T.C.
MİLLİ EĞİTİM BAKANLIĞI

MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİN
GÜÇLENDİRİLMESİ PROJESİ)

BİLİŞİM TEKNOLOJİLERİ

GÖRSEL PROGRAMLAMA KOMUTLARI

ANKARA 2007

Milli Eğitim Bakanlığı tarafından geliştirilen modüller;

- Talim ve Terbiye Kurulu Başkanlığının 02.06.2006 tarih ve 269 sayılı Kararı ile onaylanan, Mesleki ve Teknik Eğitim Okul ve Kurumlarında kademeli olarak yaygınlaştırılan 42 alan ve 192 dala ait çerçeve öğretim programlarında amaçlanan mesleki yeterlikleri kazandırmaya yönelik geliştirilmiş öğretim materyalleridir (Ders Notlarıdır).

- Modüller, bireylere mesleki yeterlik kazandırmak ve bireysel öğrenmeye rehberlik etmek amacıyla öğrenme materyali olarak hazırlanmış, denenmek ve geliştirilmek üzere Mesleki ve Teknik Eğitim Okul ve Kurumlarında uygulanmaya başlanmıştır.

- Modüller teknolojik gelişmelere paralel olarak, amaçlanan yeterliği kazandırmak koşulu ile eğitim öğretim sırasında geliştirilebilir ve yapılması önerilen değişiklikler Bakanlıkta ilgili birime bildirilir.

- Örgün ve yaygın eğitim kurumları, işletmeler ve kendi kendine mesleki yeterlik kazanmak isteyen bireyler modüllere internet üzerinden ulaşılabilirler.

- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.

- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

AÇIKLAMALAR

KOD	**482BK0069**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Veri Tabanı Programcılığı**
MODÜLÜN ADI	**Görsel Programlama Komutları**
MODÜLÜN TANIMI	Komutların tanıtımı ile ilgili öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Görsel Programlama Kod Parçaları modülünü bitirmiş olmak.
YETERLİK	Görsel programlama dilinin komutlarını kullanmak.
MODÜLÜN AMACI	**Genel Amaç** Gerekli ortam sağlandığında, programlama dilinin komutlarını yazabileceksiniz. **Amaçlar** **1.** Nesnelerin ortak üyeleri ile program yazabileceksiniz. **2.** Basit veri giriş ve çıkış diyalog kutuları ile çalışabileceksiniz. **3.** Sık kullanılan fonksiyon komutları ile çalışabileceksiniz. **4.** Döngü komutları ile program yazabileceksiniz. **5.** Alt program ve fonksiyon yazabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Bilgisayar laboratuvarı ve bu ortamda bulunan, görsel programlama için gerekli donanıma sahip bilgisayar, lisanslı işletim sistemi programı, kâğıt ve kalem hazır bulundurulmalıdır.
ÖLÇME VE DEĞERLENDİRME	Her faaliyet sonrasında o faaliyetle ilgili değerlendirme soruları ile kendi kendinizi değerlendireceksiniz. Modül içinde ve sonunda verilen öğretici sorularla edindiğiniz bilgileri pekiştirecek, uygulama örneklerini ve testleri gerekli süre içinde tamamlayarak etkili öğrenmeyi gerçekleştireceksiniz. Sırasıyla araştırma yaparak, grup çalışmalarına katılarak ve en son aşamada alan öğretmenlerine danışarak ölçme ve değerlendirme uygulamalarını gerçekleştiriniz.

GİRİŞ

Sevgili Öğrenci,

Bilgi dünyasının vazgeçilmezi olarak yerini alan bilgisayarlar ile ortaya çıkan yeni programlar günlük hayatta insanların işlerinde kolaylıklar sağlamaktadır. Büyük ticari kuruluşlardan küçük işletmelere kadar kullanılan paket programlar, bilgisayar destekli programlar ve pek çok alanda kullanılan birçok program, programlama dilleri kullanılarak yazılır.

Programlama dilleri zaman içerisinde gelişmiş, yerini yeni gelen versiyonlara devretmiş ve eski diller geçerliliğini yitirip yeni diller ortaya çıkmıştır. Bu yeni dillerden biri de Microsoft'un birkaç yıldır geliştirmekte olduğu "kişileri, kurumları ve sistemleri birbirine bağlayan yazılımlar" olarak tanımlanan .NET teknolojisidir.

Visual Studio .NET paketinin önemli bir parçasını oluşturan Visual Basic .NET, ilk çıktığı günden bu yana dünyanın dört bir yanında geniş kullanıma ulaştı ve .NET platformunun gözdesi haline geldi. .NET ile birlikte nesne yönelimli programlama desteğine de sahip olan Visual Basic, her kademedeki programcılar için her tür uygulamayı geliştirmede kullanılabilecek bir dil niteliği halini almıştır.

Bu modülün içerisinde ise Visual Basic.NET dilinin temel taşlarını oluşturan bilgileri bulacaksınız. .Net dilinde karşılaşılan hataları, bu hataları düzeltme yöntemlerini, değişkenler üzerinde işlem yapma ve bütün programlama dillerinde programcılığın önemli bir boyutunu oluşturan matematiksel ve mantıksal operatörleri bulacaksınız.

Bu modülde verilen bilgilerin programcılık hayatınızda yeni ufuklar açmasını temenni ederek derslerinizde başarılar dilerim.

ÖĞRENME FAALİYETİ– 1

AMAÇ

Uygun ortam sağlandığında nesnelerin ortak üyeleri ile program yazabileceksiniz.

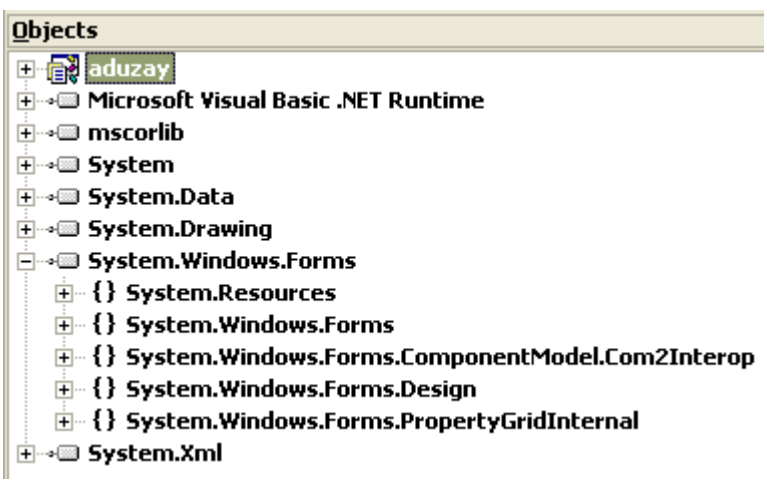
ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Farklı bir programlama dilinde kullanılan matematiksel fonksiyonları araştırınız.
- Windows işletim sisteminde fare işaretçisinin alacağı şekilleri araştırınız.

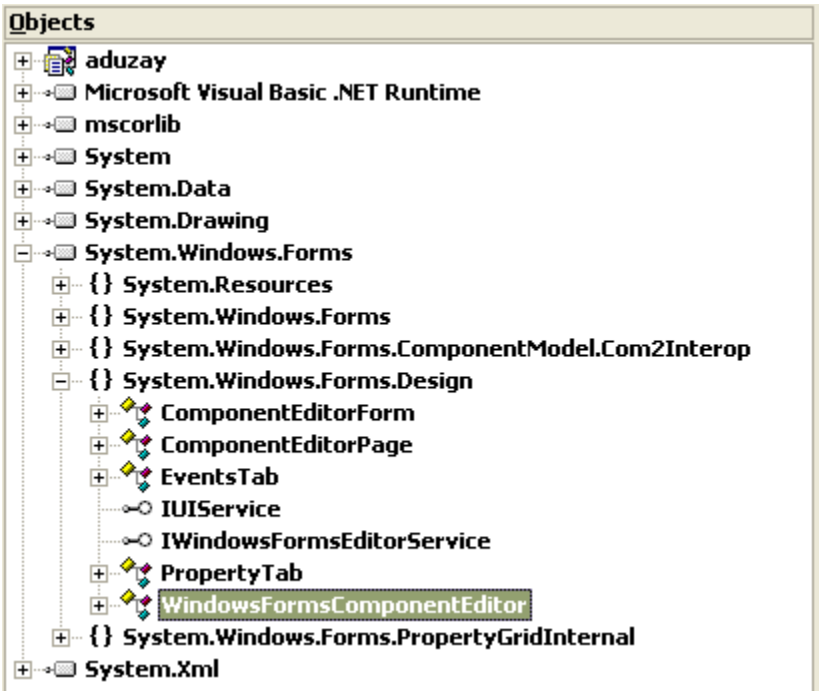
1. NESNELERİN ORTAK ÜYELERİ

Ad uzayları, "**Assembly"** nesnelerini oluşturan görev birimleridir. Assembly (Object Browser) ilk açıldığında Resim 1.1'de görülen, bağımsız çalıştırılabilen dosyalar veya DLL dosyalarına karşılık gelen kod birimleridir. Derlenmiş her program en az bir Assembly'e sahiptir. Bir Assembly'nin birçok ad uzayı vardır. Örneğin, "System.Windows.Forms" Assembly'nin ad uzaylarından birisi "System.Windows.Forms.Design"dır.



Resim 1.1: Object Browser

Ad uzaylarının bir alt düzeyi olan yani görev birimi olan nesneler ise sınıflardır. Sınıfları görebilmek için, Assembly'lerde olduğu gibi alt sınıflarını görmek istediğiniz "Ad Uzayı" nın Resim 1.2'de sol taraftaki "+" simgesine tıklamanız yeterli olacaktır.



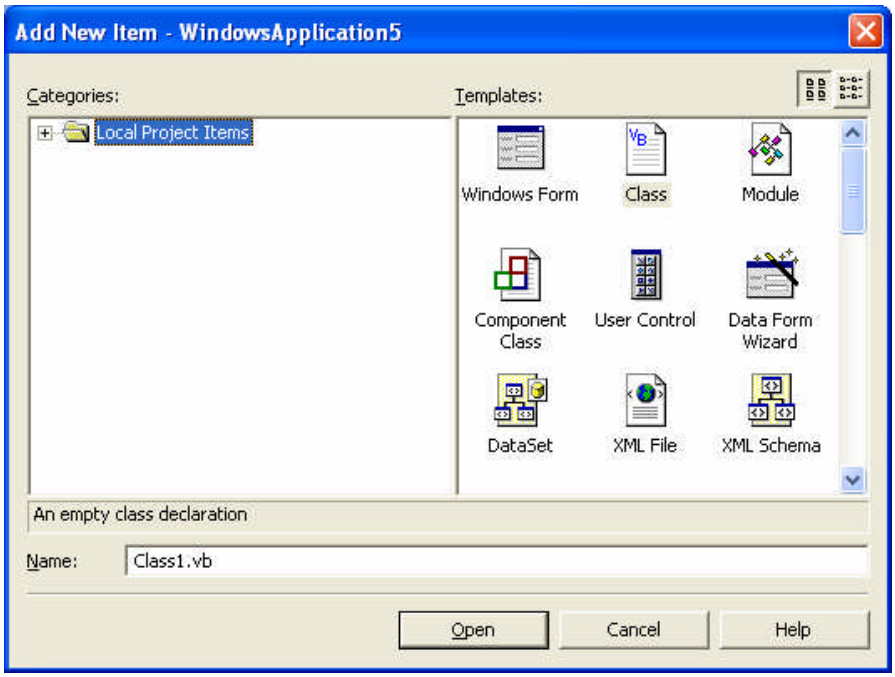
Resim 1.2: Object Browser

1.1. Sınıf Adı.Üye Adı Kullanımı

Visual Basic.NET içinde bir Class (Sınıf) tanımlamak için, aşağıdaki adımların sırasıyla yapılması gerekir.

- Project menüsünden Add New Item seçilerek gelen ekrandan Class Template seçilir ve isim verilerek Open tuşuna basılır.

- Verilen isim hem dosyanın, hem de Class'ın ismidir. Eklerden dosya ismini değiştirmediyseniz, kod penceresinde Class ismini değiştirebilirsiniz.



Resim 1.3: Add New Item penceresi

```
Public Class ders
End Class
```

- Özellikler deklare(tanımlama) edilir.
- Methodlar ve Eventlar (olaylar) deklare edilir.

Visual Basic.NET'te bir proje içinde birden fazla Class tanımlanabilir, bunun için bir limit yoktur.

Eğer bağımsız olarak Class kütüphanesi oluşturulacaksa, yeni proje oluşumu sırasında Templates kısmında Class Library seçilmesi gerekir. Burada diret "**dll**" uzantılı dosya oluşturulur.

Access Modifier: Class içinde tanımlanan değişken ve Procedure'ler için Access Modifier'ler kullanılır. Daha önceki versiyonlarda bulunan Public, Private, Friend bu versiyonda da kullanılmaktadır. Bunlara ilaveten Protected ve Protected Friend adlı iki yeni Access Modifier eklenmiştir.

Access Modifier (Erişim Şekli Tanımı)	Açıklama
Public	Her yerden ulaşılabilir.
Private	Sadece kendi içinde ulaşılabilir.
Friend	Tüm isim tanımlamaları ve kodlar aynı Assembly içindedir. Aynı Assembly içinde her yerden ulaşılabilir.
Protected	Kendi içinde ulaşılabilir ve diğer Class'lar bu Class yapısından miras alarak (inheriting) oluşmuş ise orada da kullanılabilir.
Protected Friend	Protected ve Friend birleşimidir.

Tablo 1.1: Access Modifier seçenekleri

Methodların Deklarasyonu : Visual Basic 6.0'da kullanılan method deklarasyon yöntemi, aynen burada da kullanılır.

Public Class ders1
Public Sub deneme (ByVal x As Integer)
….
End Sub
Public Function getir ( ) As Integer
….
End Function
End Class

Özelliklerin Deklarasyonu :

[Default │ReadOnly │WriteOnly │ Property değişkenadı ([Parametre listesi] [As veritipi]
Get
….
End Get
Set (ByVal value As veritipi)
…
End Set
End Property

Örnek : Yeni bir proje oluşturunuz ve Add New Item seçeneğini kullanarak içine bir Class ekleyelim.

- Class içine aşağıdaki kodu yazınız.

```
Public Class Class1
    Public Sub deneme(ByVal a As Integer, ByVal b As Integer,
Optional ByVal c As Integer = 0)
        c = a + b
        MsgBox(c)
```

```
    End Sub
End Class
```

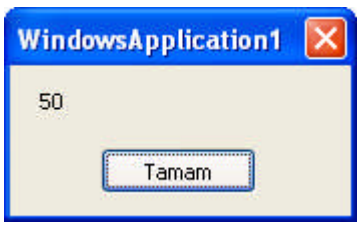
- Form üzerine koyacağımız Button içine aşağıdaki kodu yazalım. Dikkat edilirse modülden farklı olarak, New ifadesi ile Class'ı çağırarak bir değişkene atıyoruz ve ardından bu harfi yazıp noktaya basınca, Class içinde tanımlı olan yapılar karşımıza geliyor.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Button1.Click
        Dim z As New Class1()
        z.deneme(20, 30)
```

- Bu Class dosyası istenirse başka bir proje içince Add Existing Item seçeneği kullanılarak eklenebilir ve o projede de kullanılabilir.

Not: Modüle ve Class arasında farklardan birisi de Client kulanımındadır. Modüle direkt olarak çağrılır. Ancak Class, New ile tanımlanır.

- Programı çalıştırıp Button1'e tıkladığımız zaman girilen sayıların toplamını alan "deneme" Procedure'u çalışarak, bir mesaj kutusu şeklinde karşımıza gelecektir.



Resim 1.4: Class kullanımı

1.2. Sınıfların Ortak Metotları

- **Overloading Methods**

Aynı isimli fakat farklı parametreli methodların kullanılmasına imkan veren güçlü bir özelliktir. Overloads, **Key**(anahtar) ile kullanılabilir. Eğer bu key yazılmaz ise derleyici aynı isime sahip tüm methodları kontrol eder. **Inherited Class** kullanılmış ise muhakkak Overloads keyin kullanılmasını zorunlu kılar.

- **Constructors**

Visual Basic 6.0'da Initialization kodu Class'ın Class_Initialize Event'ına yazılır. Visual Basic.NET içinde bu kodların yazıldığı yere **Constructors** adı verilir. Sub New Constructor ile Class_Initialize Event'ının yerini aldığında oluşan farklılıklar şunlardır.

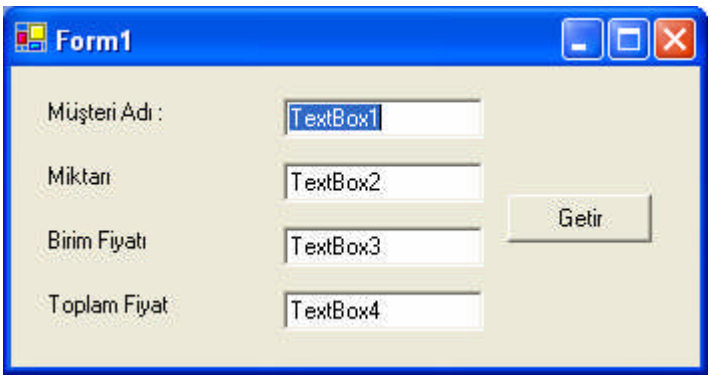
Sub New bloğu; Kod Class içindeki tüm kodlardan önce çalışır. Form'un Load Event'ına benzetebiliriz. Bir obje oluşturulduğunda, sadece bir kez çalışacaktır. Diğer bir Constructor içindeki kodun ilk satırından, aynı veya farklı bir Class içinden MyBase Key'i kullanılarak çağrılabilir.

Örnek: Yeni bir Windows uygulaması oluşturup, içine Add New Item seçeneğini kullanarak "Zeytin" isimli bir Class ekleyelim.

Zeytin isimli Class içine aşağıdaki kodları yazınız.

```
Public Class Class1
    Public Class zeytin
        Private ad As String
        Private kg As Integer
        Private fiyat As Integer
        Public Property gad() As String
            Get
                Return ad
            End Get
            Set(ByVal Value As String)
                ad = value
            End Set
        End Property
        Public Property gkg() As Integer
            Get
                Return kg
            End Get
            Set(ByVal Value As Integer)
                kg = value
            End Set
        End Property
        Public Property gfiyat() As Integer
            Get
                Return fiyat
            End Get
            Set(ByVal Value As Integer)
                fiyat = value
            End Set
        End Property
    End Class
End Class
```

Form'umuzu şekildeki gibi tasarlayalım.



Resim 1.5: Form veri giriş ekranı

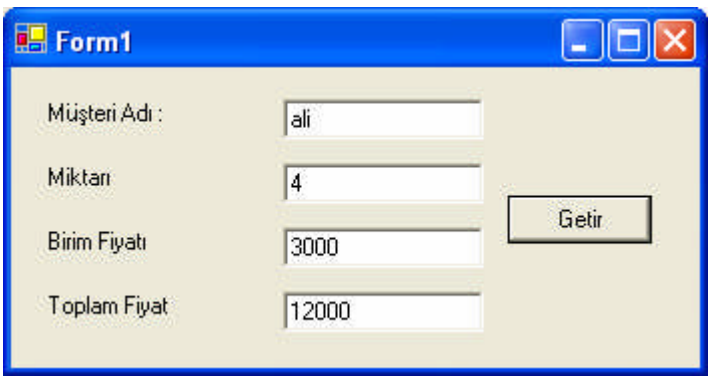
Button1_Click Event'ına aşağıdaki kodu yazalım.

```
Dim b As New zeytin()
        TextBox1.Text = b.gad
        TextBox2.Text = b.gkg
        TextBox3.Text = b.gfiyat
        TextBox4.Text = b.gkg * b.gfiyat
```

Programı çalıştırınız. Sonuca hiçbir değer atanmayacak, neden?

Dördüncü adımdaki kodu aşağıdaki gibi değiştirdiğinizde, TextBoxlara değer atanacaktır.

```
Dim b As New Class1.zeytin()
        b.gad = "ali"
        b.gkg = 4
        b.gfiyat = 3000
        TextBox1.Text = b.gad
        TextBox2.Text = b.gkg
        TextBox3.Text = b.gfiyat
        TextBox4.Text = b.gkg * b.gfiyat
```



Resim 1.6: Form veri giriş ekranı

gad ( ) Property kısmını aşağıdaki şekilde değiştirelim.

```
Public ReadOnly Property gad() As String
        Get
            Return ad
        End Get
    End Property
```

Programı çalıştırdığımızda “getir” Button’daki kod içinde hata oluşturacaktır. Bunun sebebi: ReadOnly Keyi kullanan bir Property için, başka bir yerden değer atanamaz olmasıdır. Ancak Class içinde sabit bir değer verilebilir.

gkg ( ) Property kısmını aşağıdaki şekilde değiştirelim.

```
Public WriteOnly Property gkg() As Integer
        Set(ByVal Value As Integer)
            Kg = value
        End Set
    End Property
```

Programı çalıştırdığımızda “getir” butonundaki kod içinde hata oluşacaktır. Bunun sebebi ise WriteOnly keyi kullanan bir Property’nin, bir değişkene atanamaz olmasıdır. Ancak Class bünyesinde bir işlemiçinde kullanılabilir.

Bunu test etmek için “getir” Buttonun kodunu aşağıdaki şekilde yazalım.

```
Dim b As New Class1.zeytin()
        TextBox1.Text = b.gad
        TextBox3.Text = b.gfiyat
```

Daha sona “zeytin.vb” içine aşağıdaki kodu ilave edelim ve programı çalıştıralım. Bu bir constructor’dır. Yani, tüm Class içindeki kodlardan önce çalışır. Bir Class’ı New ile çağırmanız bu yapının devreye girmesi için yeterlidir.

```
Public Sub New()
        ad = "Ahmet can"
        kg = 3
        fiyat = 5000 * kg
    End Sub
```

Zeytin Class’ı içindeki kodu ikinci adımdaki hale, “getir” butonunun içindeki kodu dördüncü adımlardaki hale getirelim.

Zeytin Class’ı içine aşağıdaki kodu ilave edelim. Programı çalıştıralım.

```
Public Sub New()
        ad = "Ahmet can"
```

```
        kg = 3
        fiyat = 5000 * kg
    End Sub
```

Program çalıştığında Text içindeki değerlerin, Button içinde tanımlanan değerler olduğunu gördük. Şimdi button içindeki kodu aşağıdaki şekilde getirip programı tekrar çalıştırdığımızda, Sub New'de tanımlanan değerlerin TextBox'lara atandığını göreceğiz.

```
Dim b As New zeytin()
        TextBox1.Text = b.gad
        TextBox2.Text = b.gkg
        TextBox3.Text = b.gfiyat
        TextBox4.Text = b.gkg * b.gfiyat
```

Zeytin Class'ı içine aşağıdaki kodu ilave edelim. Programı çalıştıralım.

```
Public Sub New(ByVal a As String, ByVal b As String, ByVal c As
Integer)
        ad = a
        kg = b
        fiyat = c
    End Sub
```

Hiçbir değişiklik olmadığını izledikten sonra "getir" Button'unun içindeki kodu aşağıdaki gibi değiştirelim ve programı çalıştıralım.

```
Dim b As New Zeytin("Dinçer GÜNDOĞDU", 23, 6000)
TextBox1.Text = b.gad
TextBox2.Text = b.gkg
TextBox3.Text = b.gfiyat
TextBox4.Text = b.gkg * b.gfiyat
```

Bu sefer TextBoxlara atanan değerlerin, ikinci Sub New'den geldiğini izleyelim.

Son olarak "zeytin" Class'ı aşağıdaki kodu ilave edelim ve programı çalıştıralım.

```
Protected Overrides Sub Finalize()
        MsgBox("güle güle")
    End Sub
```

Programı kapattığımız zaman yukarıdaki kod devreye girerek, "güle güle" mesajının karşımıza çıkmasını sağlar.

Instantiating – Initializing

Daha önceki versiyonlarda Dim a as New prestige ile başlayan tanımlamalar, Visual Basic.NET içinde bazı değişikliklere uğramıştır. Bunlar;

Tanımlama ve hafızaya yüklemenin, ayrı ayrı zamanlarada gerçekleştirilmesi.

Dim c1 as Prestige Deklare edilmesi

C1 =New Prestige ( ) Instantiate

Bu yapıda ilk önce tanımla yapılıyor, daha sonra ise program yapısının herhangi bir anında değişken hafızaya yükleniyor.

Default Constructor kullanılarak, tek bir satırda yapılması.

Dim c1 as Prestige = New Prestige ( )

Burada tanımlama ve hafızaya yükleme işlemi aynı anda yapılıyor.

Default Constructor kullanılara, tek bir satırda yapılması.

Dim c1 as New Prestige ( )

Burada tanımlama ve hafızaya yükleme işlemi aynı anda yapılıyor.

Alternatif Constructor kullanılması;

Dim c1 as New Prestige (5)

Dim c2 as New Prestige = New Prestige (5)

- **Garabage Collection**

Önceki sürümlerde referanslar, objelere Nothing Key'ini Set edince silinirdi. Bu etkili bir uygulama idi ancak bazen ilişkili referanslar varsa bunlar yok edilmeden kalırlardı.

Visual Studio.NET içinde Nothing Key'i kullanıldığında, garabage collection aktif duruma gelir ve ilişkili referansları da tarayarak, bulur ve yok eder. Hafıza için önemli bir rahatlama özelliğidir. Devreye girdiğinde Finalize methodu'nu çalıştırır.

- **Dispose**

Birçok Visual Basic.NET objesi Dispose yöntemini kullanarak, kaynakları temizler. Client programları direkt olarak objelerin kendi özelliğinde bulunan, Dispose methodu'nu kullanabilir. Eğer Client; Garabage Collection meydana gelmeden önce Dispose Methodu'nu çağırmaz ve ise Finalize Method Dispose method'unu çağırır. Onun için güvenli bir şekilde Dispose methodu kullanılabilir. Dispose Method'u kullanıldığında Garabage Collection Finalize methodu'nu Execute yapmaya ihtiyaç duymaz. İstenirse GC Obje method'u üzerinde Suppress finalize çağırılarak, Execute olayı disable ettirilebilir. Finalize Execute yapılacaksa GC.Collect method'u uygulanır.

- **Public Interface**

Bu sayede Class içindeki yapıları ayrı ayrı tutabilir ve gerektiği zaman Class içindeki parçalardan sadece istediklerimizi değiştirebiliriz. Katman şeklinde program yazmak için, tercih edilmesi gereken bir yapıdır.

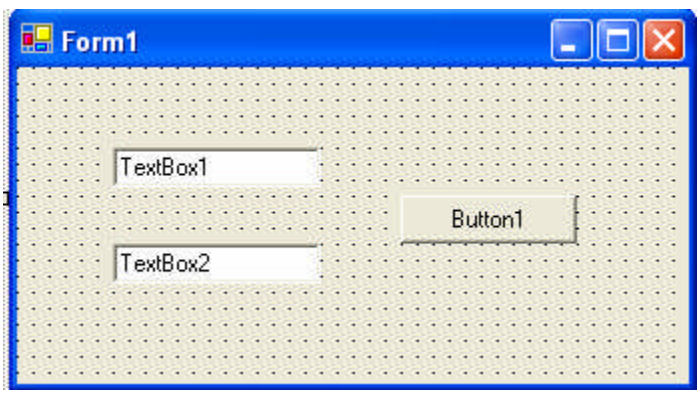
1.3. “ Math ” Sınıfı

Visual Basic.NET ortamında matematiksel fonksiyonlar Math Class’ında bulunur. Bu Class’ı kullanarak, istediğimiz matematiksel fonksiyonu çalıştırabiliriz. Aşağıda tüm fonksiyonlar tek tek incelenmiştir. Parantez içerisine ondalık sayı girilen fonksiyonlarda ondalıklı sayı yerine, tam sayı da kullanılabilir.

Math.Abs (Ondalıklı Sayı)

Mutlak değer hesabı yapan bir fonksiyondur. Bu fonksiyon sayesinde istediğiniz bir sayının pozitif değerini döndürebiliriz. Fonksiyonda dikkat edilecek olan nokta pozitif sayılar için aynı değer, negatif sayılar için pozitife çevirerek döndürmesidir.

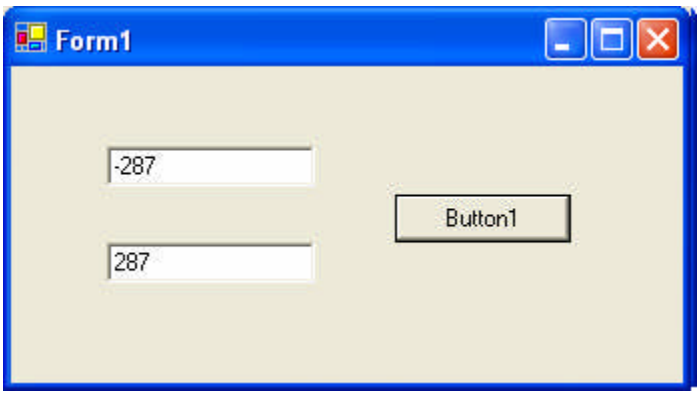
Örnek : Form üzerine iki adet TextBox ve bir adet button koyarak, Button1_click Event’ına aşağıdaki kodu yazalım.



Resim 1.7: Math.Abs fonksiyonu kullanımı

```
TextBox2.Text = Math.Abs(Convert.ToDecimal(TextBox1.Text))
```

Programı çalıştırıp, TextBox1 içine “- 287” yazarak Button1’e tıklayalım. Bunun sonucunda mutlak değeri olan “287” değerinin TextBox2’ye yazıldığını görürüz.



Resim 1.8: Math.Abs fonksiyonu kullanımı

- **Math.Ceiling (Ondalık Sayı)**

Fonksiyonun, geriye tamsayı tipi döndürür. Parametre olarak girilen değişkenin veya ondalıklı sayının büyüğe yuvarlatılarak geriye döndürülmesini sağlar. Yani sonuç "10.2" olsa bile "11" değerini döndürür. Sayının negatif olması bir şey değiştirmez ve yine büyüğe yuvarlar.

```
TextBox2.Text = Math.Ceiling(TextBox1.Text)
```

- **Math.Exp (Ondalık Sayı)**

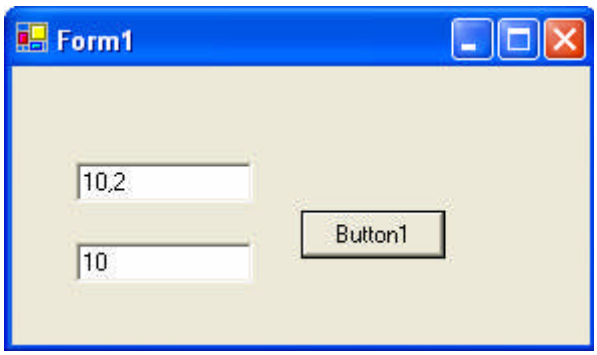
Parametre olarak girilen değişkeni veya sayıyı e'nin kuvveti olarak hesaplar. Logaritmik işlemlerde kullanılan bir fonksiyondur. Fonksiyonun geriye döndürdüğü değerin tipi doubledir.

```
TextBox2.Text = Math.Exp(TextBox1.Text)
```

- **Math.Floor (Ondalık Sayı)**

Ceiling fonksiyonu, girilen ondalı sayıyı büyük tamsayıya yuvarlıyordu. Floor fonksiyonu da küçük tamsayıya yuvarlar. Fonksiyonun geriye döndürdüğü değer tamsayı tipidir.

```
TextBox2.Text = Math.Floor(TextBox1.Text)
```

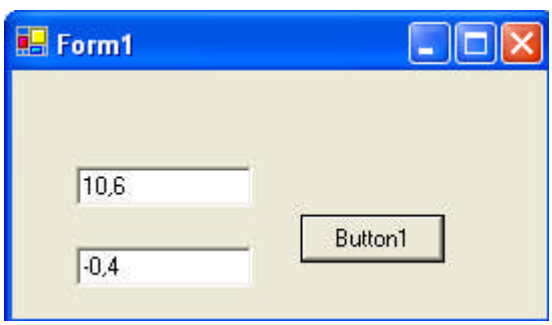


Resim 1.9: Math.Floor fonksiyonu kullanımı

- **Math.IEEERemainder (Ondalık Sayı, Ondalık Sayı)**

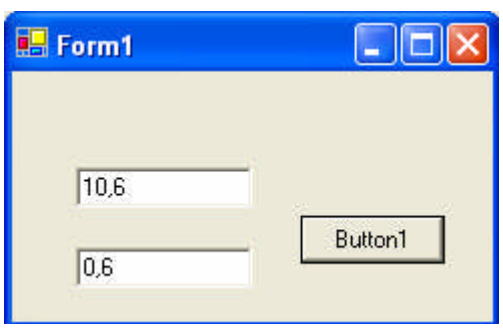
Parametre olarak girilen Reel sayının ondalıklı kısmını veya tamsayı olabilmesi için gerekli olan ondalıklı sayıyı döndürür. Hangisini döndüreceği ikinci girilen parametre ile belirlenir. İkinci parametre olarak "2" kullanılırsa ondalıklı kısmı alır. "1" kullanılırsa tamsayıya tamamlamak için gerekli olan ondalık değerini gösterir.

```
TextBox2.Text = Math.IEEERemainder(TextBox1.Text, 1)
```



Resim 1.10: Math.IEEERemainder fonksiyonu kullanımı

```
TextBox2.Text = Math.IEEERemainder(TextBox1.Text, 2)
```

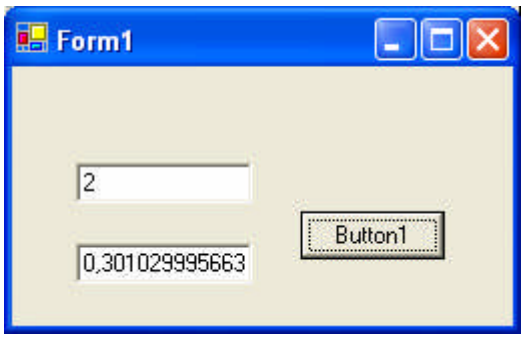


Resim 1.11: Math.IEEERemainder fonksiyonu kullanımı

- **Math.Log (Ondalık sayı, Toplam)**

Parametre olarak girilen double sayının, ikinci parametrede verilen tabana göre logaritmasını alır.

```
TextBox2.Text = Math.Log(TextBox1.Text, 10)
```



Resim 1.12: Math.Log fonksiyonu kullanımı

- **Math.Log10(Ondalıklı Sayı)**

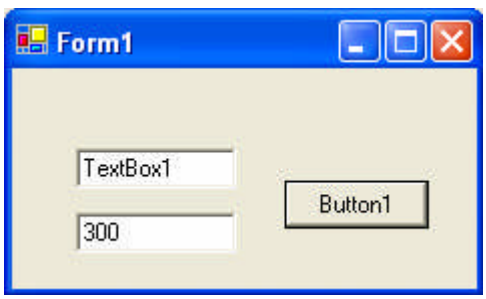
Parametre olarak girilen sayının 10 tabanına göre logaritmasını alır. Bu fonksiyon ile 10 tabanı dışında başka tabanlarda logaritma alınamaz. Öyle bir durumda log fonksiyonunu kullanmak gerekir.

```
TextBox2.Text = Math.Log10(TextBox1.Text)
```

- **Math.Max (Ondalık sayı, Ondalık sayı)**

Parametre olarak girilen değişkenlerden veya sayılardan büyüğünü döndürür.

```
TextBox2.Text = Math.Max(300, 200)
```

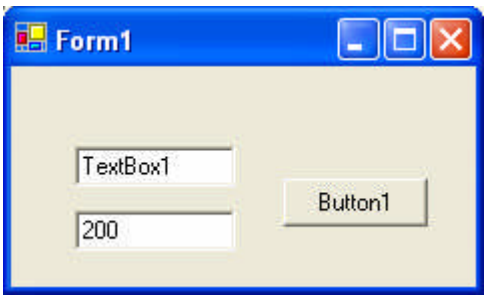


Resim 1.13: Math.Max fonksiyonu kullanımı

- **Math.Min (Ondalık sayı, Ondalık sayı)**

Parametre olarak girilen değişkenlerden veya sayılardan küçüğünü döndürür.

```
TextBox2.Text = Math.Max(300, 200)
```



Resim 1.14: Math.Min fonksiyonu kullanımı

- **Math.PI**

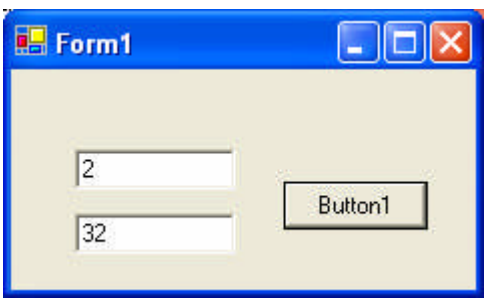
Matematiksel π (Pi) sayısını ifade eder. Bu bir fonksiyon değildir. Parametre içermez. Değeri 22/7'dir.

```
TextBox2.Text = Math.PI
```

- **Math.Pow (Ondalık sayı, Ondalık sayı)**

Üs işlemi için kullanılan bir fonksiyondur. İki parametre içerir. Birinci parametre üs'sü alınacak sayıyı, ikinci parametre ise kuvvet'i ifade eder. Tüm sayılar için kullanılabilir.

```
TextBox2.Text = Math.Pow(TextBox1.Text, 5)
```



Resim 1.15: Math.Pow fonksiyonu kullanımı

- **Math.Round (Ondalık sayı, Hassasiyeti)**

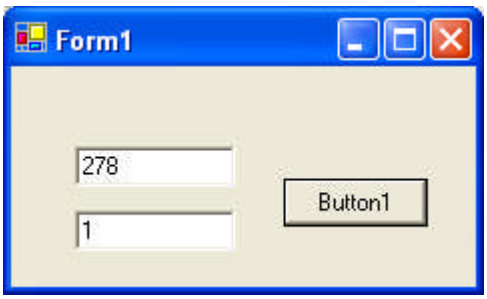
Ondalık kısmından kaç rakamın gösterileceğini belirleyen bir fonksyondur. İlk parametresi uygulanacak olan sayı, ikinci parametresi ise ondalıklı kısımdan gösterilecek olan rakam sayısıdır. Ondalıklı kısımda geri kalan rakamlar yuvarlatılır. Sayısal işlemlerde çok kullanılan bir fonksiyondur. Aşağıdaki yazılım ondalıklı kısım ne olursa olsun 3 hassasiyete yuvarlar. Bu yuvarlama işlemini yaparken dördüncü ondalıklı sayı 5'ten küçük ise üçüncü rakam aynı kalır, aksi takdirde bir üste yuvarlanır.

```
TextBox2.Text = Math.Round(Convert.ToDouble(TextBox1.Text), 3)
```

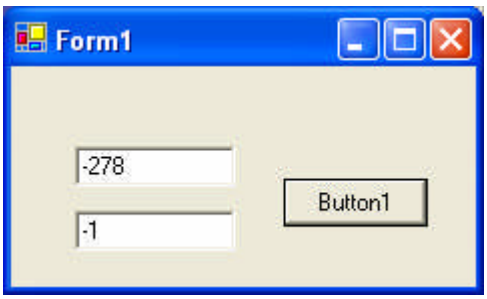
- **Math.Sign (Ondalıklı sayı)**

Parametre olarak belirten sayının pozitif, negatif veya sıfır olup olmadığını gösteren fonksiyondur. Eğer sayı pozitif ise geriye "1" değerini negatif ise "-1" değerini, sıfır ise "0" değerini döndürür.

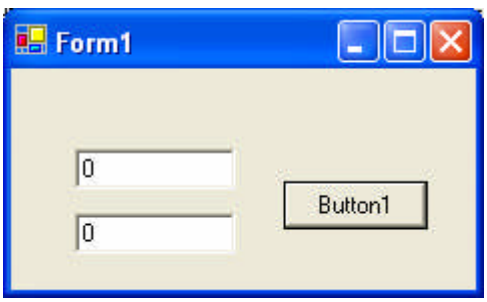
```
TextBox2.Text = Math.Sign(Convert.ToDouble(TextBox1.Text))
```



Resim 1.16: Math.Sign fonksiyonu kullanımı



Resim 1.17: Math.Sign fonksiyonu kullanımı

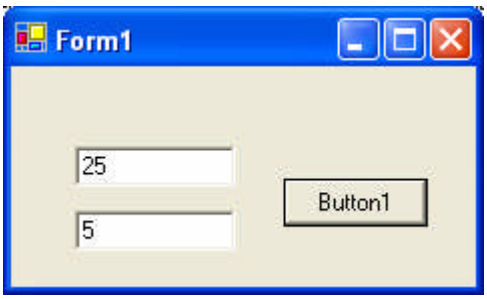


Resim 1.18: Math.Sign fonksiyonu kullanımı

- **Math.Sqrt (Ondalıklı sayı)**

Parametre ile belirtilen sayının karekökünü geriye döndürür. Sayının reel veya tam olması önemli değildir. İstenirse sayının karekökü Pow fonksiyonu ile de hesaplanabilir.

```
TextBox2.Text = Math.Sqrt(TextBox1.Text)
```



Resim 1.19: Math.Sqrt fonksiyonu kullanımı

1.4. " Cursor " Sınıfı

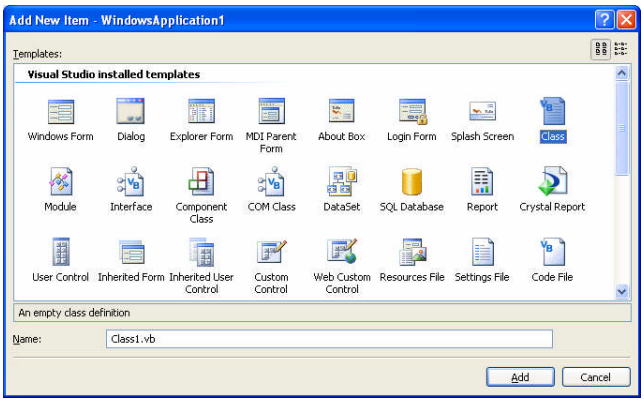
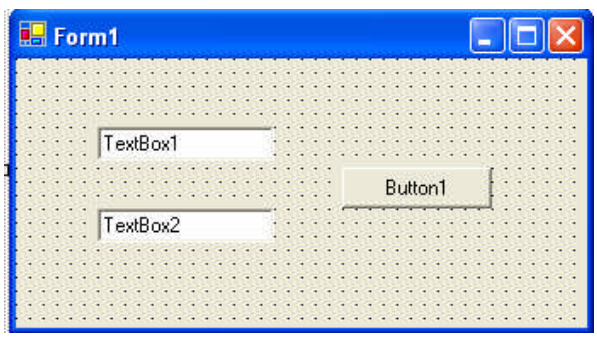
Kursör TextBox'ın üzerine geldiğine, Mouse ikonunun alacağı şekil belirlenir. Cursors parametresi ile kullanılır.

Kullanımı:
TextBox1.Cursor = Cursors.AppStarting

Cursors Parametresinin Alabileceği Değerler
Cursors.AppStarting
Cursors.Arrow
Cursors.Cross
Cursors.Default
Cursors.Hand
Cursors.Help
Cursors.Hsplit
Cursors.Ibeam
Cursors.No
Cursors.NoMove2D
Cursors.NoMoveHoriz
Cursors.NoMoveVert
Cursors.PanEast
Cursors.PanNE
Cursors.PanNorth
Cursors.PanNW
Cursors.PanSE
Cursors.PanSouth
Cursors.PanSW
Cursors.PanWest

Tablo 1.2: Cursors sınıfı parametreleri

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
Projenize yeni bir Class ekleyiniz. 	Ekleyeceğiniz Class ismini kendiniz belirleyiniz.
Form üzerine iki tane TextBox ve bir tane Buton ekleyiniz. TextBox1'e girdiğiniz sayının karekökünü TextBox2'de görüntüleyecek şekilde Buton1 içerisine gerekli kod satırlarını yazınız. 	
TextBox1 kutucuğu üzerine geldiğinizde kürsörünüzün  işaretini alması için gerekli kod satırını yazınız.	

ÖLÇME VE DEĞERLENDİRME

A- Objektif Testler (Ölçme Soruları)

Aşağıdaki sorulardan; ilk 5 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. ( ) Visual Basic.NET'te Class eklemek için Project menüsünden Add New Item seçilir.

2. ( ) Access Modifier içerisinde Public deyimi kullanılabilir.

3. ( ) Math.Pow fonksiyonu bir sayının logaritmasını almak için kullanılır.

4. ( ) Visual Basic.NET'te bir proje içinde birden fazla Class tanımlanabilir, bunun için bir limit yoktur.

5. ( ) Overloading aynı isimli fakat farkı parametreli methodların kullanılmasına imkan veren güçlü bir özelliktir.

6. Access Modifier içerisinde aşağıdaki deyimlerden hangisi kullanılmaz?
A) Public
B) Private
C) Class
D) Friend

7. Class yapıları aşağıdakilerden hangisi ile tanımlanır?
A) New
B) Cursor
C) Math
D) System

8. Bir değişken ya da sayının üs ifadesini almak için hangi fonksiyon kullanılır?
A) Math.Abs
B) Math.Max
C) Math.Floor
D) Math.Pow

Değerlendirme

Sorulara verdiğiniz cevaplar ile cevap anahtarını karşılaştırınız, cevaplarınız doğru ise bir sonraki öğrenme faaliyetine geçiniz. Yanlış cevap verdiyseniz öğrenme faaliyetinin ilgili bölümüne dönerek konuyu tekrar ediniz.

ÖĞRENME FAALİYETİ– 2

AMAÇ

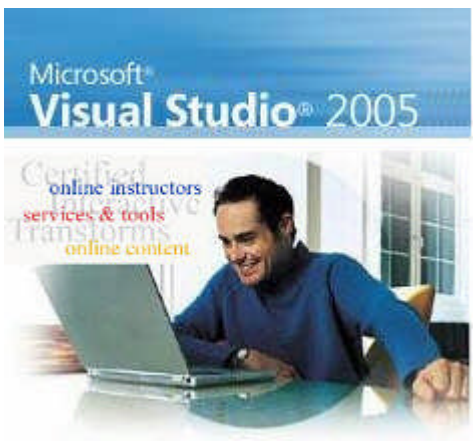
Uygun ortam sağlandığında, basit veri giriş ve çıkış diyalog kutuları ile çalışabileceksiniz.

ARAŞTIRMA

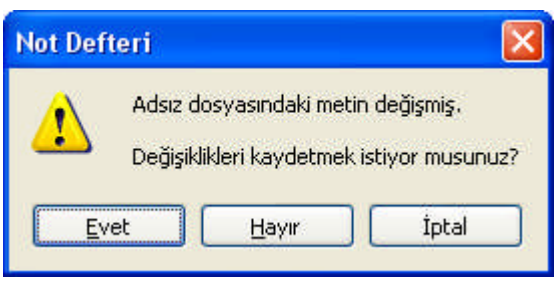
Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Diğer programlama dillerinde kullanılan mesaj komutlarını araştırınız.
- Günlük hayatımızda yer alan bankamatiklerde kullanılan programlardaki diyalog kutularını inceleyiniz.

2. DİYALOG KUTULARI



Windows ortamında çalıştığımızda hemen hemen her programda mesaj pencereleri ile karşılaşırız. Örneğin Windows'un not defteri programını açıp bir hatırlatma mesajı yazın ve programı kapatın. Mesaj penceresi sizi değişiklik hakkında uyaracaktır (Resim 2.1).



Resim 2.1: Program penceresini kapattıktan sonra gelen mesaj penceresi

Burada eğer evet düğmesine tıkarlarsanız kaydet diyalog penceresi açılacak ve dosyayı kaydetmek için sizden bir isim girmeniz istenecektir. Eğer hayır düğmesini tıklarsanız yapılan değişiklikler dikkate alınmadan programdan çıkılacaktır. İptal düğmesine ya da ESC tuşuna basarsanız mesaj penceresi kapatılacak ve programdan çıkmayacaksınız.

Visal BASIC.net ortamında bu tür mesaj pencerelerini kolaylıkla geliştirebileceksiniz.

2.1. "MessageBox" Sınıfı

Diğer bütün görsel programlarda olduğu gibi gerektiği zaman kullanıcıyı uyaran, kullanıcıdan onay alan, kullanıcıyı bilgilendiren daha genel anlamda kişiyi yönlendiren mesaj pencereleri oluşturulabilmektedir.

Kullanımı :

MsgBox ("Çıkacak Mesaj Bilgisi" , "Mesaj Kutusunun Özellikleri" , "Başlık Bilgisi")

Örnek : Aşağıdaki programı beraber inceleyelim.

```
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
        MsgBox("Mesaj penceresi oluşturmak ister misin?",
MsgBoxStyle.YesNo, "Visual Studio")
    End Sub
End Class

MsgBox("Mesaj penceresi oluşturmak ister misin?",
MsgBoxStyle.YesNo, "Visual Studio")
```

satırını formunuzun kod editörüne yazıp çalıştıracak olursanız ilk yazılan bilginin mesaj kutusunun ortasına yazıldığını, ikinci yazılan bilginin Evet/Hayır butonlarını çıkardığını ve son bölümün ise başlık kısmını oluşturduğunu göreceksiniz.

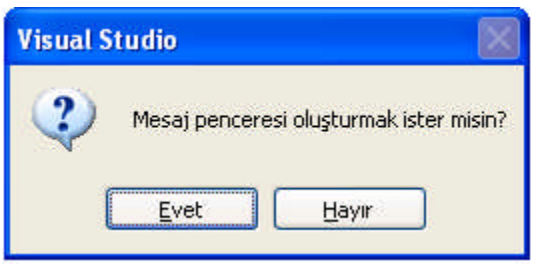
Resim 2.2: Mesaj penceresi oluşturma

Not: Burada birinci ve üçüncü bölümler aynı özelliği taşımasına rağmen, ikinci kısımlarda çeşitli değişiklikler yapılabilir.

Aynı örneği aşağıdaki gibi değiştirelim.

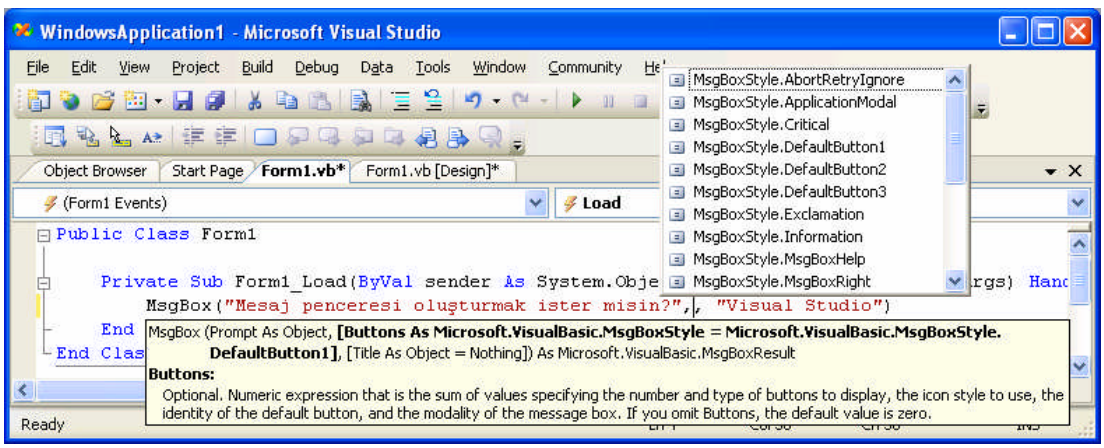
```
MsgBox("Mesaj penceresi oluşturmak ister misin?",
MsgBoxStyle.Question + MsgBoxStyle.YesNo, "Visual Studio")
```

komut satırlarını yazıp çalıştırdıkan sonra ekrana gelen mesaj kutusu görüntüsünde  simgesinin bulunduğunu göreceksiniz. Bu işlemi MsgBox komutununun ikinci parametresine + işareti (`MsgBoxStyle.Question + MsgBoxStyle.YesNo`) konularak yapılabilir.



Resim 2.3: Mesaj kutusuna Information simgesi ekleme

MsgBoxStyle özelliği ilk parametre yazıldıktan sonra virgül konulunca çıkan listeden seçim yapılarak da kazandırılabilir.



Resim 2.4: MsgBoxStyle özelliği penceresi

 **Not**: Altta verilen özellikleri ise sizler uygulayıp mesaj kutusu görüntülerini inceleyin.

MsgBoxStyle Listesi	Açıklama
OKOnly (0)	Sadece OK butonunu çıkartır.
Critical (16)	Çarpı (X) işareti çıkarır.
Exclamation (48)	Ünlem (!) işareti çıkarır.
Information (64)	Bilgi anlamına gelen şekli çıkarır.
MsgBoxHelp (16384)	OK ve Help butonlarının çıkartır.
OKCancel (1)	OK ve Cancel butonlarının çıkartır.
RetryCancel	Retry ve Cancel butonlarını çıkartır.
AbortRetryIgnore (2)	Abort, Retry ve Ignore butonlarını çıkartır.
YesNo (4)	Yes ve No butonlarını Çıkartır.
YesNoCancel (3)	Yes, No, ve Cancel butonlarıı Çıkartır.
SystemModal	Modal olarak çalışır. Yani bu ekran her şeyin üstünde bulunur.
MsgBoxRight (524288)	Yazılar sağa yaslı olarak yazılır. Özellikle arap harflerinin kullanıldığı programlarda tercih edilebilir.
MsgBoxRtlReading (1048576)	Birden fazla button olduğu durumlarda hangi buttonun aktif olacağını belirtir.
DefaultButton1 (0) DefaultButton2 (256) DefaultButton3 (512)	Birden fazla button olduğu durumlarda hangi buttonun aktif olacağını belirtir.

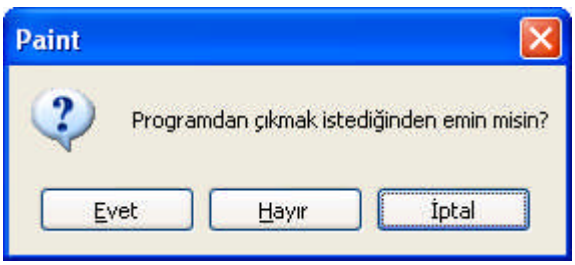
Tablo 2.1: MsgBoxStyle listesi ve özellikleri

Not: Ayrıca direkt numaraların toplamı kullanılarak da uygulanacak biçim belirlenebilir.

Örnek : Üzerinde Yes- No- Cancel butonları ile beraber işaretinin bulunduğu ve üçüncü butonun aktif olduğu bir mesaj iletisi çıkartmak istersek kod satırını şu şekilde düzenlemeliyiz.

```
MsgBox("Programdan çıkmak istediğinden emin misin?", 547,
"Paint")
```

komut satırını yazıp programı çalıştırdığımızda aşağıdaki ekran görüntüsü karşımıza gelecektir.



Resim 2.5: MsgBoxStyle özelliğini değer vererek kullanma

Not: Burada biçim kısmında kullanılan **547** rakamı, bu üç özelliği ifade eden 3+32+512 sayılarının toplamından oluşmuştur.

2.2. Diyalog Kutuları (Modal Form, ShowDialog Komutu)

Birden fazla form ile çalıştığımız zaman, bir form içinden diğer formu iki şekilde açabiliriz.

Burada kullanılan method'lar **Show** veya **ShowDialog** olacaktır. Show ile açılan ikinci form Modeless Form'dur. Bu formun özelliği öndeki formu kapatmadan, arkadaki formu kapatabiliriz. ShowDialog kullandığımız zaman, açılan formu kapatmadan, arkadaki formaları kapatamayız.

Moddless Form açmak için;

Kullanımı :

Dim a As New Form2 ( )

a.Show ( )

Modal Form açmak için;

Kullanımı :

Dim a As New Form2 ( )

a.ShowDialog ( )

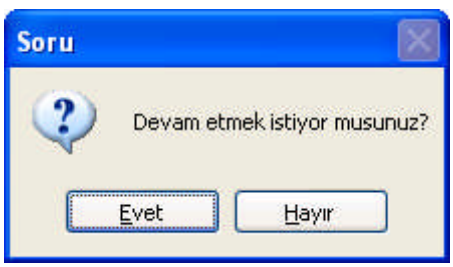
2.3. “DialogResult Seçenekleri:“ OK, Cancel, Yes, No, Abort, Retry, Ignore ve None”

Bu özellik ile mesaj kutusu üzerinde basılan tuşları kontrol edebiliriz. Bunu örnekle açıklayacak olursak;

Örnek : Üzerinde iki butonu bulunan bir mesaj iletisinin her iki butonunun da kontrolünü yapabiliriz. Bunun için aşağıdaki komut satırlarını yazalım.

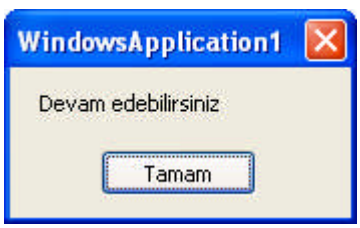
```
If MsgBox("Devam etmek istiyor musunuz?", MsgBoxStyle.YesNo +
MsgBoxStyle.Question, "Soru") = _
        MsgBoxResult.Yes Then
            MsgBox("Devam edebilirsiniz")
        Else
            MsgBox("Program kapatılıyor")
        End If
```

- Programı çalıştırdığınız zaman Resim 2.6’da verilen mesaj iletisini alacaksınız.



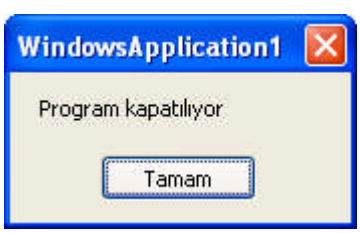
Resim 2.6: DialogResult seçenekleri

- Evet butonuna basarsak, şekildeki mesaj iletisi ile karşılaşırız.



Resim 2.7: Evet butonuna basınca gelen mesaj iletisi

- Eğer Hayır butona basarsak o zaman, ikinci mesaj iletisi ile karşılaşırız.



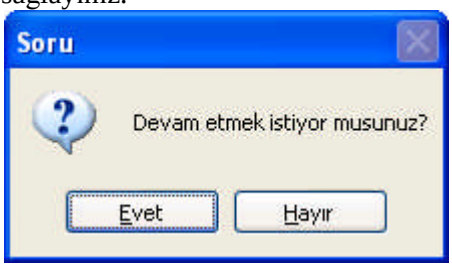
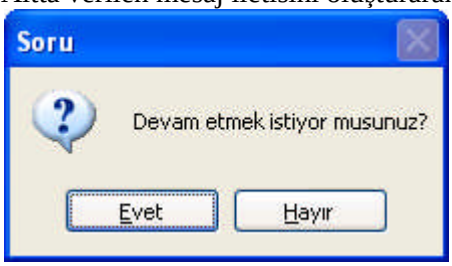
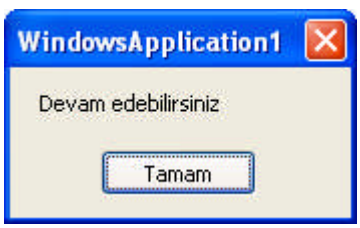
Resim 2.8: Hayır butonuna basınca gelen mesaj iletisi

Msgbox fonksiyonu ile kullanıcının hangi butona bastığını bildiren bir tamsayı değeri geri döndürülür. Bu değerler aşağıdaki tabloda verilmiştir.

Geri dönen sembolik değer	Geriye dönen sayısal değer	Açıklama
MsgBoxResult.Ok	1	Tamam düğmesi seçildi.
MsgBoxResult.Cancel	2	İptal düğmesi seçildi
MsgBoxResult.Abort	3	İşlemi durdur düğmesi seçildi.
MsgBoxResult.Retry	4	Tekrar dene düğmesi seçildi.
MsgBoxResult.Ignore	5	Göz ardı et düğmesi seçildi.
MsgBoxResult.Yes	6	Evet düğmesi seçildi.
MsgBoxResult.No	7	Hayır düğmesi seçildi.

Tablo 2.2: MsgBox fonksiyonunda geriye döndürülen değerler

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
Altta verilen mesaj iletisini oluşturunuz. 	MsgBox komutunun kullanımını inceleyiniz.
Şekildeki gibi mesaj kutusu oluşturarak üzerindeki butonların yer almasını sağlayınız. 	Information simgesine dikkat ediniz.
Altta verilen mesaj iletisini oluşturarak,  evet butonuna bastığımızda,  mesaj iletisini oluşturunuz.	

ÖLÇME VE DEĞERLENDİRME

A- Objektif Testler (Ölçme Soruları)

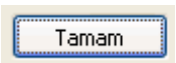
Aşağıdaki sorulardan; ilk 4 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. ( ) Visula Basic.NET'te mesaj iletileri oluşturmak için MsgBox fonksiyonu kullanılır.

2. ( ) MsgBoxStyle.YesNo komut satırı 

butonlarını çıkarmak için kullanılır.

3. ( ) Bir form üzerinden başka bir formu Show komutu yardımıyla çağırabiliriz.

4. ( ) MsgBoxResult fonksiyonunda geri dönen değer 1 ise 

butonuna basılmıştır.

5. MsgBoxStyle.Information komutu aşağıdaki simgelerden hangisini çıkarır?

A) 

B) 

C) 

D) 

6. MsgBoxResult fonksiyonunda geriye dönen değer 2 ise aşağıdaki butonlardan hangisi seçilmiştir?

A) Ok
B) Abort
C) Retry
D) Cancel

Değerlendirme

Sorulara verdiğiniz cevaplar ile cevap anahtarını karşılaştırınız, cevaplarınız doğru ise bir sonraki öğrenme faaliyetine geçiniz. Yanlış cevap verdiyseniz öğrenme faaliyetinin ilgili bölümüne dönerek konuyu tekrar ediniz.

ÖĞRENME FAALİYETİ– 3

AMAÇ

Gerekli ortam sağlandığında, sık kullanılan fonksiyon komutları ile çalışabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Diğer programlama dillerinde sık kullanılan komutların yazım ve kullanım formatlarını araştırınız.

3. SIK KULLANILAN KOMUTLAR

Visual Basic.NET ortamında da diğer dillerde olduğu gibi matematiksel, tarihsel, dizi ve string işlemleri için çok sayıda fonksiyon bulunmaktadır. Bu fonksiyonların birçoğunu kendiniz yazabileceğiniz gibi, kendi kütüphanesini kullanarak da çalıştırabilirsiniz. Zaten yeni versiyonların temel özelliği bu kütüphanelerin zenginliğidir.

3.1. “ CDec, CInt, CStr ” Fonksiyonları

3.1.1. CDec Fonksiyonu

Sayısal içerikli bir değeri Decimal tipine dönüştürür.

3.1.2. CInt Fonksiyonu

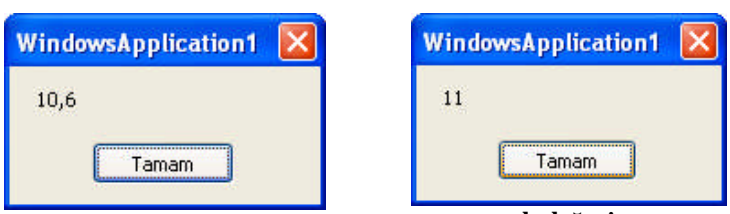
Ondalıklı bir değeri integer tipine dönüştürür. Ondalık sayıları yuvarlar.

Not: Ondalıklı kısım 5 ve altında ise aşağı, 5’in üzerinde ise yukarı yuvarlar.

Örnek :

```
Dim a, b
a = 10.6
b = CInt(a)
MsgBox(a)
MsgBox(b)
```

programı çalıştırdığını zaman sonuç şu şekilde olacaktır.



a değeri **b değeri**

Resim 3.1: CInt fonksiyonu kullanımı

3.1.3. CStr Fonksiyonu

Bir ifadeyi String tipine dönüştürür.

3.2. “ Len, Left, LTrim ” Fonksiyonları

3.2.1. Len Fonksiyonu

Metin içindeki karakter sayısını verir. Bu sayıya boşluklarda dahildir.

Kullanımı :
Len (metin)

Örnek :
```
Dim ad As String
ad = "Emrullah"
    MsgBox(Len(ad))
```

komut satırlarını yazıp çalıştırdığımız zaman;

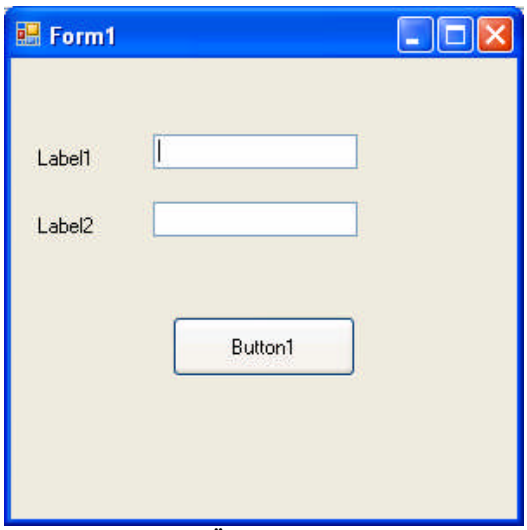
Resim 3.2: Len fonksiyonunun kullanımı

3.2.2. Left Fonksiyonu

Microsoft.visualbasic.strings sınıfın bir üyesi olup metin içinde soldan n karakteri verir.

Kullanımı :
Left (metin, n)

Örnek : Şekil 3.3 de verilen formu bilgisayaranızda tasarlayıp gerekli kontrolleri formunuza yerleştirin.

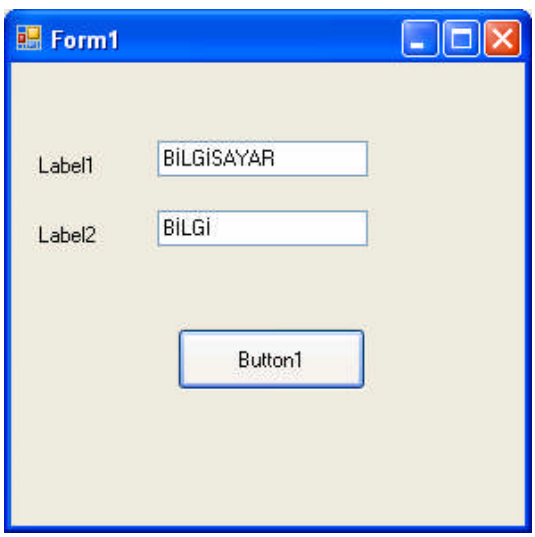


Resim 3.3: Örnek form tasarımı

Öncekli olarak TextBox1 kutucuğuna "BİLGİSAYAR" verisini girelim. Daha sonra Buton_Click fonksiyonuna,verilen komut satırını yazalım.

```
TextBox2.Text = Microsoft.VisualBasic.Left(TextBox1.Text, 5)
```

Daha sonra programımızı çalıştırıp Button1'e tıklayalım. Ekran görüntüsü şöyle olacaktır.



Resim 3.4: Left fonksiyonunun kullanımı

3.2.3. LTrim Fonksiyonu

Microsoft.visualbasic.strings sınıfının bir üyesi olup verilen metnin başındaki boşlukları kaldırır.

Kullanımı :
Ltrim(metin)

Uygulama: LTrim fonksiyonunu kullanarak kendiniz uygulama yapınız ve sonucunu öğretmeninize gösteriniz.

3.3. " DateDiff, WeekDay, WeekDayName " Fonksiyonları

Visual Basic.NET programlama dilinde tarih ve saat işlemlerini kolaylıkla yapabilmeniz için birçok fonksiyon bulunmaktadır. Bu fonksiyonları çalıştırmak için yaygın olarak kullanılan DateTime Class'ını kullanmalıyız.

3.3.1. DateDiff Fonksiyonu

Tarih fonksiyonları içerisinde en yaygın olarak kullanılan fonksiyondur. **Datediff** fonksiyonu iki zaman arasındaki farkı gösterir.

Kullanımı :
Datediff(fark_türü, tarih1, tarih2[,haftanın_günü[,yılın_haftası]])

Fark_turu → string ya da dateinterval değerlerinden birini alabilir. Farkın alınacağı zaman türünü belirler.

Zaman türü	Birinci kullanım biçimi	İkinci kullanım biçimi
Yıl	Dateinterval.Year	"yyyy"
Çeyrek	Dateinterval.Quarter	"q
Ay	Dateinterval..Month	"m"
Yılın günü	Dateinterval.Dayofyear	"y"
Gün	Dateinterval.Day	"d"
Hafta günü	Dateinterval.Weekday	"w"
Hafta	Dateinterval.Week	"ww"
Saat	Dateinterval.Hour	"h"
Dakika	Dateinterval.Minute	"n"
Saniye	Dateinterval.Second	"s"

Tablo 3.1: DateDiff fonksiyonu parametreleri

Tarih1 → Başlangıç tarihini belirler

Tarih2 → Bitiş tarihini belirler.

Haftanın_günü → Kullanılması isteğe bağlı bir parametredir. Haftanın gününü tanımlar.

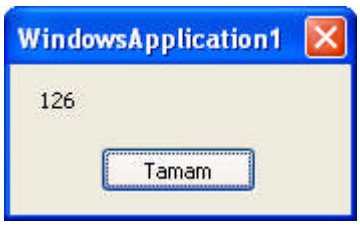
Yılın_haftası → Kullanılması isteğe bağlı bir parametredir. Yılın haftasını tanımlar.

Örnek : Verilen iki tarih arasında farkı hesaplayan programı beraber hazırlıyalım.

Not: En temel biçimde tarih2'den tarih1'i gün bazından çıkarmak için aşağıdaki yöntemi kullanırız.

```
Dim tarih1, tarih2 As Date
        tarih1 = #4/16/2006#
        tarih2 = #8/20/2006#
             MsgBox(DateDiff(DateInterval.Day, tarih1, tarih2))
```

komutunu yazıp çalıştırdıktan sonra girilen iki tarih arasındaki gün sayısı mesaj iletisi olarak ekrana gelecektir.

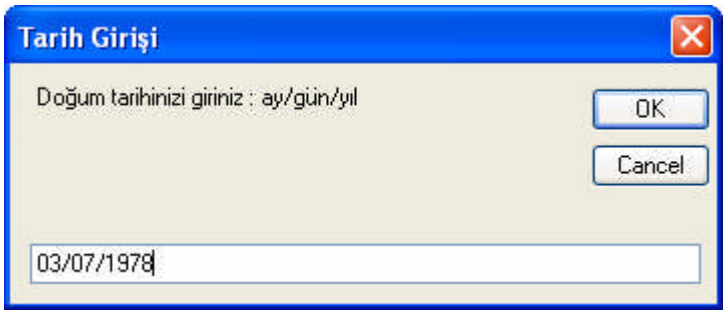


Resim 3.5: DateDiff fonksiyonunun kullanımı

Örnek : Şimdi de doğum tarihimizi girerek bu tarih ile programı yazdığımız gün arasındaki tarih farkını bumaya çalışalım.

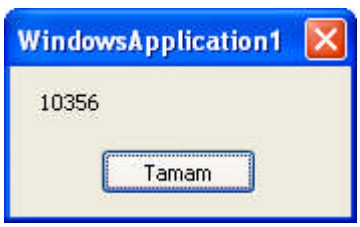
```
Dim tarih1 As Date
        Dim tarih2 As Date
        tarih1 = InputBox("doğum tarihinizi girin :
ay/gün/yıl", "tarih girişi", "")
        tarih2 = DateAndTime.Now.Date
             MsgBox(DateDiff(DateInterval.Day, tarih1, tarih2)
```

- Programımızı çalıştırdıktan sonra Resim 3.6'da yer alan tarih girişi isimli InputBox penceresi ekrana gelecektir.



Resim 3.6: Tarih girişi isimli InputBox penceresi

- Tarih girişini Ay/Gün/Yıl olarak girdikten sonra Ok butonuna tıklayalım.



Resim 3.7: DateDiff fonksiyonunun kullanımı

Not: Burada tarih formatı bilgisayarınızdan gelen formattır. Kendi bilgisayarımda Ay/Gün/Yıl olduğu için, tarih yazılımlarını bu şekilde yaptım. Sizle rde bu formatı, kendi bilgisayarınızda tanımlı olan yapıya göre yapmalısınız.

3.3.2. Weekday Fonksiyonu

Microsoft.Visualbasic.DateAndTime sınıfının bir üyesi olup verilen tarihteki haftanın gün sırasının bulur.

Kullanımı :
Weekday (Tarih)

Dönderilecek sayısal sonuçlar aşağıdaki günlere karşılık gelmektedir.

Gün adı-ing	Gün no	Gün adı-Tr
Sunday	1	Pazar
Monday	2	Pazartesi
Tuesday	3	Salı
Wednesday	4	Çarşamba
Thursday	5	Perşembe
Friday	6	Cuma
Saturday	7	cumartesi

Tablo 3.2: Weekday fonksiyonu parametreleri

Örnek : Programı yazdığımız günü beraber bulalım.

```
MsgBox(Weekday(Today))
satırlarını programınıza eklerseniz çalışmakta olduğunuz günün
sıra numarasını görebilirsiniz.
```

Resim 3.8: Weekday fonksiyonunun kullanımı

3.3.3. Weekdayname Fonksiyonu

Microsoft.Visualbasic.DateAndTime sınıfının bir üyesi olup 1 ile 7 arasında girilen sayısal değerlere karşılık gelen gün adını bulmaktadır. Aşağıdaki biçimde kullanılmaktadır.

WeekdayName(Gün_no)

Örnek :

```
MsgBox("Bugün Günlerden " & WeekdayName(Weekday(Today)))
```

Resim 3.9: WeekdayName fonksiyonunun kullanımı

3.4. "FormatNumber, FormatCurrency, FormatPercent, FormatDateTime, InputBox ve IsNumeric" Fonksiyonları

Çeşitli yapıları dönüştürmek için kullanacağımız fonksiyon türleridir. Bunlar sayı, para birimi, tarih ve saat gibi yapılarda sık kullanılan fonksiyonlardır. Sayısal ve sayısal olmayan verileri formatlı bir biçimde gösteren bir fonksiyon olup Microsoft.VisualBasic.Strings sınıfına aittir. Özellikle tarih ve zaman fonksiyonlarının kullanılmasında önemli rol oynar. Fonksiyondan dönen değer formatlı bir bilgidir.

3.4.1. FormatNumber Fonksiyonu

Control Panel içinde tanımlanmış sayı formatını uygular. İlk parametresi format uygulanacak sayıyı belirlerken, ikinci parametre ise ondalıklı kısmın kaç hassasiyet ile gösterileceğini belirtir.

Örnek :

```
Dim sayi As Double
        sayi = TextBox1.Text
             Label1.Text = FormatNumber(sayi, 2)
```

Resim 3.10: FormatNumber fonksiyonunun kullanımı

- Eğer ondalıklı kısmın görünmesini istemiyorsak o zaman son parametreyi 0 (sıfır) olarak yazmamız gerekir. Bu yazılım Control Panel içinden gelecek olası bir Digit sayısını önlemiş olur.

```
Dim sayi As Double
        sayi = TextBox1.Text
             Label1.Text = FormatNumber(sayi, 0)
```

Resim 3.11: FormatNumber fonksiyonunun kullanımı

- İstenirse sonuna istenilen string ifade eklenebilir. Para birimini eklemek istersek kod satırlarını aşağıdaki gibi düzenlememiz gerekir.

```
Dim sayi As Double
        sayi = TextBox1.Text
               Label1.Text = FormatNumber(sayi, 0) & "TL"
```

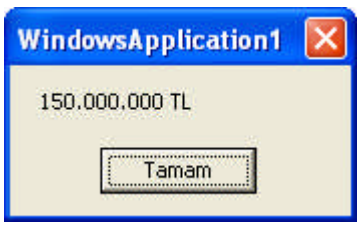
3.4.2. FormatCurrency Fonksiyonu

Control Panel içinde tanımlanmış para birimi formatını uygular.

Örnek :

```
Dim a As Integer = 3000000
            MsgBox(FormatCurrency(a, 0))
```

- En sondaki 0 digit numarası iptal etmek için kullanılır. Yani ondalıklı kısım görünmez.



Resim 3.12: FormatCurrency fonksiyonunun kullanımı

3.4.3. FormatPercent Fonksiyonu

Girilen ifadeyi yüzde biçiminde gösterir.

Örnek :

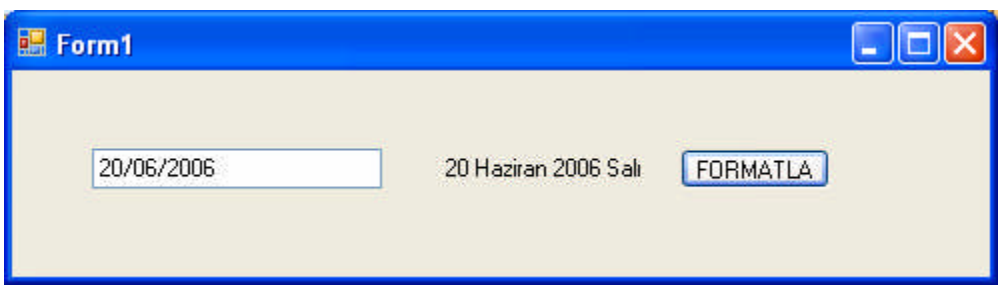
```
Dim x As Long
        x = 0.89123
        MsgBox(FormatPercent(x, 2))
```

3.4.4. FormatDateTime Fonksiyonu

Tarih ve saati biçimlendirmek için kullanılır.

Örnek :

Label1.Text = FormatDateTime(TextBox1.Text, DateFormat.LongDate)



Resim 3.13: FormatDateTime fonksiyonunun kullanımı

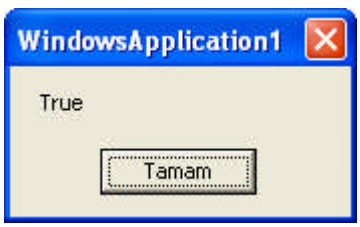
3.4.5. IsNumeric Fonksiyonu

Microsoft.Visualbasic.Information sınıfının bir üyesi olup girilen ifadenin sayı olarak değerlendirilip değerlendirilmeyeceğini belirler. Geriye True ya da False değerlerinden birini döndürür.

Kullanımı :
Isnumeric(ifade)

Örnek :

```
Dim a, b
        b = 23
        a = IsNumeric(b)
             MsgBox(a)
```



Resim 3.14: IsNumeric fonksiyonunun kullanımı

3.4.6. InputBox Fonksiyonu

Kullanıcının dışarıdan değer girmesi için Visual Basic.NET'te **InputBox** fonksiyonu geliştirilmiştir. InputBox fonksiyonu ile ekrandan bilgi alarak herhangi bir değişkene aktarma yapabiliriz. Bu fonksiyon standart olarak Ok ve Cancel düğmeleri bulunan bir pencere açar ve değer girişi bu pencereden yapılır.

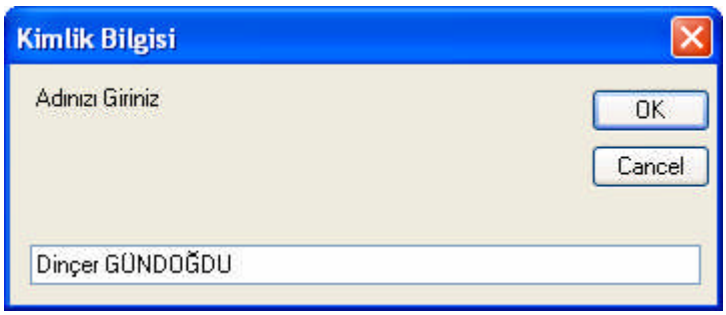
Kullanım formatı :

InputBox ("Çıkacak mesaj bilgisi", "başlık", "varsayılan değer", ekrana yatay uzaklığı, Ekrana dikey uzaklığı)

Örnek : Bu fonksiyonu kavramak için aşağıda verilen adımları sırasıyla gerçekleştirelim.

```
Dim ad As String
        ad = InputBox("Adınızı Giriniz", "Kimlik Bilgisi")
        MsgBox(ad)
```

- Bu kodu çalıştırdığımız zaman Resim 3.15'deki gibi, bilgi girişi yapmanızı isteyen bir ekran gelir. Buraya adımızı soyadımızı girelim.



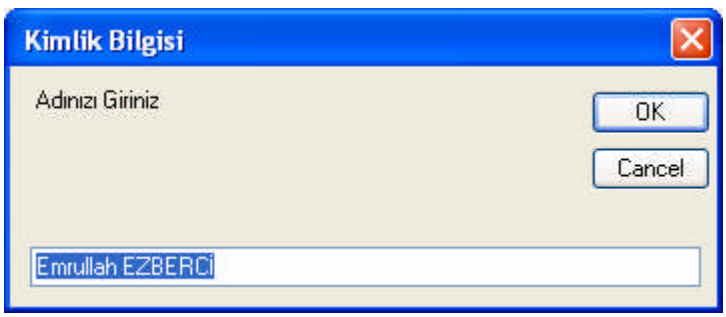
Resim 3.15: InputBox fonksiyonu kullanımı

- İstenilen bilgiyi girdiğinizde bunu, ad isimli string değişkene atayacaktır. Eğer bu değişkeni bir mesaj iletisinde kullanırsak görüntü şu şekilde olacaktır.

Resim 3.16: InputBox fonksiyonu kullanımı

- Şayet ilk çalıştığında Default (varsayılan) bir isim bilgisinin gelmesini istiyorsanız bunu, üçüncü alanan yazmanız gerekir. Yine çalıştığında ekrandaki yatay mesafesini dördüncü ve dikey mesafesini ise beşinci parametrelerde aşağıdaki gibi belirtmeniz gerekir.

```
Dim ad As String
        ad = InputBox("Adınızı Giriniz", " Kimlik Bilgisi",
"Emrullah EZBERCİ", 200, 100)
        MsgBox(ad)
```



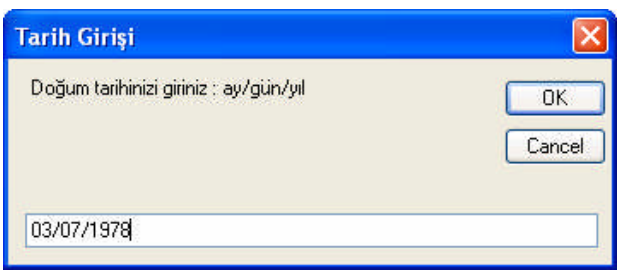
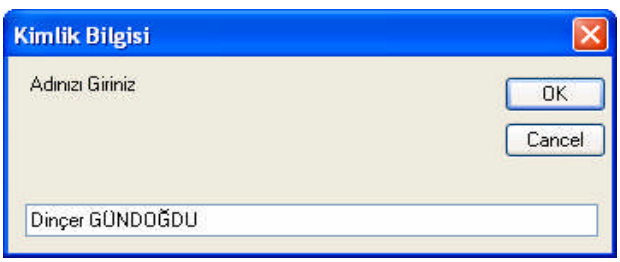
Resim 3.17: InputBox fonksiyonu kullanımı

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler

Ad değişkenine kendi adınızı atayarak isminizin uzunluğunu bulunuz.
`Dim ad As String`
`ad = "Kendi Adınız"`
`MsgBox(Len(ad))`	

Alttaki satırda a değişkeninin içeriğine göre b'nin alacağı değeri bulunuz.
`Dim a, b`
`a = 23.2`
`b = CInt(a)`
`MsgBox(a)`
`MsgBox(b)`	

Kendi doğum tarihinizi girerek o günden programı yazdığınız gün arasındaki tarih farkını hesaplayınız. 	
Kendi adınızı varsayılan değer olarak belirleyip mesaj kutusu içerisinde görüntüleyiniz. 	

ÖLÇME VE DEĞERLENDİRME

A- Objektif Testler (Ölçme Soruları)

Aşağıdaki sorulardan; ilk 4 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. ( ) Verilen metnin başındaki boşlukları kaldırmak için LTrim Fonksiyonu kullanılır.

2. ( ) Metin içindeki karakter sayısını bulmak için Left komutu kullanılır.

3. ( ) 1 ile 7 arasında girilen sayısal değerlere karşılık gelen gün adını bulmak için WeekDayName komutu kullanılır.

4. ( ) CInt fonksiyonu integer tipi bir sayıyı reel sayıya dönüştürür.

5. Girilen iki zaman arasındaki farkı bulan komut aşağıdakilerden hangisidir?
A) DateDiff
B) WeekDay
C) WeekDayName
D) DateName

6. WeekDay fonksiyonunda geri dönen değer "1" ise hangi güne karşılık gelir?
A) Wednesday
B) Tuesday
C) Monday
D) Sunday

7. Para birimi formatını uygulayan komut hangisidir?
A) FormatNumber
B) FormatCurrency
C) FormatPercent
D) FormatDateTime

8. InputBox fonksiyonunda yer alan 3.parametre aşağıdakilerden hangisine aittir?
A) Mesaj Bilgisi
B) Başlık
C) Varsayılan Değer
D) Koordinat

Değerlendirme

Sorulara verdiğiniz cevaplar ile cevap anahtarını karşılaştırınız, cevaplarınız doğru ise bir sonraki öğrenme faaliyetine geçiniz. Yanlış cevap verdiyseniz öğrenme faaliyetinin ilgili bölümüne dönerek konuyu tekrar ediniz.

ÖĞRENME FAALİYETİ– 4

AMAÇ

Gerekli ortam sağlandığında, döngü komutları ile program yazabileceksiniz.

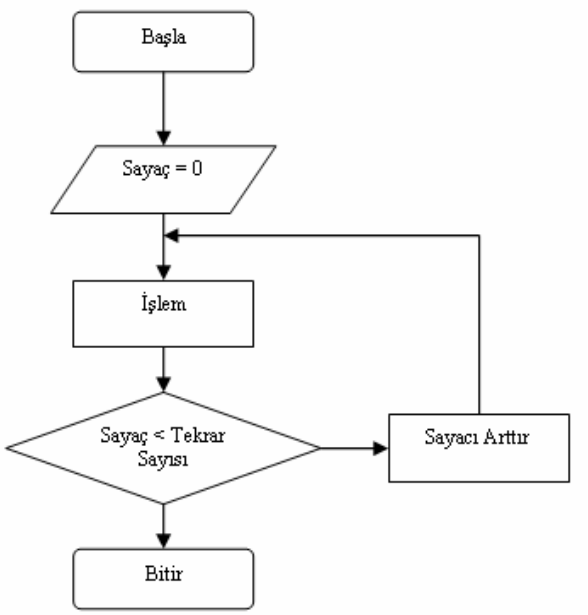
ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Döngülerin programlarınızda nerelerde kullanılacağını araştırınız.

4. DÖNGÜ KOMUTLARI

Algoritmalarda bazı işlemlerin tekrar çalışması için onları her seferinde yazmak gerekir. Ancak bu çözüm, çok fazla tekrar için hem yazmayı hem de okumayı zorlaştırır. Örneğin yüz elemanlı bir diziye ras gele sayı atanması için işlemin yüz defa yazılması gerekir. Döngüler ile işlem sadece bir defa yazılır ve tekrar sayısına göre bu işlem yinelenir.



Resim 4.1: Döngü işleminin akış şeması üzerinde gösterimi

4.1. “For Next ” Döngü Komutu

For….Next döngüsüyle bir olay yordamındaki ya da kod modülündeki belirli bir ifade ya da ifadeler grubunun çalışmasını belirlenmiş bir sayıda tekrarlayabilirsiniz. Bir döngünün birçok kez çalışmasını istiyorsanız, o zaman kullanacağınız en uygun yöntemdir. For…Next döngüsü istenilen sayıda ve aralıkta işlemleri yapar, daha sonra blok dışına çıkar.

Kullanımı :

For değişken = Başlangıç To BitişDeğeri [Step (artım miktarı)]
Komutlar
Next

Not: Şayet Step belirtilmez ise artım miktarı 1 kabul edilir.

Kullanım ifadesinde **For, To** ve **Next** gerekli anahtar sözcüklerdir, ayrıca **eşittir** (=) işleci de gereklidir. **Değişken** yerine geçerli döngü sayısının hesabını tutan bir sayı değişkeni ve **Başlangıç** ile **Bitişdeğeri** yerinde de döngünün başlama ve durma noktalarını belirten sayısal değerler yer alır.

Not: **Değişken** bildiriminin For…Next döngüsünden önce yapılmasına dikkat ediniz.

Uygulama: Aşağıda verilen komut satırlarını yazarak programı çalıştırınız ve sonucu arkadaşlarınızla tartışınız?

```
Dim i As Integer
For i = 1 To 4
   MsgBox("Visual Basic.NET")
Next
```

Örnek: Aşağıdaki For…Next döngüsü bilgisayarın hoparlöründen ardı ardına hızla altı bip sesi verir.

```
Dim i As Integer
        For i = 1 To 6
            Beep()
             Next
```

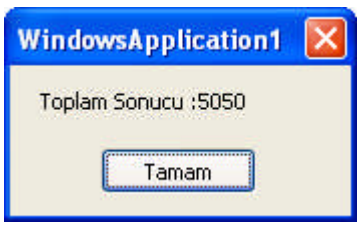
- Bu döngünün işlevsel olarak, Beep ifadesini bir yordam içinde altı kez yazmaya eşittir. Derleyici bu döngüyü

Beep ( )
Beep ( )
Beep ( )
Beep ( )
Beep ( )
Beep ( )

Örnek : 1'den 100'e kadar olan sayıların toplamını hesaplayalım.

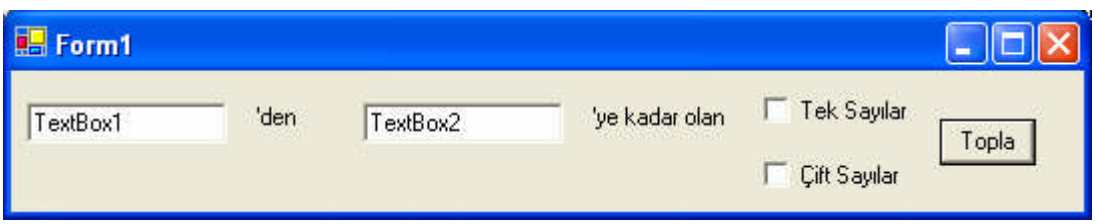
```
Dim i, Toplam As Integer
        For i = 1 To 100
            Toplam = Toplam + i
        Next
        MsgBox("Toplam Sonucu :" & Toplam)
```

- Yapılan örnekte 1'den 100'e kadar olan sayılar dediğimiz için döngünün başlangıç değeri 1 olarak bitiş değeri ise 100 olarak belirlendi. Bu aralıkta yer alan tüm sayıların toplamı ise **Toplam** isimli değişkene aktarıldı. Programı çalıştırdığımızda ise;



Resim 4.1: For... Next kullanımı

Örnek : Kullanıcının gireceği iki aralık arasındaki tek, çift veya bütün sayıları toplayacak bir programı yapalım. Örneğimiz için aşağıdaki formu oluşturalım.



Resim 4.2: For…Next kullanımı

```
Dim Bas, Son, top, i
    Bas = Val(TextBox1.Text)
    Son = Val(TextBox2.Text)
    If CheckBox1.Text And CheckBox2.Text Then
        ' her ikiside işaretli ise bütün sayıları
        For i = Bas To Son
            Top = top + i
        Next
    End If

    If CheckBox1.Text And Not CheckBox2.Text Then
        If Bas Mod 2 = 0 Then
            'Başlangıç Çift ise tek yap
            Bas = Bas + 1
        End If
```

```
        For i = Bas To son Step 2
            Top = top + i
        Next
    End If
    If Not CheckBox1.Text And CheckBox2.Text Then
        ' Check2 işaretli Check1 değilse çift sayıları
        If Bas Mod 2 <> 0 Then
            'Başlangıç tekse çift yap
            Bas = Bas + 1
        End If
        For i = Bas To son Step 2
            Top = top + i
        Next
    End If
    LabelBox3.text = top
     End Sub
```

4.2. “ Step ” Komutu

For döngüsü sayacın başlangıç değerinden başlayarak bitiş değerine kadar sayacı birer artırarak blok içinde komutları çalıştırır. Eğer artım miktarının bir değilde sizin belirleyeceğiniz aralıkta artmasını istiyorsanız step parametresinden sonra artım miktarını yazmanız gerekiyor. Başlangıç değeri, bitiş değerinden küçükse döngüye hiç girilmeyecektir.

Sayacın artarak değil de azalarak çalışması için Step’ten sonra negatif sayı vermeniz gerekir. Bu durumda ise başlangıç değeri, bitiş değerinden büyükse döngüye hiç girilmeyecektir.

Örnek : Bir metin kutusu içinde aşağıdaki satır numaraları sırasını görüntüleme

```
Dim i As Integer
        Dim Wrap As String
        Wrap = Chr(13) & Chr(10)
        For i = 5 To 25 Step 5
            TextBox1.Text = TextBox1.Text & "Line" & i & Wrap
             Next
```

Sonuç : Line 5

Line 10

Line 15

Line 20

Line 25

4.3. “ Do Until” veya “While ” Döngüsü

Bir blok içerisindeki komutları, belirtilen koşul sağlanıncaya kadar çalıştırma mantığına dayanır. **Until** veya **While** keyleri (anahtar) ile birlikte bir koşul belirtilerek kullanılır. Bu keyler **Do**’dan sonra olabileceği gibi **Loop**’dan sonra da olabilir. Aradaki fark ise başta olursa döngüye girmeden önce koşulu kontrol eder, eğer koşul doğru ise döngü içindeki komutları uygular. Sonda olursa ilk önce komutu uygular, daha sonra koşulu kontrol eder, eğer koşul doğru ise tekrar döngü içinden devam eder. While, genellikle “**Not**” keyi ile beraber kullanılır.

Kullanımı :

Do Until Şart

Komutlar

Loop

Bir şart gerçekleştiği sürece çalışması gereken program bloklarında kullanılır.

Örnek:

```
Dim a, b As Integer
b = 1
Do Until a = 5
    a = a + 1
    b = b * 2
Loop
MsgBox(a & vbTab & b)
```

Resim 4.3: Do Until-Loop kullanımı

Kullanımı :

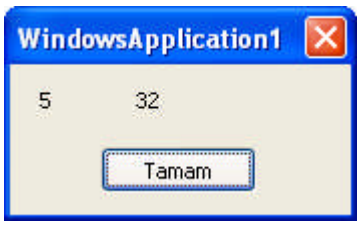
Do While Şart

Komutlar

Loop

Örnek:

```
Dim a, b As Integer
b = 1
Do While Not (a = 5)
    a = a + 1
    b = b * 2
Loop
MsgBox(a & vbTab & b)
```



Resim 4.4: Do While-Loop kullanımı

4.4. " Loop Until" veya "While" Döngüsü

Bu döngü yapılarında şart ifadesi sonda yer aldığı için komutlar mutlaka bir kez çalışır. Sonda kontrol edilen şart ifadesi doğru ise döngüye devam edilir, şart yanlış ise döngü dışına çıkılır.

Kullanımı :

Do

Komutlar

Loop Until Şart

Kullanımı :

Do

Komutlar

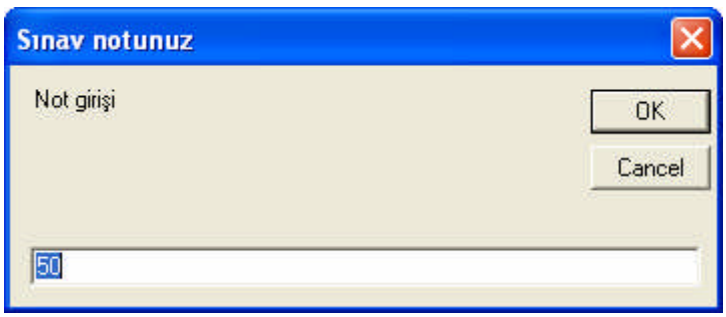
Loop While Şart

Bu döngülerin Do Until ya da Do While döngülerinden tek farkı döngüye girerken değil çıkarken şart kontrol edilir. Yani döngü içerisindeki kod en az bir defa çalışır.

Örnek: Kullanıcıya sınav notunu sorduğumuzu düşünelim. Gireceği not 0-100 aralığı dışında ise notu bu aralıkta girinceye kadar tekrar – tekrar sorulması gerekir. Bu iş Do-Loop Until yapısı uygundur. Çünkü döngü içerisinde not sorulduktan sonra not 0-100 aralığı dışında ise tekrar sorulması gerekir.

```
Dim nott
    Do
    nott = InputBox("Not girişi", "Sınav notunuz", "50")
    Loop Until (nott > 0) And (nott < 100)
```

Not: Visual Basic.NET'te not değişkeni özel bir deyim olduğu için nott olarak tanımlanmıştır. Programınızı yazarken bu kurala dikkat etmelisiniz.



Resim 4.5: Do-Loop Until kullanımı

4.5. "While End While " Komutu

Bir şart gerçekleştiği sürece çalışması gereken program bloklarında kullanılır.

Kullanımı :

While Şart

Komutlar

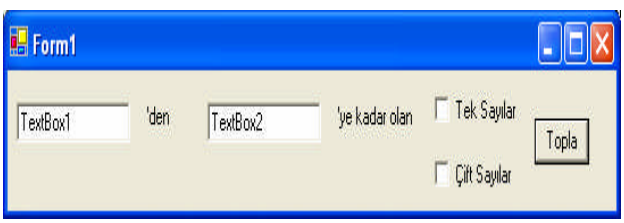
Wend

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler

Sizde Beep ( ) komutunu 100 defa çalışacak şekilde değiştiriniz.
`Dim i As Integer`
`For i = 1 To 6`
`Beep()`
`Next`	

1 ile 250 arasındaki sayıların toplamını hesaplayıp sonucu mesaj kutusu üzerinde görüntüleyiniz.
`Dim i, Toplam As Integer`
`For i = 1 To 100`
`Toplam = Toplam + i`
`Next`
`MsgBox("Toplam Sonucu :" & Toplam)`	

Girilen sayı aralığındaki bütün tek ve çift sayıların toplamını hesaplayan komut satırlarını yazınız. 	

ÖLÇME VE DEĞERLENDİRME

A- Objektif Testler (Ölçme Soruları)

Aşağıdaki sorulardan; ilk 5 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. ( ) Döngülerde kullanacağımız sayaç değişkenini döngülerden sonra da yapabiliriz.

2. ( ) Bir döngünün ne kadar yineleneceğini genellikle önceden bilemeyeceğimizden For…Next döngüleri kullanışlıdır.

3. ( ) Sayaç'ın artarak değil de azalarak çalışması için Step'ten sonra pozitif sayı vermeniz gerekir.

4. ( ) While döngüleri genellikle "**No**" keyi ile birlikte kullanılır.

5. ( ) Do-Loop Until ya da Do-Loop While döngülerinde komutlar en az bir defa çalışırlar.

6. For…Next döngülerinde artım miktarını belirleyen sayaç aşağıdakilerden hangisidir?
A) For
B) Next
C) Step
D) Until

7. Aşağıdakilerden hangisi bir döngü deyimi değildir?
A) Do Until-Loop
B) Do While-Loop
C) Do-Loop While
D) While-Loop While

Değerlendirme

Sorulara verdiğiniz cevaplar ile cevap anahtarını karşılaştırınız, cevaplarınız doğru ise bir sonraki öğrenme faaliyetine geçiniz. Yanlış cevap verdiyseniz öğrenme faaliyetinin ilgili bölümüne dönerek konuyu tekrar ediniz.

ÖĞRENME FAALİYETİ– 5

AMAÇ

Gerekli ortam sağlandığında, altprogram ve fonksiyon yazabileceksiniz.

ARAŞTIRMA

- Programlarımızda kullandığımız bu program parçacıklarının kullanım amaçlarını internet ve kaynak kitaplardan araştırınız.
- Visual Basic programlama dilinin bir önceki sürümlerine ait kitaplardan alt programları araştırınız.

5. ALT PROGRAMLAR

Bir fonksiyon veya bir alt program belirli bir işi yapmak için oluşturulan küçük program parçalarıdır. Fonksiyon ve alt program tanımlamanın bir çok avantajı vardır. Bir çok yerde aynı kodları kullanmanız gerektiğinde bunları her sefer yeniden yazmak yerine bu kodları bir prosedüre tanımlayıp kullanacağımız yerden bunu çağırmanız yeterlidir.

Örneğin programınızda birçok yerde faktöriyel hesabı yapmanız gerekiyorsa, her seferinde faktöriyel hesaplayacak kodu yazmaktansa bu işlemi bir fonksiyonda yaptırıp gerekli yerlerden bu alt programı çağırabilirsiniz.

Bir standart modül bir programda geniş kapsamlı ya da genel değişkenler ile Function ve Sub yordamlarını içeren ayrı bir kaptır. Bu bölümde genel değişkenlerin bildiriminin nasıl yapılacağını ve onların nasıl kullanılacağını öğreneceksiniz. Ayrıca kendi yordamlarınızı nasıl oluşturup çağıracağınız da öğreneceksiniz.

Uzun program yazdığınızda aynı değişken ve yordamların bazılarını kullanan çeşitli formlarınızın ve fonksiyon yordamlarınızın bulunması olasıdır. Değişkenler varsayılan olarak bir fonksiyon yordamında yereldir, yani yalnızca içinde oluşturuldukları olay yordamında okunabilir ya da değiştirilebilirler. Değişken bildirimlerini bir formun program kodunun en üstünde yapıp değişkenlere form boyunca daha geniş bir kapsam da kazandırabilirsiniz. Ancak, bir proje içinde birden çok form oluştursanız bir formun en üstünde bildirimi yapılan değişkenler yalnız o form içinde geçerli olacaktır. Aynı biçimde, fonksiyon yordamları bildirimi varsayılan olarak özeldir ve içinde oluşturuldukları form için yereldir. Örneğin, Button1_Click fonksiyon yordamının bildirimi özel olarak form1 içinde yapılmışsa bu yordamı Form2 adlı ikinci bir formdan çağıramazsanız.

Değişkenleri ve yordamları bir proje içindeki tüm formlarda ve fonksiyon yordamlarında paylaşmak için onların bildirimlerini projenin bir ya da birden çok standart modülünde yapabilirsiniz. Bir standart modül (ya da kod modülü), dosya uzantısı .vb olan ve programın her tarafında kullanabilen değişken ve yordamlar içeren özel bir dosyadır.

Yordamlar bir görevi yerine getirmek için ilişkili bir ifadeler kümesini gruplandırma yoludur. Visual Basic.NET'te esas olarak iki tür yordam vardır: İşlev yordamları ve Sub yordamları. İşlev yordamları olay yordamlarından ya da diğer yordamlardan adlarıyla çağrılırlar. İşlev yordamları bağımsız değişkenler alabilirler ve her zaman işlev adı içinde değer döndürürler, genel olarak da hesaplamalarda kullanılırlar.

İşlev yordamları ve Sub yordamları bir formun program kodunda tanımlanabilir, ancak çoğu kullanıcılar için yordamların tüm projeyi kapsamaları bakımında bir standart modül içinde oluşturulmaları en kullanışlı yaklaşımdır. Bu, özellikle genel amaçlı yordamlar olarak adlandırılabilecek ve çeşitli programlama işleri kapsamında kullanılması yeterince esnek ve yararlı olan kod blokları için doğrudur.

5.1. "Sub-End Sub" Alt Programı

Belirli bir tekrar gerektiren komutları her seferinde yazmak yerine, bir **Procedure** içinde tanımlayıp ihtiyacımız olduğunda çağırmak, programlama hızını arttıran önemli etkenlerdendir. Bunlardan ilki **Sub-End Sub** yapısıdır.

Tüm çalışabilen kodlar mutlaka Sub Procedure'ü içinde olmalıdır. Sub yapısını; bir Module, Class, Interface veya Structure içinde tanımlayabiliriz. Ancak, bir Sub içinde başka bir Sub tanımlanamaz.

Sub Procedure'lerinde varsayılan olarak yani, hiçbir şey yazılmadığında Public açma Mod'u istenirse diğer seçenekler de kullanılabilir (protected, Friend, Protecte Friend, Private)

Bir Sub yapısını acilen sonlandırmak için; Exit Sub veya Return ifadesi kullanılabilir. Sub Procedure'leri parametreli veya parametresiz olarak yazılabilir. Çağırmak için **Call** ifadesi kullanılır.

Olay yordamlarından ya da diğer yordamlardan adlarıyla çağrılırlar, bağımsız değişkenler alabilirler ve değer de döndürebilirler. Anca Sub yordamları işevlerden (Funciton) farklı olarak kendi özel Sub yordam adlarıyla ilişkili değerler döndüremezler. Sub yordamları yaygın olarak girdi almak ya da işlemek, çıktı görüntülemek veya özellik ayarlamak için kullanılır.

Kullanımı :

Sub *isim(Parametreler)*

Komut veya komutlar

End Sub

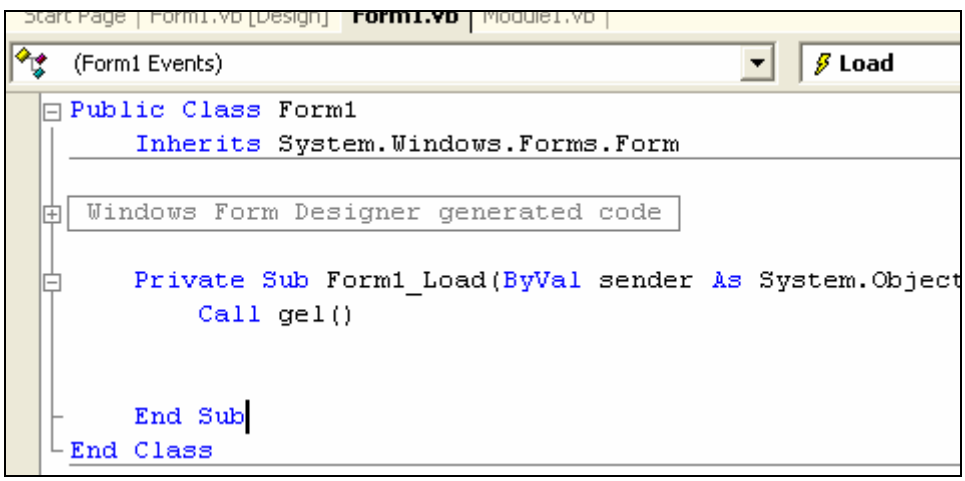
Yukarıdaki yazılımda da görüldüğü gibi Sub ifadesi mutlaka, End Sub ifadesi ile bitmek zorundadır. Sub için bir isim verildikten sonra parantez açılır ve kapanır. Eğer bu yapı için dışarıdan bir parametre yollanacak ise o zaman, bu kullanılacak parametrelerin ve veri tiplerinin parantez içinde belirtilmesi gerekir.

Örnek : Sub Procedure'e isim olarak "gel" verilmiştir ve hiçbir parametre tanımlanmadığı için parantez içi boş bırakılmıştır. İçinde ise basit bir mesaj iletisi kullanılmıştır.

```
Sub gel()
        MsgBox(" Kaydedecek misiniz?")
End Sub
```

Oluşturulan Sub'ı aşağıdaki şekilde herhangi bir program yapısı içinde çağırdığımızda buradaki komut çalışacaktır. Aynı yapı içinde çağırma yapılırken istenirse, başına Call ifadesi yazılmayabilir.

Sub'ı eğer bir form içine yazıyorsak, Form'un Declaration Event'ını seçmemiz gerekmektedir. Çağırma işlemini ise herhangi bir kontrol içinde, herhangi bir program komutu arasında yapabiliriz. Şekilde Button1_Click Event'ı kullanılarak yapılmıştır.



Resim 5.1: Sub-End Sub kullanımı

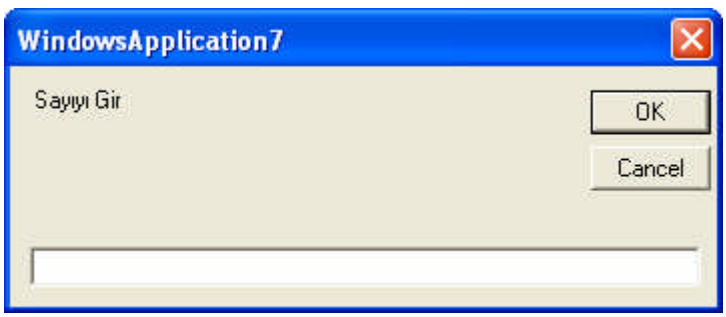
Örnek : Form1_Declaration Event'ına aşağıdaki Kod'u yazalım.

```
Sub kare()
    Dim a, b As Integer
    a = InputBox("Sayıyı Gir")
    b = a * a
    MsgBox(a & " sayısının karesi :" & b)
End Sub
```

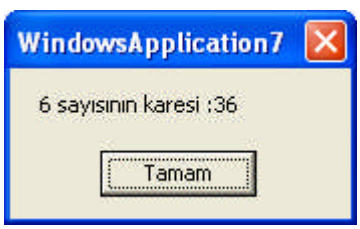
- Bunu çağırmak için Button1_Click Event'ına aşağıdaki Kod'u yazalım.

```
Call kare()
```

- Programı çalıştırarak Button1'e tıkladığımızda, gelen InputBox iletisine şekildeki gibi karesini almak istediğimiz sayıyı girelim ve OK butonuna basalım.



Resim 5.2: Sub-End Sub kullanımı



Resim 5.3: Sub-End Sub kullanımı

- Bunun sonucunda şekilde gibi, sayının karesini bize gösterecektir. Belki çoğunuz bu kadar uzun işleme ne gerek var diye, düşünebilir. Ancak "**kare**" isimli Sub'ı her zaman ve her yerde çağırabileceğimizi unutmayınız.

Örnek : Şimdi de girilen üç sayının ortalamasını alan Sub Procedure'ünü oluşturmak için, Form1_Declaration Event'ına aşağıdaki kodu yazalım;

```
Sub ortalama(ByVal a As Integer, ByVal b As Integer, ByVal c As
Integer)
        Dim ortalama As Integer = (a + b + c) / 3
        MsgBox(" Sayıların Ortlaması : " & ortalama)
    End Sub
```

- Bu kodu çağırmak için aşağıdaki kodu Button1_Click Event'ına yazalım.

```
ortalama(50, 30, 90)
```

- Bu programı çalıştırdığımızda şekildeki gibi sayıların ortalaması karşımıza çıkar. Burada ortalama ifadesini yazdıktan sonra, parantezi açınca tüm girmemiz gereken parametreler sırası ile karşımıza gelir yalnız bu işlemde mutlaka, parametrelerin hepsini girmek gerekir.

Resim 5.4: Sub-End Sub kullanımı

5.2. " Function-End Function "

Bu tür Procedure'ler, Sub Procedure'lerin tüm özelliklerini taşır. Ancak en büyük farkı; Procedure içinde meydana gelecek değerin, Procedure çağrıldıktan sonra herhangi bir değişken veya kontrole atanmasıdır. Bunun için; fonksiyon içinde döndürülecek değer Return ifadesi ile gönderilmelidir. Ayıraca, döndürülecek değerin veri tipi argümanlarının tanımlandığı parantezden sonra **As** ifadesi ile belirtilebilir. Fonksiyonlarda ifadeler genellikle metin işleme, girdi işleme ya da bir matematiksel değer hesaplama gibi anlamlı işleri yapar.

Kullanımı :

Function isim (Argümanlar) as Döngülecek veritipi

Komutlar

Return döndürücelek değer

End Funciton

Bir fonksiyon End Function satırı ile biter. Ancak bazı şartlar gerçekleştiğinde fonksiyonun çalışmasını bitirmeden çıkmak için Exit Function kullanılabilir.

Fonksiyon çağrısı

Programın herhangi bir yerinde bir fonksiyonu şu şekilde çağırabiliriz.

Donendeger=FonksiyonAdi([Giris degerleri])

Örnek :Faktöriyel hesaplayan bir fonksiyon yazalım. Bunun için kod penceresine aşağıdaki fonksiyonu yazalım.

```
Public Function Fak(ByVal x As Integer)
        Dim i, s As Integer
        s = 1
        For i = 1 To x
```

```
            s = s * i
        Next
        Return s
    End Function
```

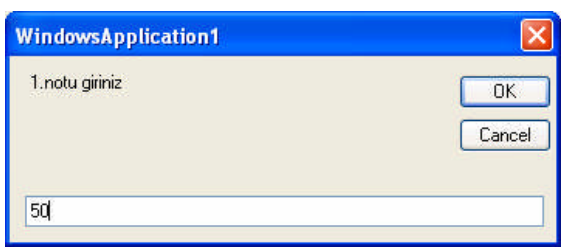
Örnek : Form1_Decleration kısmına aşağıdaki kodu yazalım. Burada ise döndürülecek değer bir dizidir.

```
Function dizi(ByVal notsayisi As Integer) As Array
        Dim i, s(notsayisi) As Integer
        For i = 0 To notsayisi - 1
            s(i) = InputBox(i + 1 & ".notu giriniz")
        Next
        Return s
    End Function
```

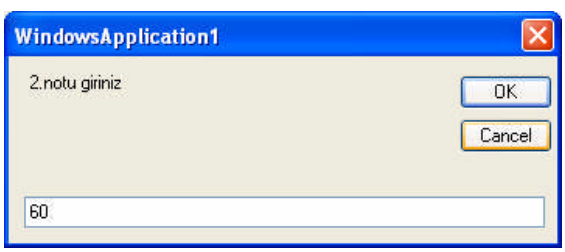
- Button1_Click Event'ına verilen kodu yazalım.

```
Dim toplam, k(10) As Integer
        k = dizi(3)
        toplam = k(0) + k(1) + k(2)
        MsgBox(toplam)
```

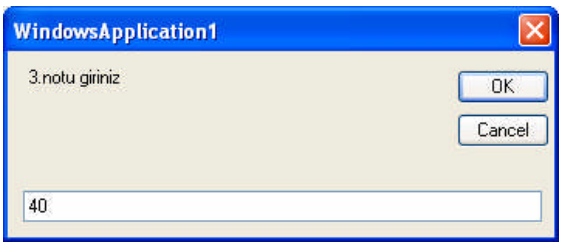
- Programı çalıştırıp butona bastığımızda, bizden 3 adet sayı girmemizi isteyecektir ve girilen bu 3 sayının toplamını, ekranda bir mesaj iletisi ile gösterilecektir.



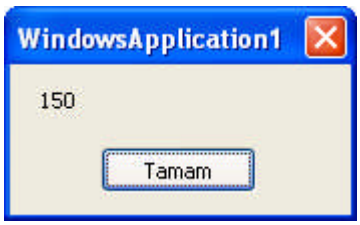
Resim 5.5: Function-End Function kullanımı



Resim 5.6: Function-End Function kullanımı



Resim 5.7: Function-End Function kullanımı



Resim 5.8: Function-End Function kullanımı

5.3. “ Private ve Public ” Anahtar Kelimeleri

Prosedürler **Public** olarak tanımlanırsa programdaki bütün form ve modüller bu prosedürü çağırabilir. Private olarak belirlenen prosedürler ise yalnızca tanımlandığı form veya modülden çağrılabilir.

Örneğin Form1’de Public oalarak tanımladığınız prosedürü Form2’den veya diğer modürllerin birinden çağırırken önce Form1’in ismini vermeniz gerekir.

5.4. “ Call ” Deyimi

Sub alt programlarını çağırmak için kullanılır.

```
Sub kare()
        Dim a, b As Integer
        a = InputBox("Sayıyı Gir")
        b = a * a
        MsgBox(a & " sayısının karesi :" & b)
    End Sub
```

Bu alt programı çağırmak için aşağıdaki kodu yazmalıyız ;

```
Call kare()
```

5.5. “ ByVal” (Varsayılan) ve “ ByRef ” Kelimeleri

Tanımlanmış bir prosedüre değer gönderirken tanımlanan değişkenlerin değerlerinden **ByVal** ya da bellekteki adreslerinden çağırmak için **ByRef** kullanılır.

ByRef olarak tanımlanmış parametrelerde parametrenin değeri değil, o parametrenin bellekteki adresi gönderilir. Dolayısıyla bu parametreye atanan değer aynı bellek bölgesinde değişikliğe sebep olacağı için fonksiyonu çağıran yer de bu değişimden etkilenir.

ByVal olarak tanımlanmış parametrelerde ise fonksiyona parametrenin adresi değil bir kopyası gönderilir. Bu iki kopya ayrı bellek bölgelerini kullanacaklarından parametrenin fonksiyon içindeki değişiminden çağıran yer etkilenmez.

```
Public Function f(ByVal z As Integer)
        z = 7
        Return 3
    End Function

Private Sub Button1_Click(ByVal Sender As System.Object, ByVal
e As System.EventArgs) Handles Button1.Click
        Dim t As Integer
        t = 5
        Debug.WriteLine(t)                              'sonuc 5
        Debug.WriteLine(f(t))               'sonuc 3
        Debug.WriteLine(t)                              'sonuc 5
    End Sub
```

Örnek : Aşağıdaki kodu Form1_Declaration kısmına yazalım. Dikkat edilirse “iv” isimli değişken ByVal olarak tanımlanmıştır.

```
    Sub cv(ByVal iv As Integer)
        iv = 13
    End Sub
```

- Button1_Click Event’ına ise yukarıda oluşturduğumuz Sub’ı, aşağıdaki şekilde çağıralım;

```
Dim c As Integer
        c = 1
        cv(c)
        MsgBox(c)
```

- Sonuçta ekrana “1” değeri gelir. Çünkü “iv” isimli değişken ByVal olarak tanımlandığı için, programın herhangi bir yerinde değiştirme hakkına sahibiz ve en son “1” ataması yapıldığı için, bu son değer geçerli olacaktır.

- İlk adımda yazdığımız kod içindeki “iv” değişkeninin ifadesini ByRef olarak aşağıdaki şekilde değiştirelim;

```
    Sub cv(ByRef iv As Integer)
        iv = 13
    End Sub
```

- Bu seferde sonuç “13” olarak karşımıza gelir. Bunun sebebi ise Procedure içinde “iv” isimli değişkene “13” ataması hafızaya yerleşir ve değeri ne kadar değiştirilmek istenirse istensin, bu ilk değer her zaman geçerli olur.

Not: ByRef olursa sonuç: “13”, ByVal olursa sonuç “1” çıkar.

Örnek : Verilen Procedure’leri oluşturalım.

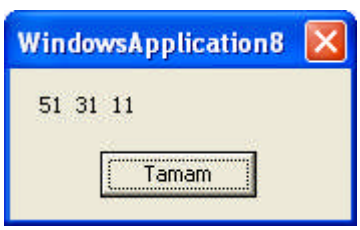
```
Sub son(ByVal a As Integer, ByRef b As Integer, ByRef c As
Integer)
        a = a + 1
        b = b + 1
        c = c + 1
        MsgBox(a & "" & b & "" & c)
    End Sub


Sub son2()
        Dim x As Integer = 50
        Dim y As Integer = 30
        Dim z As Integer = 10
        Son(x, y, z)
        MsgBox(x & "" & y & "" & z)
```

- Button içine aşağıdaki kodu yazalım;

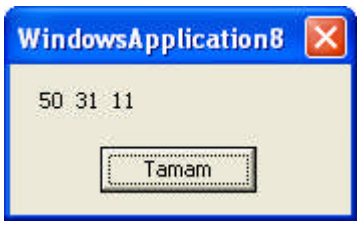
son2

- Programı çalıştırıp, butona bastığımızda ilk olarak “son” yapısındaki mesaj kutusu şekildeki gibi karşımıza çıkacaktır.



Resim 5.9: ByVal kullanımı

- Yukarıda bir sorun yok, ancak OK ile devam ettiğimiz zaman, bu seferki görüntü şekildeki gibi olacaktır. Çünkü ilk parametre ByVal olduğu için, bulunduğu yerdeki değer olan 50 değeri atanmıştır. Diğerleri ByRef olduğu için hafızadaki yerleşik değerler atanmış, dolayısı ile bir önceki Sub’da hesaplanan değerler karşımıza gelmiştir.



Resim 5.10: ByRef kullanımı

Not: Ne zaman ByVal ve ne zaman ByRef kullanacağıyla ilgili bazı ana noktalar şu şekildedir.

- Bir yordamın, kendisine bir bağımsız değişken aracılığıyla aktarılan bir değişkeni düzeltmesini istemiyorsanız ByVal kullanın.
- Bir yordamın kendisine aktarılan bir değişkeni düzeltmesine izin vermek istediğinizde ByRef kullanın.
- Kuşkulu durumlarda ByVal anahtar sözcüğünü kullanın.

5.6. “ Return ” Deyimi

Bir fonksiyonda geriye değer döndürmek için return komutu kullanılır.

Örnek : Aşağıdaki kodu Form1_Declaration kısmına yazalım.

```
Function dizi(ByVal notsayisi As Integer) As Array
        Dim i, s(notsayisi) As Integer
        For i = 0 To notsayisi - 1
            s(i) = InputBox(i + 1 & ".notu giriniz")
        Next
        Return s
    End Function
```

- Button1_Click Event’ına aşağıdaki kodu yazalım.

```
Dim toplam, k(10) As Integer
        k = dizi(3)
        toplam = k(0) + k(1)
        MsgBox(toplam)
```

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler

Altta verilen program yapısını inceleyerek siz de klavyeden girilen sayının karekok ( ) isimli fonksiyonda karekökünü bulunuz.
`Sub kare()`
`  Dim a, b As Integer`
`  a = InputBox("Sayıyı Gir")`
`  b = a * a`
`MsgBox(a & " sayısının karesi :" & b)`
`  End Sub`
.	

Aynı örneği 5 sayı için yapınız.
`Sub ortalama(ByVal a As Integer, ByVal b As Integer, ByVal c As Integer)`
`  Dim ortalama As Integer = (a + b + c) / 3`
`  MsgBox(" Sayıların Ortlaması : " & ortalama)`
`  End Sub`	

ÖLÇME VE DEĞERLENDİRME

A- Objektif Testler (Ölçme Soruları)

Aşağıdaki sorulardan; ilk 5 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. ( ) Bir fonksiyonda geriye değer döndürmek için return komutu kullanılır.

2. ( ) Tanımlanmış bir prosedüre değer gönderirken tanımlanan değişkenlerin değerlerinden (As) ya da bellekteki adreslerinden (Call) faydalanılabilir.

3. ( ) Bir fonksiyon Exit Function satırı ile biter. Ancak bazı şartlar gerçekleştiğinde fonksiyonun çalışmasını bitirmeden çıkmak için End Function kullanılabilir.

4. ( ) Tüm çalışabilen kodlar mutlaka Sub Procedure'ü içinde olmalıdır. Sub yapısını; bir Module, Class, Interface veya Structure içlinde tanımlayabiliriz. Ancak, bir Sub içinde başka bir Sub tanımlanamaz.

5. ( ) Prosedürler **Public** olarak tanımlanırsa programdaki bütün form ve modüller bu prosedürü çağırabilir.

6. Sub alt programını çağırmak için aşağıdaki komutlardan hangisi kullanılır?
A) Return
B) Private
C) Call
D) ByVal

7. Tanımlanmış bir prosedüre değer gönderirken bellekteki adreslerinden çağırmak için aşağıdakilerden hangisi kullanılır?
A) ByRef
B) ByVal
C) Call
D) Return

Değerlendirme

Sorulara verdiğiniz cevaplar ile cevap anahtarını karşılaştırınız, cevaplarınız doğru ise bir sonraki öğrenme faaliyetine geçiniz. Yanlış cevap verdiyseniz öğrenme faaliyetinin ilgili bölümüne dönerek konuyu tekrar ediniz.

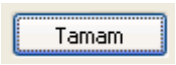
MODÜL DEĞERLENDİRME

Aşağıdaki sorulardan; ilk 10 soruda verilen ifadeye göre parantez içine doğru ise “D”, yanlış ise “Y” yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. ( ) Visual Basic.NET’te Class eklemek için Project menüsünden Add New Item seçilir.

2. ( ) Access Modifier içerisinde Public deyimi kullanılabilir.

3. ( ) Math.Pow fonksiyonu bir sayının logaritmasını almak için kullanılır.

4. ( ) MsgBoxResult fonksiyonunda geri dönen değer 1 ise 

butonuna basılmıştır.

5. ( ) Bir form üzerinden başka bir formu Show komutu yardımıyla çağırabiliriz.

6. ( ) Verilen metnin başındaki boşlukları kaldırmak için LTrim Fonksiyonu kullanılır.

7. ( ) 1 ile 7 arasında girilen sayısal değerlere karşılık gelen gün adını bulmak için WeekDayName komutu kullanılır.

8. ( ) Bir döngünün ne kadar yineleneceğini genellikle önceden bilemeyeceğimizden ı For…Next döngüleri kullanışlıdır.

9. ( ) Sayaç’ın artarak değil de azalarak çalışması için Step’ten sonra Pozitif sayı vermeniz gerekir.

10. ( ) Tüm çalışabilen kodlar mutlaka Sub Procedure’ü içinde olmalıdır. Sub yapısını; bir Module, Class, Interface veya Structure içlinde tanımlayabiliriz. Ancak, bir Sub içinde başka bir Sub tanımlanamaz.

11. Access Modifier içerisinde aşağıdaki deyimlerden hangisi kullanılmaz?
A) Public
B) Private
C) Class
D) Friend

12. Class yapıları aşağıdakilerden hangisi ile tanımlanır?
A) New
B) Cursor
C) Math
D) System

13. MsgBoxStyle.Information komutu aşağıdaki simgelerden hangisini çıkarır?

A) 

B) 

C) 

D) 

14. MsgBoxResult fonksiyonunda geriye dönen değer 2 ise aşağıdaki butonlardan hangisi seçilmiştir?

A) Ok
B) Abort
C) Retry
D) Cancel

15. InputBox fonksiyonunda yer alan 3.parametre aşağıdakilerden hangisine aittir?

A) Mesaj Bilgisi
B) Başlık
C) Varsayılan Değer
D) Koordinat

16. For…Next döngülerinde artım miktarını belirleyen sayaç aşağıdakilerden hangisidir?

A) For
B) Next
C) Step
D) Until

17. Aşağıdakilerden hangisi bir döngü deyimi değildir?

A) Do Until-Loop
B) Do While-Loop
C) Do-Loop While
D) While-Loop While

18. Sub alt programını çağırmak için aşağıdaki komutlardan hangisi kullanılır?

A) Return
B) Private
C) Call
D) ByVal

19. Tanımlanmış bir prosedüre değer gönderirken bellekteki adreslerinden çağırmak için aşağıdakilerden hangisi kullanılır?

A) ByRef
B) ByVal
C) Call
D) Return

CEVAP ANAHTARLARI

Öğrenme Faaliyeti – 1 Cevap Anahtarı

1	D
2	D
3	Y
4	D
5	D
6	C
7	A
8	D

Öğrenme Faaliyeti – 2 Cevap Anahtarı

1	D
2	Y
3	D
4	D
5	C
6	D

Öğrenme Faaliyeti – 3 Cevap Anahtarı

1	D
2	Y
3	D
4	Y
5	A
6	D
7	C
8	C

Öğrenme Faaliyeti – 4 Cevap Anahtarı

1	Y
2	D
3	Y
4	Y
5	D
6	C
7	D

Öğrenme Faaliyeti – 5 Cevap Anahtarı

1	D
2	Y
3	Y
4	D
5	D
6	C
7	A

Cevaplarınızı cevap anahtarları ile karşılaştırarak kendinizi değerlendiriniz.

ÖNERİLEN KAYNAKLAR

- http://bilisim-kulubu.com
- www.ceturk.com
- www.dotnetturk.com
- www.findikkurdu.com
- www.freevbcode.com
- www.godoro.com
- http://msdn.microsoft.com/netframework/
- www.programlama.com
- www.vbturk.net
- www.wikipedia.org
- www.yazgelistir.com
- www.yazilimgrubu.com
- www.yazilimuzmanı.com
- www.startvbdotnet.com
- www.vbasicmaster.com

KAYNAKÇA

- ÇÖMLEKÇİ Mehmet(Çeviren), **Visual Basic 6 Temel Kullanım Klavuzu**, Alfa Basım Yayım Dağıtım San. Ve Tic. Ltd. Şti, 1999.
- GÜLEÇ Hakan (Çeviren), **Visual Basic 2005**, Alfa Yayınları, 2006.
- HALVORSON Michael, **Microsoft Visual Basic .NET Step By Step**, Microsoft Pres, A Divicion of Microsoft Corporation One Microsoft Way Redmond, 2002.
- HOCAOĞLU Özgür, **Visual Basic .NET**, Pusula Yayıncılık, 2005.
- İNAN Yüksel, Nihat DEMİRLİ, **Visual Basic. NET 2003**, Palme Yayıncılık, 2005.
- KARAGÜLLE İhsan, **Visual Basic.NET Başlangıç Rehberi**, Türkmen Yayınevi, 2003.
- PALA Zeydin, **Microsoft Visual Basic.NET,** Türkmen Kitabevi, 2003.
- YANIK Memik, **Visual Basic 5.0,** Beta Basım Yayım Dağıtım A.Ş, 1997.
- http://www.bmsoftware.net/programlama/vbnet/vbnet02.aspx
- http://www.hazirkod.com/default.asp?fform_kategori_id=20&fform_kategori=VISUAL%20BASIC.NET
- http://www.msakademik.net/makaleler.aspx?grup=VBN
- http://www.vbasicmaster.com
- http://www.vbturk.net/
- Mastering, Visual Basic .NET
- Evongelos Petroutsos, SYBEX, Inc., Alameda, CA, 2002