T.C.
MİLLİ EĞİTİM BAKANLIĞI

MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİN
GÜÇLENDİRİLMESİ PROJESİ)

BİLİŞİM TEKNOLOJİLERİ

GÖRSEL PROGRAMLAMA YARDIMCI KODLARI

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	**482BK0070**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Veritabanı Programcılığı**
MODÜLÜN ADI	**Görsel Programlama Yardımcı Kodları**
MODÜLÜN TANIMI	Görsel programlamada yardımcı kodlar ile ilgili öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Görsel Programlama Komutları modülünü bitirmiş olmak
YETERLİK	Görsel programlama dilinde yardımcı komutları yazmak
MODÜLÜN AMACI	**Genel Amaç:** Gerekli ortam sağlandığında, istisnai durumları yakalayabilecek, yapı ve sınıf tanımlayabilecek döngü komutlarını kullanabilecek ve nesneye yönelik kod yazabileceksiniz. **Amaçlar:** 1. İstisnai durumları yakalayabileceksiniz. 2. Yapı ve sınıflar kütüphaneleri ile çalışabileceksiniz. 3. Döngü komutlarını kullanabileceksiniz. 4. Nesneye yönelik kod yazabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Bilgisayar laboratuvarı ve bu ortamda bulunan; Bilgisayar, yazıcı, internet bağlantısı, bilgisayar masaları, kâğıt, kalem
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,

Programcılıkta karşımıza çıkacak sorunlar hiçbir zaman bizleri yıldırmamalıdır. Önemli olan kişide öğrenme isteği olmasıdır ve öğrenen kişinin hemen vazgeçmemesidir.

Yapılan işlerin hepsinde başarılı olunamayabilir. Ancak gayret, çalışma ve sabır sonrasında tüm sorunların üstesinden gelinebilir. Bu bazıları için kısa bazıları için ise uzun zaman alabilir. Hedefine ulaşmadan pes etmemeniz gerekir.

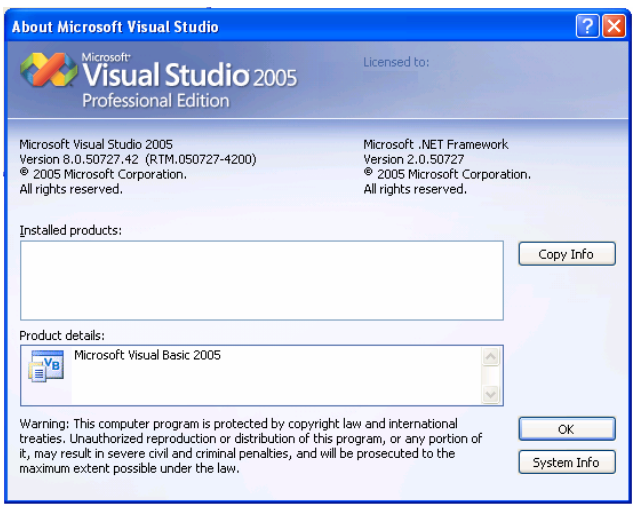
Bu modül ile öğreneceğiniz konular istisnai durumlar, yapı ve sınıflar, FV, with ve For Each komutları ve nesneye yönelik kodlardır. Modülü bitirdiğinizde anlamadığınız yerleri tekrar okuyup uygulayınız. Konuları tam olarak kavramadan diğer modüle geçmeyiniz.

Konular kapsamlı olarak, derinlemesine anlatılmamıştır. Ne kadar çok uygulama, araştırma yapar iseniz kendinizi o kadar geliştirirsiniz ve ilerletirsiniz.

Her dilin kendine göre avantajı bulunmaktadır. Modülde verilen örnekleri yaptığınızda o dili öğrenmiş olmayacaksınız. Asıl istenen, anlatılmak istenen konunun uygulanmasıdır, somut hale gelmesidir, anlaşılır hale gelmesidir.

Bu modülde verilen programlarda bilerek veya bilmeyerek yapılmış hatalar olabilir. Bulduğunuz hataları ve yeni önerilerinizi arkadaşlarınız ile paylaşınız.

Belli yerlerde geçen araştırma konuları için "Önerilen Kaynaklar" kısmından yardım almayı unutmayınız.



ÖĞRENME FAALİYETİ–1

AMAÇ

Gerekli ortam sağlandığında, istisnai durumları yakalayabilecek ve bu durumlara göre hata mesajları yazdırabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde yapmanız gereken araştırmalar şunlardır:

1. İstisnai durumun ne olduğunu araştırınız.
2. Hata sınıfları ve bunların oluşması için gerekli şartları araştırınız.

Araştırma işlemleri için internet ortamı ve görsel programlama dilini anlatan kaynak kitaplardan faydalanınız.

1. İSTİSNAİ DURUMLAR

Tüm programlama dillerinde (Visual Basic, Visual C#, Visual Java vs.) program akışını etkileyen hatta durdurabilen istisnai durumlar (Exceptions) bulunabilir. İyi bir bilgisayar programcısı yaptığı programlarda hata çıkmayan, kullanıcıyı sıkmayan, sistem kaynaklarını gereğinden fazla kullanmayan ve her platformda çalışan programlar üreten kişidir.

İstisnai durum, programda ortaya çıkması beklenmeyen bir durumdur. Mesela, bir bölme işleminde bölen 0 (sıfır) olursa istisnai durum oluşur. Bu durum programın çalışması esnasında ortaya çıkar ve her ortamda programın kırılmasını sağlar. Bu durumda programın çalışmasını durdurduğu için programın güvenirliliği tartışılır hale gelir. Önemli olan bu hataların ortaya çıkacağı komut satırlarını bulup gerekli önlemleri almaktır.

Bu öğrenme faaliyetinde Visual Basic.NET programlama dilinde istisnai durumların hangi şartlarda oluşabileceği ve programın kırılmasının engellenmesi için alınacak tedbirler hakkında bilgi verilecektir.

1.1. Try Catch (Finally) Komutu

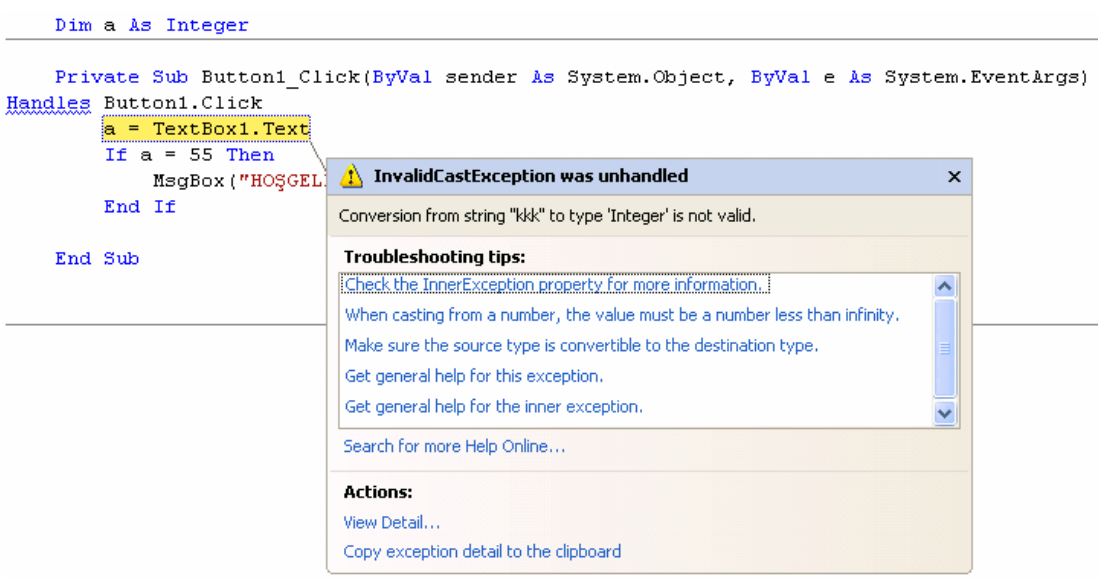
Ortaya çıkabilecek istisnai durumları yönetmek için kullanılan komuttur. Basit bir örnekle istisnai durumun nasıl oluştuğunu anlatım.

Öncelikle klavyeden girdiğimiz bilginin 55 sayısına eşit olup olmadığını kontrol eden basit bir program yazıp buna uygun bir form oluşturalım.

```
Dim a As Integer
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
   a = TextBox1.Text
   If a = 55 Then
      MsgBox("HOŞGELDİNİZ")
   End If
       End Sub
```

Resim 1.1: İstisna oluşumu örnek form

Oluşturduğumuz programı çalıştırdığımızda TextBox1 kutucuğuna bilgi yazıp kontrol butonu ile girilen bilginin 55 olup olmadığını kontrol edebiliriz. Kullanıcı bu kutucuğa yanlışlıkla rakam yerine string (harf, kelime) ifadesi yazarsa karşımıza bir istisnai durum çıkacaktır.



Resim 1.2: İstisnai durum görüntüsü

Yukarıdaki istisnai durum TextBox1 kutucuğuna rakam girmemiz gerekirken string bilgisi girdiğimiz için oluşmuş ve program durdurulmuştur. Programcı programında böyle bir hata ortaya çıkacağını tahmin edip bunun önlemini almalıdır.

İşte try…catch komutu bu durumda kullanılır. **Try** ifadesi çalışma hatası olabilecek komut satırlarından hemen önce, **catch** ifadesi ise çalışma hatası oluştuğunda çalıştırılacak ifadelerin önüne yazılır. Ayrıca programımızda hata olup olmadığına bakılmaksızın çalıştıracağımız komutlar varsa bunlar **finally** bloğu altına yazılarak çalıştırılabilir. Finally bloğu isteğe bağlı olarak kullanılır.

Try …Catch …Finally ifadesinin genel kullanım şekli aşağıdaki gibidir.

Try

İstisnai durum oluşturabilecek kod/kodlar

Catch

İstisnai durum oluştuğunda çalışacak kod/kodlar

Finally *(isteğe bağlı)*

İstisnai durum oluşsun veya oluşmasın çalışacak kod/kodlar

End Try

Şimdi başta yaptığımız örneği Try…Catch…Finally komutunu kullanarak tekrar yapalım.

```
Dim a As Integer
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)Handles Button1.Click
    Try                                          1
      a = TextBox1.Text
      If a = 55 Then
        MsgBox("HOŞGELDİNİZ")
      End If
    Catch                                        2
      MsgBox("HATALI BİLGİ")
    Finally                                      3
      MsgBox("TEŞEKKÜRLER...")
    End Try                                      4
  End Sub
```

Örnekte görüldüğü gibi istisnai durum oluşturabilecek komut satırları 1 ile 2 numaralı satırlar arasına yani **Try…Catch** deyimleri arasına, istisnai durum oluştuğunda çalıştırılacak komut veya komutlar 2 ile 3 numaralı satırlar arasına yani **Catch…Finally** deyimleri arasına yazılır. Finally komutu kullanılmazsa 3 nu.lı satır programda yazılmaz, komut veya komutlar 2 ile 4 nu.lı satırlar arasına yani C**atch…End Try** deyimleri arasına yazılır. İstisnai durum oluşsa da oluşmasa da çalışmasını istediğimiz komutlar 3 ile 4 nu.lı satırlar arasına yani **Finally…End Try** deyimleri arasına yazılır.

Programın çalışma mantığını kısaca açıklayacak olursak;

Resim 1.1'deki form oluşturulup program çalıştırdığımızda girilen bilgi 55 ise ekrana "HOŞGELDİNİZ" , 55'in dışında bir sayı ise "TEŞEKKÜRLER", string ise önce "HATALI BİLGİ" sonra "TEŞEKKÜRLER" yazdırılacaktır.

Not: Dikkat edilirse istisnai durum oluşsa da oluşmasa da ekrana "TEŞEKKÜRLER" bilgisi yazdırılır. Ayrıca istisnai durum oluştuğunda program kırılmaz çalışmasına devam eder.

1.2. Exception Türü

Exception tüm hata türlerini yakalamak ve yönetmek için kullanılan hata sınıfıdır. Bu sınıf içinde birçok hata türünü barındırır. Bu sınıfı kullanarak yakalanan hatalar bir değişkene aktarılıp ekrana yazdırılabilir. Ayrıca yeni istisnalar oluşturulabilir.

Kullanım şekli;

Catch değişken *As* Exception

Catch bloğu hata yakalanınca yapılacak işlemler listesinin hemen öncesinde bulunuyordu. Burada herhangi bir istisna yakalandığında bu istisna bilgisi değişkene aktarılıyor ve değişken üzerinden işlem yapılıyor. Ayrıca kullanılacak değişken de hata bilgisini tutmak için aşağıdaki şekilde tanımlanmalıdır.

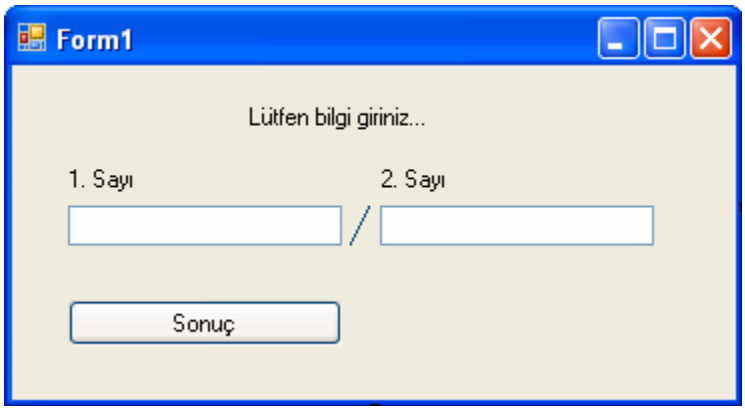
Dim değişken *As* Exception

Exception nesnesine ait özellik veya metotlar aşağıda listelenmiştir.

Özellik veya Metot	**Açıklama**
HelpLink	Hata hakkında detaylı bilgi veren URL bilgisi barındırır.
InnerException	Catch bloğuna yazdığımız kodda oluşacak hatalara erişmek için kullanılır.
Message	Kullanıcıyı bilgilendirmek üzere hata hakkında detaylı açıklama içerir.
Source	Hatayı oluşturan sınıf hakkında bilgi verir.
SatckTrace	Hatanın nerede oluştuğunun detaylı bilgisini içerir. Kaynak dosya ismi ve hatanın oluştuğu satır numarasını verir.
TargetSite	Güncel istisnayı yöneten metottur.

Örnek: Exception nesnesine ait özellikleri içeren program.

```
Public Class Form1
    Dim a, b As Integer
    Dim c As Double
    Dim e As Exception
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Try
            a = TextBox1.Text
            b = TextBox2.text
            c = a / b
            Label2.Text = c
        Catch e As Exception
            MessageBox.Show(e.Message)
            MessageBox.Show(e.Source)
            MessageBox.Show(e.StackTrace)
        End Try
    End Sub

End Class
```

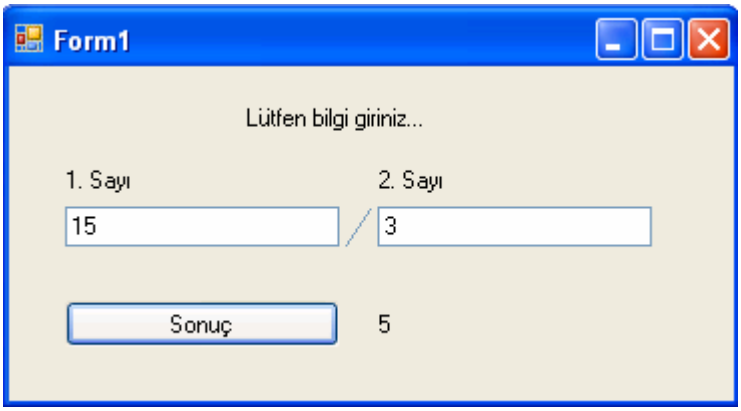


Resim 1.3: Program için hazırlanmış form

Örnekte resim 1.3'teki formdan girilen iki değerin bölümü sonuç butonuna basıldığında ekrana yazdırılır. Burada hem tanımlama hatası hem de sıfıra bölme hatası oluşabilir.

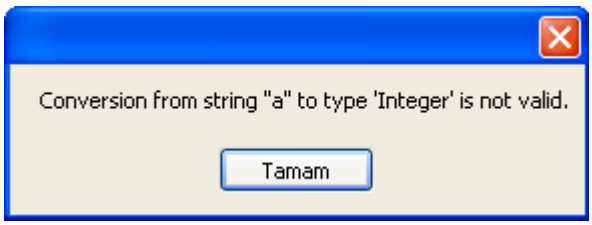
Textbox'lardan herhangi birine integer tipin dışında bilgi yazılırsa tanımlama hatası, 2. sayının yazılacağı Textbox'a sıfır yazılırsa sıfıra bölme hatası oluşur. Oluşan hatalar Exception komutu yardımı ile yakalanıp " e " değişkenine aktarılır. Message özelliği ile hata hakkında ayrıntılı bilgi, Source özelliği ile hatayı oluşturan sınıf hakkında bilgi ve StackTrace özelliği ile de hatanın oluştuğu yer ve kaynak dosya bilgileri ekrana sırasıyla aktarılır.

Programı çalıştırıp 1. sayıya 15, 2. sayıya 5 yazılıp sonuç butonuna basıldığında karşımıza resim 1.4'teki ekran çıkacaktır.

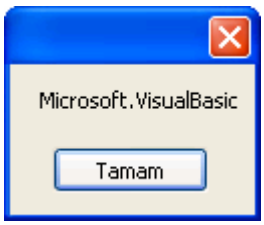


Resim 1.4: Formun işlem yapılmış hali

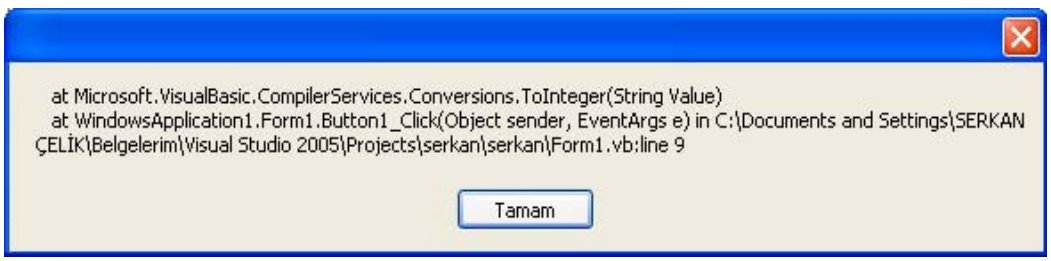
Programı çalıştırıp 1. sayıya 15, 2. sayıya a yazılıp sonuç butonuna basıldığında ise karşımıza sırasıyla aşağıdaki mesajlar çıkar.



Resim 1.5: Sıfıra bölme hatası uyarı mesajı



Resim 1.6: Hata kaynağı uyarı mesajı



Resim 1.7: Sıfıra bölme hatası ayrıntılı uyarı mesajı

- Resim 1.5'teki hata mesajında a bilgisinin integer tipte olmadığı belirtilmektedir. Bu mesajın görüntülenmesini sağlayan komut MessageBox.Show (e.Message) dur.
- Resim 1.6'da hatanın kaynağının Microsoft.VisualBasic olduğu belirtilmektedir. Bu mesajın görüntülenmesini sağlayan komut MessageBox.Show(e.Source) dur.
- Resim 1.7'de ise hata hakkında ayrıntılı bilgi verilmektedir. Burada hatanın hangi satırda oluştuğu belirtilmektedir. Bu mesajın görüntülenmesini sağlayan komut MessageBox.Show(e.StackTrace) dır.

1.3. Sık Görülen Hata Sınıfları

1.3.1. ArgumentException

Argument Exception (Bağımsız değişken hatası) bir metot çalıştırılırken en az bir bağımsız değişken söz konusu metodun parametre şartlarını karşılamadığında bu istisna oluşur.

ArgumentExcepion istisnasının başlıca iki alt sınıfı bulunur. Bunlar ArgumentNullException ve ArgumentoutOfRangeexception'larıdır. İki türemiş sınıfın kabul edilmediği durumlar dışında ArgumentException yerine bu sınıflar kullanılabilir. Bu istisnai durumlar aşağıda belirtilen durumlarda oluşur.

- ArgumentNullException; boşluğun geçerli bir bağımsız değişken olarak kabul edilmediği bir metoda geçildiğinde oluşur.

- ArgumentOutOfRange; bağımsız değişkenin değerinin kabul edilebilir sınırlar dışında olduğunda oluşur.

Örnek:

```
Public Class degisken
   Dim a As String
   Dim b As Bayt
   Dim hata As Exception
   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
      Try
         a = TextBox1.Text
         b = TextBox2.Text
         Label1.Name = a.CompareTo(b)
      Catch hata As ArgumentException
         MsgBox("Lütfen Bilgileri kontrol ediniz")
      End Try
   End Sub
End Class
```

Örnekte bir string bilgisi ile bayt bilgisi karşılaştırılmak isteniyor. Böyle bir durumda Argument Exception durumu oluşur. Burada CompareTo metodunun parametre şartlarından bir karşılanmıyor. Bunun sebebi karşılaştırılacak iki bilgininde string olmamasıdır.

1.3.2. DivideByZeroException

DivideByZeroException (Sıfıra bölme hatası) matematikte de karşımıza çıkan bir hatadır. Bu hata (istisna) bir sayının 0 (sıfır) a bölünmesi sonucu ortaya çıkar. Özellikle bölme işlemi olan programlarda, programın kırılmasını önlemek için bu hatanın yakalanıp yönetilmesi (Bölünen sıfır yapılırsa bu hata ortadan kaldırılır) gerekir.

Örnek:

```
Public Class bolme
   Dim a, b As Integer
   Dim c As Double
   Dim h, k As Exception
   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
      Try
         a = TextBox1.Text
         b = TextBox2.Text
         c = a / b
      Catch h As DivideByZeroException
         MsgBox("Sıfıra bölme hatası")
         b = 0
      Catch k As Exception
         MessageBox.Show(k.Message)
      End Try
   End Sub
      End Class
```

a değeri 10, b değeri ise sıfır girildiğinde sıfıra bölme hatası ortaya çıkar. Bu hatayı yakalamak için Catch h As DivideByZeroException komut satırı kullanılır. Hata oluştuğunda yakalanıp bölüneni (b değerini) sıfır yaparak bu hatayı ortadan kaldırmış oluruz.

```
Catch k As Exception
 MessageBox.Show(k.Message)
```

Komut satırları ise oluşacak diğer hataları yakalayıp ekrana aktarır.

1.3.3. OwerflowException

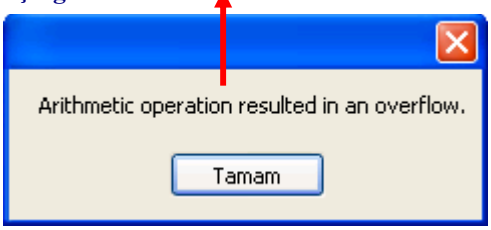
OwerflowException (taşma hatası) herhangi bir değişkene alabileceği değerden daha büyük bir değerin atanması sonucu oluşur. Her değişken tanımlandığı değişken türüne göre belli aralıklarda değer alabilir. Bu hatanın oluşabileceği programlarda hatanın yakalanıp programın kırılması önlenmelidir.

Örnek:

```
Public Class tasma
   Dim x, y, z As Short
   Dim hata As Exception
   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
      Try
         x = TextBox1.Text
         y = TextBox2.Text
         z = x * y
      Catch hata As OverflowException
         MessageBox.Show(hata.Message)
         MsgBox("Değişken değerini değiştirin")
      End Try
   End Sub
     End Class
```

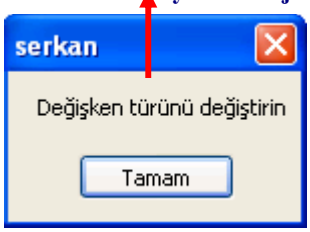
Örnekte x, y, z değişkenleri short olarak tanımlanmıştır. Short olarak tanımlanan değişkenler -32768 ile 32767 arasında değerler alabilir. Programı çalıştırıp x değişkenine 1000, y değişkenine 100 değeri atarsak 1000×100=100000>32767 olduğundan OverFlowException hatası meydana gelir. Çünkü z'nin alabileceği en büyük değer 32767 dir. Bu durumda karşımıza sırasıyla aşağıdaki mesajlar gelir.

Aritmetik işlem sonucunda taşma hatası oluştuğunu belirtir.



Resim 1.8: Taşma hatası uyarı mesajı

Hatanın oluşmaması için kullanıcı tarafından belirlenen uyarı mesajıdır.



Resim 1.9: Kullanıcı uyarı mesajı

Araştırma: Yukarıdaki örnekte y değişkenine -100 yazılırsa taşma hatası oluşur mu?

1.3.4. FieldAccessException

FieldAccesException (Alan girişi hatası) tipik olarak bir alanın bir sınıf kütüphanesindeki erişim seviyesi değişir ve kütüphaneyi temsil eden bir veya birden fazla derleyici ile yeniden derlenmezse oluşan bir istisnai durumdur. Kısaca bir sınıf kütüphanesine erişilemezse oluşan istisnai durumdur.

1.4. Throw New Komutu

İstisna fırlatmak için kullanılan komuttur. İstisna fırlatmak belli alanlarda kendi istisnalarımızı oluşturmaktır. Oluşturulan istisnalar "**Try Catch"** bloğu yardımı ile yakalanır. Bu işlem için "**Throw New"** anahtar kelimesi kullanılır.

Throw New Application.Exception ("Hata oluştu.") satırı yazılarak Application.Exception istisnai durumu oluşturulabilir.

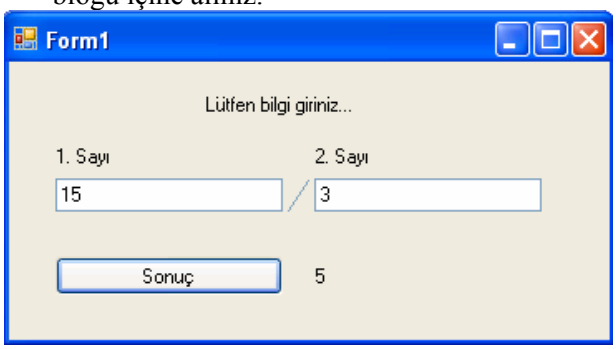
Örnek:

```
Public Class Form1
   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
      Try
         If TextBox1.Text = "100.yıl" Then                    İstisna(hata)
            Throw New Application.Exception("Yanlış bilgi girişi…")   fırlatma
         End If
      Catch ex As Exception
         MessageBox.Show(ex.Message)          İstisna yakalama
      End Try                                 Yönetme
   End Sub
End Class
```

Burada dikkat edilirse; hata oluşturacak bir program parçası bulunmamaktadır. Hatayı biz şarta bağlı olarak **"Throw New"** komutu yardımıyla oluşturup daha sonra yakalıyoruz. Normal şartlarda sadece exception kütüphanesinde bulunan hatalar yakalanabilir. Burada Exception kütüphanesinin dışında istisnai durumlar oluşturulup yönetilebilir.

Araştırma: İstisna fırlatılmasının sebebini araştırınız.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Bölme işlemi yapan bir program yazınız. Programda hata oluşturabilecek satırları Try bloğu içersine alınız. Hata oluştuğunda çalışacak komutları Cath bloğu içine alınız. 	Program için bir form oluşturulursa çalışması daha iyi anlaşılacaktır. İşlem yapıldığında hata oluşursa kullanıcı hatayı rahatlıkla düzeltip programa devam edebilir.
2. Hatadan bağımsız çalıştırılacak komutlar için Finally bloğu oluşturunuz.	
3. Hata oluşunca messagebox komutu ile ekrana mesaj yazdırınız.	Ekrana "Bölünen değeri yanlış girilmiş" uyarı mesajı yazdırınız.
4. Throw New komutu kullanılarak hata oluşturunuz.	Kendi oluşturduğunuz hatayı Catch bloğu ile yakalayıp işlem yapınız.

ÖLÇME VE DEĞERLENDİRME

A- OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; sonunda parantez olanlar doğru / yanlış sorularıdır. Verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız.

1. İstisnanın diğer ismi hatadır. ( )

2. Hata oluşabilecek komutlar Try bloğu içine alınır. ( )

3. Catch bloğunun kullanımı isteğe bağlıdır. ( )

4. Finally komutu olmadan da hatalar yakalanabilir. ( )

5. Source metodu hata kaynağını belirler.( )

6. Messagebox komutu ekrana mesaj yazdırır. ( )

7. Throw New istisna fırlatma komutudur. ( )

8. Throw new ile fırlatılan istisnalar Catch bloğu ile yakalanamaz. ( )

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

Hatasız bir şekilde yapı ve sınıflar kütüphaneleri ile çalışabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde yapmanız gereken araştırmalar şunlardır.

1. Yapı ve sınıf kütüphanelerinde hangi veri türlerinin saklandığını araştırınız.
2. Veri dönüşümü nasıl yapılır? Araştırınız.
3. Veri dönüşüm komutlarını araştırınız.

Araştırma işlemleri için internet ortamı ve görsel programlama dilini anlatan kaynak kitaplardan faydalanınız.

2. YAPI VE SINIFLAR

Kullanacağımız yapı (structure) ve sınıflar (class) genel olarak tüm .Net programlarında bulunmaktadır. Visual Basic 6.0 veya daha önceki versiyonları kullananlar bu yapılardaki farkları daha iyi anlayacaklardır. Yapılar **System** Namespace (Adalanı) içinde yer alır. Sırasıyla bu veri yapılarını inceleyelim.

2.1. Yapılar

Tüm programlama dillerinde bilgilerin saklandığı yapılar mevcuttur. Her yapının saklayacağı bilgi ve buna bağlı olarak hafızada kapladığı alan farklıdır.

2.1.1. Bayt Yapısı

8 bitlik işaretsiz tam sayıları temsil eder.

2.1.2. Int16 Yapısı

16 bitlik işaretli tam sayıları temsil eder.

2.1.3. Int32 Yapısı

32 bitlik işaretli tam sayıları temsil eder.

2.1.4. Int64 Yapısı

64 bitlik işaretli tam sayıları temsil eder.

2.1.5. Single Yapısı

32 bitlik işaretli ondalıklı sayıları temsil eder.

2.1.6. Double Yapısı

64 bitlik işaretli ondalıklı sayıları temsil eder.

2.1.7. Boolean Yapısı

16 bitlik veri olmasına rağmen sadece true ve false değerlerini temsil eder.

2.1.8. Char Yapısı

16 bitlik karakter temsil eder.

2.1.9. Decimal Yapısı

96 bitlik sayı temsil eder.

2.1.10. Date Time Yapısı

64 bitlik tarih ve saat bilgisini temsil eder.

 **Veri yapılarına ait genel metotlar aşağıdaki tabloda listelenmiştir.**

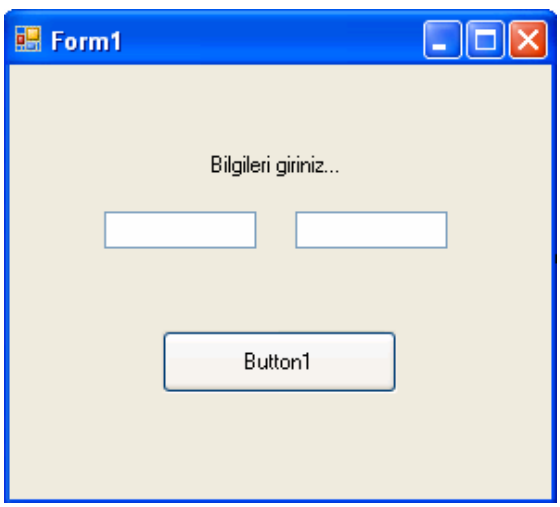
İsim	Tanım
CompareTo	Bir yapıyı kendisi veya belli bir nesne ile karşılaştırır. Sonuç eşit ise “0”, büyükse “+1”, küçükse”-1” değerlerini geri döndürür.
Equals	Bir yapının bir nesneye veya kendisiyle aynı yapıdaki veriye eşit olup olmadığını kontrol eder. Sonuçta true veya false değeri geri döndürür.
GetHashCode	Yapılardaki hata kodunu geri bildirir.
GetType	Mevcut tapının türünü alır.
GetTypeCode	Yapılar için Typecode (tür kodunu) u geri bildirir.
Parse	Bir dizinin istediğimiz herhangi bir elemanının değerini istenilen türe dönüştürüp geri döndürür.
ReferenceEquals	Belirli nesne örneklerinin aynı olup olamadığına karar verir
ToString	Yapının rakamsal değerini eş değer string gösterime dönüştürür.
TryParse	Bir dizinin istediğimiz herhangi bir elemanının değerini istenilen türe dönüştürüp dönüşümün başarılı veya başarısız olduğunu boolean türünde geri döndürür.

Veri türleri ve tür numaraları tabloda verilmiştir.

TypeName	GetType(tür)
Boolean	3
Bayt	6
Date	16
Decimal	15
Double	14
Integer	9
Long	11
Short	7
Single	13
String	18

Veri yapılarına ait metotların program içerisindeki kullanımını bir örnekle inceleyelim.

Örnek:

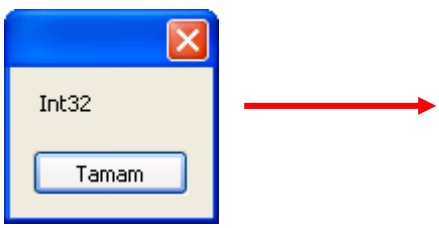


Resim 2.1: Veri yapı metotları örnek form

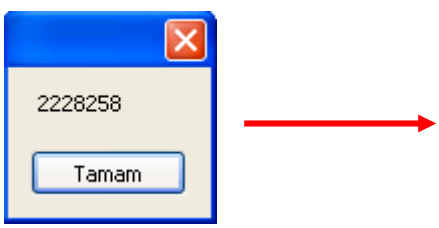
```
Public Class Form1
    Dim a As Integer
    Dim b As Short
    Dim di(5) As Integer

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)Handles Button1.Click
        a = TextBox1.Text
```

```
        b = TextBox2.Text
        di(0) = 1000
        di(1) = 2
        di(2) = 3
        di(3) = 4
        di(4) = 5
        MessageBox.Show(a.GetTypeCode.ToString)
        MessageBox.Show(b.GetHashCode)
        MessageBox.Show(a.ToString)
        MessageBox.Show(a.CompareTo(25))
        MessageBox.Show(a.Equals(b))
        MessageBox.Show((Bayt.TryParse(di(0), 13)))
        MessageBox.Show((Integer.Parse(di(4))))
    End Sub
End Class
```

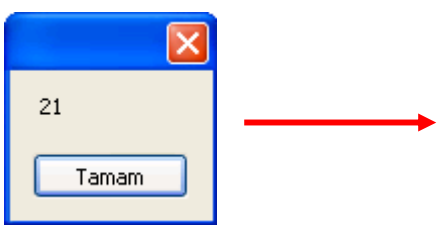
Yukarıdaki örnekte TextBox1'deki bilgi ile Textbox2'deki bilgiler a ve b değişkenine aktarılmıştır. Program çalıştığında sırasıyla aşağıdaki mesajlar ekranda görünecektir.



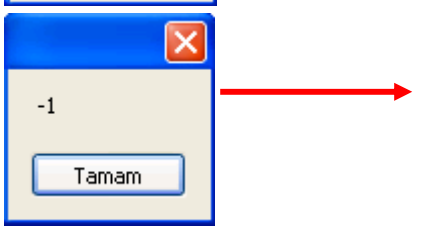
MessageBox.Show(a.GetTypeCode.ToString)
Tip kodunu stringe çevirir.



MessageBox.Show(b.GetHashCode)
Olaya uygun bir hata kodu üretir



MessageBox.Show(a.ToString)
Veriyi stringe çevirir.



MessageBox.Show(a.CompareTo(25)
TextBox1 deki bilgi ile 25 bilgisini karşılaştırır. Eşit ise 0, küçükse -1 değeri döndürür.

MessageBox.Show(a.Equals(b)))
TextBox1 deki bilgi ile TextBox2 deki bilgiyi karşılaştırır. Eşit ise true, eşit değil ise false değeri döndürür.

MessageBox.Show((Bayt.TryParse(di(0), 13)))
Dizinin 0. elemanını Bayt türüne dönüştürür. Dönüşüm sağlanmışsa True sağlanmamışsa False değeri döndürür. Burada 1000 bayt türüne dönüşmediğinden false değeri döndürmüştür.

MessageBox.Show((Integer.Parse(di(4))))
Dizinin 4. elemanını Integer türüne dönüştürüp ekrana aktarır

Resim 2.2: Veri yapıları örnek mesajlar

2.2. Sınıflar

Visual Basic.net ortamındaki tüm komutlar belirli sınıflarda tanımlanmış ve o sınıfın bir üyesi olmuştur. Object ve string sınıfları system adalanının birer üyesidir.

2.2.1 Object Sınıfı

.Net sınıf hiyerarşisinin altındaki tüm sınıfları destekler ve türemiş sınıflara düşük seviyeli hizmetler sağlar. Object sınıfı .Net çatısı altındaki tüm sınıfların nihai temel sınıfıdır; tür hiyerarşisinin temelidir (köküdür). Object sınıfı ile tanımlanmış bir değişkene her türde bilgi kaydedilebilir. Object sınıfının metotları tüm sınıflar tarafından kullanılabilir.

Dim *Değişken İsmi* As Object

Bir object nesnesi ile tanımlanan değişkene her türlü bilgi kaydedilebilir.

Aşağıdaki tabloda object sınıfının üyeleri listelenmiştir.

Genel metotlar

İsim	Tanım
Equals	İki nesne örneğinin eşit olup olmadığını belirtir.
GetHashCode	Özel bir türün bozuk kodunu bulur.
GetType	Mevcut örneğin türünü elde eder.
ReferenceEquals	Belirli nesne örneklerinin referans türe eşit olup olmadığına karar verir.
ToString	Mevcut nesneyi temsil eden bir dizeyi geri döndürür.

Yardımcı metotlar

İsim	Tanım
Finalize	Bir nesnenin serbest kaynakları denemesine ve çöp koleksiyonuna geri dönmeden önce diğer temizleme işlemlerinin gerçekleşmesine izin verir.
MemberwiseClone	Mevcut nesnenin basit bir kopyasını oluşturur.

Örnek: Object metotlarının kullanımı

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    Dim obj1, obj2 As Object
    obj1 = 5
    obj2 = "ali"
    MsgBox(obj1.Equals(obj2))
    MsgBox(obj1.ToString)
    MsgBox(Object.ReferenceEquals(obj1, 5))
  End Sub
```

Programı çalıştırıldığında sırasıyla **false** bilgisi (iki nesne eşit değil), **5** (1. nesnenin string türde değeri) ve **false** (1. nesne türü referans türle aynı değil) ifadeleri ekranda görülür.

2.2.2. String

String sınıfı karakterleri temsil eden bir kütüphanedir. Bu sınıftaki üyeler yardımıyla stringlerle ilgili tüm işlemler (birleştirme, karşılaştırma, büyük küçük harf dönüşümü, karakter ekleme karakter silme vb.) yapılabilir.

Aşağıdaki tablolarda string sınıfına ait üyeler ve görevleri listelenmiştir.

String sınıfının genel özellikleri

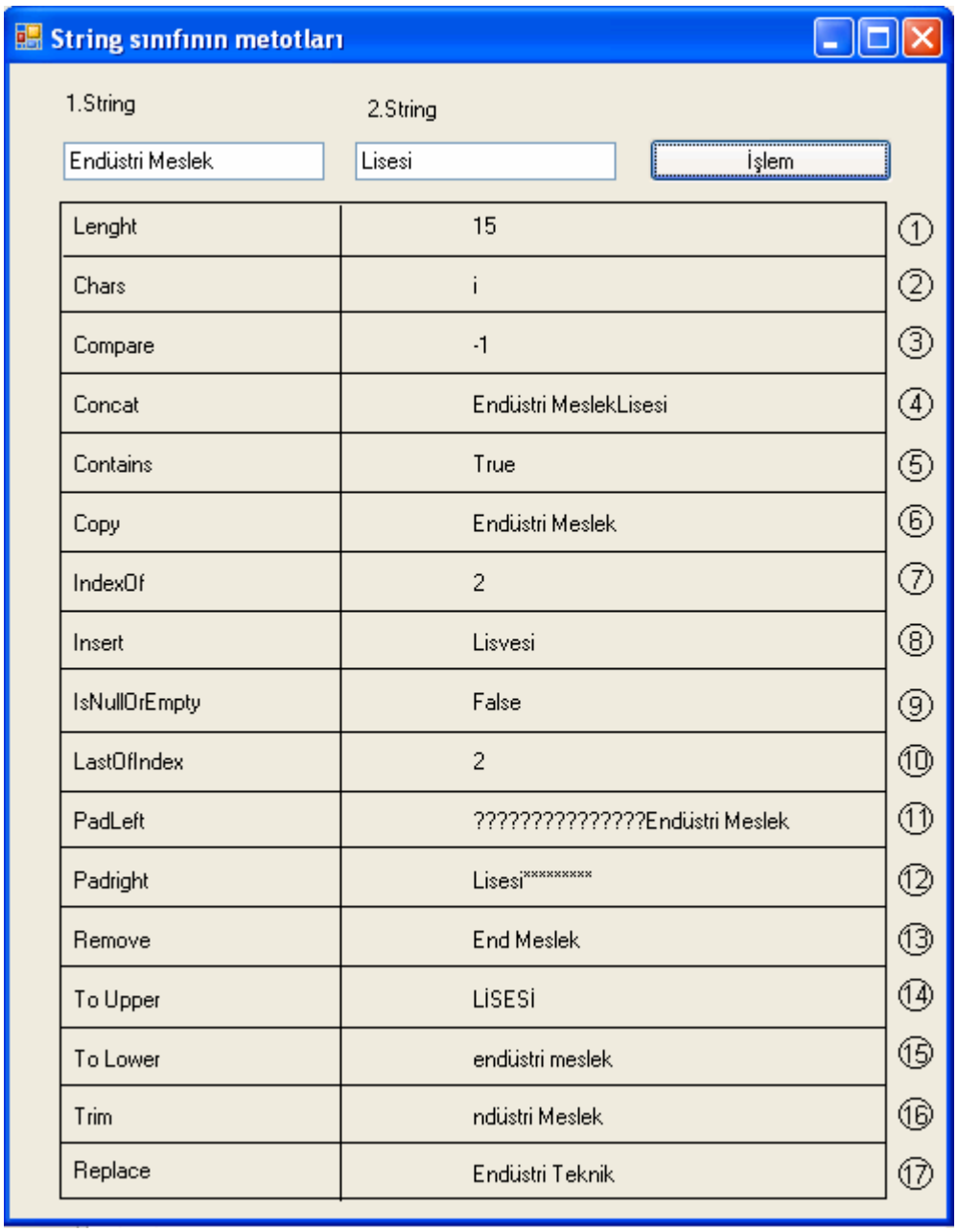
Name	Description
Chars	Belirlenmiş bir karakterin yerine karakter ekler.
Length	Stringdeki karakter sayısını verir.

String sınıfının sık kullanılan metotları

Name	Description
Compare	Belirlenmiş iki string nesnesini karakter sayılarına göre karşılaştırır. Sonuç eşit ise "0", büyükse "+1", küçükse"-1" değerlerini döndürür.
CompareTo	Belirlenen iki stringi alfabetik sıraya göre karşılaştır. Sonuç eşit ise "0", eşit değil ise "-1" değeri geri döndürür.
Concat	Stringi bir veya birden fazla string ile birleştirir.
Contains	String içerisinde belirlenmiş başka string nesnelerinin bulunup bulunmadığını kontrol eder.
Copy	Belirlenmiş bir string karakterlerini kullanarak yeni bir örnek string oluşturur.
EndsWith	Stringin sonunda belirli bir string örneğinin olup olmadığına karar verir.
Equals	İki string nesnesinin aynı değerlere sahip olup olmadığı kontrol eder. True veya false değerleri geri döndürür.
Format	Stringi bilgiyi belli bir biçime dönüştürür.
GetEnumerator	String içerisinde tekrar eden bir nesne bulur.
GetHashCode	Stringdeki bozuk kodu bulur.
GetType	Örneğin tipini verir.
GetTypeCode	Stringin Type code (Tür kodunu) çağırır.
IndexOf	Stringin içindeki dizenin başlangıç noktasını bulur.
IndexOfAny	Stringdeki herhangi bir karakterin başlangıç noktasını bulur.
Insert	String içerisine karakter ekler.
IsNullOrEmpty	Stringin boş olup olmadığını gösterir.
LastIndexOf	Bir karakter veya karakter grubunun stringin içinde bulunduğu pozisyonu sayısal olarak ifade eder. (Eğer karakter veya karakter grubu string içerisinde birden fazla ise en sondaki karakter veya karakter grubunun sayısal olarak bulunduğu yeri belirtir.)
PadLeft	Stringin solundan başlayarak stringin boyutu verilen değere eşit olana kadar istenilen karakter eklenir.
PadRight	Stringin sağından başlayarak stringin boyutu verilen değere eşit olana kadar istenilen karakter eklenir.
ReferenceEquals	Belirlenmiş nesne örneklerinin referans türle aynı olup olmadığına karar verir.
Remove	String içerisinde belirli sayıda karakter siler.
Replace	Stringin istediğimiz karakterlerini değiştirmek için kullanılır.
StartsWith	Belirlenen bir dizenin stringin başlangıcı olup olmadığını bulur. Boolean tipte bir değer döndürür.
Substring	Stringde belirtilen bir karakterden başlayarak sabit sayıda karakter döndürür.
ToCharArray	Stringi dizi karakterlerine çevirir.
ToLower	Stringdeki harfleri küçük harfe dönüştürür.
ToString	Bilgileri string türüne dönüştürür.
ToUpper	Stringdeki harfleri büyük harfe dönüştürür.
Trim	Stringin başında ve sonundaki boşlukları veya belirtilen karakterleri kaldırır.
TrimEnd	Stringin sonundaki boşlukları veya belirtilen karakterleri kaldırır.
TrimStart	Stringin başındaki boşlukları veya belirtilen karakterleri kaldırır.

Örnek:

```
Public Class Form1

    Dim a, b As String

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)Handles Button1.Click
        a = TextBox1.Text
        b = TextBox2.Text
        Label18.Text = a.Length
        Label19.Text = b.Chars(1)
        Label20.Text = a.CompareTo(b)
        Label21.Text = String.Concat(a, b)
        Label22.Text = a.Contains("eslek")
        Label23.Text = String.Copy(a)
        Label24.Text = a.IndexOf("düstri")
        Label25.Text = b.Insert(3, "v")
        Label26.Text = String.IsNullOrEmpty(b)
        Label27.Text = a.LastIndexOf("d")
        Label28.Text = a.PadLeft(30, "?")
        Label29.Text = b.PadRight(15, "*")
        Label30.Text = a.Remove(3, 5)
        Label31.Text = b.ToUpper
        Label32.Text = a.ToLower
        Label33.Text = a.Trim("En")
        Label36.Text = a.Replace("Meslek", "Teknik")

    End Sub

End Class
```



Resim 2.3: String sınıfının metotları örnek form penceresi

Numaralandırılmış satırları sırasıyla inceleyelim, böylece string sınıfı metotlarının kullanımını daha iyi anlamış olacağız.

1. a.Length = 1. stringin karakter uzunluğunu verir. Uzunluk bulunurken boşluklar dâhil edilir.

2. b.Chars(1) = 2 . stringin 1. elemanını ekrana yazdırır. Dikkat edilecek husus stringin ilk karakteri 0. elemanıdır.

3. a.CompareTo(b) = 1 . string ile 2. stringi karşılaştırır. İki string eşit -1 değeri döndürür.

4. String.Concat(a, b) = 1 . stringin sonuna 2. stringi ekler. 1. stringin sonunda veya 2. stringin başında boşluk olmadığından stringler birleştirildiğinde arada boşluk olmaz.

5. a.Contains("eslek") = 1 . string içerisinde "eslek" kelimesi olduğundan true değeri döndürür.

6. String.Copy(a) = 1. stringin kopyasını oluşturur.

7. a.IndexOf("düstri") = "dustri" stringinin 1. stringin kaçıncı karakterinden itibaren başladığını gösterir. stringin ilk elemanı 0. karakterdir.

8. b.Insert(3, "v") = 2. stringin 3. karakterine v harfini ekler.

9. String.IsNullOrEmpty(b) = 2. Stringin boş olup olmadığını kontrol eder. Boş olmadığı için False değeri döndürür.

10. a.LastIndexOf("d") = d harfinin 1. stringin kaçıncı karakteri olduğunu bulur.

11. a.PadLeft(30, "?") = 1 . stringin sol tarafına toplam uzunluk 30 karakter olana kadar "?" işareti ekler.

12. b.PadRight(15, "*") = 2 . stringin sağ tarafına toplam uzunluğu 15 karakter olana kadar "*" işareti ekler.

13. a.Remove(3, 5) = 1 . stringin 3. karakterinden başlayarak 5 karakter siler.

14. b.ToUpper = 2 . stringin bütün karakterlerini büyük harfe çevirir.

15. a.ToLower = 1 . stringin bütün karakterlerini küçük harfe çevirir.

16. a.Trim("En") = 1 . stringin başında veya sonunda En kelimesi varsa bulup siler.

17. a.Replace("Meslek", "Teknik") = 1 . string içerisinde "Meslek" kelimesini arayıp bulduğunda "Teknik" kelimesi ile yer değiştirir.

2.3. Veri Türleri

Tüm programlama dillerinde standart veri türleri mevcuttur. Bunlar tam sayı veri türleri, ondalıklı veri türleri ve stringler olarak ayrılır. String veri türü .Net programlama dilinde ayrı bir sınıf olarak oluşturulmuştur. Bu yüzden bir önceki konularda ayrıntılı olarak incelenmiştir.

2.3.1. Bayt (Bayt) Veri Türü

1 baytlık (hafızada kapladığı alan) işaretsiz tam sayı tipidir. 0-255 arasında bir değer alabilir. System adalanının içerisinde bulunur. Veri türünün .NET altındaki karşılığı System.Bayt'tır.

Kullanımı:

Dim *değişken_ismi* As **Bayt**

2.3.2. Short (Int16) Veri Türü

2 baytlık işaretli tam sayı tipidir. -32768 ile 32767 arasında bir değer alabilir. System adalanının içerisinde bulunur. Veri türünün .NET altındaki karşılığı System.Int16'dır.

Kullanımı:

Dim *değişken_ ismi* As **short**

2.3.3. Integer (Int32) Veri Türü

4 baytlık işaretli tam sayı tipidir. -2,147,483,648 ile 2,147,483,6487 arasında bir değer alabilir. Sistem adalanının içerisinde bulunur. Veri türünün .NET altındaki karşılığı System.Int32'dir. Değişken sonuna % işareti konularak da Integer veri türü tanımlanabilir.

Kullanımı:

Dim *değişken_ ismi* As **Integer**
Dim *değişken_ ismi* **%**

2.3.4. Long (Int64) Veri Türü

8 baytlık işaretli tam sayı tipidir. -9,223,372,036,854,775,808 ile 9,223,372,036,854,775,807 arasında bir değer alabilir. Sistem adalanının içerisinde bulunur. Veri türünün .NET altındaki karşılığı System.Int64'tür. Değişken sonuna & işareti konularak da Long veri türü tanımlanabilir.

Kullanımı:

Dim *değişken_ ismi* As **Long**
Dim *değişken_ ismi* **&**

2.3.5. Single Veri Türü

4 baytlık işaretli ondalıklı sayı tipidir. $\pm 3.402823 \times 10^{38}$ ile $\pm 1.401298 \times 10^{-45}$ arasında bir değer alabilir. Sistem adalanının içerisinde bulunur. Veri türünün .NET altındaki karşılığı System.Single'dır. Değişken sonuna "!" işareti konularak da Single veri türü tanımlanabilir.

Kullanımı:

Dim *değişken_ ismi* As **single**
Dim *değişken_ ismi* **!**

2.3.6. Double Veri Türü

8 baytlık işaretli ondalıklı sayı tipidir. $\pm 1.79769313486232 \times 10^{308}$ ile $\pm 4.94065645841247 \times 10^{-324}$ arasında bir değer alabilir. Sistem adalanının içerisinde bulunur. Veri türünün .NET altındaki karşılığı System.Double'dır. Değişken sonuna # işareti konularak da Double veri türü tanımlanabilir.

Kullanımı:

Dim *değişken_ ismi* As **double**
Dim *değişken_ ismi* **#** =1.25478744

2.3.7. Boolean Veri Türü

2 baytlık bir veri türüdür. Sadece true (doğru) ve false (yanlış) değerlerini alabilir. Bu tür değişkenlere true ve false değerlerinin yanında 0 veya 1 de atanabilir. 0 false 1 true olarak kabul edilir. Sistem adalanının içerisinde bulunur. Veri türünün .NET altındaki karşılığı System.Boolean'dir.

Kullanımı:

Dim *değişken_ ismi* As **boolean**

2.3.8. Char Veri Türü

2 baytlık bir veri türüdür. Tek bir karakter bilgisi saklayabilir. 0 ile 65535 arasındaki herhangi bir Unicode (evrensel kod) içerir. Sistem adalanının içerisinde bulunur. Veri türünün .NET altındaki karşılığı System.Char'dir.

Kullanımı:

Dim *değişken_ ismi* As **char**

2.3.9. Decimal veri türü

12 baytlık veri türüdür. ±79,228,162,514,264,337,593,543,950,335 arasındaki tam sayıları ve ±7.9228162514264337593543950335 arasındaki ondalıklı sayılardan herhangi bir değer alabilir. 28 başamaklı bir sayıyı saklayabilir. Bu veri türünün alabileceği en düşük değer ±0.0000000000000000000000000001'dir. Sistem adalanının içerisinde bulunur. Veri türünün .NET altındaki karşılığı System. Decimal'dir. Değişken sonuna @ işareti konularak Decimal veri türü tanımlanabilir.

Kullanımı:

Dim *değişken_ ismi* As **Decimal**

Const *değişken_ ismi* **@** =3.75

2.3.10. Date Veri Türü

8 baytlık veri türüdür. Tarih v saat bilgilerini saklayabilir. Bu veri türüne 1/1/100 ile 31/12/9999 arasındaki tarih ve 0:00:00 ile 23:59:59 arasındaki saat bilgilerinden bir değer alabilir. Tarih ve saat bilgileri "… " veya # … # ifadeleri arasında değişkenlere atanmalıdır. Sistem adalanının içerisinde bulunur. Veri türünün .NET altındaki karşılığı System.DateTime dir.

Kullanımı:

Dim *değişken_ ismi* As **Date**

Dim *değişken_ ismi* As **Time**

Dim *değişken_ ismi* As **Date = # 15/07/2007#**

Dim *değişken_ ismi* As **Time = "23:27:00"**

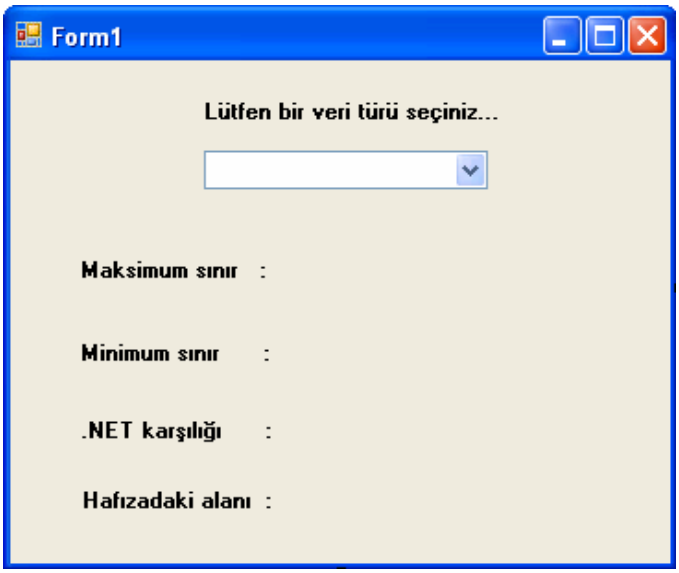
Açıklama: *Int16,Int32,Int64 veri türleri önüne U harfi getirilerek kapsamı değiştirilmeden işaretsiz veri türüne dönüştürülebilir (UInt16,UInt32,UInt64). Ayrıca Bayt veri türünün önüne S harfi getirilerek işaretli hale dönüştürülebilir (SBayt).*

Veri türü	Sınırları
UInt16	0 ile 65535
UInt32	0 ile 4,294,967,295
UInt64	0 ile 18,446,744,073,709,551,615
SBayt	-128 ile +127

Aşağıdaki tabloda tüm veri türleri hakkında özet bilgi verilmiştir.

Veri türü	Kapladığı alan (Bayt)	Açıklama	.NET karşılığı
Bayt	1	8-bit işaretsiz tam sayı	System.Bayt
Char	2	16-bit Unicode (Evrensel) karakterler	System.Char
Integer	4	32-bit işaretli tam sayı	System.Int32
Double	8	64-bit ondalıklı sayı	System.Double
Long	8	64-bit işaretli tam sayı	System.Int64
Short	2	16-bit işaretli tam sayı	System.Int16
Single	4	32-bit ondalıklı sayı	System.Single
Date	8	Tarih ve saat bilgisi	System.DateTime
Boolean	2	Numerik olmayan tip	System.Boolean
Decimal	16	128-bit ondalıklı sayı	System.Decimal

Örnek: Combobox ile seçilen herhangi bir veri türünün kapladığı alan, .Net karşılığı ve alabileceği değerlerin sınırlarını kullanıcıya aktaran bir form oluşturup buna uygun bir program yazalım.



Resim 2.4: Veri türleri örnek program formu

```
Public Class Form1
    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
        If ComboBox1.SelectedIndex = 0 Then
            Label5.Text = "0"
            Label6.Text = "255"
```

```
    Label7.Text = "System.Bayt"
    Label8.Text = "1"
End If
If ComboBox1.SelectedIndex = 1 Then
    Label5.Text = "ASCII kodu 0 olan karakter"
    Label6.Text = "ASCII kodu 255 olan karakter"
    Label7.Text = "System.char"
    Label8.Text = "2"
End If
If ComboBox1.SelectedIndex = 2 Then
    Label5.Text = "-2,147,483,648 "
    Label6.Text = "+2,147,483,647 "
    Label7.Text = "System.Int32"
    Label8.Text = "4"
End If
If ComboBox1.SelectedIndex = 3 Then
    Label5.Text = "±1.79769313486232×10308"
    Label6.Text = "±4.94065645841247×10-324"
    Label7.Text = "System.Double"
    Label8.Text = "8"
End If
If ComboBox1.SelectedIndex = 4 Then
    Label5.Text = "-9,223,372,036,854,775,808"
    Label6.Text = "+9,223,372,036,854,775,807"
    Label7.Text = "System.Int64"
    Label8.Text = "8"
End If
If ComboBox1.SelectedIndex = 5 Then
    Label5.Text = "-32768"
    Label6.Text = "+32767"
    Label7.Text = "System.Int16"
    Label8.Text = "2"
End If
If ComboBox1.SelectedIndex = 6 Then
    Label5.Text = "±3.402823×1038"
    Label6.Text = "±1.401298×10-45"
    Label7.Text = "System.Single"
    Label8.Text = "4"
End If
```

```
        If ComboBox1.SelectedIndex = 7 Then
            Label5.Text = "1/1/100 - 0:00:00"
            Label6.Text = "31/12/9999 - 23:59:59"
            Label7.Text = "System.DateTime"
            Label8.Text = "8"
        End If
        If ComboBox1.SelectedIndex = 8 Then
            Label5.Text = "±0.0000000000000000000000000001"
            Label6.Text = "±7.9228162514264337593543950335"
            Label7.Text = "System.Decimal"
            Label8.Text = "16"
        End If
        If ComboBox1.SelectedIndex = 9 Then
            Label5.Text = "0(False)"
            Label6.Text = "1(True)"
            Label7.Text = "System.Boolean"
            Label8.Text = "2"
        End If
    End Sub
Private Sub Form1_Activated(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Activated
        ComboBox1.Items.Add("Bayt")
        ComboBox1.Items.Add("Char")
        ComboBox1.Items.Add("Integer")
        ComboBox1.Items.Add("Double")
        ComboBox1.Items.Add("Long")
        ComboBox1.Items.Add("Short")
        ComboBox1.Items.Add("Single")
        ComboBox1.Items.Add("Date")
        ComboBox1.Items.Add("Decimal")
        ComboBox1.Items.Add("Boolean")
    End Sub
End Class
```

Araştırınız: Tek bir veri türü ile tüm veri türlerinin görevlerini yerine getirmenin dezavantajları nelerdir?

2.4. Otomatik Veri Dönüşümleri (Widening Conversions)

Belirli kurallar (Tüm veri türleri birbirine güvenli olarak dönüştürülemez) dâhilinde veri türleri birbirine dönüştürülebilir. Tür dönüşümü ile değişkenlerin içeriği genişletiliyorsa, sınırları ve kapladığı alan artıyorsa bu tip dönüşümlere widenning conversions (genişletme dönüşümleri) denir. Bir veri türünün diğer bir veri türüne kolaylıkla dönüştürülmesi için **Option Strict** değeri **on** olmalıdır. Hangi veri türleri arasında güvenli dönüşüm yapabileceği aşağıdaki tabloda gösterilmiştir.

Veri Türü	**Veri kaybı olmadan güvenli dönüşüm yapılacak tür**
Bayt	UInt16, Int16, UInt32, Int32, UInt64, Int64, Single, Double, Decimal
SBayt	Int16, Int32, Int64, Single, Double, Decimal
Int16	Int32, Int64, Single, Double, Decimal
UInt16	UInt32, Int32, UInt64, Int64, Single, Double, Decimal
Char	UInt16, UInt32, Int32, UInt64, Int64, Single, Double, Decimal
Int32	Int64, Double, Decimal
UInt32	Int64, Double, Decimal
Int64	Decimal
UInt64	Decimal
Single	Double

Bazı türlerin single veya double türüne dönüşümlerinde kesinlikle veri kaybı oluşur. Bu türlere ait tablo aşağıda verilmiştir.

Veri türü	**Dönüştürülebilecek tür**
Int32	Single
UInt32	Single
Int64	Single, Double
UInt64	Single, Double
Decimal	Single, Double

2.5. Ters Yöndeki Dönüşümler (Narrowing Conversions)

Ters yönde dönüşüm veri sınırları büyük olan bir türden daha küçük veri sınırlarına sahip bir türe dönüşümdür. Başka bir deyişle genişletme dönüşümlerinin tersidir. Bu işlem sonucunda daralma (veri kaybı) meydana gelir. Hangi veri türlerinin ters yönde veri dönüşümü yapabileceği aşağıdaki tabloda verilmiştir. **Option scrict** değeri **off** durumunda ise ters dönüşüme izin verilmez. Ters yönde veri dönüşümü yapıldığında genellikle veri daralması meydana gelir. Tür dönüşümünde System.convert sınıfı kullanılırsa dönüşüm yapılacak veri türünün maksimum ve minimum sınırları hedef veri türünün sınırlarından büyükse ve bu durum kontrol edilmek isteniyorsa OwerFlowException istisnası fırlatılarak kullanıcı bu durumdan haberdar edilir.

Tür	**Dönüşüm yapılabilecek tür**
Bayt	SBayt
SBayt	Bayt, UInt16, UInt32, UInt64
Int16	Bayt, SBayt, UInt16
UInt16	Bayt, SBayt, Int16
Int32	Bayt, SBayt, Int16, UInt16, UInt32
UInt32	Bayt, SBayt, Int16, UInt16, Int32
Int64	Bayt, SBayt, Int16, UInt16, Int32, UInt32, UInt64
UInt64	Bayt, SBayt, Int16, UInt16, Int32, UInt32, Int64
Decimal	Bayt, SBayt, Int16, UInt16, Int32, UInt32, Int64, UInt64
Single	Bayt, SBayt, Int16, UInt16, Int32, UInt32, Int64, UInt64
Double	Bayt, SBayt, Int16, UInt16, Int32, UInt32, Int64, UInt64

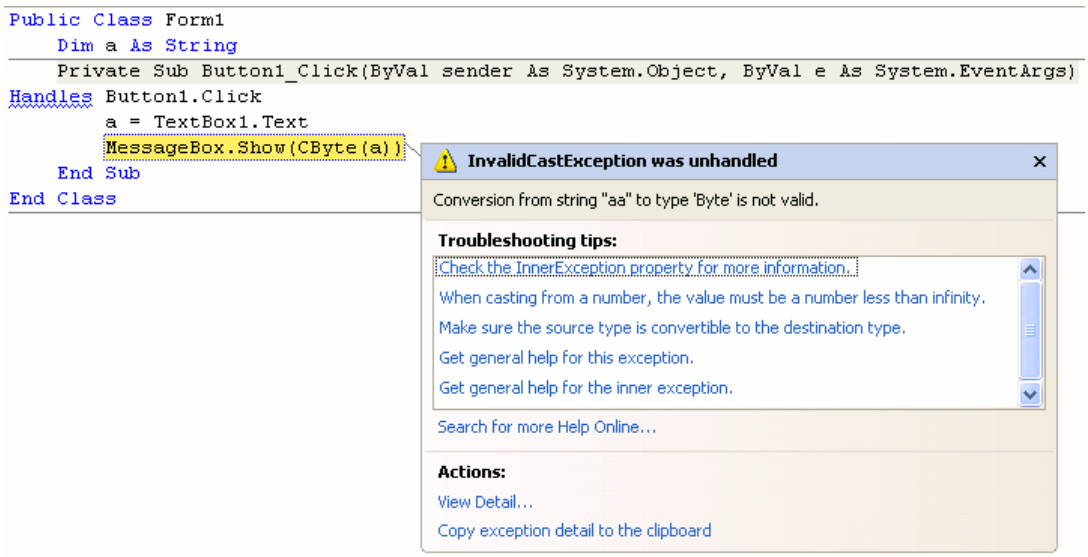
2.6. InvalidcastException Hatası

InvalidcastException (tip uyuşmazlığı) hatası genellikle rakamsal bilgilerle string bilgilerin karıştırılması sonucu ortaya çıkar. Örnek olarak string tipte tanımlanmış bir değişkeni bayt veri türüne dönüştürmek istediğimizde bu hata meydana gelir.

Örnek1: Bu hata türünün nasıl oluştuğunu anlamak için bir örnek program yapalım.

```
Public Class Form1
   Dim a As String
   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
Handles Button1.Click
      a = TextBox1.Text
      MessageBox.Show(CBayt(a))
   End Sub
      End
```

Programa dikkat edilirse a değişkeni string tipinde (metin) tanımlanmıştır. Bu nedenle değişkene aktarılacak bilgi karakterlerden oluşmalıdır. Programı çalıştırıp Textbox1 kutusuna "123a" bilgisi yazılıp button1 tıklanırsa karşımıza aşağıdaki hata çıkacaktır. Bunun sebebi CBayt komutu bayt veri türüne dönüşüm için kullanılır. Fakat a bilgisi string türde olduğu için dönüşüm yapılamaz ve InvalidCastException hatası oluşur. Program bu hata nedeniyle durdurulur. Bu hata, dönüşümü mümkün olmayan veri türleri arasında dönüşüm yapılmak istendiğinde veya yanlış dönüşüm komutu kullanıldığında karşımıza çıkar.



Resim 2.5: InvalidCastException hatası

InvalidCastException istisnai durumu oluştuğunda bu durumun yakalanıp programın çalışmasına zarar vermeden yönetilmesi için Try…Catch bloğu kullanılmalıdır. Aşağıdaki örnekte try… catch bloğunun kullanımı görülmektedir.

Örnek2:

```
Public Class Form1
    Dim a As String
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
  System.EventArgs)
  Handles Button1.Click
    Try
      a = TextBox1.Text
       MessageBox.Show(CBayt(a))
       Catch hata As InvalidCastException
          MessageBox.Show(hata.Message)
          MsgBox("Yanlış dönüşüm yaptınız")
    End Try
    End Sub
  End Class
```

Örnek incelenirse hata oluşabilecek satırlar try…catch bloğu arasına alınmıştır. Böylece bir hata oluşması durumunda hata yakalanıp kullanıcı uyarılacak ve hatayı düzeltmesi sağlanacaktır.

2.7. Dönüşüm Komutları

Bu bölümde veri türü dönüşüm komutlarını inceleyeceğiz. Dönüşüm komutlarının bir kısmı system.convert sınıfının üyeleridir. Bu komutlarının kullanımı diğer tip dönüşüm komutlarından farklıdır. Fakat sonuçta işlevleri aynıdır.

2.7.1. ToString, CStr Komutları

Bir ifadeyi String türüne dönüştürür.

Kullanımı:

CStr(değişken_ ismi veya bilgi)
System.Convert.ToString(değişken_ ismi veya bilgi)

Örnek1:

```
Dim a As Bayt
Dim c As String
a = 150
c = System.Convert.ToString(a)
MsgBox(c)
```

Örnek2:

```
Dim c As String
c = CStr(150)
MsgBox(c)
```

Not: Programlar çalıştığında ekranda "150" (bir beş sıfır) stringi görülür.

2.7.2. ToDouble, CDbl Komutları

Sayısal içerikli bir değeri Double veri türüne dönüştürür.

Kullanımı:

CDbl(değişken_ ismi veya değeri)
System.Convert.ToDouble(değişken_ ismi veya değeri)

Örnek1:

```
Dim a As Integer
Dim c As Double
a = 15000
c = System.Convert.ToDouble(a)
MsgBox(c)
```

Örnek2:

```
Dim a As Integer
Dim c As Double
a = 15000
c = CDbl(a)
MsgBox(c)
```

Not: Programlar çalıştığında ekranda 15000 değeri görülür.

2.7.3. ToSingle, CSng komutları

Sayısal içerikli bir değeri single türüne dönüştürür.

Kullanımı:

CSng(değişken_ ismi veya değeri)
System.Convert.ToSingle(değişken_ ismi veya değeri)

Örnek1:

```
Dim a As Long
Dim c As Single
a = 1987564
c = System.Convert.ToSingle (a)
MsgBox(c)
```

Örnek2:

```
Dim c As Single
c = CSng(1987564)
MsgBox(c)
```

Not: Programlar çalıştığında ekranda 1987564 değeri görülür.

2.7.4. ToInt16, CShort Komutları

Sayısal içerikli bir değeri Short veri türüne dönüştürür. Sayının ondalıklı kısmını (varsa) tam sayıya yuvarlar.

Kullanımı:

CShort(değişken_ ismi veya değeri)
System.Convert.ToInt16(değişken_ ismi veya değeri)

Örnek1:

```
Dim a As Double
Dim c As Short
a = 250.25
c = System.Convert.ToInt16 (a)
MsgBox(c)
```

Örnek2:

```
Dim a As Double
Dim c As Short
a = 250.25
c = CShort(a)
MsgBox(c)
```

Not: Programlar çalıştığında ekranda 250 değeri görülür. Çünkü 250.25 tam sayıya dönüştürüldüğünde 250'ye yuvarlanır.

2.7.5. ToInt32, CInt Komutları

Sayısal içerikli bir değeri Integer veri türüne dönüştürür. Sayının ondalıklı kısmını (varsa) tam sayıya yuvarlar.

Kullanımı:

CInt(değişken_ ismi veya değeri)
System.Convert.ToInt32(değişken_ ismi veya değeri)

Örnek1:

```
Dim c As Integer
c = System.Convert.ToInt32 (15.60)
MsgBox(c)
```

Örnek2:

```
Dim a As Single
Dim c As Integer
a = 15.60
c = CInt(a)
MsgBox(c)
```

Not: Programlar çalıştığında ekranda 16 değeri görülür. Çünkü 15.60 tam sayıya dönüştürüldüğünde 16'ya yuvarlanır.

2.7.6. ToInt64, CLng Komutları

Sayısal içerikli bir değeri Long veri türüne dönüştürür. Sayının ondalıklı kısmını (varsa) tam sayıya yuvarlar.

Kullanımı:

```
CLng(değişken_ ismi veya değeri)
System.Convert.ToInt64(değişken_ ismi veya değeri)
```

Örnek1:

```
Dim a As Bayt
Dim c As Long
a = 36
c = System.Convert.ToInt64 (a)
MsgBox(c)
```

Örnek2:

```
Dim a As Bayt
Dim c As Long
a = 36
c = CLng(a)
MsgBox(c)
```

Not: Programlar çalıştığında ekranda 36 değeri görülür.

2.7.7. ToBayt, CBayt Komutları

Sayısal içerikli bir değeri Bayt veri türüne dönüştürür. Sayının ondalıklı kısmını (varsa) tam sayıya yuvarlar.

Kullanımı:

CBayt(değişken_ ismi veya değeri)
System.Convert.ToBayt(değişken_ ismi veya değeri)

Örnek1:

```
Dim c As Bayt
c = System.Convert.ToBayt (250)
MsgBox(c)
```

Örnek2:

```
Dim a As Integer
Dim c As Bayt
a = 250
c = CBayt(a)
MsgBox(c)
```

Not: Programlar çalıştığında ekranda 250 değeri görülür.

2.7.8. ToDateTime, CDate Komutları

Sayısal içerikli bir değeri Date veri türüne dönüştürür.

Kullanımı:

CDate(*değişken_ ismi veya tarih saat bilgisi*)
System.Convert.ToDateTime(*değişken_ ismi veya tarih saat bilgisi)*

Örnek1:

```
Dim c As Date
Dim d As Date
d= "1 Oca 1985 3:20:59 pm"
c = System.Convert.ToDateTime(d)
MsgBox(c)
```

Not: Program çalıştığında ekranda 01.01.1985 15:20:59 değeri görülür.

Örnek2:

```
Dim c As Date
c = System.Convert.ToDateTime("3:20:59 am")
MsgBox(c)
```

Not: Program çalıştığında ekranda 30.07.2006 03:20:59 değeri görülür. Burada dikkat edilecek husus ToDateTime komutunda sadece saat bilgisi girilirse önüne otomatik olarak bilgisayarın tarih bilgisi eklenir. Bu durum CDate komutunda geçerli değildir.

Örnek3:

```
Dim c As Date
c = CDate("Aug 6, 1969")
MsgBox(c)
```

Not: Program çalıştığında ekranda 06.08.1969 değeri görülür.

Örnek4:

```
Dim c As Date
c = CDate("15 tem 1976 13:20:19")
MsgBox(c)
```

Not: Program çalıştığında ekranda 15.07.1976 13:20:19 değeri görülür.

Örnek5:

```
Dim c As Date
c = CDate("1 feb 79")
MsgBox(c)
```

Not: Program çalıştığında ekranda 01.02.1979 değeri görülür.

2.7.9. ToDecimal, CDec Komutları

Sayısal içerikli bir değeri Decimal veri türüne dönüştürür.

Kullanımı:

```
CDecimal(değişken_ ismi veya değeri)
System.Convert.ToDecimal(değişken_ ismi veya değeri)
```

Örnek1:

```
Dim a As Long
Dim c As Decimal
a = 9500000000
c = System.Convert.ToDecimal (a)
MsgBox(c)
```

Örnek2:

```
Dim c As Decimal
c = CDecimal(950000000)
MsgBox(c)
```

Not: Programlar çalıştığında ekranda 9500000000 değeri görülür.

2.7.10. ToChar, CChar Komutları

CChar komutu string ifadenin ilk karakterini alarak Char veri türüne dönüştürür. ToChar komutu ise tek karakterlik string ifadeyi char veri türüne dönüştürür.

Kullanımı:

CChar(değişken_ ismi veya string ifade)
System.Convert.ToChar(değişken_ ismi veya karakter)

Örnek1:

```
Dim a As String
Dim c As Char
a = "b"
c = System.Convert.ToChar (a)
MsgBox(c)
```

Not: Program çalıştığında ekranda "b" karakteri görülür.

Örnek2:

```
Dim c As Char
c = CChar("eml")
MsgBox(c)
```

Not: Program çalıştığında ekranda "e" karakteri görülür.

2.7.11. ToBoolean, CBool Komutları

Sayısal veya string ifadeleri Boolean veri türüne dönüştürür. Burada önemli husus sayısal bilgi sıfırdan farklı ise True, sıfıra eşit ise False değeri döndürür. Dönüştürülecek string ifade yalnızca True veya False olabilir.

Kullanımı:

CCBool(*değişken_ ismi veya bilgi*)
System.Convert.ToBoolean(*değişken_ ismi veya bilgi)*

Örnek1:

```
Dim c As Boolean
c = System.Convert.ToBoolean (125)
MsgBox(c)
```

Not: Program çalıştığında ekranda True ifadesi görülür.

Örnek2:

```
Dim c As Boolean
c = System.Convert.ToBoolean (0)
MsgBox(c)
```

Not: Program çalıştığında ekranda False ifadesi görülür

Örnek3:

```
Dim c As Boolean
Dim a As String
a = "False"
c = System.Convert.ToBoolean(a)
MsgBox(c)
```

Not: Program çalıştığında ekranda False ifadesi görülür.

2.7.12. CType Komutu

Genel amaçlı tip dönüşümü için kullanılır. Bu komut yardımı ile bir veriyi istediğimiz veri tipine dönüştürebiliriz.

Kullanımı:

CType (*veri, dönüştürülecek tip ismi***)**

Örnek1:

```
Dim veri As Integer
veri = 100
MsgBox(CType(veri, String))
```

Not: Program çalıştığında ekranda "100" stringi görülür. Integer tip veriyi string tip veriye dönüştürür.

Örnek2:

```
Dim veri As Bayt
veri = 5
MsgBox(CType(veri, Boolean))
```

Not: Program çalıştığında ekranda True değeri görülür. Bayt tipindeki veriyi Boolean tipi veriye dönüştürür. Boolean tipine dönüşümde değer sıfırdan farklı ise "True", sıfıra eşit ise "False" değeri oluşur.

Örnek3:

```
Dim veri As Double
veri = 5.15
MsgBox(CType(veri, Integer))
```

Not: Program çalıştığında ekranda 5 değeri görülür. Integer tam sayı veri tipi olduğundan 5.15 sayısı 5'e yuvarlanır.

2.7.13. Parse Komutu

String türdeki bir veriyi sayısal veri türüne çevirir.

Kullanımı:

Sayısal veri türü**. Parse (**değişken_ ismi veya string ifade**)**

Örnek1:

```
Dim a As String
Dim c As Integer
a = "125"
c = Integer.Parse(a)
MsgBox(c)
```

Not: Program çalıştığında ekranda 125 sayısı görülür.

Örnek2:

```
Dim c As Bayt
c = Bayt.Parse("2")
MsgBox(c)
```

Not: Program çalıştığında ekranda 2 sayısı görülür.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Bir sınıfın veya yapının metot ve özelliklerini kulanınız.	Mesela örnek bir program yapınız. Burada yapı ve sınıflara ait özellikleri kullanarak sonuçları ekrana yazdırınız.
2. Veri türlerinin alabileceği değerleri ekrana yazdırınız.	Bunun için bir program oluşturunuz ve sınırlar dışında değer girildiğinde ekrana uyarı mesajı yazdırınız.
3. Veri türlerini birbirine dönüştürünüz.	Mesela double veri türünü integer türüne, bayt veri türünü Long veri türüne ve single veri türünü short veri türüne dönüştüren bir program yazabilirsiniz.
4. Bir veriyi string türe çeviriniz.	ToString komutunu kullanınız.
5. Bir metni sayıya çeviriniz.	Parse komunu kullanınız.

ÖLÇME VE DEĞERLENDİRME

A- OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; ilk 7 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. Bayt bir yapı türüdür. ( )

2. Decimal bir sınıf türüdür. ( )

3. String bir sınıf türüdür. ( )

4. Narrowing conversions genişleme dönüşümleri anlamına gelir. ( )

5. Narrowing conversions işlemi için **Option scrict off** olmalıdır. ( )

6. InvalidCastException tip dönüşüm hatasıdır. ( )

7. Bir sayıyı double türe CStr komutu ile çevirebiliriz. ( )

8. Genel amaçlı tip dönüşümü için kullanılan komut Ctype komudur. ( )

9. ToString ile Parse komutları aynı işi yapar. ( )

10. 101.65 ondalıklı sayısının integer türde karşılığı aşağıdakilerden hangisidir?
A) 101 B) 102
C) 101.65 D) 101,7

11. "MEB" stringi char tipine dönüştüğünde hangi ifade oluşur?
A) MEB B) B
C) M D)

12. Aşağıdaki komut ikililerinden hangileri aynı işi yapmaz?
A) ToString-CStr B) ToDouble-CDbl
C) ToInt64-CLng D) ToInt16-CInt

13. Aşağıdaki değişken veri ikililerinden hangisi yanlıştır?
A) Int16 – 39412 B) Single – 255.65
C) Bayt – 12 D) Long – 213545

14. Aşağıdaki değişken veri ikililerinden hangisi yanlıştır?
A) Int32 – 256235 B) Single – "Ali"
C) Char – "SS" D) DateTime – #21/06/2006#

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

Döngü komutlarını öğrenip program içerisinde kullanabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

13. Bir yatırımın gelecekteki değeri nasıl hesaplanır?

14. For next döngüsü nasıl çalışır?

15. Nesne özellikleri nelerdir?

Araştırma işlemleri için internet ortamı ve görsel programlama dilini anlatan kaynak kitaplardan faydalanınız.

3. FV, WITH VE FOR EACH KOMUTLARI

Bu öğrenme faaliyetinde finansal uygulamalar için kullanılan FV komutu, kod yazmayı kolaylaştıran With bloğu, nesne özelliklerini daha kolay değiştirmeyi sağlayan For Each komutu ve TypeOf Is komutları hakkında ayrıntılı bilgi verilecektir.

3.1. FV Komutu ve Parametreleri

FV (**F**uture **V**alue) gelecekteki değer anlamına gelmektedir. Periyodik sabit ödemeler (taksitlendirme) ve belli kazanç oranını hesaplama üzerine kurulmuş bir komuttur. Genellikle finansal işlemler (bir yatırımın gelecekteki değerini bulmak) için kullanılır. İşlem sonuçları büyük ve ondalıklı olabileceğinden geri dönen değer double tiptedir. Taksitlendirme belli bir zamanda yapılan sabit ödemeler serisidir. Bu, ev ipoteği gibi bir borçlanma ya da aylık birikim planı gibi bir yatırım olabilir. FV komutunun parametrelerinden olan Rate ve NPer bağımsız değişkenleri, aynı birimlerle ifade edilen ödeme dönemleri kullanılarak hesaplanmalıdır. Örnek olarak Rate değişkeninde aylar baz alınmışsa Nper değişkeninde de aylar baz alınmalıdır. Tüm bağımsız değişkenler için; harcanan para negatif değerle, kazanılan para ise pozitif değerle ifade edilir.

FV komutu Microsoft.VisualBasic.Financial sınıfının bir üyesidir.

Kullanımı:

FV (rate, nper, pmt [, pv [, due]])

Parametreler:

Rate: Her dönemde elde edilen double tipinde belirli kazanç oranıdır. Mesela yıllık %10 oranında kârla bir arabaya borçlanıp ve aylık ödemeler yaparsanız, her dönem (ay) için 0,10/12 oranında veya 0,0083 kazanç elde edersiniz. Bu parametrenin kullanımı zorunludur.

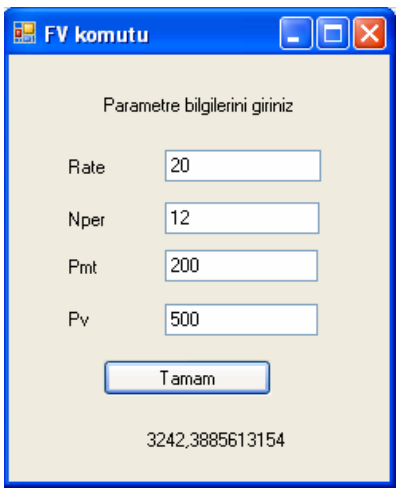
NPer: Taksitlendirme içindeki toplam ödeme süresini belirten double tipinde veridir. Mesela 4 yıl vade ile araba aldığınızda, aylık ödemeler yapacaksanız borcunuzu toplam 4x12 (48) ödeme döneminde ödemeniz gerekir. Bu parametrenin kullanımı zorunludur.

Pmt: Her ay yatırım yapılacak sabit miktarı belirten double tipte veridir. Birikim yapılırken aynı zamanda bu paraya faiz de uygulanır. Bu parametrenin kullanımı zorunludur.

PV: Bileşik faiz hesabında anaparayı temsil eden double tipte veridir. Sabit miktarda bir paranın faizle işletilmesi için kullanılır. Bu parametre kullanımı isteğe bağlıdır, kullanılmazsa 0 alınır.

Due: Ödemelerin yapıldığı tarih bilgisidir. Bu bağımsız değişken, ödemeler ödeme sürecinin sonunda ise dönem sonu DueDate.EndOfPeriod, başında ise dönem başı DueDate.BegofPeriod olmalıdır. Bu parametrenin kullanımı isteğe bağlıdır.

Örnek1:



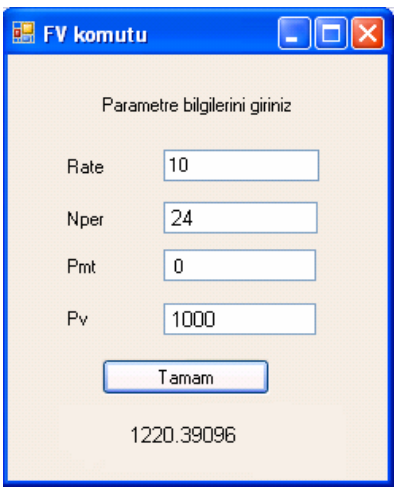
Resim 3.1: Fv komutu örnek1 form görünümü

```
Public Class Form1
   Dim Nper, Rate, pmt, pv As Double
   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
      Nper = (TextBox2.Text)
      If TextBox1.Text > 1 Then Rate = TextBox1.Text / 100
      pmt = TextBox3.Text
      pv = TextBox4.Text
      Label5.Text = - (Financial.FV(Rate /12, Nper, pmt, pv, DueDate.EndOfPeriod))
   End Sub
      End Class
```

Örnekte TextBox1 yıllık faiz oranını, TextBox2 birikimin yapılacağı toplam süreyi, TextBox3 aylık olarak biriktirilecek miktarı, TextBox4 ise anaparayı ifade etmektedir. Programı çalıştırılıp resim 3.1'deki değeler girildiğinde FV komutu 12 ay sonunda elimize geçecek birikim miktarını hesaplayıp ekrana yazdıracaktır. Bu miktar yaklaşık 3242.38 YTL dir. FV komutunun önündeki – işareti sonucu pozitif yapmak için kullanılmıştır. Normalde işlem sonucunda çıkan miktar harcana para kabul edildiğinden sonuç "–" çıkar.

Açıklama: Bu örnekte dikkat edilirse iki ayrı işlemi aynı anda yaptık. Hem anaparamıza hem de aylık birikimlerimize faiz uyguladık. Bunları ayrı ayrı da yapabiliriz. Pv değerini sıfır girerek sadece aylık birikimlerimizi artırabilir veya Pmt parametresine sıfır girerek anaparamıza faiz uygulayabiliriz. Bankalarda buna benzer hesaplamalar kullanılmaktadır.

Örnek2: Elimizde bulunan 1000 YTL parayı yıllık %10 faizle 2 yıllığına bankaya yatırırsak süre sonunda paramızın miktarı ne kadar olur?



Resim 3.2: FV komutu örnek2 form görünümü

```
Public Class Form1
    Dim Nper, Rate, pmt, pv As Double
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Nper = (TextBox2.Text)
    If TextBox1.Text > 1 Then Rate = TextBox1.Text / 100
    pmt = TextBox3.Text
    pv = TextBox4.Text
    Label5.Text = - (Financial.FV(Rate /12, Nper, pmt, pv, DueDate.EndOfPeriod)) &
"YTL"
    End Sub
        End Class
```

Resim 3.2'deki değerler forma girilip tamam butonu tıklandığında 2 yıl sonunda paramızın değerinin 1220.39096 YTL olacağı görülür.

Araştırma: Bankalarda bileşik faizin nasıl hesaplandığını araştırınız.

3.2. With…End With Komutu

With anahtar kelimesi nesnenin ismini yazmadan sürekli olarak nesneye başvurma kolaylığı sağlayan bir metottur. With anahtar kelimesi programlama dillerine çok önemli kolaylıklar sağlamıştır. Bu kelime yardımı ile tanımlanmış bir nesnenin ismini kullanmadan özelliklerini kullanabiliriz.

Kullanımı:

With *nesne ismi*
Komu veya komutlar…
End With

With komutu ile tanımlanan nesne özelliklerini End With komutuna kadar kullanabiliriz. End With komutu With komutuna son verir. Örneklerle bu komutun çalışmasını inceleyelim.

Örnek1:

```
With Label1
    .Text = "MEB"
End With
```

Burada label1 nesnesi with ile tanımlanmıştır. Bu nesneye ait text özelliğini kullanmak için .Text yazmak yeterlidir.

Örnek2:

```
With TextBox1
    .Text = "a"
    .Width = 100
    c = CInt(.text) * 2
    b = .TextLength
End With
```

TextBox1 nesnesine ait özelliklerin nesne ismi yazılmadan kullanılması örnekte görülmektedir.

Örnek3:

```
With ComboBox1
    .Items.Add("Okul")
    .Refresh()
    .Hide()
End With
```

ComboTextBox1 nesnesine ait özelliklerin nesne ismi yazılmadan kullanılması örnekte görülmektedir.

With anahtar sözcüğü özellikle program içerisinde belli nesneler sıklıkla kullanılıyorsa kullanıcıyı bu nesnelerin isimlerini sürekli yazmaktan kurtarır.

3.3. For Each Komutu

Visual Basic görsel programlama dilinde bir form üzerindeki tüm nesnelere controls collection (denetim kümesi) denir. Nesnelerle çalışmanın en kolay şekli onları bir grup olarak ele almaktır. Koleksiyonların varlık nedeni nesne gruplarını hızlı bir biçimde işleyebilmektir. Örnek olarak, tüm nesneler grubunu bir defada görüntülemek, taşımak, sıralamak, gizlemek, yeniden adlandırmak ya da yeniden boyutlandırmak isteyebiliriz. Bu işlemleri yapmak için For Each…Next döngüsü kullanılır. For Each…Next döngüsü For…next döngüsüne benzer, tek farkı kümeler ile beraber kullanılmasıdır. Denetim kümesi kullanılırken projemizde birden fazla form varsa tanımlamalarda form ismi belirtilmelidir. Örneğin form1'in denetim kümesi kullanılacaksa form1.controls adı kullanılmalıdır.

Kullanımı:

Dim *Değişken* **As control**

For Each *Değişken* **In controls** *// eğer birden fazla form varsa form ismi ile birlikte kullanılır. Örn: Form1.controls, Form3.controls gibi //*

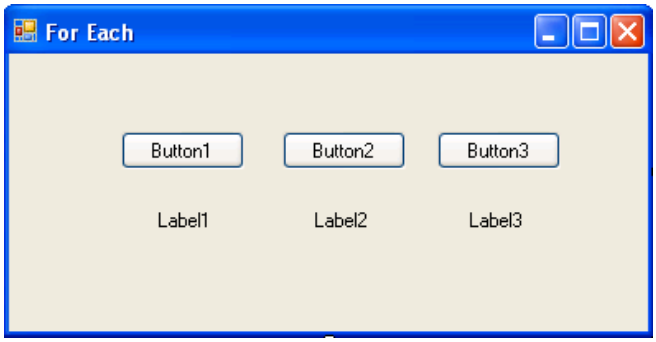
Nesneler…

Next *Değişken*

Dikkat edilirse işlenecek nesne grupları For Each…Next döngüsü içersinde yer alır. Denetim (control) bir değişkene aktarılır. Değişken yardımı ile işlemler yapılır. Değişken koleksiyondaki geçerli nesneyi temsil eder. Controls kelimesindeki "s" eki formdaki tüm denetim nesnelerini temsil eden koleksiyon sınıfıdır. Döngü koleksiyondaki üyeleri tek tek işlemek için kullanılır. Örneğin, koleksiyondaki nesnelerin Enabled, Left, Top, Text ya da Visible özelliklerini değiştirmek ya da bir liste kutusundaki her nesnenin adını listelemek gibi.

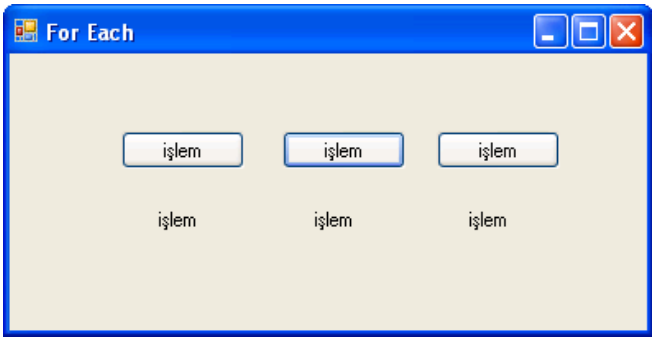
Örnek 1: Formumuzda bulunan tüm nesnelerin isimlerini değiştirmek için bir program yazalım.

```
Dim ad As Control

  Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
    For Each ad In Controls
      ad.Text = "işlem"
    Next ad
  End Sub
```



Resim 3.3: For Each komutu örnek1 form görünrüsü

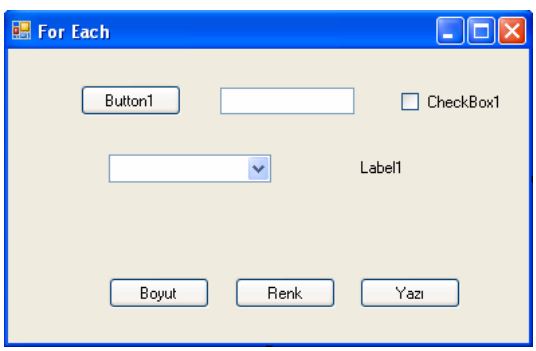
Şeklinde bir form oluşturup aşağıdaki programı çalıştırırsak formdaki tüm nesnelerin isimlerini "işlem" olarak değiştirmiş oluruz.



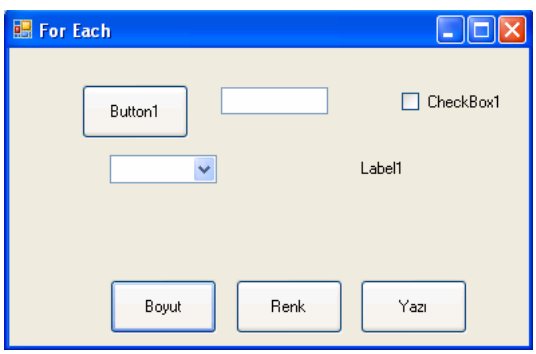
Resim 3.4: For Each komutu örnek1 çalıştırıldıktan sonraki form görüntüsü

Örnek 2:

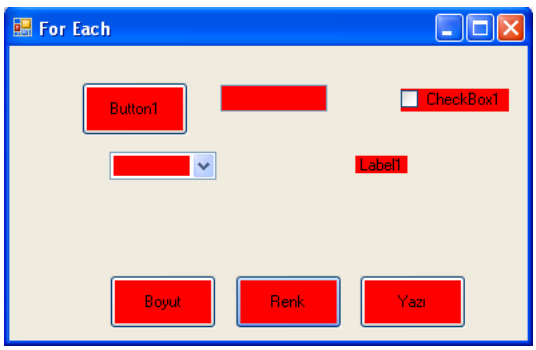
```
Dim ykslt As Control
   Dim renk As Control
   Dim yaz As Control
   Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
      For Each ykslt In Controls
         ykslt.Height = 40
         ykslt.Width = 80
      Next ykslt
   End Sub
   Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
      For Each renk In Controls
         renk.BackColor = Color.Red
      Next renk
   End Sub
   Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
      For Each yaz In Controls
         yaz.Text = "FOR EACH"
      Next yaz
   End Sub
```



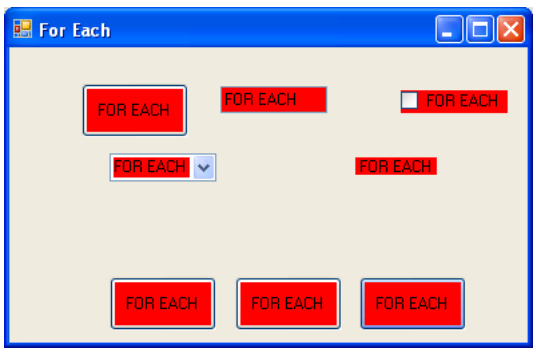
Resim 3.5: Program çalıştırılmadan önceki form



Resim 3.6: Boyut butonuna basılmış form



Resim 3.7: Renk butonuna basılmış form



Resim 3.8: Yazı butonuna basılmış form

Program yardımı ile form üzerindeki nesnelerin çeşitli özellikleri değiştirilmiştir.

Açıklama: Form üzerindeki belli nesnelerin özellikleri değiştirilmek isteniyorsa nesnelerin name özelliğinden faydalanılabilir.

```
For Each a In Controls
   If a.name = "button1" then a.text = "OK"
Next a
```

Örnekte görüldüğü gibi sadece ismi button1 olan nesnenin text özelliği değiştirilmiştir.

3.4. TypeOf…Is Komutu

Object olarak tanımlanmış bilginin belli bir veri türüne ait olup olmadığına karar verir. Eğer bilgi türe ait ise true, değilse false değeri üretir.

Kullanımı:

TypeOf *değişken_ ismi* **Is** *karşılaştırılacak veri türü*

Örnek1:

```
Dim a As Object

   Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
      a = 45.45
      MsgBox(TypeOf a Is Double)
         End Sub
```

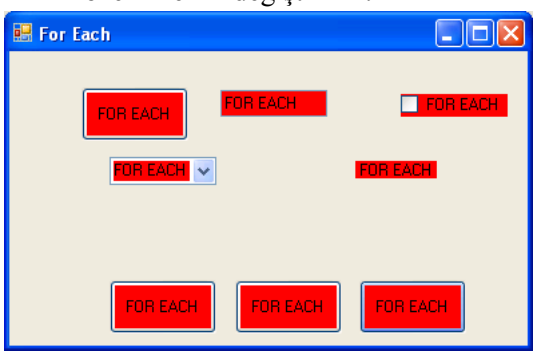
Program çalıştırıldığında ekranda true bilgisi görülür.

Örnek 2:

```
Dim a As Object

    Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        a = 45
        MsgBox(TypeOf a Is Double)
            End Sub
```

Program çalıştırıldığında ekranda false bilgisi görülür. Çünkü 45 double türde bir veri değildir.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Bir yatırımın gelecekteki değerini FV (Future Value) komutu ile hesaplayınız. 	Öncelikle parametrelere girilecek değerlerin ne olacağını düşünüz ve parametreleri tanımlayınız.
2. "With…End With" komutları arasına nesne ismini yazmadan kod yazınız.	Nesne özelliklerinin neler olduğunu nesne isminden sonra "." karakterini yazarak görebilirsiniz.
3. Döngü kurarak form üzerinde yerleştirilmiş olan nesnelerin özelliklerini değiştiriniz. 	Öncelikle basit bir form oluşturup forma nesne ekleyiniz.
4. TypeOf…Is komutu ile değişken tipi kontrolü yapınız.	İki değişken türünü karşılaştırabilirsiniz.

ÖLÇME VE DEĞERLENDİRME

A- OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; sonunda parantez olanlar doğru / yanlış sorularıdır. Verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Şıklı sorularda uygun şıkkı işaretleyiniz.

1. FV komutu yatırımlarımızın gelecekteki değerini hesaplar. ( )

2. FV komutunda Pmt parametresinin kullanımı zorunlu değildir. ( )

3. FV komutu bir fonksiyondur. ( )

4. Aşağıdaki ifade 100 YTL'nin yıllık %10 faizle 2 yıl sonraki değerini bulur. ( )

FV(0.1/12,2,0,100)

5. With…End With komutunu kullanarak bir nesnenin ismini sürekli yazmaktan kurtuluruz. ( )

6. For Each…Next bir döngü komutudur. ( )

7. For Each…Next komutu ile 100 tane nesnenin ismi aynı anda değiştirilebilir.( )

8. For Each…Next komutunda değişken control tipte tanımlanır. ( )

9. TypeOf 0.15 Is String komutu ile 0.15 bilgisinin string türde olup olmadığı kontrol edilir. ( )

10. TypeOf…Is komutu string tipte değer üretir. ( )

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

Programlama dilinde sınıf oluşturup bu sınıf içerisinde nesneye yönelik kod yazabileceksiniz.

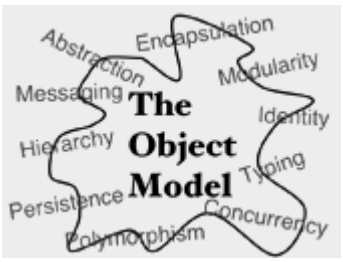
ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

1. Nesne yönelimli programlamanın kullanıcıya faydaları nelerdir?
2. Sınıf ve object ifadelerini araştırınız.

Araştırma işlemleri için internet ortamı ve görsel programlama dilini anlatan kaynak kitaplardan faydalanınız.

4. NESNEYE YÖNELİK KOD

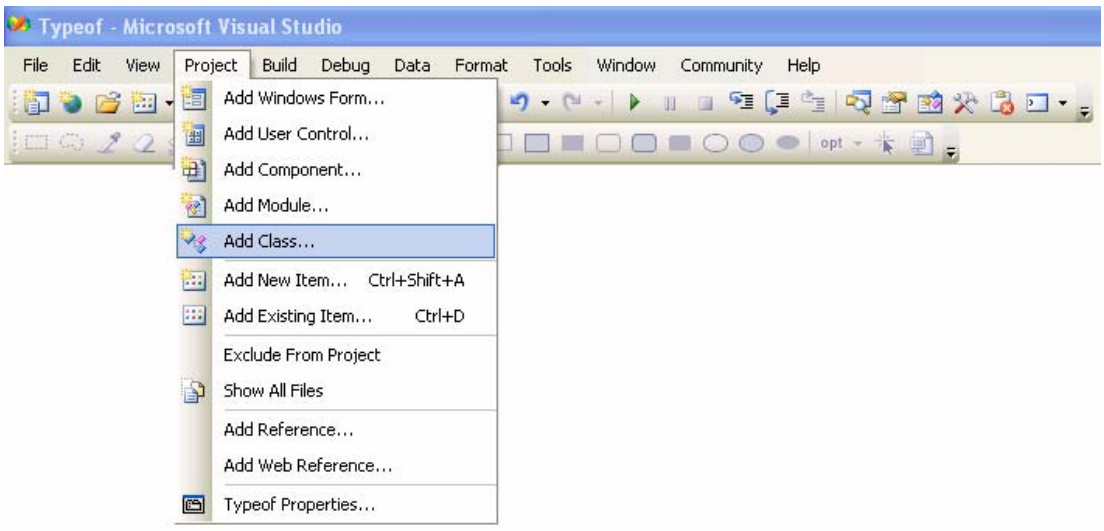


Nesneye yönelik programlama (Object Oriented Programming) çözülmesi istenen bir problemi parçalara ayırıp her bir parça arasındaki ilişkiyi gerçeğine uygun şekilde belirleme tekniğine denir. Başka bir deyişle kullandığımız yordamları direk uygulama koduna değil, sınıf içine yazıp bu sınıflardan türettiğimiz nesneler üzerinden çağırıyorsak bu işlem Nesneye Yönelik Programlamadır. Bu öğrenme faaliyetinde nesneye yönelik kod yazmak için gerekli bilgiler verilecektir.

4.1. Kullanıcı Tanımlı Sınıflar

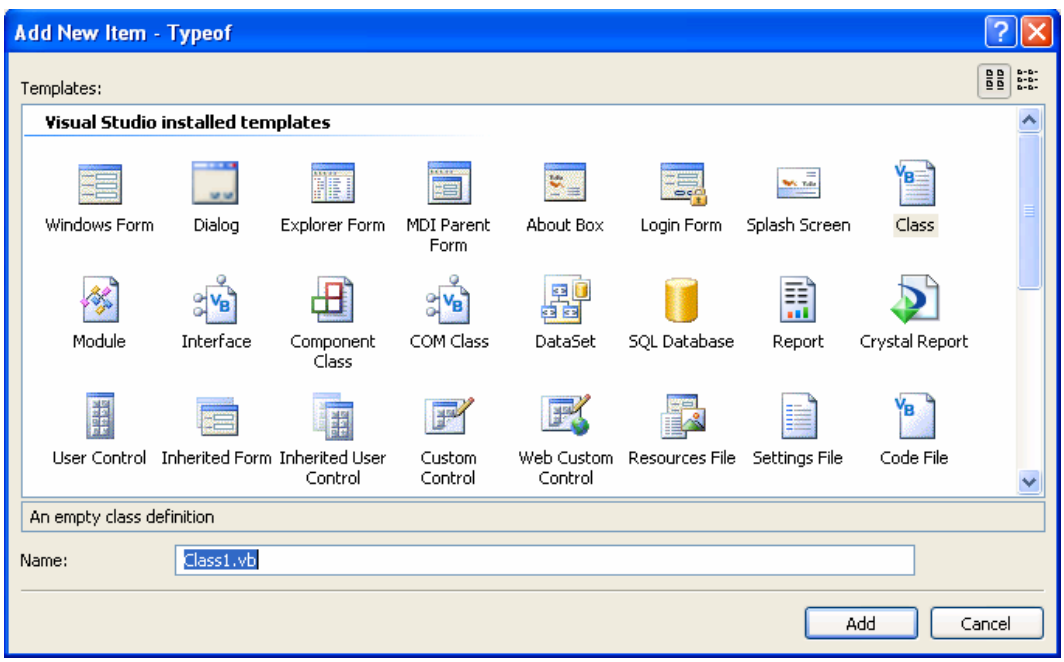
Sınıflar (classes) nesne yönelimli programlamanın en önemli kavramıdır. Sınıf kendisinden nesneler türetilen, temel özellikleri ve işlevi önceden kendisini yazan programcı tarafından belirlenmiş nesne şablonlarıdır. Bir sınıfa, ait olduğu adalanı (namespace) hiyerarşisi içinden erişilebilir. Kullanıcı tanımlı sınıf programda kendi nesnelerimizi tanımlamamızı sağlar. Oluşturulan sınıfın özellikleri, yöntemleri ve olayları bulunur. Örnek olarak otomobil sınıfını düşünelim. Bu sınıfın Araçlar.otomobil adalanı içinde olduğunu

varsayalım. Otomobilleri sınıflara ayırırsak, SporOtomobil, AileOtomobili, YarışOtomobili gibi sınıflar (classes) oluşur. Bu sınıfların bazı özellikleri (properties) önceden bu sınıfları oluşturanlar tarafından belirlenmiştir. Bunlar boyut, model, tip, aksesuar gibi özelliklerdir. Bunlara müdahale edemeyiz fakat istediğimiz sınıfı kullanabiliriz. Bize spor otomobil nesnesi lazımsa bunu SporOtomobil sınıfından türetiriz. Projemize yeni bir sınıf eklemek için Project menüsünden Add Class komutunu çalıştırırız.



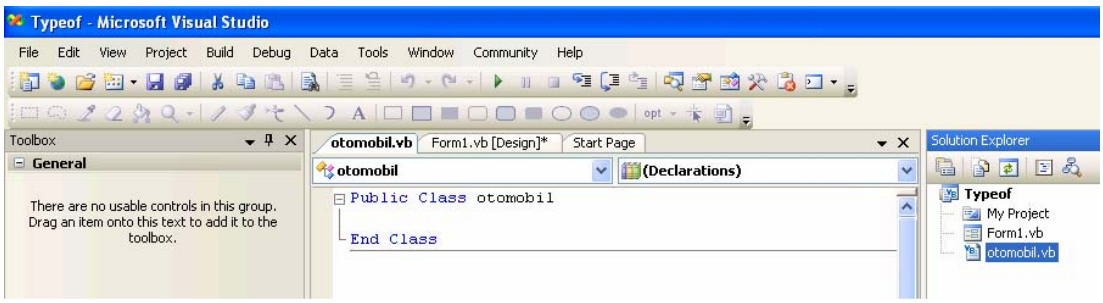
Resim 4.1: Sınıf oluşturma işlemi

Bu komut çalıştırıldıktan sonra karşımıza Add New Item penceresi gelir.



Resim 4.2: New Add Item penceresi

Add New Item penceresi yardımıyla projemize birçok öğe ekleyebiliriz. Class (sınıf) öğesi seçip name kutucuğundan isim verdikten sonra Add butonuna tıklarız. Böylece belirlediğimiz isimde bir sınıf oluşturmuş oluruz.



Resim 4.3: Oluşturulmuş sınıf görünümü

Oluşturduğumuz sınıf Solution Explorer penceresine yerleşecek ve sınıfa ait kodlar kod penceresinde görünecektir.

4.2. "Class…End Class" Bloğu

Programlamama dilindeki tüm sınıflar **Class** sözcüğü ile başlar, **End Class** sözcükleri ile sona erer. **Class…End Class** sözcükleri arasında değişkenler, metotlar, özellikler ve olaylar yer alabilir.

Proje içerisinde birden fazla sınıfı tanımlamak mümkündür. İstediğimiz zaman **Class…End Class** blokları arasına gerekli kodları yazarak sınıf oluşturabiliriz.

Kullanımı:

```
Public Class isim
Komut/komutlar
End Class
```

İsim: Oluşturulacak sınıfın ismidir.

Örnek:

```
Public Class otomobil
……..
End Class
```

4.3. "Get…End Get" ve "Set…End Set" Blokları

"Get…End Get" ve "Set…End Set" blokları projemiz içerisindeki sınıfa, özellik eklendiğinde oluşur. Get…End Get bloğu özellikten bilgi okunabileceğini, Set…End Set bloğu ise özelliğe bilgi yazılabileceğini gösterir.

4.4. “New” Anahtar Kelimesi

Program içerisinde yeni bir nesne oluşturmak için kullanılır. Bu nesne program çalıştıktan sonra oluşturulur.

Dim form3 As New Form( ) → form3 ismi ile yeni bir form oluşturur.

Dim labe7 As New Label( ) → labe7 ismi ile yeni bir label oluşturur.

Dim ad As New Textbox( ) → ad ismi ile yeni bir textbox oluşturur.

4.5. “Encapsulation” Deyimi

Veriler ve fonksiyonlar bir bütün olarak ele alınır ve tek bir varlık olarak düşünülür. Verilerin ve fonksiyonların bir araya gelmesi nesneyi oluşturur. Encapsulation **(**Verilerin korunması / paketlenmesi **)** nesneyi dışarıdan gelebilecek zararlara karşı korumaktadır. Başka bir deyişle; programın bütün unsurlarını, çalışma mantıklarını kendi içinde saklayan kapsüller halinde yazılmasıdır. Nesnenin verileri ve fonksiyonları kendine özeldir. Dolayısıyla sadece kendisi kullanabilir. Ancak kullanılan erişim metotları ile nesnenin bazı kısımları dış kullanıma da açıktır. Nesnenin bazı üyelerinin private olması bu üyelerin, nesnenin kendisine özel olduğunu ve sadece kendisinin kullanacağı anlamına gelir. Bazı üyelerin public olması ise dışarıdan ulaşılabilmesini sağlar. Dışarıya açık olan bu kısımlar nesnenin özel alanlarına ulaşmak için kullanılır.

4.6. Metods (Hareketler) ve Properties (Özellikler)

Bir sınıf metot ve özelliklerden oluşur. Metotlar sınıfa ait alt programlardır. Sınıf içine istediğimiz kadar özellik ve metot ekleyebiliriz. Görsel programlama dilinde bir sınıfa metot eklemek için “**Sub…EndSub**” veya “**Function End Fuction**” kalıpları kullanılır. Aralarındaki fark Sub ile başlayan metotlar geriye değer döndürmezken Function ile başlayan metotlar geriye değer döndürür.

```
Sub vb()
  …
End Sub

Function vb() As Integer
  …
Return geri dönen değer

End Function
```

Function’da geriye dönen değerin türü de tanımlanmalıdır. Örnekte Integer olarak tanımlanmıştır. Metotlar tanımlanırken kapsam sözcükleri ile birlikte kullanılır. Genel olarak kullanılan kapsam sözcükleri aşağıda açıklanmıştır.

Private: Sadece tanımlanan sınıf içerisinde erişilebilir.

Friend: Sadece tanımlanan proje içerisinde erişilebilir.

Public: Sınıfın içinden veya dışından erişilebilir.

Protected: Sınıftan türetilen tüm diğer sınıflardan erişilebilir.

Protected Friend: Sadece tanımlanan projedeki diğer sınıf ve alt sınıflardan erişilebilir.

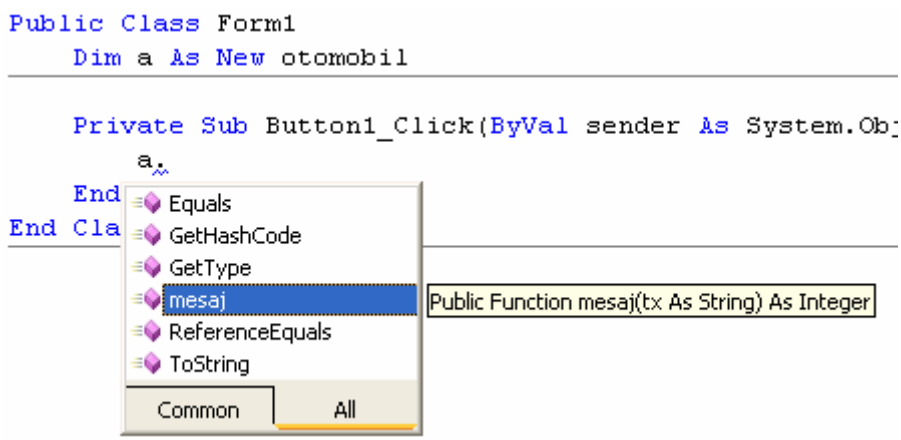
En geniş kapsamlı sözcük "public" tir. Public olarak tanımlanmış metoda her yerden ulaşılabilir.

Örnek: Sınıfımız içerisine bir metot ekleyip bunun kullanımını görelim.

```
Public Class otomobil
   Public Function mesaj(ByVal tx As String) As Integer
      MsgBox("kelimeninz " & tx.Length & " karakterden oluşur")
  Return (tx.lenght)
   End Function
End Class
```

Şeklinde sınıfımıza bir metot eklemiş olduk.

Bu metot ismi, sınıfla ilişkilendirilen bir değişkenin ismi yazılıp "." karakterine basılırsa otomatik olarak gelecektir.



Resim 4.4: Oluşturulan metodun programda kullanımı

```
Private Sub Button1_Click (ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        b=  a.mesaj("ahmet")
   End Sub
```

Programı yazıldığında sınıf içerisinden mesaj metoduna gönderilen "ahmet" bilgisinin kaç karakter olduğu "kelimeniz 5 karakterden oluşur" şeklinde bir mesajla ekranda görünecek, 5 değeri ana programa döndürülüp b değişkenine atanacaktır.

Property (özellik) metotların özel bir türü olup sınıftan değer okumak veya değer göndermek için kullanılır. Özellik oluşturmak için öncelikle oluşturulan sınıfın kod bloğu arasına (Class…End Class) Public Property *özellik adı* As *özellik türü* şeklinde komut satırı yazmamız gerekir. Bu işlem yapıldığında Get…End Get ve Set…End Set blokları otomatik olarak oluşturulur.

```
Public Property ismi() As String
Get

End Get
Set(ByVal value As String)

End Set
End Property
```

Bizim yapmamız gereken Get…End Get ve Set…End Set blokları arasına gerekli komutları yazmaktır. Set…End Set özelliğe eklenen bilginin tutulduğu yeri, Get…End Get ise oluşturduğumuz özellikten geri dönecek değeri belirler.

Örnek:

```
Public Property ismi() As String
    Get
        Return ism1
    End Get
    Set(ByVal value As String)
        ism1 = value
    End Set
End Property
```

ism1değişkeni geri dönecek bilginin tutulduğu değişkendir.

ism1= value ifadesi özelliğe aktarılacak bilginin türünü ve değişken ismini belirtir.

Buradaki ism1 değişkeni "Public Property ismi() As String" komutundan önce "Private ism1 As string" şeklinde tanımlanmalıdır. Bu oluşturduğumuz özelliği projemizde kullanmak için,

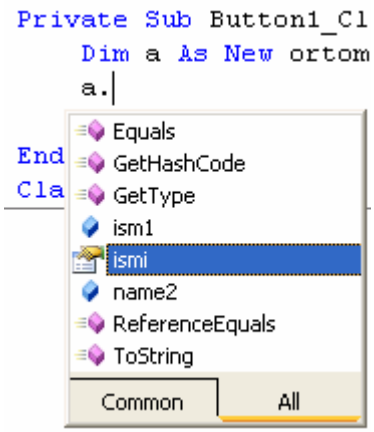
```
Dim a As New otomobil
    a.ismi = TextBox1.Text
     MsgBox(a.ismi)
```

Şeklinde basit bir program oluşturmamız gerekir. Dikkat edilirse a değişkeni oluşturduğumuz sınıf (otomobil) için tanımlanmıştır. Bu tanımlamadan sonra sınıfın özelliklerini kullanabiliriz. a değişkeninin ismi özelliğine klavyeden girdiğimiz bilgi aktarılır ve hafızada saklanır. Msgbox(a.ism1i) komutu ile hafızadaki bu bilgi çağrılıp ekrana aktarılabilir veya program içerisinde kullanılabilir. Özellikler tanımlanırken Public veya Private ön adları kullanılır. Bunlar özelliğin dışarıdan görülüp görülemeyeceğini belirler. Aşağıdaki örnekte bu durum açıkça görülmektedir.

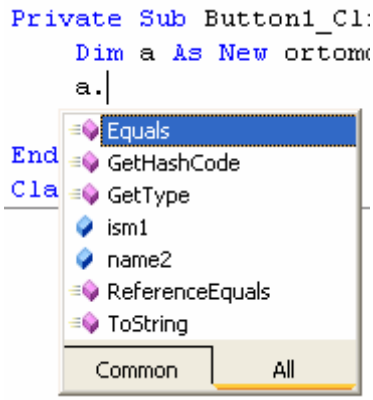
Örnek:

```
Public Property ismi() As String
    Get
        return ism1
    End Get
Set(ByVal value As String)
        ism1 = value
End Set
End Property
```

```
    Private Property ismi() As String
    Get
    return  ism1
    End Get
Set(ByVal value As String)
        ism1 = value
    End Set
    End Property
```



Resim 4.5: Public özelliği



Resim 4.6: Private özelliği

4.7. Sadece Yazılabilir veya Sadece Okunabilir Özellikler

Oluşturulan bir özelliğin sadece yazılabilir (write only) veya sadece okunabilir (read only) özellik olması istenebilir. ReadOnly ve WriteOnly özellikleri kullanılarak Get…End Get veya Set…End Set bloklarından biri otomatik olarak kaldırılır. Böylece özellikten sadece bilgi okunabilir veya özellik içerisine sadece bilgi gönderilebilir.

```
ReadOnly Property ismi() As String
    Get
      …
    End Get
End Property
```

Bu şekilde yalnız okunabilir özellik oluşturulur.

```
Writeonly Property ismi() As String
    Set(ByVal value As String)
      …
    End Set
End Property
```

Yalnız yazılabilir özellik oluşturulur.

4.8. Kullanıcı Arabirim Sınıfları

Arabirimi diğer sınıflar için rehber olarak tanımlayabiliriz**.** .Net programlama dilinden önce, bir sınıf temel alınarak birçok sınıf oluşturulabilmekteydi (çok kalıtımlılık). Yani bir sınıf, kalıtımsal olarak birden fazla sınıftan türetilebiliyordu. Ancak sınıflar türedikçe kodlar karmaşıklaşıyor ve anlaşılabilirliği azalıyordu. Bu nedenle .Net programlama dillerinde bir sınıf temel alınarak sadece tek sınıfın oluşturulabileceği kısıtlaması getirilmiştir. Çok kalıtımlık görevi ise anlaşılması daha kolay olan arabirimlere bırakılmıştır. Arabirimlerin kullanmasının en büyük nedenlerinden birisi budur.

.Net programlama dilinde arabirim tanımlamak için Interface… End Interface blokları kullanılır.

```
Interface okul
…
End Interface
```

Yukarıdaki komut satırları ile okul isminde bir arabirim tanımlanmıştır. Bu arabirime metot, özellik veya olay eklenebilir.

```
Interface okul
Event degistir(ByVal Success As Boolean)
Property bol() As String
Function mat() As Integer
End Interface
```

Oluşturulan interface program içerisinde kulanmak için önce implements deyimi kullanılarak sınıfla ilişkilendirilmelidir. Bu deyim arabirim içinde tanımlanan olay, özellik ve metotların sınıf içerisinde kullanılabileceğini bildirir.

```
Class egitim
  Implements okul
End Class
```

Arayüz oluştururken uyulması gereken kurallar.

- Bir arayüz'ün tüm üyeleri public kabul edilir.
- Metotları public olarak tanımlayamayız. (Bütün üyeler public tanımlanmış kabul edildiğinden hata oluşur.)
- Bir arayüz, bir yapı (Struct) dan veya bir sınıf(class) tan kalıtımla türetilemez.
- Arayüz elemanları static olarak tanımlamaz.
- Arayüzlerin uygulandığı sınıflar, arayüzde tanımlanan bütün üyeleri kullanmak zorundadır.

Örnek: Bir arabirim oluşturmayı, oluşturulan arabirime özellik eklemeyi ve başka bir sınıfta kullanmayı adım adım uygulayalım.

- Öncelikle arabirimi tanımlayalım.

```
Interface arabirim
   Property uz() As String
   Function deger() As Integer
End Interface
```

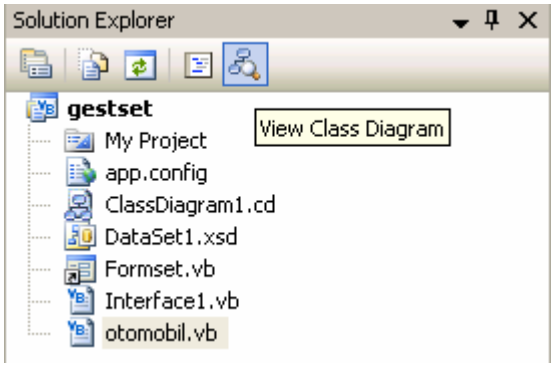
- Oluşturulan arabirime bir sınıf içerisinde özellik ekleyelim.

```
Class aritmetik
   Implements arabirim
   Private c As Integer
   Private d, e As String

   Public Property uz() As String Implements arabirim.uz
      Get
         e = String.Concat(d, "oooooo")
         Return e
      End Get
      Set(ByVal value As String)
         d = value
      End Set
   End Property

   Public Function deger() As Integer Implements arabirim.deger
      c = 150
      Return c
   End Function
End Class
```

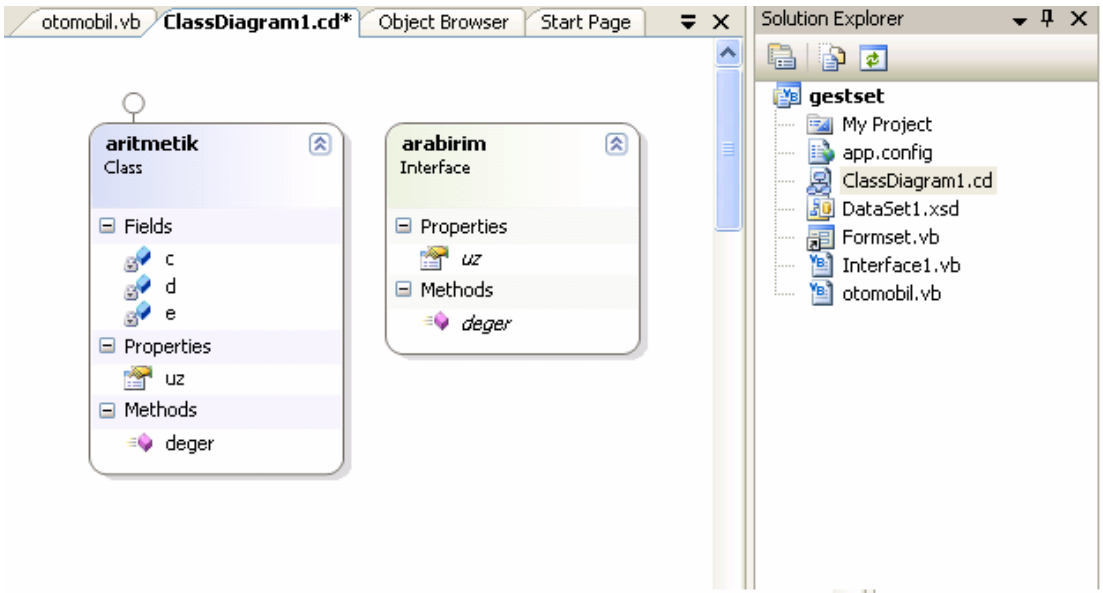
- Uz isimli fonksiyon ve deger isimli fonksiyonu programımızda kullanalım.

```
Public Class Form1
   Dim b As Integer
   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
      Dim a As New aritmetik
      MsgBox(a.deger)
      a.uz = "ali"
      MsgBox(a.uz)
   End Sub
End Class
```

Not: Oluşturduğumuz sınıf, arabirim ve özellikler ve metotlar solution Explorer penceresindeki class view seçeneği yardımı ile görülebilir.



Resim 4.7: View Class seçeneği



Resim 4.8: Class view penceresinde oluşturduğumuz sınıf ve arabirimin görünümü

4.9. “Garbage collection”

VB.net programlama dilinde garbage collection (çöp toplayıcı) uygulamalarımız için hafızayı boşaltır ve kullanıma hazır hale getirir. Kısaca hafızayı yönetir. Nesne oluşturulurken new operatörü kullanılmasının sebebi, yığın (veri topluluğu) yönetiminden nesneler için çalışma sırasında hafızadan yer ayrılmasıdır. Yığın yönetimi her oluşturulan nesne için hafızadan yer ayırır. Fakat hafızalar sonsuz değildir. Bu sebeple garbage collector koleksiyonun performansını düzenlemek için bir miktar hafızayı sürekli olarak boş tutar. Uygulamalar ve işlemler yapılırken hafıza dolarsa otomatik olarak hafızayı temizlemek için kullanılmayan nesneleri siler. Basit bir örnek verirsek;

```
Dim i As Integer
   i=textbox1.text
    msgbox(i)
              i=5
```

programında i değişkeninin ilk değeri textbox1 nesnesine yazılan bilgidir. Daha sonra i değişkenine 5 değeri aktarınca önceki bilgi silinir. İşte önceki bilginin silinmesini sağlayan garbage collection’dur. Aynı şekilde nesneleri de kontrol eder, kullanılmayan nesneleri otomatik olarak siler.

4.10. “Nothing Deyimi”

Herhangi bir veri türünün veya objenin default (varsayılan) değerini geri döndürür. Değişkene daha önce hangi bilgi aktarılmış olursa olsun değeri default olarak belirlenir (sıfırlanır).

Örnek:

```
Dim i As Integer
          i = 155
         MsgBox(i)
          i = Nothing
         MsgBox(i)
```

Burada i değişkeni integer tipte tanımlanmıştır. 155 değeri i değişkenine yüklenir, msgbox komutu yardımı ile bu bilgi ekrana aktarılır. Daha sonra i değişkenin değeri Nothing deyimi kullanılarak default değerini alır, yani sıfır olur. Ekrana gelen ikinci mesajda “0” bilgisi görülür. Bu deyim yardımı ile değişkenlere bilgi aktarılıp aktarılmadığı da kontrol edilebilir.

```
If i IsNot Nothing Then i = 0 // i değişkeninin değeri değişmişse sıfırla//

If i Is Nothing Then i = 15 // i değişkeninin değeri değişmemişse 15 değerini ata//
```

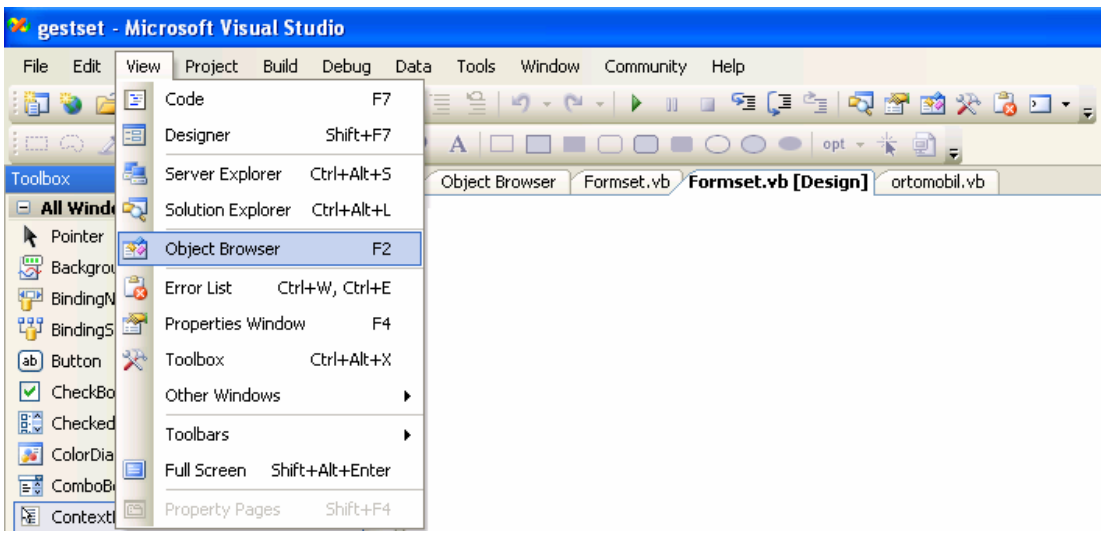
4.11. Object Browser Penceresi

Object browser, .NET nesnelerini, sınıflarını ve nesneler aralarındaki ilişkileri gösteren penceredir. Browser'ı programımızda mevcut olan nesneler hakkında bilgi edinmek için kullanabiliriz.

Object Browser'ı .NET Framework'ün yapısını, sınıf hiyerarşisini ve sınıf üyelerini öğrenmek, nesnenin özellik, olay ve metotlarını ayırt etmek için kullanırız. Genel anlamda Object Browser kullanıcının görsel programla dilini keşfetmesini sağlar.

Object Browser'ı açmak için birkaç metot vardır. Bunlar;

- VB .NET View menüsünden resim 4.9'da görüldüğü gibi Object Browser seçeneğini tıklamak.

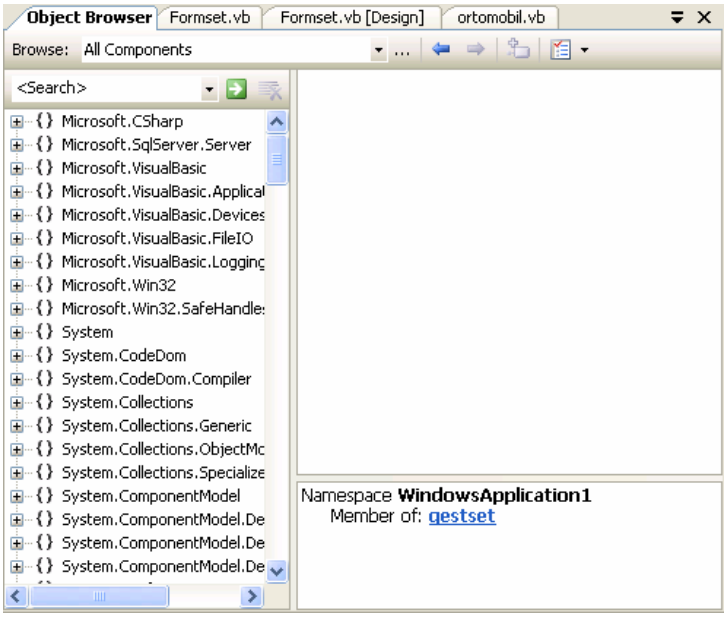


Resim 4.9: Object Browser'ın açılışı

- **Ctrl + Alt + J** veya F2 kısayol tuşlarını kullanmak.
- Kod editöründe imleci herhangi bir nesnenin üzerine getiriniz, sağ tıklayıp Go To Definition öğesini seçiniz.

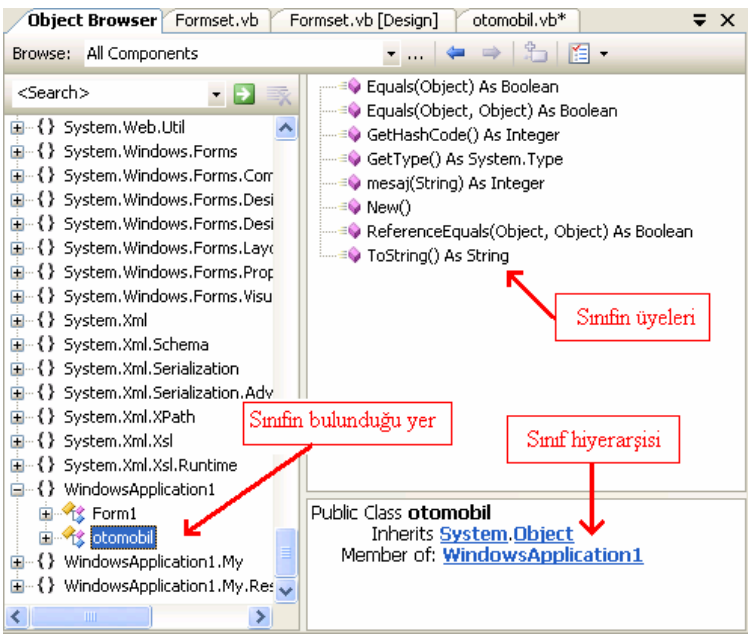
Object Browser penceresi açıldığında karşımıza aşağıdaki pencere çıkar.

Araştırma : Visual Basic.Net programlama dilindeki sınıfları araştırınız.



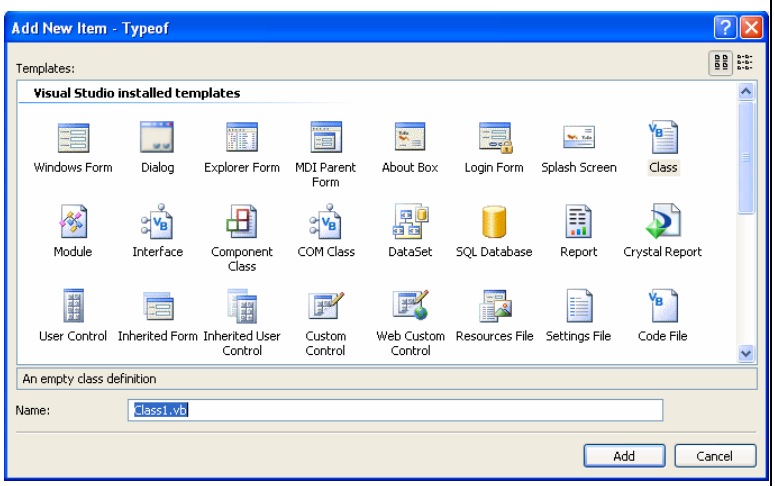
Resim 4.10: Object Browser penceresi

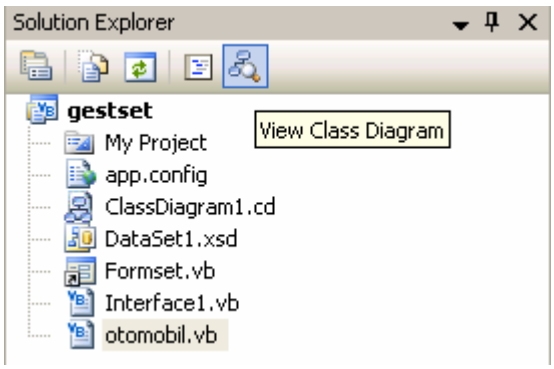
Browser yardımı ile .NET diline ait tüm sınıflar, nesneler, sınıflara ait özellikler ve metotlar görülebilir. Ayrıca istediğimiz sınıfı, nesneyi veya özelliği rahatlıkla aratabiliriz. Kendi oluşturduğumuz sınıf, metot ve özelliklere de browser yardımı ile ulaşabiliriz. Mesela; daha önce oluşturduğumuz otomobil isimli sınıfı ve özelliklerini görüntüleyelim. Önce Object Browser açılır, WindowsApplication1 şablonundan otomobil sınıfı bulunur.



Resim 4.11: Object Browser'da oluşturduğumuz sınıfın görünümü

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Yeni bir sınıf tanımlayınız. Oluşturduğunuz sınıf içerisinde bir nesne tanımlayınız.	Her şey döngü halinde olmak zorunda değildir. Tekrar eden işlemleri döngü haline getiriniz. Mesela: ekrana 3'er 3'er, 1'den 20'ye kadar sayıları yazdırınız.
2. Nesnelere ulaşıp özelliklerini değiştiriniz.	
3. Project menüsünden Add Class komutu ile sınıf oluşturunuz. 	Sınıf oluştururken istediğiniz ismi verebilirsiniz.
4. Oluşturulan sınıf içerisine özellik (Properties) ve alt program (metot) yazınız.	Oluşturduğunuz nesneleri public tanımlarsanız her yerde kullanabilirsiniz..

5.	Solution Explorer penceresindeki Class View sekmesini kullanarak oluşturulan sınıfı inceleyiniz.  	Özellik ve metotların simgelerini inceleyiniz.

ÖLÇME VE DEĞERLENDİRME

A- OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; ilk 7 soru doğru / yanlış sorularıdır. Verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorularda uygun seçeneği işaretleyiniz.

1. Class…End Class bloğu ile nesne oluşturabiliriz. ( )

2. Sınıflara nesne eklemek için Get komutunu kullanabiliriz. ( )

3. Set…End Set bloğu sadece okunabilir özelliği temsil etmektedir. ( )

4. Function…End Function bloğu ile alt program oluşturabiliriz. ( )

5. Garbage Collection bellek düzenlemek için kullanılır. ( )

6. Nothing deyimi ile arabirim oluşturabiliriz. ( )

7. Aşağıdakilerden hangisi nesne değildir?
A) name
B) textbox
C) label
D) combobox

8. Bir arabirimde aşağıdakilerden hangisi bulunmaz?
A) olay
B) özellik
C) metot
D) sınıf

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

MODÜL DEĞERLENDİRME

PERFORMANS TESTİ (YETERLİK ÖLÇME)

Modül ile kazandığınız yeterliği aşağıdaki kriterlere göre değerlendiriniz.

DEĞERLENDİRME ÖLÇÜTLERİ	Evet	Hayır
Programda hata oluşturabilecek satırları Try bloğu içersine, hata oluştuğunda çalışacak komutları Cath bloğu içine aldınız mı?		
Hatadan bağımsız çalıştırılacak komutlar için Finally bloğu oluşturdunuz mu?		
Hata oluşunca messagebox komutu ile ekrana mesaj yazdırdınız mı?		
Throw New komutu kullanılarak hata oluşturdunuz mu?		
Bir sınıfın veya yapının metot ve özelliklerini kulandınız mı?		
Veri türlerinin alabileceği değerleri ekrana yazdırdınız mı?		
Veri türlerini birbirine dönüştürdünüz mü?		
Bir veriyi string türe çevirdiniz mi?		
Bir metni sayıya çevirdiniz mi?		
Bir yatırımın gelecekteki değerini FV (Future Value) komutu ile hesapladınız mı?		
"With…End With" komutları arasına nesne ismini yazmadan kod yazdınız mı?		
Döngü kurarak form üzerinde yerleştirilmiş olan nesnelerin özelliklerini değiştirdiniz mi?		
TypeOf…Is komutu ile değişken tipi kontrolü yaptınız mı?		
Yeni bir sınıf tanımlayınız. Oluşturduğunuz sınıf içerisinde bir nesne tanımladınız mı?		
Nesnelere ulaşıp özelliklerini değiştirdiniz mi?		
Project menüsünden Add Class komutu ile sınıf oluşturdunuz mu?		
Oluşturulan sınıf içerisine özellik (Properties) ve alt program (metod) yazdınız mı?		
Solution Explorer penceresindeki Class View sekmesini kullanarak oluşturulan sınıfı incelediniz mi?		

DEĞERLENDİRME

Yaptığınız değerlendirme sonucunda eksikleriniz varsa öğrenme faaliyetlerini tekrarlayınız.

Modülü tamamladınız, tebrik ederiz. Programlamada önemli bir konu olduğundan belli zamanlarda bu modülü tekrar gözden geçiriniz.

Öğretmeniniz size çeşitli ölçme araçları uygulayacaktır. Öğretmeninizle iletişime geçiniz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1 CEVAP ANAHTARI

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

ÖĞRENME FAALİYETİ-2 CEVAP ANAHTARI

1	D
2	Y
3	D
4	Y
5	Y
6	D
7	Y
8	D
9	Y
10	B
11	C
12	D
13	A
14	C

ÖĞRENME FAALİYETİ-3 CEVAP ANAHTARI

1	D
2	Y
3	D
4	Y
5	D
6	D
7	D
8	D
9	D
10	Y

ÖĞRENME FAALİYETİ-4 CEVAP ANAHTARI

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

Cevaplarınızı cevap anahtarları ile karşılaştırarak kendinizi değerlendiriniz.

ÖNERİLEN KAYNAKLAR

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

KAYNAKÇA

- BOWMA Richard, **Visual Basic.NET**, Hungry Minds inc., New York 2002.
- DAVIS Harold, **Visual Basic.NET Programlama Kılavuzu**, ALFA Yayınevi, İstanbul 2002.
- EVJEN Bill, Jason Beres, **Visual Basic.NET Bible,** Hungry Minds inc., New York 2002.
- GRUNDGEIGER Dave, **Programming Visual Basic.NET**, O'Reillly 2002.
- HALVORSON Michael, **Adım Adım Microsoft Visual Basic.Net**, Arkadaş Yayınevi, Ankara 2002.
- HALVORSON Michael, **Adım Adım Microsoft Visual Basic 6.0 Professional,** Arkadaş Yayınevi, Ankara 2002.
- PALA Zeydin, **Microsoft Visual Basic.NET**, Türkmen Kitapevi, 2003.