorted**

Sadece tasarım kipinde ayarlanabilecek bir özelliktir. Listedeki elemanları otomatik olarak sıralar.

#### **AddItem**

Liste kutularında kullanılan en önemli metottur. Listeye yeni elemanlar eklemek için kullanılır.

```
List1.AddItem "Uğur ŞAHİN"
```

#### **RemoveItem**

Listeden elemanları çıkartmak için kullanılan bir metottur.

```
List1.RemoveItem List1.ListIndex
```

Yukarıdaki kod listeden seçili elemanı siler.

#### **Clear**

Liste boşaltır.

```
List1.Clear
```

**ComboBox**

Açılır liste kutularında liste kutuları ile aynı özellik ve metotları paylaşır. 3 değişik görünüm sunarlar; Bu görünümleri style özelliğini kullanarak değiştirebiliriz. Alabilecekleri değerler

0-Dropdown Combo ; İçerisine yazılabilir ComboBox

1-Simple Combo ; TextBox Görünümünde

2-Dropdown List ; Salt okunur ComboBox.

Combo1.Style = 2

**Shape**

Form üzerine dikdörtgen, kare, elips, çember ve oval çizmek için kullanılır.

**Shape**

Shape kontrolunun çizeceği şekli belirler.

0: Dikdörtgen

1: Kare

2: Elips

3:Çember

4: Oval Diktörgen

5: Oval Kare

**BorderStyle**

Nesnenin çerçeve biçimini belirler.

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

**BorderWidth**

Çerçeve kalınlığını belirler. BorderStyle özelliği Solid ve Inside Solid iken kullanılır.

**FillColor**

Şeklin iç boyama rengini belirler.

**FillStyle**

Şeklin içini boyamak için kullanılacak deseni belirler.

**Line**

Form üzerine çizgi çizmek için kullanılır.

**X1, X2, Y1, Y2**

X1 ve Y1 başlangıç noktasını X2 ve Y2 ise bitiş noktasını belirler.

**DriveListBox**

Sürücü listelerini çağırmak için kullanılır. Dizin ve dosya liste kutuları ile birlikte kullanılır.

Name dışında önemli tek özelliği Drive'dır.

**Drive**

Kullanıcının seçmiş olduğu sürücüyü gösterir. Çalışma kipinde kullanılabilir.

**Change**

En popüler olayıdır. Kullanıcı bir sürücü seçtiğinde gerçekleşir.

**DirListBox**

Dizinleri listeletmek için kullanılır. Sürücü ve dosya liste kutuları ile birlikte kullanılır.

**Path**

Gösterilecek dizini ayarlayan özelliğidir. Çalışma kipinde kullanılabilir.

**Change**

En popüler olayıdır. Kullanıcı bir dizin seçtiğinde gerçekleşir.

**FileListBox**

Dosyaları listeletmek için kullanılır. Sürücü ve dizin liste kutuları ile birlikte kullanılır.

**FileName**

Seçili dosyanın adını verir.

**Path**

Gösterilecek dosyaların hangi dizin veya sürücüde olduğunu belirler. Çalışma kipinde kullanılır.

**Pattern**

Hangi dosyaların gösterileceğini belirleyen bir özelliktir. Bir filtre görevi görür. Gösterilecek dosyaların hangi dizin veya sürücüde olduğunu belirler. Çalışma kipinde kullanılır.

**Archive**

True yada false değerini alabilir. Arşiv dosyalarının görünümünü sağlar. Başlangıç değeri True'dur

**Hidden**

True yada false değerini alabilir. Gizli dosyalarının görünümünü sağlar. Başlangıç değeri False'dir

**ReadOnly**

True yada false değerini alabilir. Salt okunur dosyalarının görünümünü sağlar. Başlangıç değeri True'dur

**System**

True yada false değerini alabilir. Sistem dosyalarının görünümünü sağlar. Başlangıç değeri False'dur

**Change**

Kullanıcı bir dosya seçtiğinde gerçekleşir.

**PathChange**

Path değeri değiştiğinde gerçekleşir.

**PatternChange**

Pattern değeri değiştiğinde gerçekleşir.

Yandaki şekilde görüldüğü gibi bir form dizayn edelim.

Formumuza bir sürücü listesi, bir dizin listesi, bir dosya listesi ve bir de image denetimi ilave edelim ve bu denetimlerin özelliklerini aşağıdaki gibi değiştirelim sonrada aşağıda görülen kodu girelim.

Nesne	Özellik	Değeri
File1	Pattern	*.bmp;*.wmf;*.ico
Image1	Stretch	True
Image1	BorderStyle	1-Fixed Single

```
Private Sub Drive1_Change()
```

```
Dir1.Path = Drive1.Drive
```

```
End Sub
```

```
Private Sub Dir1_Change()
```

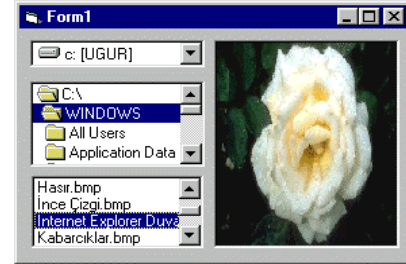
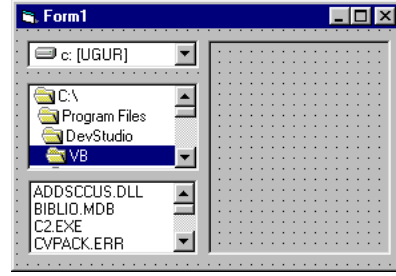
```
File1.Path = Dir1.Path
```

```
End Sub
```

```
Private Sub File1_Click()
```

```
Image1.Picture = LoadPicture(File1.Path & "\" & File1.filename)
```

```
End Sub
```

**Drag & Drop**

Drag&Drop olayı bir nesnenin bir yerden başka bir yere sürüklenmesi olayıdır. Bu olayları takip edebilmek için Visual Basic bir takım özellik olay ve yöntem sunar.

**Drag Icon**

Bir nesne taşınırken alacağı şekli belirler.

**Drag Mode**

Bu özellik iki farklı değer alabilir.

0 : vbManual ; Bu durumda kullanıcı bir nesneyi sürükleyip bıraktığında, nesnenin taşınabilmesi için üzerine bırakılan nesnenin DragDrop olayına kod yazılır. Nesnelere taşımak için Drag yönteminden faydalanılır.

1: vbAutomatic; Bu durumda nesne otomatik olarak sürüklenir. Bu yöntem kullanıldığında nesnenin click olayı çalışmaz.

DragMode özelliği manual olarak belirtilmişse kontrollerde Drag olayı, o kontrolün MouseDown olayında Drag metodu ile başlar.

```
Private Sub Text1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    'Farenin sol tuşu basılı ise drag olayını başlat
    If Button = 1 Then
        Text1.Drag
    End If
End Sub
```

### DragDrop(Source As Control, X As Single, Y As Single)

Taşınan bir nesnenin bir kontrol üzerine bırakılması olayıdır. Bırakma işlemi için kod bu olaya yazılmalıdır.

Source parametresi üzerine bırakılan nesneyi ifade eder. X ve Y parametreleri ise sürüklenen elemanın bırakıldığı koordinatları verir.

### DragOver(Source As Control, X As Single, Y As Single, State As Integer)

Taşıma esnasında hedef nesne üzerinden geçerken bu olay meydana gelir.

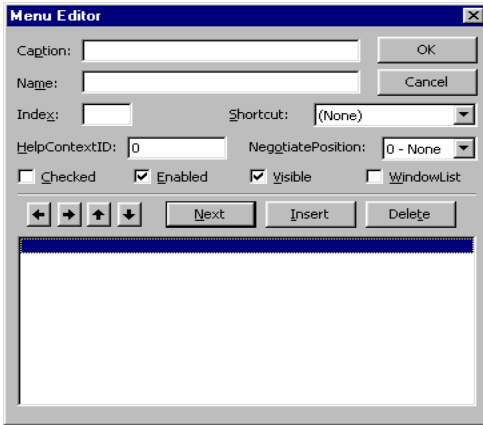
Source parametresi üzerine bırakılan nesneyi ifade eder. X ve Y parametreleri sürüklenen elemanın bırakıldığı koordinatları verir. State parametresi ise 3 farklı değer alabilir.

0: vbEnter; Taşınan nesne hedef üzerine giriş yaptı

1: vbLeave; Taşınan nesne hedef üzerinden ayrıldı

2: vbOver; Taşınan nesne hedef üzerinde geçiş yapıyor.

## Formlarımıza Menü Ekleyelim



Bütün profesyonel Windows programlarında menüler bulunmaktadır. Visual Basic formlarımıza menü ekleyebilmemiz için bize imkan sağlar. Bir forma menü eklemek için Tools-Menü Editor komutunu veririz. Bu komut verildikten sonra karşımıza yandaki gibi Menü Editor penceresi gelir.

Formlarımıza menü eklerken ve menüleri düzenlerken yanda görülen pencereden faydalanacağız. Oluşturmak istediğimiz menünün başlığını Caption adlı bölüme gireriz. Name kısmına ise bu menü için bir ad yazarız. İlk menümüzü belirledikten sonra Next düğmesine basarız. Visual Basic Caption ve Name kutucuklarını bir sonraki menüyü ilave edebilmemiz için boşaltır. Yeni menü öğeleri yaratıldıkça, aşağıda bulunan liste kutusunda listelenirler. Alt menüleri oluşturmak için ise Next düğmesinin sol tarafında bulunan ok düğmelerini kullanırız. Var olan bir menünün silinmesi için liste kutusundan menüyü seçer ve Delete butonuna tıklarız. Yeni menüler ilave ederken ise Insert butonundan faydalanırız.

Microsoft Word'de Düzen menüsüne tıkladığımız zaman Kopyala için Ctrl + C, Kes için Ctrl + X, Yapıştır için Ctrl + V kısayol tuşları görülür. Visual Basic'de de bu tip menüler oluşturmak

istediğimizde Menü Editor penceresindeki liste kutusundan bir menü seçtikten sonra Shortcut bölümünü kullanabiliriz.

### Popup Menüler

Windows'ta sağ düğmeye bastığımızda çıkan menülere popup menüler adını veririz. Visual Basic'de hazırladığımız projelerde de popup menüler oluşturabiliriz. Popup menü tasarımı normal menü tasarımı gibi gerçekleşir. Sadece bu menünün visible özelliğine false değeri verilir ve açılması istenilen yerde **PopupMenu** yöntemi kullanılır.

PopupMenu MenüAdı,Flags, X, Y, Bold

**MenüAdı**; Menü Editörde tasarlanan menünün adı

**Flags**; X parametresinin nasıl kullanılacağını ve açılan menüde farenin hangi tuşunun click olayını gerçekleştireceğini belirler.

Flags parametresi aşağıdaki gibi kullanılır;

Flags=KoorX + Fare

KoorX aşağıdaki değerleri alır

0: Menü X kordinatının solunda yer alır

4: Menü X kordinatını ortalayacak

8: Menü X kordinatının sağında yer alır

Fare aşağıdaki değerleri alır

0: Sol fare tuşu ile menüden eleman seçilebilecek

2: Sağ fare tuşu ile menüden eleman seçilebilecek

**Bold**; Bu parametre ile menüdeki elemanlardan biri kalın yapılabilir

```
Private Sub Text1_MouseDown(Button As Integer,Shift  
As Integer, X As Single, Y As Single)
```

```
    'Sağ fare düğmesine basıldı ise
```

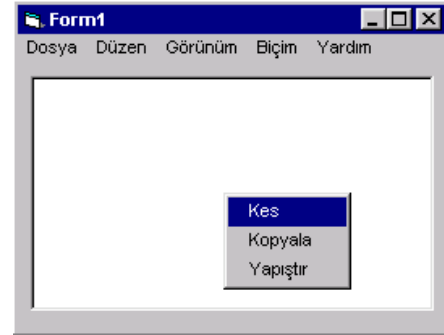
```
    If Button = 2 Then
```

```
        PopupMenu mnuDuzen, 4, Text1.Left + X,
```

```
        Text1.Top + Y
```

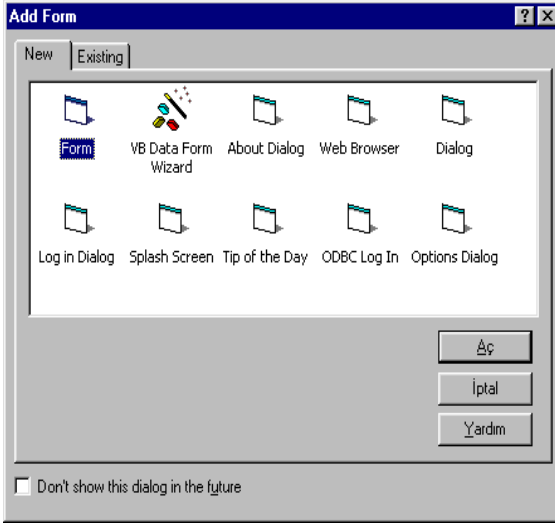
```
    End If
```

```
End Sub
```



## Formlarla Çalışmak,

Hazırladığımız projelerde çoğu zaman kullanıcı ile iletişim kurabilmek için birden fazla forma ihtiyaç duyarız. Visual Basic bu konuda bize yardımcı olmakta, projelerimize birden fazla form eklememize imkan sağlamaktadır. Projeye eklenen her yeni forma Visual Basic bir isim vermektedir. Yeni bir proje başlattığımızda Visual Basic bir tane form oluşturmakta ve buna Form1 adını vermektedir. Daha sonradan bizim eklediğimiz formlar ise sırası ile Form2, Form3 gibi isimler alacaktır.



Bir projeye yeni bir form eklemek için Project menüsünü kullanabileceğimiz gibi Project Explorer penceresine sağ tıklayarak gelen menüden Add seçeneğini kullanarak da yeni bir form ekleyebiliriz. Bu komutlar verildikten sonra Visual Basic bize aşağıdaki gibi bir pencere sunacak.

Bu pencerede Visual Basic ile beraber gelen birçok hazır form şablonu bulunmaktadır. Eğer istersek bu şablonlardan birini seçerek projemize ilave edebilir veya Form seçeneğini kullanarak boş bir form ekleyebiliriz.

Projemize yeni bir form ekledikten sonra, eklenen yeni form adı Project Explorer penceresinde görülebilir. Projemizde birden fazla form varsa eğer, hangi formla çalışmak istiyorsak Project Explorer penceresinden formu seçip View Object butonuna basarız.

Eklediğimiz yeni formu projemize kaydetmek için File-Save Form komutunu verebiliriz. Karşımıza gelen pencereden formun bulunmasını istediğimiz klasör ve forma vermek istediğimiz ismi belirtiriz. Visual Basic formu **.frm** uzantısı ile kaydeder.

Visual Basic'de eğer bir forma ihtiyacımız kalmadı ise, o formu projemizden çıkarmak için File-Remove Form komutunu kullanırız. Böylece form projeden ve Project Explorer penceresinden çıkartılır.

Visual Basic başka bir projede kullandığımız bir formu üzerinde çalıştığımız projeye eklememize izin verir. Böylece tekrar tekrar aynı tip formları dizayn etmekle uğraşmak zorunda kalmayız. Add Form komutunu verdiğimizde gelen pencerede Existing sekmesini kullanarak istediğimiz **.frm** uzantılı dosyayı seçerek projemize ekleyebiliriz.

## Program Kodları ile Formları Yönetmek

Projelerimizde yeni bir formu yüklemek için aşağıdaki yöntem kullanılır.

Load FormAdı

Burada FormAdı yerine yüklemek istediğimiz formun adını yazmalıyız. Daha önceden yüklenmiş olan bir formu göstermek için ise aşağıdaki sözdizimi kullanılır.

FormAdı.Show Mode

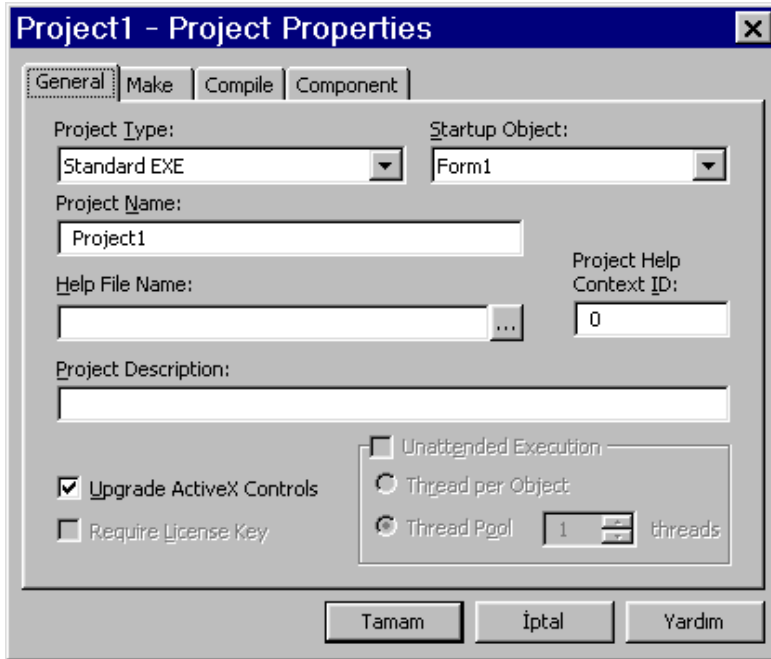
Mode parametresine 1 verilirse form kalıcı olacak, 0 verilirse kalıcı olmayacaktır. Eğer mode bildirilmezse Visual Basic varsayılan değeri olan 0'ı kullanacaktır. Kalıcı formlara örnek olarak ekrana gelen mesaj pencerelerini verebiliriz. O pencerede kapanmadan başka bir işlem yapamayız.

Bir formu henüz yüklemeyen Show yöntemini kullanırsak Visual Basic önce formu yükleyecek yani load olayını gerçekleştirecek ardından ise formu gösterecektir.

Açık olan veya yüklenmiş olan bir formu kapatmak için ise aşağıdaki yöntemler kullanılır.

Unload FormAdı 'Formu hafızadan kaldırır.

FormAdı.Hide 'Formu gizli hale getirir. Form hala hafızada yer işgal eder.



Visual Basic'de birden fazla form ile çalışırken projeyi çalıştırdığımızda hangi formun ilk olarak çalışacağını belirlemek için Project -Project Properties komutunu kullanmalıyız. Bu komutu verdikten sonra karşımıza yukarıdaki gibi bir pencere gelecek. Gelen pencereden Startup Object adlı açılan liste kutusundan projenin hangi form ile açılacağını belirleyebiliriz.

## MDI Formlar

Hazırlayacağımız projelerde genellikle birden fazla form kullanacağız. Bu formlardan birisi Ana Formumuz olacak, diğerleri ise bu form içerisinde çalışan Yavru Formlar olacaktır. Örneğin Microsoft Word, Microsoft Excel bu şekilde tasarlanmış uygulamalardır. Word ve Excel içerisinde birden fazla form açılmaktadır. Uygulama simge durumunda küçültüğünde veya kapatıldığında içerisinde bulunan formlar da otomatik olarak kapanmaktadır. Visual Basic'de bu tip Formlara MDI Form ve MDI Child Form adı verilmektedir.

Projemize bir MDI Form eklemek için Project-Add MDI Form komutunu veririz. Bir projede en fazla bir tane MDI Form olabilir. Projemizde bulunan diğer formları ise yavru form olarak tanımlamak için Properties penceresinden MDIChild özelliklerine True değerini atamamız gerekir. MDI Form aşağıdaki özellikler dışında diğer formlarla aynı işlevleri sağlarlar.

- Bütün yavru formlar ait oldukları MDI Form içerisinde görülürler.
- Bir yavru form simge durumunda küçültüğünde, görev çubuğunda değilde ait olduğu MDI Form içerisinde küçük bir başlık çubuğuna dönüşür.
- MDI Form simge durumunda küçültüğünde, yavru formlarda kendisi ile beraber simge durumunda küçülür ve hepsi görev çubuğunda tek bir düğme olarak gösterilirler.
- Bütün yavru formlarda bulunan menüler, MDI Formun menü çubuğunda görülür.

## Visual Basic'de Çizim

Visual Basic'de çizim yapabilmek için Line, Circle, Pset yöntemlerini kullanırız. Bu yöntemler Form ve Printer için kullanılabilir. Form üzerine çizim yapılırken formun AutoRedraw özelliğini True yapmalıyız.

Visual Basic'de çizim yaparken kordinat sistemini anlamalıyız. Çizim yaparken X ve Y kordinatlarını kullanırız. Y kordinatının yukarıdan aşağı inerken artar ve başlangıç değeri 0'dır. X kordinatı ise soldan sağa giderken artar ve başlangıç değeri 0'dır. Formun sol üst köşesi kordinatları 0,0 dır.

Form üzerinde çizim yapılacak alan formun iç kısımlarıdır. Width ve Height özellikleri ise formun çerçeveler ve başlık çubuğu dahil olmak üzere genişlik ve yüksekliğini verir. Formun iç yükseklik ve genişliğini ise **ScaleWidth** ve **ScaleHeight** özelliklerini kullanarak öğrenebiliriz.

**DrawWidth**; Çizgi kalınlığını belirler.

### Line Yönetimi

**Line (X1,Y1) - (X2,Y2), renk**;Line yöntemi form üzerine çizgi çizmek için kullanılır. X1 ve Y1 başlangıç, X2 ve Y2 bitiş noktasını belirler. Renk paramtresi seçimlidir. Eğer istenirse ForeColor özelliğine değer atayarakda renkli çizimler yapılabilir.

```
Private Sub Form_Load()  
    DrawWidth = 5  
    For i = 0 To Form1.ScaleHeight Step Form1.ScaleHeight / 10  
        Line (0, i)-(Form1.ScaleWidth, i), vbRed  
    Next  
End Sub
```

### Circle Yönetimi

**Circle (MX,MY), R , renk**; Circle yöntemi merkezi MX ve MY olan çemberi R yarıçapında verilen renkte çizer. FillStyle özelliği 0 yapıldıktan sonra ve FillColor özelliğini kullanılarak içleri renklendirilebilir.

```
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, vbRed
```

```
Next
```

```
End Sub
```

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
FillStyle = 0
```

```
FillColor = Rnd * 16777216
```

```
Circle (X, Y), 500
```

```
End Sub
```

## **Pset Yönetimi**

**Pset(X,Y), renk;** Pset yöntemi X ve Y olarak verilen kordinatlara bir nokta koyar.

```
Private Sub Form_Load()
```

```
Dim x, y, i, r
```

```
r = ScaleHeight / 2
```

```
While r > 0
```

```
    i = (i + 1) Mod 360
```

```
    r = r - 1 'Yarıçığı 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
```

## **Yazıcı Kullanımı**

Er yada geç projelerimizde yazıcıyı kullanmak zorunda kalacağız. Visual Basic ile yazıcıdan döküm alabilmek için Windows ortamında tanıtılmış bir yazıcıya ihtiyaç duyulur. Visual Basic yazıcı kullanımı için Printer nesnesini sunar. Printer nesnesinin önemli özellikler aşağıda verilmiştir;

<b><u>Özellik</u></b>	<b><u>Tanımı</u></b>
• DeviceName	Yazıcı ismi bu özellikle öğrenilebilir
• FontName	Metinde kullanılacak yazı tipinin adını belirler
• FontSize	Metinde kullanılacak yazı tipi boyutunu belirler
• FontBold	Metinde kullanılacak yazı tipi biçiminin kalın olup olmayacağını belirler
• FontItalic	Metinde kullanılacak yazı tipi biçiminin eğik olup olmayacağını belirler
• Page	Basılmakta olan sayfa numarasını belirler

- CurrentX Yazdırma işlemine sayfanın üstünden ne kadar boşluk bırakılarak başlanacağını belirler.
- CurrentY Yazdırma işlemine sayfanın sol tarafından ne kadar boşluk bırakılarak başlanacağını belirler.
- ColorMode Yazdırmanın renklimi siyah beyazını olacağını belirler.
- Copies Alınacak kopya sayısı belirlenir.
- Orientation Sayfanın yatayını, dikeyini kullanılacağını belirler
- Papersize Sayfa boyutunu belirler.

### Yöntem

### Tanımı

- Print Belirlenen metnin yazıcıdan çıkmasını sağlar.
- Circle, Line, Pset Yazıcıda çizi yapmayı sağlar
- NewPage Baskı işlemine yeni bir sayfadan devam edileceğini belirler
- EndDoc Baskı işleminin bittiğini belirler
- KillDoc Sürmekte olan baskı işlemini sona erdirir.

Visual Basic'de kullanılan varsayılan ölçü birimi Twip'dir. Bir Twip 1/1440 inç'e tekabül etmektedir. Buna göre A4 ebatındaki bir kağıtın boyutları Twip cinsinden 11909 X 16834'dür.

```
Printer.CurrentY = 300
```

```
Printer.CurrentX = 400
```

```
Printer.FontName = "Times New Roman"
```

```
Printer.Font.Size = 16
```

```
Printer.Font.Bold = True
```

```
Printer.Print "Brain Bilişim Akademisi "
```

```
Printer.EndDoc
```

Yukarıda kod ile üstten ve yandan ne kadar boşluk bırakılacağını belirledikten sonra yazı tipini ve biçimini belirledik. Ardından Printer.Print yazdırmak istediğimiz metni belirledik. Son olarak da Printer.EndDoc komutunu vererek yazdırma işinin son bulduğunu belirledik.

Circle, PSet gibi işlevler Printer nesnesiyle kullanılabilir. Bu yöntemler kullanılarak sayfalarımıza şekillerde çizdirebiliriz.

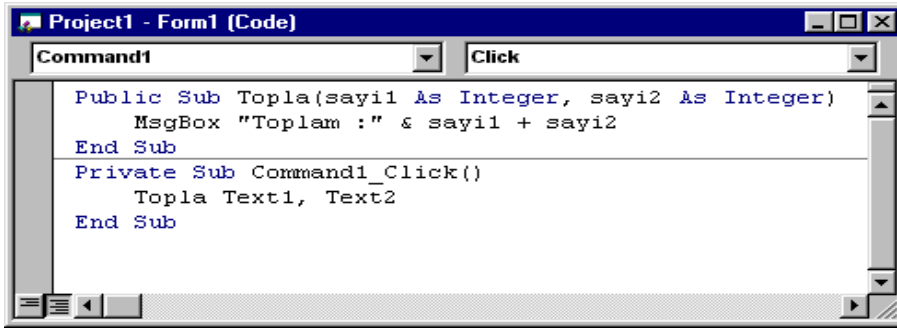
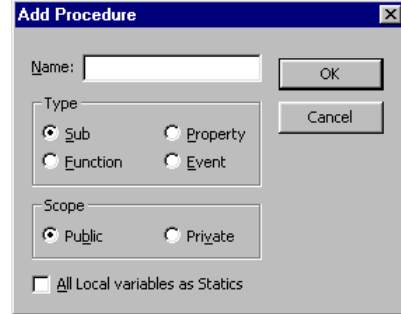
## **Yordamlar ve Fonksiyonlar**

Visual Basic'de sık sık tekrarlanan kodlar yazmak yerine yordam ve fonksiyonlar oluşturulabilir. Oluşturulan bu yordam veya fonksiyonlar program içerisinde istenilen yerden ismi kullanılarak çağrılabilir. Bu sayede tekrar eden kodlar yazmaktan kurtulabiliriz. Eğer oluturduğumuz yordam geriye bir değer gönderiyorsa fonksiyon adını veririz.

### Procedure (Yordam)

Projelerimizde şimdiye dek hep nesnelere ait olaylar üzerine kodlar yazdık. Eğer proje içerisinde birçok kez tekrarlanan kodlar varsa sürekli olarak aynı kodları yazmak yerine bu işlemleri gerçekleştiren yordam'lar yazarız. Visual Basic'de yazılan bir yodama sadece yordam adı belirterek çağırabiliriz.

Visual Basic'de yeni bir yordam oluşturmak için Tool-Add Procedure komutunu veririz. Bu komut verildikten sonra yandaki gibi bir pencere ile karşılaşırız. Name kutucuğuna yordama vermek istediğimiz adı yazarız. Type bölümünden tür seçeriz. Yordamlar için Sub Fonksiyonlar için ise Function seçilir. Scope bölümünden ise Public işaretlenir. Tüm bu işlemler yapıldktan sonra Visual Basic formumuzun General Declarations bölümünde belirlediğimiz isimde bir yordam oluşturur. Eğer bu yordamın paramtre almasını istiyorsak parantezler içerisinde değişkenler tanımlanır.

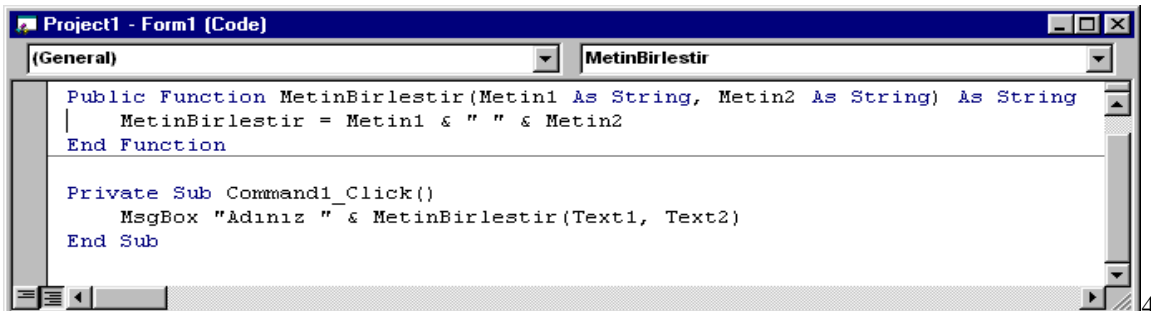


Yukarıda tanımlanmış bir yordam örneği var. Command1 nesnesinin Click olayı gerçekleştiğinde yordam çalışıyor.

### Function (Fonksiyon)

Fonksiyonların oluşturulması ve kullanılması tıpkı yordamlara benzer. Fonksiyonlar yordamlardan farklı olarak geriye bir değr gönderirler.

Bir fonksiyon oluşturmak için Tool-Add Procedure komutunu veririz. Name kutucuğuna fonksiyona vermek istediğimiz adı yazarız ve Function seçeneğini işaretleriz. Fonksiyonun sadece form üzerinde çalışmasını istiyorsak Private Bütün Projede çalışmasını istiyorsak Public işaretlenmelidir. Tüm bu işlemler yapıldktan sonra Visual Basic formumuzun General Declarations



bölümünde bir fonksiyon oluşturur. Eğer bu fonksiyona paramtre vermek istiyorsak parantezler içerisine değişkenler tanımlanır. Parantez sonrasında ise fonksiyonun geriye döndüreceği veri tipini belirlememiz gerekir.

Yukarıda tanımlanmış bir fonksiyon örneği var. Command1 nesnesinin Click olayı gerçekleştiğinde fonksiyon çalışıyor.

## Diziler

Aynı tür bilgileri bellekte tutmak için kullanabileceğimiz listelere dizi adı verilir. Dizi kullanmanın avantajı, aynı tür bilgileri bir listede tutarak daha hızlı işlem yapılabilmesidir. Visual Basic'de dizi şu şekildedir

### Örnek

```
Private Sub Form_Load()  
    Dim Ad(2) As String  
    Ad(0) = "Ali"  
    Ad(1) = "Ahmet"  
    Ad(2) = "Ebru"  
End Sub
```

## Option Base

Eğer diziyi 0'dan değilde 1'den itibaren başlatmak istersek diziyi tanımlamadan önce **Option Base 1** satırını eklemeliyiz.

## Statik Diziler

Bu tip dizilerde kullanılacak yer sayısı sabittir. Bu tip diziler sadece tanımlandıkları modül içerisinde kullanılabilirler. Statik dizi şu şekilde tanımlanabilir

Dim dizi\_adı(sayı) As Veri\_Tipi

Tüm proje içinde kullanılacak bir dizi tanımlanmak isterse standart modülün General Declarations bölümünde yukarıdaki şekildeki gibi tanımlanmalıdır

### Örnek

```
Option Base 1  
Dim a(5) As Integer  
Private Sub Form_Load()  
    a(1) = 10  
    a(2) = 20  
End Sub
```

```
Private Sub Command1_Click()  
    Text1.Text = a(1) + a(2)  
End Sub
```

Bu örnekte görüldüğü gibi projenin General Declarations kısmında a(5) dizisi tanımlanıyor. Bu dizi tanımlanmadan önce dizi indislerinin 1 den itibaren başlayacağını belirten Option Base 1 satırı koda dahil edilmiştir. Form1 yüklendiğinde bu dizinin ilk elemanına 10 sayısı ikinci elemanına 20 sayısı atanıyor. Eğer kullanıcı Command1 isimli butona tıklarsa dizinin ilk ve ikinci elemanları toplanarak Form üzerinde Text1 adlı nesnenin Text özelliğine atanıyor. Yani TextBox'ın bu sayıların toplamını göstermesi sağlanıyor.

### **Dinamik Diziler**

Bu tip dizilerde kullanılacak yer sayısında bir sınırlama yoktur. Bu tip diziler ilk başta aşağıdaki şekildeki gibi bir tanımlamaya ihtiyaç duyarlar

```
Dim dizi_adi( ) As Veri_Tipi
```

Daha sonra bu dizi kullanılacak iken indis sayısını belirtmek gerekir bunun içinde aşağıdaki gibi bir tanımlama yapılmalıdır.

```
ReDim dizi_adi(boyut ) As Veri_Tipi
```

Artık bu veri dizisini projemiz içinde kullanabiliriz.

### **Örnek**

```
Option Base 1  
Private Sub Form_Load()  
    Dim s() As String  
End Sub
```

```
Private Sub Command1_Click()  
    ReDim s(10) As String  
    s(1) = "Selam "  
    s(2) = "Ayşe"  
    Text1.Text = s(1) + s(2)  
End Sub
```

Bu dizi tanımlanmadan önce dizi indislerinin 1'den itibaren başlayacağını belirten Option Base 1 satırı koda dahil edilmiştir. Form1 yüklendiğinde s adlı bir dinamik dizi tanımlanmaktadır. Kullanıcı Command1 isimli butona tıkladığında, ReDim anahtar sözcüğü ile s dizisinin boyutu belirlenir. Bu dizinin ilk elemanına "Selam " değeri ikinci elemanına "Ayşe" değeri atanıyor. Dizinin

ilk ve ikinci elemanları birleştirilerek Form üzerinde Text1 adlı nesnenin Text özelliğine atanıyor. Yani TextBox'ın bu değerlerin toplamını göstermesi sağlanıyor.

Dim Ad As String \* 20

## Clipboard

Windows işletim sistemi Kopyala-Yapıştır ve Kes-Yapıştır yöntemini kullanırken Clipboard nesnesinden faydalanır. Visual Basic'de Clipboard'u nesnesini kullanabilmemiz için birkaç yöntem sunar.

### *Clear*

Clipboard nesnesinin içeriğini temizler.

### *SetText*

Clipboard nesnesinin içerisine gidecek yazıyı belirler. Kopyalama işlemlerinde kullanılır.

### *GetText*

Clipboard nesnesinin içerisindeki yazının alınacağını belirler. Yapıştır işlemlerinde kullanılır.

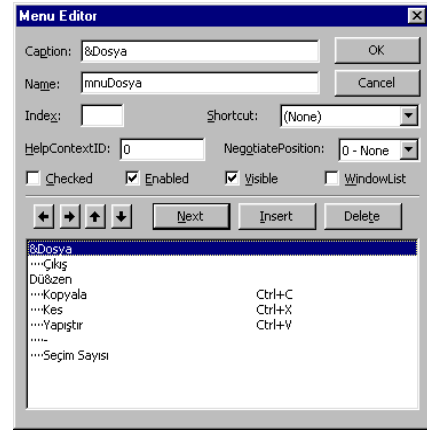
### *SetData*

Clipboard nesnesinin içerisine gidecek resini belirler. Kopyalama işlemlerinde kullanılır.

### *GetData*

Clipboard nesnesinin içerisindeki resmin alınacağını belirler. Yapıştır işlemlerinde kullanılır.

Visual Basic'de Clipboard nesnesini kullanarak kes, kopyala ve yapıştır işlemlerini gerçekleştiren yeni bir proje tasarlayalım. İlk önce yeni bir proje başlatalım. Projenin formuna bir TextBox nesnesi yerleştirip Text özelliğindeki değeri boşaltalım, Left ve Top özelliklerinede 0 değerini verelim. Bu sayede formun sol üst köşesinde içi boş bir TextBox nesnesi oluşturmuş olduk. Ardından Menü editörü açarak yanda görüldüğü gibi menüleri ekleyelim. Formumuzu tasarladıktan sonra kes, kopyala ve yapıştır işlemlerini gerçekleştirecek kodları Code Editörü açarak aşağıdaki gibi girelim. Eğer istenirse yeni menüler ilave ederek kullanıcının yazı biçimlerini değiştirmesinide sağlayabilirsiniz.



```
Private Sub Form_Load()  
    Text1.Text = ""  
    Text1.Left = 0  
    Text1.Top = 0  
    Text1.Width = Form1.ScaleWidth  
    Text1.Height = Form1.ScaleHeight  
End Sub  
  
Private Sub Form_Resize()  
    Text1.Width = Form1.ScaleWidth  
    Text1.Height = Form1.ScaleHeight  
End Sub  
  
Private Sub mnuCikis_Click()  
    End  
End Sub  
  
Private Sub mnuDuzen_Click()  
    If Text1.SelLength = 0 Then  
        mnuKes.Enabled = False  
        mnuKopyala.Enabled = False  
    Else  
        mnuKes.Enabled = True  
        mnuKopyala.Enabled = True  
    End If  
End Sub
```

```
Private Sub mnuKes_Click()  
    Clipboard.Clear  
    Clipboard.SetText Text1.SelText  
    Text1.SelText = ""  
End Sub  
  
Private Sub mnuKopyala_Click()  
    Clipboard.Clear  
    Clipboard.SetText Text1.SelText  
End Sub  
  
Private Sub mnuSecim_Click()  
    Dim Mesaj As String  
    Dim KarakterSayisi As Integer  
    KarakterSayisi = Text1.SelLength  
    Mesaj = "Seçili karakter sayısı : "  
    Mesaj = Mesaj & Str(KarakterSayisi)  
    MsgBox Mesaj  
End Sub  
  
Private Sub mnuYapistir_Click()  
    Text1.SelText = Clipboard.GetText  
End Sub
```