

Görsel Programlama Dilleri

C# ile Görsel Programlama

Dr. Öğr. Üyesi Hayri Volkan Agun

Kaynaklar:

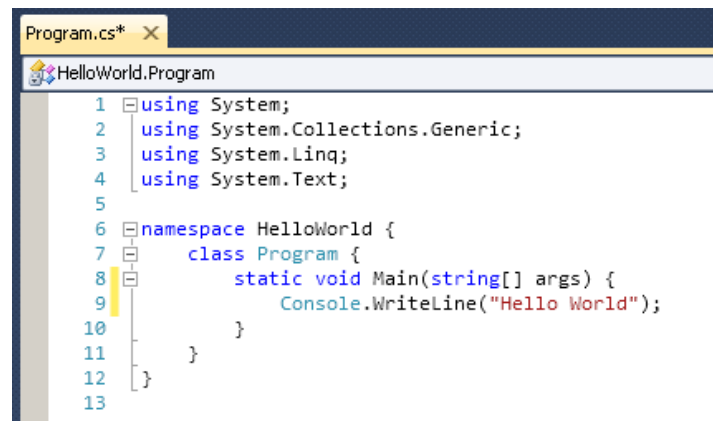
[Doç. Dr. Erdinç Uzun - Ders Notları](#)

C# Kod Geliştirme Ortamları

- Visual Studio .NET 2003, 2005, 2008, 2010, 2012, 2013, 2015 ...
- ASP.NET Web Matrix
- SharpDevelop
- PrimalCode
- Antchechinus C# Editor
- Eclipse
- Visual SlickEdit
- MonoDevelop

Merhaba C#

- Sınıf bilgisi
 - Using
- Main fonksiyonu
- Console
 - Konsol uygulamaları için standart giriş ve çıkış hata akışlarına temsil eder.
- Açıklama satırı yazma
 - `//` ve `/* ... */`

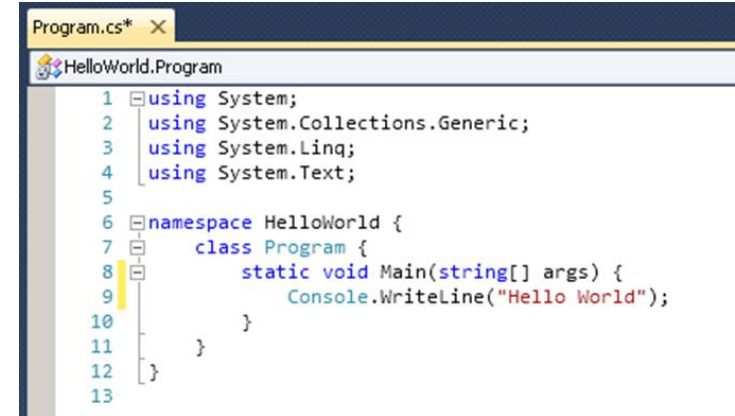


The screenshot shows a code editor window titled 'Program.cs*' with a sub-header 'HelloWorld.Program'. The code is as follows:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace HelloWorld {
7     class Program {
8         static void Main(string[] args) {
9             Console.WriteLine("Hello World");
10        }
11    }
12 }
13
```

Using

Using kelimesi sıklıkla isim uzayları (namespace) içerisindeki sınıfların kod içerisinde kolayca kullanılabilmesini sağlamak amacıyla kullanılır.



```
Program.cs* X
HelloWorld.Program
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace HelloWorld {
7     class Program {
8         static void Main(string[] args) {
9             Console.WriteLine("Hello World");
10        }
11    }
12 }
13
```

Command Ekranından Derlenmiş Kodun Çalıştırılması

İşletim sisteminin RUN bölümüne cmd yazıp veya Donatılar bölümünden console uygulamasını bulup çalıştırın.

Programınızın kaydedildiği yerde Debug ve Bin dizinini gidin.

- cd.. Üst dizine geçmek.
- cd dizin_ismi: belirlenen dizine gitmek
- dir: dizin içindeki dizin ve dosyaları görüntülemek
- Bin dizinine ulaştınca exe uzantılı dosyanın ismini yazıp enter'a basın.

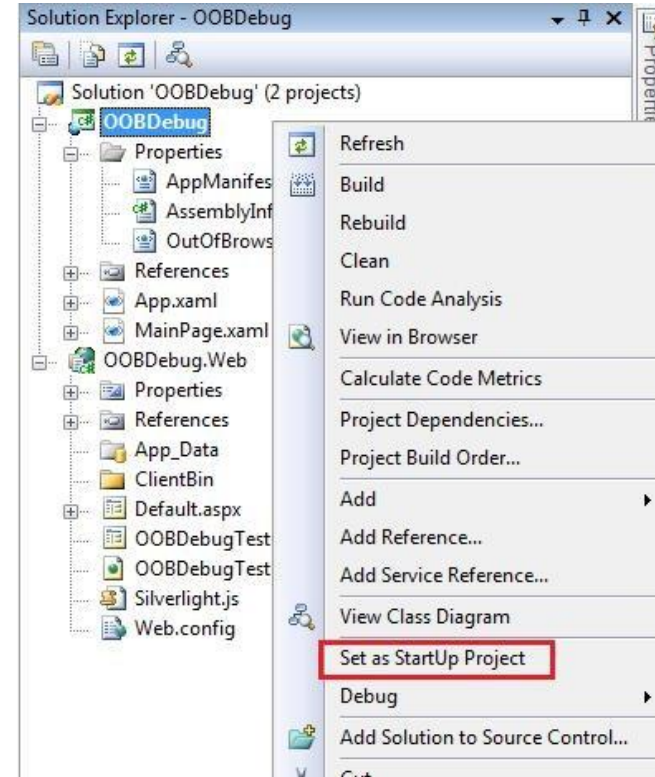
Birden fazla proje ile çalışma

Solution Explorer'dan tüm projelerinizi görebilirsiniz.

Solution üzerine gelip farenin sağ tuşu ile yeni bir proje eklenebilir. (Add->New Project)

Proje üstüne gidip «Set as Startup Project» ile çalışacak proje belirlenir.

Derste her yazılan kod için bir proje oluşturulacaktır. Evde çalışırken seçili projeyi değiştirmeniz gerektiğini unutmayın.

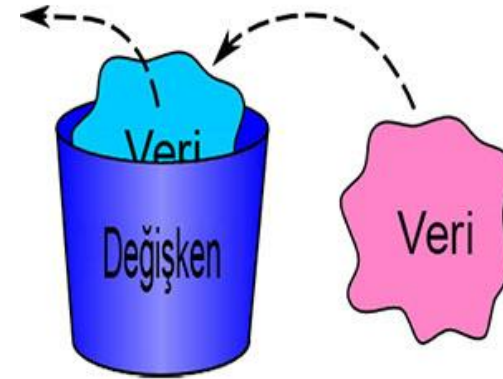


Veri Türleri ve Değişken

Değişken, programın çalışması için gerekli verilerin tanımlanarak, bellek üzerinde tutulduğu bölgelere verilen isimlerdir.

Tanımlama o bellek bölgesinde tutulacak olan verinin türünün belirtilmesidir.

Bir değişken tanımlandıktan sonra aynı türden değer atamak koşuluyla değeri değiştirilebilir.



Veri Türleri

Veri Türü	CTS Karşılığı	Açıklama	Max ve Min aralık yada değeri
sbyte	System.Byte	8 bit işaretli tamsayı	-128 : 127
short	System.Int16	16 bit işaretli tamsayı	-32.768 : 32.767
int	System.Int32	32 bit işaretli tamsayı	-2.147.483.648 : 2.147.483.647
long	System.Int64	64 bit işaretli tamsayı	-9.223.372.036.854.775.808 : -9.223.372.036.854.775.807
byte	System.Byte	8 bit işaretsiz tamsayı	0 : 255
ushort	System.UInt16	16 bit işaretsiz tamsayı	0 : 65.535
uint	System.UInt32	32 bit işaretsiz tamsayı	0 : 4.294.967.295
ulong	System.UInt64	64 bit işaretsiz tamsayı	0 : 18.446.744.073.709.551.615
float	System.Single	32 bit tek kayan sayı	+yada - $1,5 \cdot 10^{-45}$: + ya da - $3,4 \cdot 10^{38}$
double	System.Double	64 bit çift kayan sayı	+yada - $5 \cdot 10^{-324}$: + ya da - $1,7 \cdot 10^{308}$
decimal	System.Decimal	128 bit ondalıklı sayı	+yada - $1,5 \cdot 10^{-28}$: + ya da - $7,9 \cdot 10^{28}$
bool	System.Boolean		true ya da false
char	System.Char	Karakterleri temsil eder	16 Unicode karakterleri
object	System.Object	Bütün veri türlerinin türediği kök eleman	
string	System.String	Unicode karakterlerinden oluşan string	

Değişken Tanımlama

<DeğişkenTürü> <DeğişkenAdı>;

<DeğişkenAdı> = <DeğişkenDeğeri>;

//veya

<DeğişkenTürü> <DeğişkenAdı> = <DeğişkenDeğeri>;

Değişken Tanımlama Örnekleri

`int sayi = 5;`

`int x, y = 8, z; //Aynı türdeki değişkenler aynı anda tanımlanabilir.`

`bool aktif = true;`

`float f = 5.4f; // Değerin sonuna eklediğimiz f harfi değişkenin float türünde olduğunu gösterir.`

`double d = 3.2;`

`long l = 123456789;`

`short s = -312;`

`decimal dec = -5.26m; //Değerin sonundaki m harfi değişkenin decimal türünde olduğunu gösterir.`

`char ch = 'c'; //Char tipinde ki değişkenler tek tırnak içerisinde yazılır.`

`string deger = "merhaba";`

Const ve Readonly

Uygulama çalışırken değeri değiştiremeyecek değişken tanımlamamızı sağlarlar.

Const (Sabit): Class seviyesinde tanımlanır ve tanımlanma anında değeri verilmek zorundadır. Sonradan değeri değiştirilemez.

Readonly (Sadece-Okunabilir): Class seviyesinde tanımlanır. Tanımlandığı anda değeri verilebilir veya Class Constructor'ında değeri verilebilir. Sonradan değeri değiştirilemez.

Veri Türleri Arasında Dönüşüm

`X = Y` (bilinçsiz dönüşüm-implicit type conversion)

`X = (int) Y` (bilinçli dönüşüm-explicit type conversion)

`X = System.Convert.ToString(Y);`

Bilinçsiz Dönüşüm (implicit type conversion)

C#'ta düşük kapasiteli bir değişken, sabit ya da değişken ve sabitlerden oluşan matematiksel ifade daha yüksek kapasiteli bir değişkene atanabilir. Buna bilinçsiz tür dönüşümü denir, bunun için herhangi bir özel kod gerekmez.

```
using System;
class TurDonusumu
{
    static void Main()
    {
        byte a=5;
        short b=10;
        sbyte c=30;
        int d=a+b+c;
        string e="deneme";
        char f='k';
        object g=e+f+d;
        long h=d;
        float i=h;
        double j=i;
        double k=12.5f;
        Console.WriteLine(j+k);
    }
}
```

```
using System;
class TurDonusumu
{
    static void Main()
    {
        char a='h';
        int b=a;
        Console.WriteLine(b);
    }
}
```

```
using System;
class TurDonusumu
{
    static void Main()
    {
        byte a=5, b=10;
        short d=2, e=9;
        byte f=a+b;
        short g=d+e;
        Console.WriteLine(f+g);
    }
}
```

Bilinçli Dönüşüm (explicit type conversion)

Bilinçli tür dönüşümü genellikle derleyicinin izin vermediği durumlarda kullanılır.

Bilinçli tür dönüşümüyle küçük türün büyük türe dönüştürülmesi sağlanabilse de aslında bu gereksizdir, çünkü aynı şeyi bilinçsiz tür dönüşümüyle de yapabilirdik.

```
using System;
class TurDonusumu
{
    static void Main()
    {
        int a=5;
        byte b=(byte)a;
        Console.WriteLine(b);
    }
}
```

```
using System;
class TurDonusumu
{
    static void Main()
    {
        byte b=(byte)12.5f;
        Console.WriteLine(b);
    }
}
```

Bilinçli Dönüşüm

- object türündeki bir değişkene başka herhangi bir türdeki değişken ya da sabit (string dışında) + işaretiyle eklenemez.
- Hatalı Durum

```
using System;
class TurDonusumu
{
    static void Main()
    {
        object a='5';
        int b=4;
        Console.WriteLine(a+b);
    }
}
```

Bilinçli Dönüşüm

- object türündeki bir değişkene herhangi bir türdeki değişken ya da sabit atanabilir (bilinçsiz tür dönüşümü). Ancak object türündeki herhangi bir değişkeni başka bir türe çevirmek için tür dönüştürme işlemi kullanılır. Ayrıca da dönüştürülen türlerin uyumlu olması gerekir. Yani object türüne nasıl değer atandığı önemlidir.

- Hatalı Durum

```
using System;
class TurDonusumu
{
    static void Main()
    {
        object a="5";
        int b=(int)a;
        Console.WriteLine(b);
    }
}
```


Bilinçli Dönüşüm

- object türündeki bir değişkene herhangi bir türdeki değişken ya da sabit atanabilir (bilinçsiz tür dönüşümü). Ancak object türündeki herhangi bir değişkeni başka bir türe çevirmek için tür dönüştürme işlemi kullanılır. Ayrıca da dönüştürülen türlerin uyumlu olması gerekir. Yani object türüne nasıl değer atandığı önemlidir.

- Hatalı

```
using System;
class TurDonusumu
{
    static void Main()
    {
        object a="5";
        int b=(int)a;
        Console.WriteLine(b);
    }
}
```

- Doğru

```
using System;
class TurDonusumu
{
    static void Main()
    {
        object a='k';
        char b=(char)a;
        Console.WriteLine(b);
    }
}
```

Bilinçli Dönüşüm

- string türüyle ilgili dönüşümler

```
using System;
class TurDonusumu
{
    static void Main()
    {
        int a=6;
        string b=a.ToString();
        Console.WriteLine(b);
    }
}
```

Tür Dönüşüm

- System isim alanının altındaki Convert sınıfı içinde tür dönüşümü yapabileceğimiz birçok metot bulunur. Bu metotlarla hemen hemen her türü her türe dönüştürebiliriz. Ancak bunun için değişken türlerinin CTS karşılıklarını bilmeliyiz. Değişken türleri ve CTS karşılıkları aşağıdaki tabloda listelenmiştir.

Tür	CTS karşılığı
bool	Boolean
byte	Byte
sbyte	Sbyte
short	Int16
ushort	UInt16
int	Int32
uint	UInt32
long	Int64
ulong	UInt64
float	Single
double	Double
decimal	Decimal
char	Char

Convert.ToBoolean(x)
Convert.ToByte(x)
Convert.ToSbyte(x)
Convert.ToInt16(x)
Convert.ToUInt16(x)
Convert.ToInt32(x)
Convert.ToUInt32(x)
Convert.ToInt64(x)
Convert.ToUInt64(x)
Convert.ToSingle(x)
Convert.ToDouble(x)
Convert.ToDecimal(x)
Convert.ToChar(x)

Bir string bir integer dönüştürme yöntemleri

`Int32.Parse(string s)`

`Convert.ToInt32(string s)`

`Int32.TryParse(string s, out int result)`

Int32.Parse

- Int32.Parse metodunun 3 adet Exception'ı bulunur. Bunlar ArgumentNullException, FormatException ve OverflowException'dır.
 - ArgumentNullException: Giriş değeri null olduğu zaman gerçekleşir.
 - FormatException: Giriş değeri doğru formatta olmadığı zaman gerçekleşir. Örneğin; alfanümerik bir değer gibi.
 - OverflowException: Giriş değeri Int32 sınırlarını aştığında gerçekleşir.
- Try catch ile exception durumlar yakalanabilir. Try catch ileri ki haftalarda anlatılacaktır.

Convert.ToInt32

- Convert.ToInt32 metodunun Int32.Parse'tan en önemli farkı, yalnızca string verileri Int32 türüne çevirmek değil, bunların dışında bool, byte, char, DateTime, decimal, double, float, int, long, object, sbyte, short, string, uint, long ve short veri tiplerine de çevirebiliyor olması geliyor.
- İkinci önemli fark ise Int32.Parse'ın aksine 3 exception yerine 2 exception'a sahip olması. Bunlar FormatException ve OverflowException'dır. Fark ettiğiniz üzere ArgumentNullException Convert.ToInt32 bulunmuyor. Sebebi ise giriş verisinin null olması durumunda hata fırlatmak yerine geriye 0 değeri döndürmesinden kaynaklanıyor.

Int32.TryParse

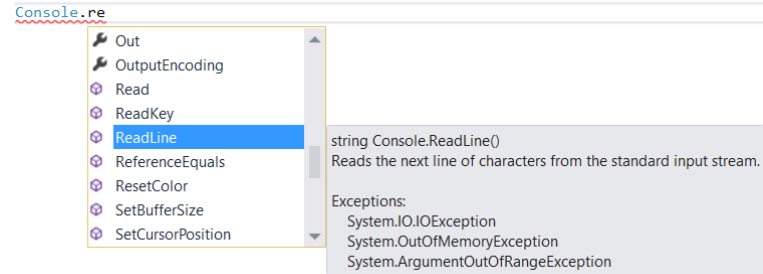
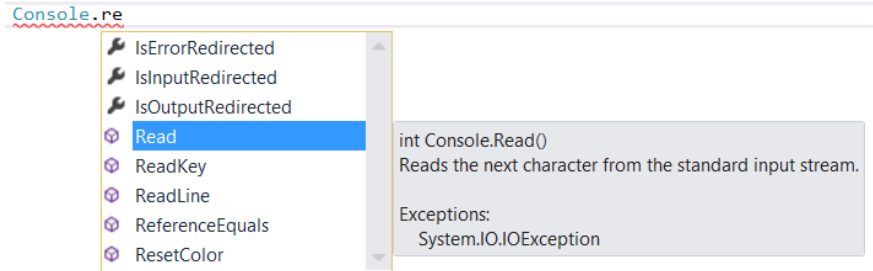
- `Int32.TryParse`; işleme sokulacak olan verinin, tipini sorgulayarak bizlere `bool` yani mantıksal bir ifade geri döndürür.
- `Int32.Parse` ve `Convert.ToInt32`'de belirttiğimiz exceptionların tümü `Int32.TryParse`'ta toplu halde bulunur. Yani `convert` yapacağınız verinin `null`, doğru formatta olup olmaması ya da `Int32` sınırlarını aşması gibi durumlarda geriye `bool` veri tipinde `false` değeri döndürür. Ancak bunu `int` veri tipinde saklar ve dolayısı ile görünen sonuç "0" olarak gönderilir.

Console Write ve WriteLine

- Console ekranına çıktı için kullanılır.
- Write işleminden sonra alt satıra geçilmez. C'deki gibi \n ile alt satıra geçilebilir.
- WriteLine işleminden sonra alt satıra geçilir.

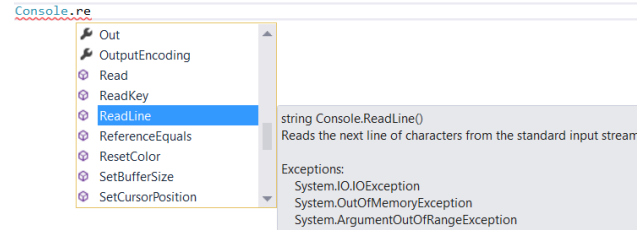
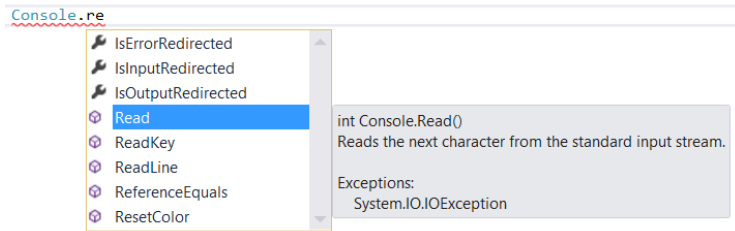
Console Read - ReadLine

- Console yazıp noktaya tıklayın ve fonksiyonların döndürdüğü değerleri bakın.



Console Read - ReadLine

- Read int türünden ve Readline'ın string türünden olduğunu dikkat edin.



Readline

```
using System;
class TurDonusumu
{
    static void Main()
    {
        string s1, s2;
        int sayi1, sayi2;
        int Toplam;
        Console.Write("Birinci sayıyı girin: ");
        s1=Console.ReadLine();
        Console.Write("İkinci sayıyı girin: ");
        s2=Console.ReadLine();
        sayi1=Convert.ToInt32(s1);
        sayi2=Convert.ToInt32(s2);
        Toplam=sayi1+sayi2;
        Console.Write("Toplam= "+Toplam);
    }
}
```

Özet

- Bu derste C# programlama dilinde tanımlanan veri türlerini ve türler arası dönüşümleri gördük.
- Tür dönüşümlerindeki hatalardan bahsettik.
- Ayrıca konsol ekranından giriş okuma ve ekrana yazmada kullanılan fonksiyonlarla bir örnek yaptık.