Metodlar

C# İçinde Program Modülleri
`Math` Sınıfı Metodları
Metodlar
Metod Tanımlamaları
C# Alan Adları
Parametre Gönderme : Call-by-Value ve Call-by-Reference
Tanımlayıcıların Süreleri
Rastgele Sayı Üretme
Scope Kuralları
Method Overloading

C# İçindeki Program Modülleri

- Modüller
  - Sınıf
  - Metod
  - Sınıfların veya Metodların sadece ne yaptıklarını bilerek ve nasıl çalıştığını bilmeye gerek kalmadan kullanmak

C# İçindeki Program Modülleri

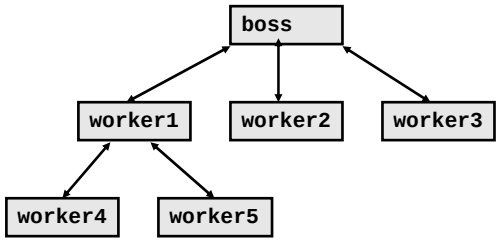


Fig. 6.1 Hiyerarşik olarak boss metodu ile worker metodu arasındaki ilişki.

`Math` Sınıfı Metodları

- **`Math`** sınıfı
  - Kullanıcıların matematiksel hesaplamaları yapmalarını sağlar
  - Metoldarın kullanımı
    - *ClassName.MethodName*(*argument1*, *arument2*, …)
  - Sabitler (Constants)
    - **`Math.PI`** = 3.1415926535…
    - **`Math.E`** = 2.7182818285…

Metod	Tanım	Örnek

Abs(x)	Mutlak değer ***x***	`Abs( 23.7 )` = `23.7`
		`Abs( 0 )` = `0`
		`Abs( -23.7 )` = `23.7`

Ceiling(x)	rounds ***x*** değerini ***x*** ' den büyük en küçük tamsayıya yuvarlar	`Ceiling( 9.2 )` = `10.0` `Ceiling( -9.8 )` = `-9.0`
Cos(x)	Kosinüs (***x*** radyan)	`Cos( 0.0 )` = `1.0`
Exp(x)	exponential	`Exp( 1.0 )` yaklaşık `2.7182818` `Exp( 2.0 )` yaklaşık `7.389056098`
Floor(x)	rounds ***x*** to the largest integer not greater than ***x***	`Floor( 9.2 )` = `9.0` `Floor( -9.8 )` = `-10.0`
Log(x)	e tabanında doğal logaritma	`Log(2.7182818)` yaklaşık `1.0` `Log(7.3890560)` yaklaşık `2.0`
Max(x, y)	***x*** ve ***y*** den büyük olana eşittir (`float`, `int` ve `long` değer alabilir)	`Max( 2.3, 12.7 )` = `12.7` `Max( -2.3, -12.7 )` = `-2.3`
Min(x, y)	***x*** ve ***y*** den küçük olana eşittir (`float`, `int` ve `long` değer alabilir)	`Min( 2.3, 12.7 )` = `2.3` `Min( -2.3, -12.7 )` = `-12.7`
Pow(x, y)	***x*** değerinin ***y*** dereceden üssünü alır	`Pow( 2.0, 7.0 )` = `128.0` `Pow( 9.0, .5 )` = `3.0`
Sin(x)	sinüs (***x*** radyan)	`Sin( 0.0 )` = `0.0`
Sqrt(x)	Karekök ***x***	`Sqrt( 900.0 )` = `30.0` `Sqrt( 9.0 )` = `3.0`
Tan(x)	Tanjant (***x*** radyan)	`Tan( 0.0 )` = `0.0`

Fig. 6.2 `Math` sınıfının yaygın kullanılan meodları

Metodlar

- Değişkenler
  - Metod içinde tanımlananlar = local variables
  - Metod dışında tanımlananlar = global variables

Metod Tanımlamaları

- Özel metod yazmak
  - Header
    - ***`ReturnType`*** ***`Properties`*** *Name*(*Param1*, *Param2*, …)
  - Body
    - Metodun yaptığı işleri içeren kod bulunur
    - Gerekliyse metodun geri döndürdüğü değer bulunur
  - Programın herhangi bir yerinde çağrılabilir
    - Gerekliyse parametre gönderilebilir
  - Bütün metodların bağlı oldukları sınıf içerisinde yazılması zorunludur

```
// Fig. 6.3: SquareInt.cs
// A programmer-defined Square method.

using System;                // includes basic data types
using System.Drawing;        // for graphics capabilities
using System.Collections;    // for complex data structures
using System.Component                                     or
using System.Windows.F
using System.Data;

// form used to display results of squaring 10 numbers
public class SquareIntegers : System.Windows.Forms.Form
{
   private System.ComponentModel.Container components = null;

   // label containing results
   private System.Windows.Forms.Label outputLabel;

   public SquareIntegers()
   {
      // Required for Windows Form Designer support
      InitializeComponent();

      int result; // store result of call to method Square

```

class SquareInteger başlangıcı `System.Windows.Forms.Form`

SquareIntegers metodunun başlangıcı

Metod değişkeni. Sadece metod içinde kullanılabilir.

```
26          // loop 10 times
27          for ( int counter = 1; counter <= 10; counter++ )
28          {
29             // calculate square of counter and
30             result = Square( counter );
31
32             // append result to output string
33             outputLabel.Text += "The square of " + counter +
34                " is " + result + "\n";
35          }
36
37       } // end SquareIntegers
38
39       // Clean up any resources being used.
40       protected override void Dispose( bool disposing )
41       {
42          // Visual Studio .NET-generated code for method Dispose
43       }
44
45       // Required method for Designer support
46       private void InitializeComponent()
47       {
48          // Visual Studio .NET generated code
49          // for method InitializeComponent
50       }
51
```

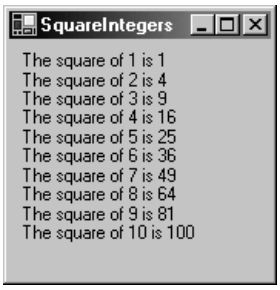
SquareIntegers metodunun body kısmı

Square adlı metodun çağrılışı. **counter** değişkeni matoda gönderiliyor ve dönen sonuç **result** değişkenine aktarılıyor

```
52       // The main entry point for the application.
53       [STAThread]
54       static void Main()
55       {
56          Application.Run( new SquareIntegers() );
57       }
58
59       // Square method definition
60       int Square( int y )
61       {
62          return y * y; // return square of y
63
64       } // end method Square
65
66    } // end of class SquareIntegers
```

Square metodu. Bir integer alır ve bir integer döndürür

Metod gelen değişkeni kendisiyle çarpıp geri gönderir



```
1    // Fig. 6.4: MaximumValue.cs
2    // Finding the maximum of three doubles.
3
4    using System;
5
6    class MaximumValue
7    {
8       // main entry point for application
9       static void Main( string[] args )
10      {
11         // obtain user input and convert to double
12         Console.Write( "Enter first floating-point value: "
);
13         double number1 = Double.Parse( Console.ReadLine() );
14
15         Console.Write( "Enter second floating-point value: "
);
16         double number2 = Double.Parse( Console.ReadLine() );
17
18         Console.Write( "Enter third floating-point value: "
);
19         double number3 = Double.Parse( Console.ReadLine() );
20
21         // call method Maximum to determine largest value
22         double max = Maximum( number1, number2, number3 );
23
24         // display maximum value
25         Console.WriteLine("\nmaximum is: " + max );
26
27      } // end method Main
```

Kullanıcı 3 değer girer

Girilen 3 değer Maximum matoduna gönderilir

```
28
29      // Maximum method uses method Math.Max to help
determine
30      // the maximum value
31      static double Maximum( double x, double y, double z )
32      {
33         return Math.Max( x, Math.Max( y, z ) );
34
35      } // end method Maximum
36
37   } // end class MaximumValue
```

Maximum metodu 3 değer alır ve en büyük olan 1 değeri gönderir

```
Enter first floating-point value: 37.3
Enter second floating-point value: 99.32
Enter third floating-point value: 27.1928

maximum is: 99.32
```

C# Alan Adları

- Alan Adı
  - Bir grup sınıf ve sınıflara ait metodlardan oluşur
  - Alan adları .dll dosyalarda saklıdır
  - Bir program içinde **`using`** anahtar kelimesiyle kullanılır

C# Alan Adları

Alan Adı	Tanımlama
`System`	Temel sınıfları ve veri tipleri bulundurur (`int`, `double`, `char`, v.b.).
`System.Data`	Veritabanı erişimi ve işlemleri için kullanılan ADO.NET sınıflarını içerir
`System.Drawing`	Çizim ve grafik işlemleri için gerekli sınıfları içerir
`System.IO`	Veri giriş ve çıkışı için gerekli sınıfları bulundurur
`System.Threading`	Multithread sınıflarını bulundurur
`System.Windows.Forms`	Grafik kullanıcı arayüzü için gerekli sınıfları içerir
`System.Xml`	XML verilerini işlemek için kullanılan sınıfları içerir

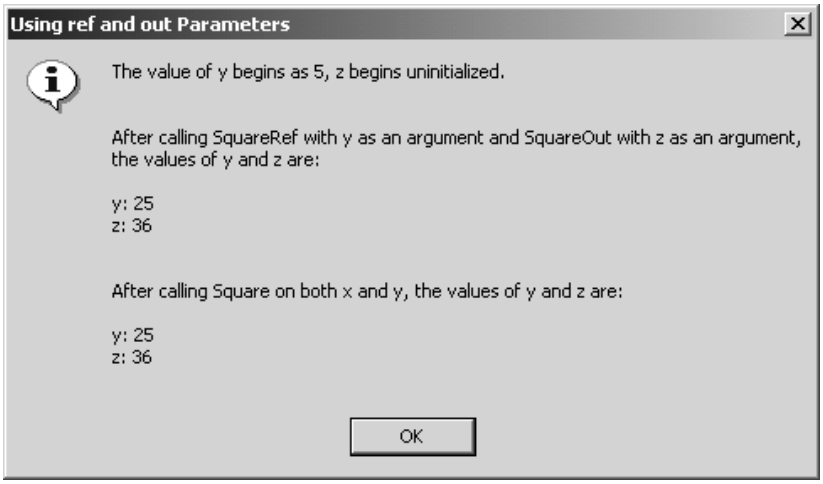
Fig. 6.6 Framework sınıf kütüphanelerindeki alan adları

Parametre Gönderme: Call-By-Value vs. Call-By-Reference

- Value ile gönderme
  - Nesnenin bir kopyası metoda gönderilir
  - Geri bir value döner
- Reference ile gönderme
  - Gerçek referans noktası gönderilir
    - Sonuçta değişkenin bütün program içindeki değeri değiştirilir
  - Geri bir referans döner
  - **`ref`** anahtar kelimesi kullanılır
  - **`out`** çağrılan metodun gönderilen değişkene başlangıç değerini vereceğini gösterir

```
1    // Fig. 6.8: RefOutTest.cs
2    // Demonstrating ref and out parameters.
3    using System;
5    using System.Windows.Forms;
6
7    class RefOutTest
8    {
9       // x is passed as a ref int (original value will change)
10      static void SquareRef( ref int x )
11      {
12         x = x * x;
13      }
14
15      // original value can be changed and initialized
16      static void SquareOut( out int x )
17      {
18         x = 6;
19         x = x * x;
20      }
21
22      // x is passed by value (original value not changed)
23      static void Square( int x )
24      {
25         x = x * x;
26      }
27
29      static void Main( string[] args )
30      {
31         // create a new integer value, set it to 5
32         int y = 5;
33         int z;   // declare z, but do not initialize it
```

```
34         // display original values of y and z
35         string output1 = "The value of y begins as "
36            + y + ", z begins uninitialized.\n\n\n";
37
38         // values of y and z are passed by value
39         RefOutTest.SquareRef( ref y );
40         RefOutTest.SquareOut( out z );
41
42         // display values of y and z after modified by methods
43         // SquareRef and SquareOut
44         string output2 = "After calling SquareRef with y as an " +
45            "argument and SquareOut with z as an argument,\n" +
46            "the values of y and z are:\n\n" +
47            "y: " + y + "\nz: " + z + "\n\n\n";
48
49         // values of y and z are passed by value
50         RefOutTest.Square( y );
51         RefOutTest.Square( z );
52
53         // values of y and z will be same as before because Square
54         // did not modify variables directly
55         string output3 = "After calling Square on both x and y, " +
56            "the values of y and z are:\n\n" +
57            "y: " + y + "\nz: " + z + "\n\n";
58
59         MessageBox.Show( output1 + output2 + output3,
60            "Using ref and out Parameters", MessageBoxButtons.OK,
61            MessageBoxIcon.Information );
62
63      } // end method Main
64
65   } // end class RefOutTest
```



Rastgele Sayı Üretme

- **`Random`** sınıfı
  - **`System`** alan adı içinde
  - Gerçek random
    - Günün kesin zamanını kullanan bir fonksiyonla oluşturulur
  - **`randomObject.Next`**()
    - **`Int32.MaxValue`** aralığında bir rastgele sayı üretir
      - Int32.MaxValue = 2,147,483,647
  - **`randomObject.Next`**(*x*)
    - 0 ile *x* aralığında bir sayı üretir
  - **`randomObject.Next`**(*x*, *y*)
    - *x* ile *y* arasında bir sayı üretir

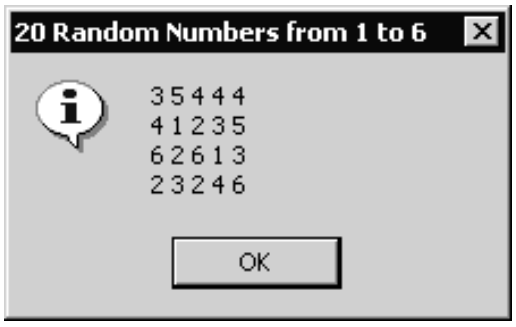
```
1    // Fig. 6.9: RandomInt.cs
2    // Random integers.
3
4    using System;
5    using System.Windows.Forms;
6
7    // calculates and displays 20 random integers
8    class RandomInt
9    {
10      // main entry point for application
11      static void Main( string[] args )
12      {
13         int value;
14         string output = "";
15
16         Random randomInteger = new Random();
17
18         // loop 20 times
19         for ( int i = 1; i <= 20; i++ )
20         {
21            // pick random integer between 1 and 6
22            value = randomInteger.Next( 1, 7 );
23            output += value + " "; // append value to output
24
25            // if counter divisible by 5, append newline
26            if ( i % 5 == 0 )
27               output += "\n";
28
29         } // end for structure
30
```

Yeni bir Random nesnesi üretir

1 ile 7 arasında 7 hariç bir rastgele sayı üretir

Her satırda 5 karakter yazılır

```
31          MessageBox.Show( output, "20 Random Numbers from 1 to 6",
32             MessageBoxButtons.OK, MessageBoxIcon.Information );
33
34       } // end Main
35
36    } // end class RandomInt
```



Tanımlayıcıların süreleri

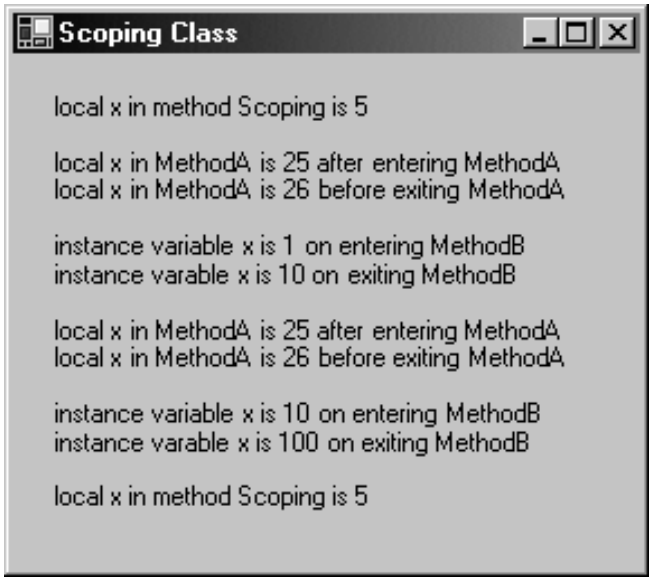
- Süre (Duration)
  - Tanımlayıcının hafızada kalma süresi
- Scope
  - Program içindeki bir nesnenin referans edildiği kısımdır
- Local variables
  - Tanımlanırken oluşturulur
  - Tanımlandığı bloktan çıkarken hafızadan atılır

Scope Kuralları

- Scope
  - Programın bir değişkene erişim yapılabilen kısmıdır
  - Class scope
    - class oluşturulduktan sonra başlar
    - class sonlanana kadar (}) devam eder
    - class içindeki tüm metodlar için global
  - Block scope
    - Blok oluşturulduktan sonra başlar
    - Blok sonlanana (}) kadar devam eder
    - Sadece tanımlandığı blok içinde kullanılır
    - Blok içinde aynı isimle ikinci bir değişken tanımlanamaz

```
1    // Fig. 6.13: Scoping.cs
2    // A Scoping example.
4    using System;
5    using System.Drawing;
6    using System.Collections;
7    using System.ComponentModel;
8    using System.Windows.Forms;
9    using System.Data;
10
11   public class Scoping : System.Windows.Forms.Form
12   {
13      private System.ComponentModel.Container components = null;
14      private System.Windows.Forms.Label outputLabel;
15
16      public int x = 1;
17
18      public Scoping()
19      {
20         InitializeComponent();
21
22         int x = 5;     // variable local to constructor
23
24         outputLabel.Text = outputLabel.Text +
25            "local x in method Scoping is " + x;
27         MethodA();    // MethodA has automatic local x;
28         MethodB();    // MethodB uses instance variable x
29         MethodA();    // MethodA creates new automatic local x
30         MethodB();    // instance variable x retains its value
31
32         outputLabel.Text = outputLabel.Text +
33            "\n\nlocal x in method Scoping is " + x;
34      }
```

```
36      // Visual Studio .NET-generated code
37
38      public void MethodA()
39      {
40         int x = 25;    // initialized each time a is called
41
42         outputLabel.Text = outputLabel.Text +
43            "\n\nlocal x in MethodA is " + x +
44            " after entering MethodA";
45         ++x;
46         outputLabel.Text = outputLabel.Text +
47            "\nlocal x in MethodA is " + x +
48            " before exiting MethodA";
49      }
51      public void MethodB()
52      {
53         outputLabel.Text = outputLabel.Text +
54            "\n\ninstance variable x is " + x +
55            " on entering MethodB";
56         x *= 10;
57         outputLabel.Text = outputLabel.Text +
58            "\ninstance varable x is " + x +
59            " on exiting MethodB";
60      }
62      // main entry point for application
63      [STAThread]
64      static void Main()
65      {
66         Application.Run( new Scoping() );
67      }
68
69   } // end of class Scoping
```

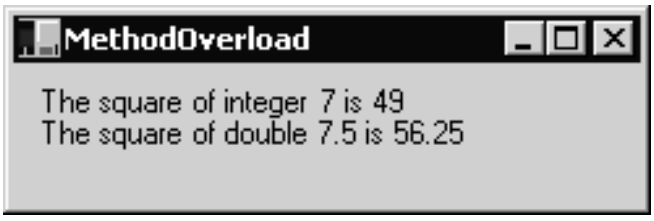


Method Overloading

- Aynı isme sahip olan metodlar
  - Aynı isme sahiptirler ancak farklı parametreler alırlar
    - Aldıkları değişkenler farklı olmak zorundadır
  - Genellikle aynı işi farklı veri tipleriyle yaparlar
    - Farklı veri tipleri kullanırlar

```
1    // Fig. 6.18: MethodOverload.cs
2    // Using overloaded methods.
3
4    using System;
5    using System.Drawing;
6    using System.Collections;
7    using System.ComponentModel;
8    using System.Windows.Forms;
9    using System.Data;
10
11   public class MethodOverload : System.Windows.Forms.Form
12   {
13      private System.ComponentModel.Container components = null;
14
15      private System.Windows.Forms.Label outputLabel;
16
17      public MethodOverload()
18      {
19         InitializeComponent();
20
21         // call both versions of Square
22         outputLabel.Text =
23            "The square of integer 7 is " + Square( 7 ) +
24            "\nThe square of double 7.5 is " + Square ( 7.5 );
25      }
26
27      // Visual Studio .NET-generated code
28
```

```
29      // first version, takes one integer
30      public int Square ( int x )
31      {
32         return x * x;
33      }
34
35      // second version, takes one double
36      public double Square ( double y )
37      {
38         return y * y;
39      }
40
41      [STAThread]
42      static void Main()
43      {
44         Application.Run( new MethodOverload() );
45      }
46
47   } // end of class MethodOverload
```

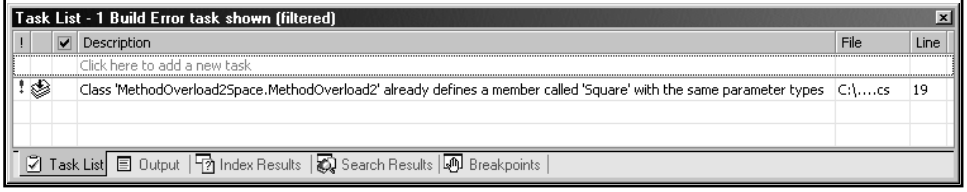



```
1    // Fig. 6.19: MethodOverload2.cs
2    // Overloaded methods with identical signatures and
3    // different return types.
5    using System;
7    class MethodOverload2
8    {
9       public static int Square( double x )
10      {
11         return x * x;
12      }
14      // second Square method takes same number,
15      // order and type of arguments, error
16      public static double Square( double y )
17      {
18         return y * y;
19      }
21      // main entry point for application
22      static void Main()
23      {
24         int squareValue = 2;
25         Square( squareValue );
26      }
28   } // end of class MethodOverload2
```

integer döndürür

double döndürür

Giden değerle hangisinin çalışacağı bilinemez ve hata verir



Haftalık Ödev

Bir otoparkta otomobil için 2YTL, kamyon için 5YTL ve otobüs için 8YTL ücret alınmaktadır. Kar oranları otomobil için %25, kamyon için %30 ve otobüs için %35 olarak belirlenmiştir. Programın tüm işlemlerine aşağıdaki gibi bir ana menü ile ulaşılmaktadır. Örneğin araç girişi için 1, Toplam araç sayıları için 2 gibi seçimlerle ilgili işlemler yapılmaktadır. Araç girişinde otomobil için "O", kamyon için "K" ve otobüs için "B" girilmektedir. Araç girişinden çıkmak için "C" harfi girilmektedir. Her araç girildiğinde yeni araç türü girilmesi istenmekte ve çıkış için "C" girilmesi gerektiği kullanıcıya bildirilmektedir. "C" ile çıkıldığında ekrana ana menü gelecektir. Aşağıdaki tüm işlemleri yapan bir program yapınız.

1-Araç Girişi
2-Toplam otomobil, kamyon ve otobüs sayıları
3-Toplam otomobil, kamyon ve otobüs ciroları
4-Otomobil, kamyon ve otobüs cirolarını yüzdelik dağılımları
5-Toplam otomobil, kamyon ve otobüs kar miktarları
6-Otomobil, kamyon ve otobüs kar miktarlarını yüdelik dağılımları
7-Toplam araç sayısı
8-Toplam gelir miktarı
9-Toplam kar miktarı
10-Programdan çıkış

Lütfen Seçiminizi Giriniz (1..10)

Not:
Program menü geçişlerinde Console ekranını temizleyecek ve istenen işlem yapıldıktan sonra ana menüye geçiş için kullanıcıdan bir tuşa basması istenecek. Ana menüye geçişte ekran temizlenip ana menü ekrana getirilecektir. Ekran temizlemek için Web Sayfamda Downlods bölümünde bulunan ClearConsole sınıfını programınızın alan adı içerisine yeni bir class olarak eklemeniz ve bir değişkenle kullanmanız gerekmektedir.