

Namespaces and .NET Class Library (İsim Alanları Sınıf Kütüphanesi)

- Programcıların işlerini kolaylaştırmak için bir takım hazır kütüphaneler vardır fakat C# dili ile gelen hazır bir takım kütüphaneleri yoktur.
- Bunun yerine Framework dediğimiz altyapıda bir takım temel türler ve sınıflar mevcuttur. Bu sınıf ve türleri organize edebilmek için Namespace kavramı kullanılır.

Namespaces and .NET Class Library (İsim Alanları Sınıf Kütüphanesi)

```
namespace Test1;  
    class testProgram  
    {  
    }  
namespace Test2;  
class testProgram  
{ }
```

Kullanımı: Test2.testProgram şeklinde.

Namespaces and .NET Class Library (İsim Alanları Sınıf Kütüphanesi)

- **System** isim alanı : .NET çalışırken gerekli temel sınıfları içerir. Ayrıca diğer tüm sınıf kütüphaneleri de bunu içinde kümelenmiştir. System hiyerarşinin tepesinde bulunur.
- Örneğin tüm veritabanı işlemleri için kullanılacak sınıf kütüphanesi “**System.Data**” dır.
- Bu sınıf kütüphanesi içindeki SQL ile işlemler için “System.Data.SqlClient” isim alanı mevcuttur.

Namespaces and .NET Class Library (İsim Alanları Sınıf Kütüphanesi)

- ***System.Net*** : HTTP ve ağ protokolleri için kullanılır.
- ***System.Xml*** : XML verileri ile çalışmak için
- ***System.IO*** : dosyalara bilgi girişi, dosyadan bilgi okuma, I/O işlemleri için kullanılır.
- ***System.Windows.Forms***: Windows tabanlı uygulamalarda kullanılan zengin grafik arabirimi kontrollerini içerir.

C# Assembly ve Çoklu Dosya

- *C# ' ta çoklu dosya kullanımı ile birden fazla kaynak kod referans edilerek veya DLL dosyası halinde diğer kaynak kodlar tarafından kullanılabilir.*
- *Örneğin: Elimizde çeşitli sayı tiplerinin dönüşümünü yapan 'Program.cs' ve 'topla' isimli fonksiyon içeren 'Program1.cs' adlı kod dosyalarımız olsun.*

C# Assembly ve Çoklu Dosya

```
using toplama;
```

```
namespace kutuphane
```

```
{
```

```
    class program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            float a,b;
```

```
            string c;
```

```
            Console.WriteLine("a sayısı gir");
```

```
            a = Convert.ToSingle(Console.ReadLine()); //float a çevrim
```

```
            Console.WriteLine("b sayısı gir");
```

```
            b =float.Parse(Console.ReadLine()); // float a çevrim
```

```
            Console.WriteLine("toplam={0}", toplama.program.topla(a, b));
```

```
            Console.ReadKey();    }
```

```
    } //program.cs
```

C# Assembly ve Çoklu Dosya

```
namespace toplama
```

```
{
```

```
class program
```

```
{
```

```
public static float topla(float a, float b)
```

```
{
```

```
return (a + b);
```

```
}
```

```
}
```

```
}
```

```
//program1.cs
```

C# Assembly ve Çoklu Dosya

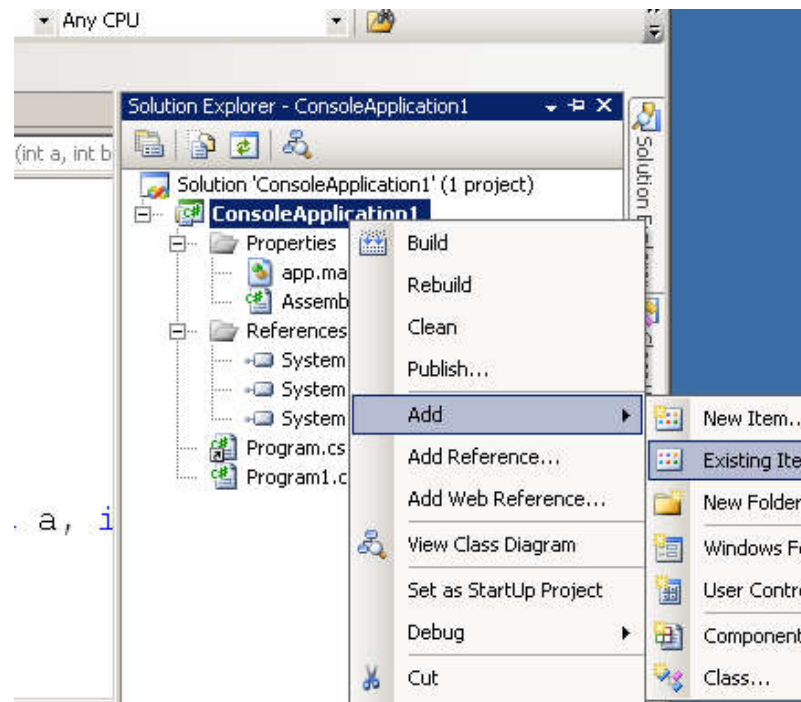
- *Program.cs kaynak kod dosyasının içindeki*

Console.WriteLine("toplam={0}",toplama.program.topla(a, b));

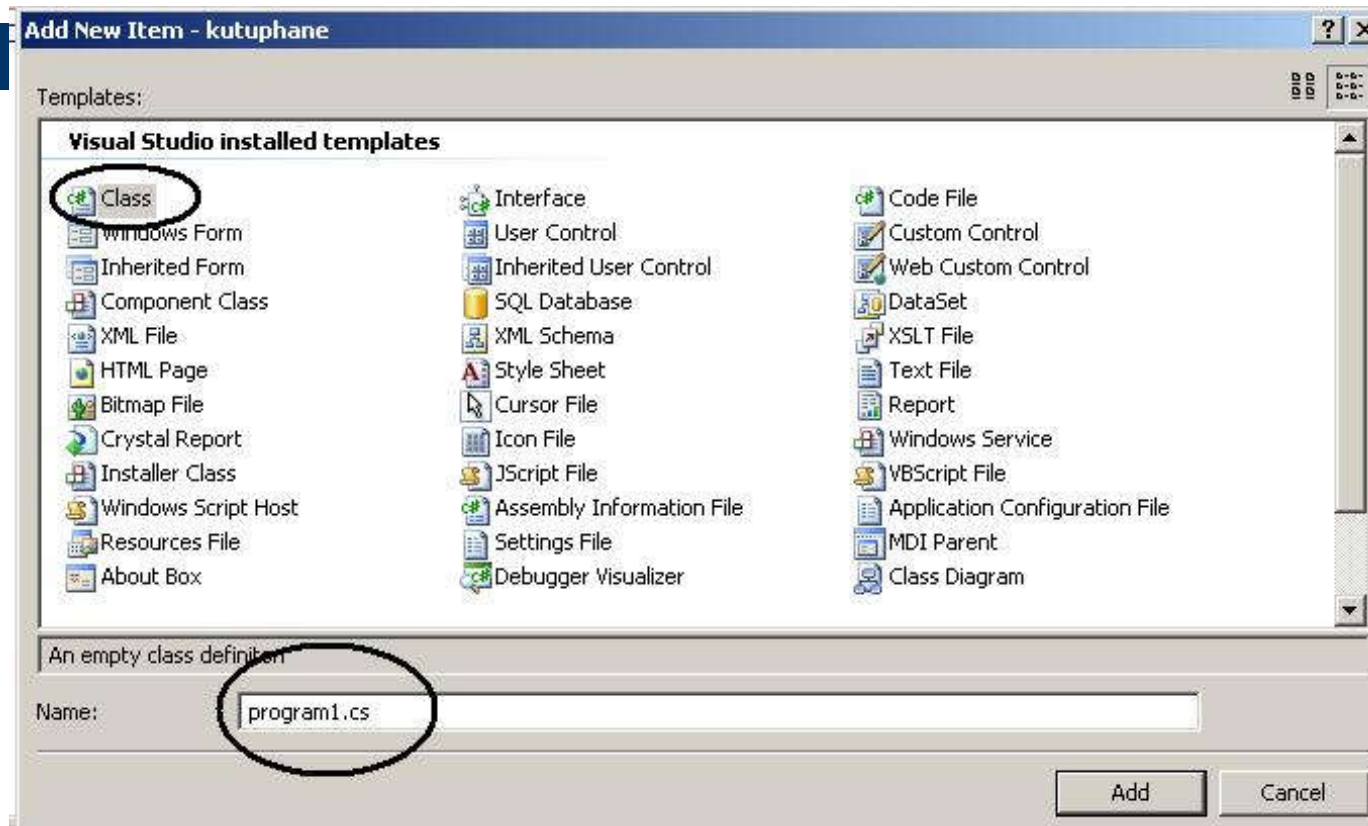
- *Yukarıdaki toplama.program.topla(a,a) kısmında bu kaynak kod içinden başka bir alanadı(namespace) içindeki 'Program' sınıfı içindeki 'topla' isimli metot çağrılmaktadır.*
- *Bu durumda hs2 alanadını içeren Program1.cs dosyasının bir şekilde Program.cs'ye dahil edilmesi gerekir. Bunun için VStudio'yu kullanalım.*

C# Assembly ve Çoklu Dosya

- İlk durumda Visual Studio.NET'i kullanarak. Sağ taraftaki 'Solution Explorer' üzerinde sağ buton yaparak 'Add New Item' ile yeni class dosyası eklenir dosyası projeye dahil edilir.

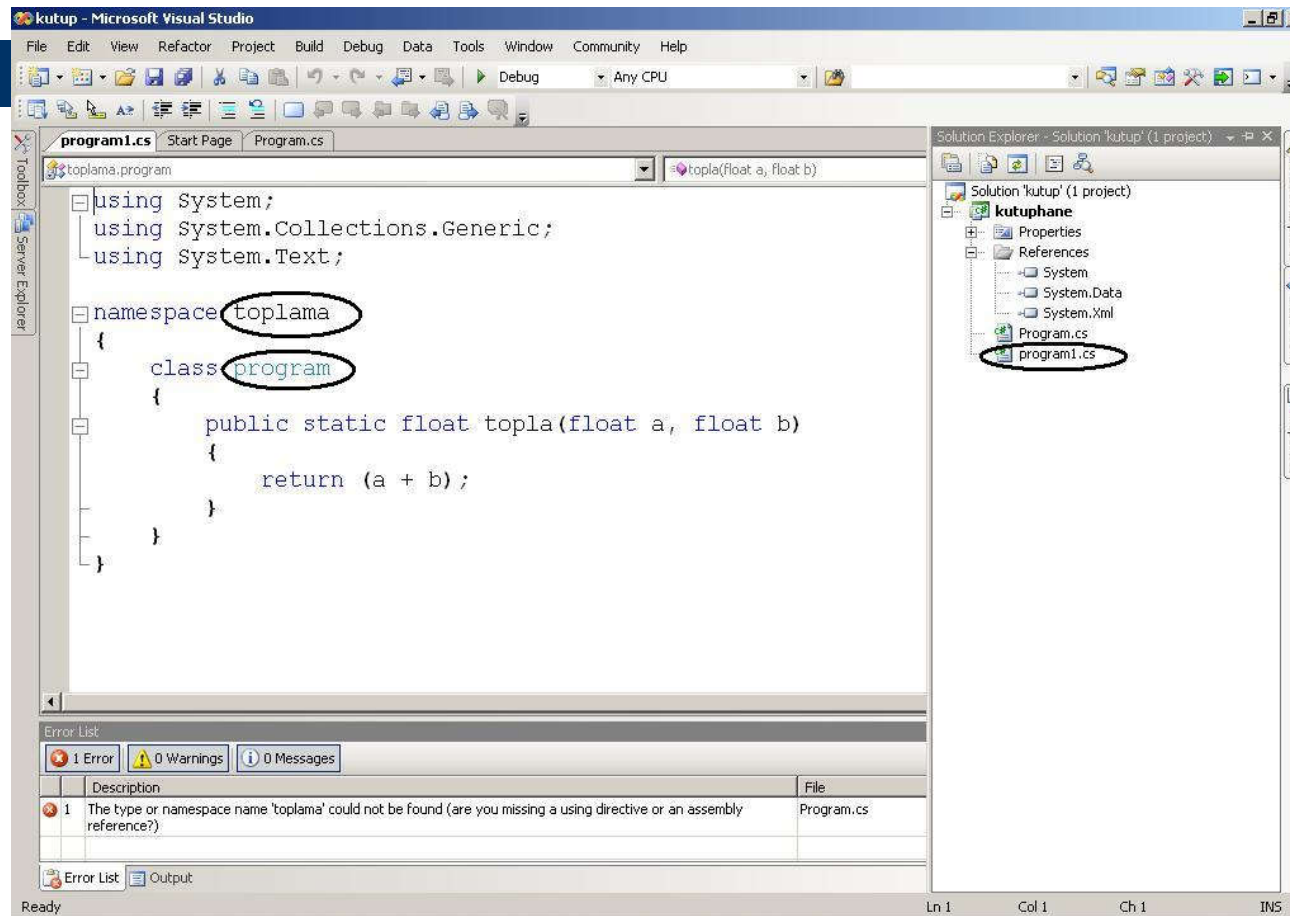


C# Assembly ve Çoklu Dosya



Projeye yeni bir sınıf dosyası ekliyoruz. Metodun saklı olacağı sınıf dosyasını

C# Assembly ve Çoklu Dosya



C# Assembly ve Çoklu Dosya

- *Projeye dahil edilen Program1.cs içindeki toplama alanadını using deyimini kullanarak artık kendi kaynak kodumuz içinde kullanabiliriz.*
- *Eğer Program1'i DLL olarak dahil etmek istersek. Yapmamız gereken komut satırından C# derleyicisini (CSC.EXE) kullanmak olacaktır. (Csc /? Yardım bilgisini verir)*



```
C:\ Visual Studio 2005 Command Prompt
Setting environment for using Microsoft Visual Studio 2005 x86 tools.

c:\Program Files\Microsoft Visual Studio 8\VC>csc
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.42
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.

fatal error CS2008: No inputs specified

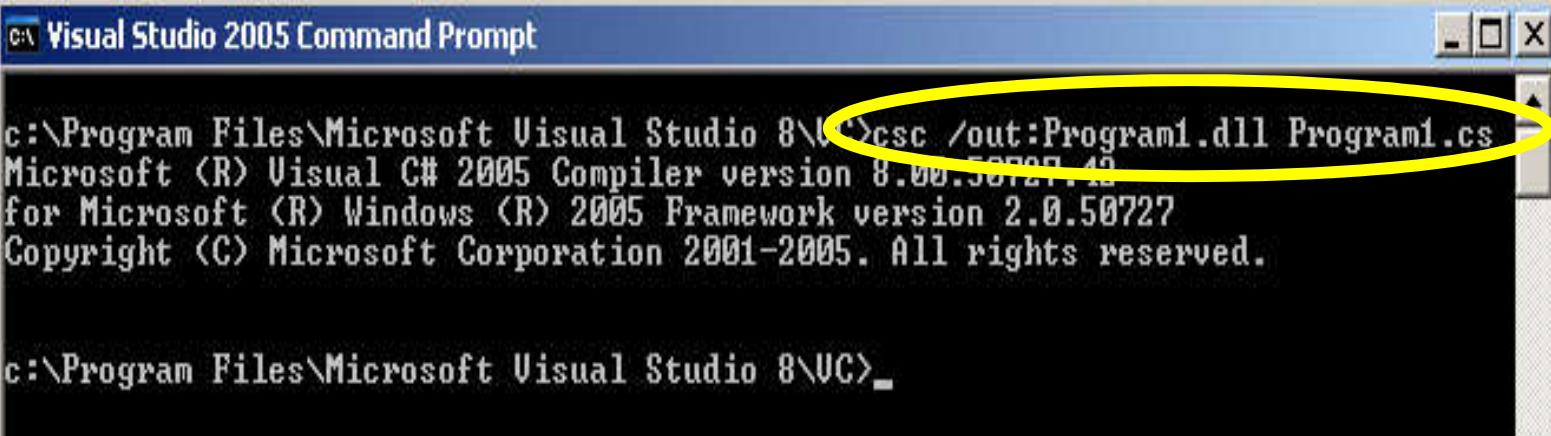
c:\Program Files\Microsoft Visual Studio 8\VC>
```

C# Assembly ve Çoklu Dosya

- İlk önce dahil edilecek olan *Program1.cs* dosyasını DLL olarak derleyeceğiz.
- *Program1.Dll* dosyasını elde ettikten sonra bu dosyayı *Program.cs* dosyasını derlerken referans olarak dahil edeceğiz.

> **csc /out:Program1.dll Program1.cs**

Artık DLL dosyamız elimizde.



```
Visual Studio 2005 Command Prompt
c:\Program Files\Microsoft Visual Studio 8\VC>csc /out:Program1.dll Program1.cs
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.42
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.

c:\Program Files\Microsoft Visual Studio 8\VC>_
```

C# Assembly ve Çoklu Dosya

- *Dll dosyamızı şimdi Program.cs dosyasına referans olarak göstereceğiz. Sonuçta referans ettiğimiz yerdeki alanadları veya metotlara ulaşabileceğiz. Buradaki ' /r ' parametresi referans edilen dosyayı göstermektedir.*

.. > csc /r:Program1.dll Program.cs

Artık çalıştırılabilir Program.exe elde edilmiş oldu.

```
c:\Program Files\Microsoft Visual Studio 8\VC>csc /r:Program1.dll Program1.cs
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.42
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.

c:\Program Files\Microsoft Visual Studio 8\VC>dir Program.exe
C sürücüsü birimi: Yavuz
Birim Seri Numarası: 94BE-CDDC

c:\Program Files\Microsoft Visual Studio 8\VC dizini
03/08/2006 06:06          3,584 Program.exe
                  1 Dosya          3,584 bayt
                  0 Dizin    18,564,861,952 bayt boş
c:\Program Files\Microsoft Visual Studio 8\VC>
```

Veri Türleri Şeması



CTS veri tipleri şeması

Veri Türleri

- **Stack Bölgesi:** Tanımlı değişkenlerin tutulduğu bellek alanıdır. Derleyici tarafından değişkenlere yapılacak yer tahsisatı önceden bilinmelidir.
- **Heap Bölgesi:** Stack'ten farklı olarak heap bölgesinde tahsisatı yapılacak nesnenin derleyici tarafından bilinmesi zorunlu değildir. Bu programlarımıza esneklik getirir.
- Heap'te bir nesne için yer tahsisatı ***new*** kelimesi ile yapılır.

Veri Türleri

- *new* ile tahsis edilen alanlar dinamiktir. Çalışma zamanında tahsisat yapılır, derleme zamanında bir yer ayrılmaz.
- **Stack**'e göre daha yavaştır.
- **Değer** veri türleri **Stack**, **Referans** veri türleri **Heap**' te tutulurlar.

Değer Tipleri (Value Types)

- Değer tiplerinde bir nesnenin değeri direkt olarak saklıdır.

```
int a=3,b;
```

```
b=a;
```

Bu noktada a üzerindeki değişikliklerden b etkilenmeyecektir.

- Değer tiplerinin tamamı **Object** denilen bir nesneden türemiştir.

Değer Tipleri (Value Types)

- Değer tiplerine ilk değer verme;

```
int a;    // deklere edildi
```

```
a=new int(); //yapıcı çalışır.(referans tip)
```

```
a=0;
```

- Yukarıdaki iki satırda aynı işlemi yapar.

```
float b; //derleyici hatası, atama yapılması gerekir.
```

```
Error2 Use of unassigned local variable b'
```

```
float b=new float(); //hata vermez yapıcı çalıştı
```

```
b=3.21f //yeni atama yapılıyor
```

Referans Tipleri (Reference Types)

- C# ' ta önceden tanımlı iki referans tipi vardır. **Object** ve **String**. Object türü C#'ta bütün türlerin türediği sınıftır.
- Object türü özelleştirilerek farklı amaçlara yönelik kullanılabilirler. Object'e eşleştirme (Boxing) işlemi ve tersi, Object'i dönüştürme (Unboxing)

Veri Tipleri

	Long Form	in Java	Range
sbyte	System.SByte	byte	-128 .. 127
byte	System.Byte	---	0 .. 255
short	System.Int16	short	-32768 .. 32767
ushort	System.UInt16	---	0 .. 65535
int	System.Int32	int	-2147483648 .. 2147483647
uint	System.UInt32	---	0 .. 4294967295
long	System.Int64	long	$-2^{63} .. 2^{63}-1$
ulong	System.UInt64	---	$0 .. 2^{64}-1$
float	System.Single	float	$\pm 1.5E-45 .. \pm 3.4E38$ (32 Bit)
double	System.Double	double	$\pm 5E-324 .. \pm 1.7E308$ (64 Bit)
decimal	System.Decimal	---	$\pm 1E-28 .. \pm 7.9E28$ (128 Bit)
bool	System.Boolean	boolean	true, false
char	System.Char	char	<u>Unicode</u> character

Tür Dönüşümleri

- **Bilinçsiz Tür Dönüşümü:** Derleyici bizim için yapıyor.

```
int s=10;
```

```
float a;
```

```
a=s;
```

```
//a=10.0
```

```
Console.WriteLine(a);
```

Tür Dönüşümleri

- **Küçük türün büyük türe dönüştürülmesi:**

Küçük tür büyük türe dönüştürülürken fazla bitler sıfır ile doldurulur.

Örn: `byte a=12;`

`int b;`

`b=a;`

Tür Dönüşümleri

- **Büyük türün küçük türe dönüştürülmesi:**
İstenmeyen durum. Ancak “()” operatörü ile yapılır.

Örn:

```
int b=256;  
byte i=(byte) b; //sonuç 0 'dır.
```


Referans-Değer Dönüşümleri

- **Referans ve Değer Türleri Arasındaki Dönüşüm**
C# dilinde değer tipindeki verileri referans tipine çevirmek önemli bir konudur. Değer veri tipleri Stack, Ref. veri tipleri Heap'te tutulur.
- C# herşey nesne(object) referans türünden türetilmiştir. Temelde bir sınıf vardır. Örneğin object sınıfının ToString() metodu bütün temel veri ve referans türlerinde kullanılır.

`string str = 345.59f.ToString()`

Referans-Değer Dönüşümleri

Örn:

```
int a=5;
```

```
int b=7
```

```
string a1=a.ToString();
```

```
string b1=b.ToString();
```

```
Console.WriteLine(a+b);
```

```
Console.WriteLine(a1+b1);
```

Sonuç: 12

57

```

int a=0; int d = (int) 6.0; //float -> integer dönüşüm
object k="merhaba"+15; //object türü, hem karakter hem sayısal
float b=10.5f; //float tanımı
double c=20.1; //double tanımı
Double dd = new double(); //referans olarak double tanımı
const double pi = 3.14; //sabit tanımı
string[] isimler ={ "Ozlem","Nesrin", "Ozge", "Fulya" }; //string dizi tanımı
object[] isim ={ "Ozlem","Nesrin", "Ozge", "Fulya" }; //object dizi tanımı
string s = "true"; //string tanımı
string dd="12.45f";
b= float.Parse(dd); //string tip float'a çevriliyor
b=Convert.ToSingle(dd); //String float'a çevriliyor
a =Convert.ToInt32(b + c); //float -> integer
bool cevap = (Convert.ToBoolean(s)); //boolean tanımı
Console.Write((float)a/d+"\n"); // () operatörü ile float dönüşümü
Console.WriteLine("cevap=" + cevap); // cevap = true yazar
Console.WriteLine(k.GetType()); //bulunduğu sınıf,alanadını verir.
a = Convert.ToSingle(Console.ReadLine()); //girilen değer float'a çevriliyor
Console.WriteLine("a={0} b= {1} c={2} d={3} ", a, b, c,d);
if (isimler[0].Equals("Ozlem")==true) //eğer dizinin ilk elemanı Ozlem ise yazar
    Console.WriteLine("birinci isim Ozlem");
foreach (string ss in isimler) // string dizi içindeki her bir eleman yazdırılıyor
{ Console.WriteLine(ss); }

```

System.Convert Sınıfı (Temel Veri Tiplerinin Dönüşümü)

Convert.ToBoolean(str)

Convert.ToByte(str)

Convert.ToInt32(str)

Convert.ToChar(str)

...

int a=50;

byte b=Convert.ToInt32(a);*//Yanlış tür dönüşümü*

string c="12.34";

a=float.Parse(c);

Referans-Değer Dönüşüm(Boxing)

Günümüzdeki popüler dillerde referans ve değer tipleri arasında dönüşüm yapılmamaktadır. Böyle bir çevrime ihtiyaç duyulduğunda “Boxing” kutulama yapılır. Bir değer tipini referans tipe atadığımızda stack'teki bilgi **bit** olarak heap'e kopyalanır ve stack'teki object türünden olan değişken heap'i gösterecek şekilde ayarlanır.

Örn:*Bilinçsiz boxing işlemi.*

```
int i=50; //değer tipi
```

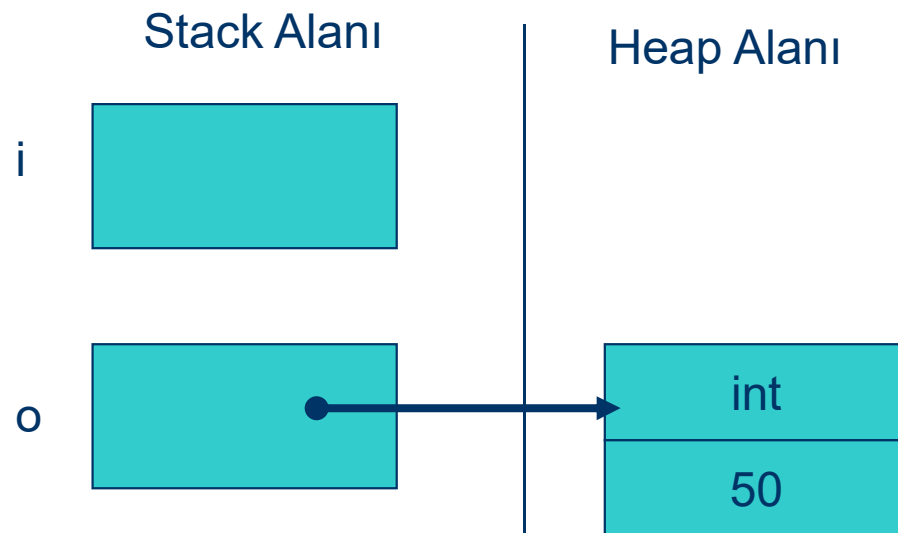
```
object o=i; //boxing
```

Referans-Değer Dönüşüm(Boxing)

Örn:Bilinçli boxing işlemi.

```
int i=50; //değer tipi
```

```
object o=(object) i; //boxing
```



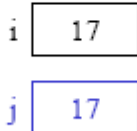
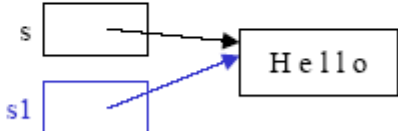
Referans-Değer Dönüşüm(Unboxing)

Boxing işleminin tam tersidir. Aşağıdaki koşullara uyularak yapılmalıdır.Bilinçsiz yapılamaz.

1. Unboxing işlemine tabi tutulacak nesnenin daha önceden boxing işlemine tabi tutulmuş olması.
2. Boxing işlemine tabi tutulmuş olan bu nesnenin unboxing işlemi sırasında doğru türe dönüştürülmesidir

Örn: `int i=50;`
`object o=i;`
`int j=(int)o;`

Değer tipleri Referans Tiplerine Karşı

	Value Types	Reference Types
variable contains	value	reference
stored on	stack	heap
initialisation	0, false, '\0'	null
assignment	copies the value	copies the reference
example	<pre>int i = 17; int j = i;</pre>  <p>The diagram shows two memory boxes. The first box is labeled 'i' and contains the value '17'. The second box is labeled 'j' and also contains the value '17'.</p>	<pre>string s = "Hello"; string s1 = s;</pre>  <p>The diagram shows three memory boxes. Two boxes on the left are labeled 's' and 's1'. Both have arrows pointing to a third box on the right labeled 'Hello'.</p>