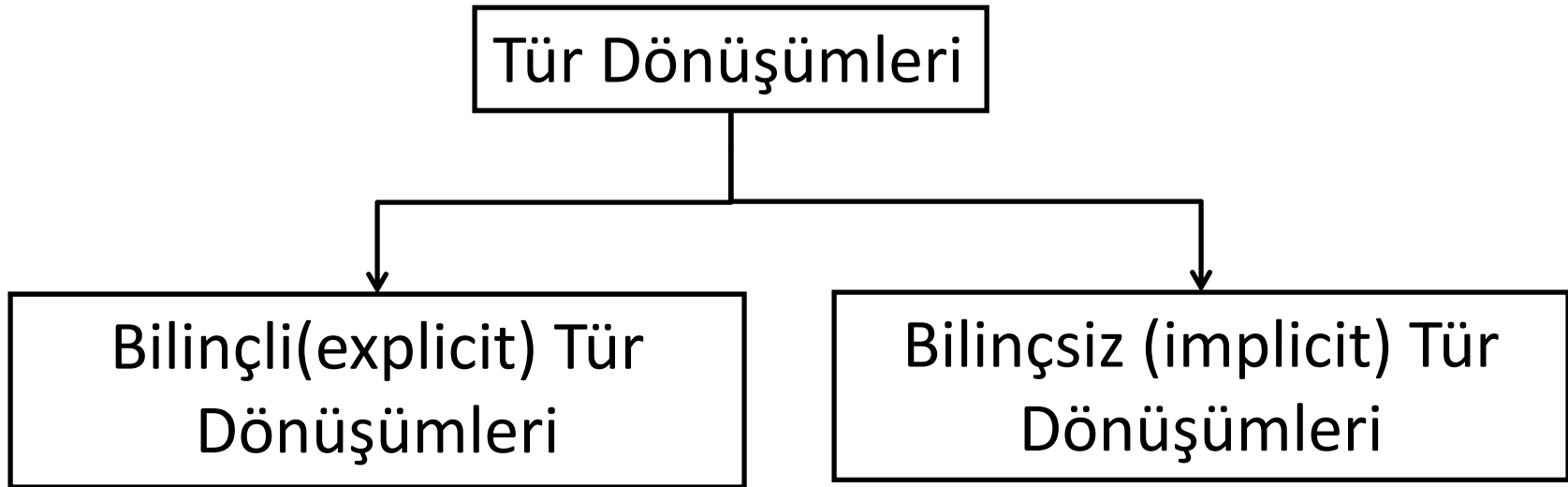


Tür Dönüşümü



Farklı türden değişkenlerin aynı ifade içinde işlem görmeleri için tür dönüşümü kullanılır. Tür dönüşümlerini aşağıdaki şekilde gruplara ayırmak mümkündür:



Bilinçsiz (Implicit) Tür Dönüşümleri



Derleyici tarafından bir değişkeni tanımladığımız türün dışında **geçici** olarak başka bir türe çevirmeye bilinçsiz tür dönüşümü denir. Örn:

```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        int x=5;
        float a;
        a=x;
        Console.WriteLine(a);
    }
}
```



Bilinçsiz yapılan tür dönüşümlerinde bir nesnenin türü asla kalıcı olarak değiştirilmez.



Bilinçsiz yapılan tür dönüşümleri 2 şekilde gerçekleştirilebilir.

- a. Küçük Türün Büyük Türe Dönüştürülmesi
- b. Büyük Türün Küçük Türe Dönüştürülmesi

Bilinçsiz (Implicit) Tür Dönüşümleri

- Küçük Türün Büyük Türe Dönüşümü

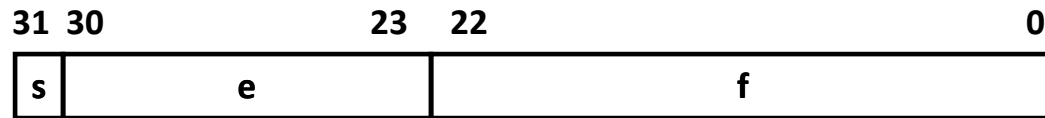
- Küçük türler büyük türlere dönüştürülürken fazla olan bitler yani büyük türden dolayı eklenen bitler sıfırla beslenirler. Küçük türün yüksek anlamlı bitlerinin sıfırla beslenmesi deęişkendeki deęeri deęiřtirmedięi için tür dönüşümünde herhangi bir veri kaybı olmaz.

```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        byte a=20;
        int b;
        b=a;
        Console.WriteLine(b);

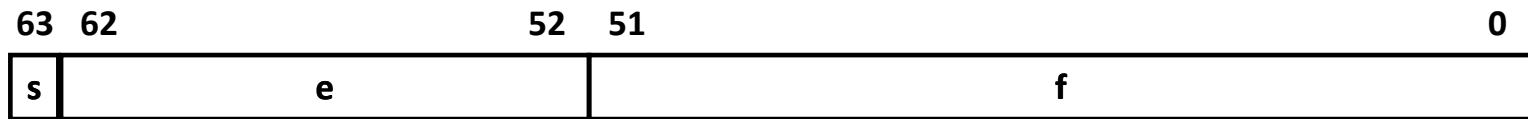
        float f=20f;
        double d;
        d=f;
        Console.WriteLine(d);

        char c='a';
        decimal m;
        m=c;
        Console.WriteLine(m);
    }
}
```

float sayı formatı



double sayı formatı



s : işaret biti

e : üsleri temsil eden bitler

f : noktalı (anamlı) kısım



Byte türünden nesnelere toplandığında sonuç olarak int türünden nesnelere üretilirler. Örnek:

```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        byte a=15;
        byte b=25;
        byte c;
        c=a+b;
        Console.WriteLine(c);
    }
}
```


Tür	Bilinçsiz Dönüşüme İzin verilen Türler
sbyte	byte, ushort, uint, ulong, veya char
byte	sbyte veya char
short	sbyte, byte, ushort, uint, ulong, veya char
ushort	sbyte, byte, short, veya char
int	sbyte, byte, short, ushort, uint, ulong, veya char
uint	sbyte, byte, short, ushort, int, veya char
long	sbyte, byte, short, ushort, int, uint, ulong, veya char
ulong	sbyte, byte, short, ushort, int, uint, long, veya char
char	sbyte, byte, veya short
float	sbyte, byte, short, ushort, int, uint, long, ulong, char, veya decimal
double	sbyte, byte, short, ushort, int, uint, long, ulong, char, float, veya decimal
decimal	sbyte, byte, short, ushort, int, uint, long, ulong, char, float, veya double



Aşağıdaki kod parçasını bilgisayarınızda deneyiniz:

```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        short b1=100;
        char c1=b1;
        bool b2=true;
        int i1=b2;
        double d1=10.2;
        int i2=d1;
        decimal m1=20.6M;
        double d2=m1;
        byte bt1=65;
        char c2=bt1;
        float f1=34.78F;
        decimal d3=f1;
    }
}
```






Bazı türler arasında tür dönüşümü yapmam mümkün değildir. Bunlar :

- a. Bool, decimal ve double türünden herhangi bir türe
- b. Herhangi bir türden char türüne
- c. Float ve decimal türünden herhangi bir türe (float türünden double türüne dönüşüm hariç)


Bilinçsiz (Implicit) Tür Dönüşümleri

- **Büyük Türün Küçük Türe Dönüşümü**
 - Büyük türlerin küçük türlere otomatik dönüştürülmesi C#'da yasaklanmıştır. Eğer bu tür bir dönüştürme(bilinçsiz olarak) mümkün olsaydı birtakım veri kayıpları yaşanacaktır.

Bilinçli (Explicit) Tür Dönüşümleri

-  Bilinçli (explicit) tür dönüşümü genellikle derleyicinin izin vermediği dönüşümlerde kullanılır.
-  Bu tür dönüşümlerde de yine küçük türler büyük türe ya da tersi dönüşümler yapılabilir.
-  Küçük türlerin büyük türlere çevrilmesi aynı bilinçsiz dönüşümde olduğu gibidir.

Tür Dönüştürme Operatörü

 Bilinçli tür dönüşümü yapılırken “tür dönüştürme operatörleri” kullanılır. Tür dönüştürme operatörü parantez içinde değişken ya da sabitten önce yazılır.

– (dönüştürülecek tür)değişken_yada_sabit;

```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        byte a=100;
        int b=(int)a;
        Console.WriteLine(b);
    }
}
```



Bu işlem tür dönüşüm operatörü kullanmadan da yapılabilirdi. Buradaki amaç daha çok okunabilirliği arttırmaktır.

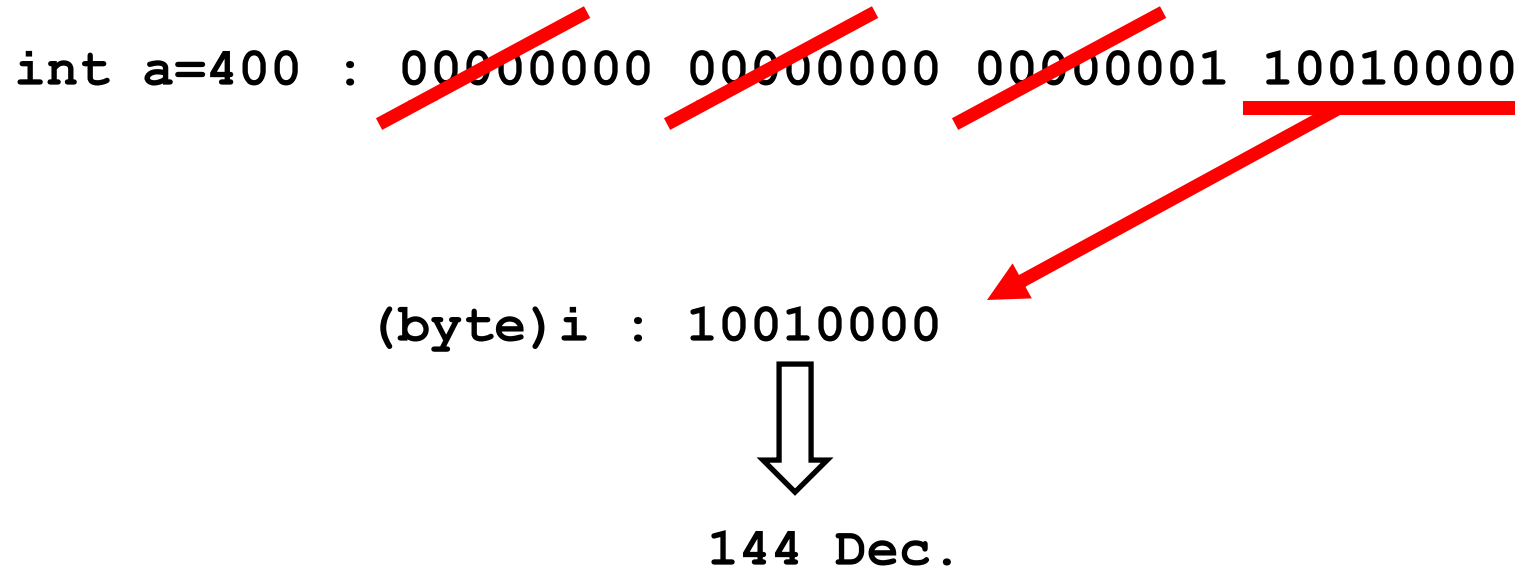


Bilinçsiz yapılan tür dönüşümlerinde büyük türler, küçük türlere dönüştürülemez. Eğer tür dönüştürme operatörü kullanılırsa bu işlem mümkün olur. **Örn:** Aşağıdaki programı deneyin:

```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        int a=400;
        byte b=(byte)a;
        Console.WriteLine(b);
    }
}
```




Programı çalıştırdığımızda ekrana 144 yazdırdı.
Neden?





```
using System;
class TurDonusumu9
{
    static void Main()
    {
        int i = 256;
        byte b = (byte)i;
        Console.WriteLine(b);

        double d = 123456.7890123456;
        float f = (float)d;
        int i3 = (int)d;
        Console.WriteLine(f);
        Console.WriteLine(i3);

        decimal m = 123456789.123456789M;
        double d2 = (double)m;
        Console.WriteLine(d2);

        int i2 = 65600;
        short s = (short)i2;
        Console.WriteLine(s);
    }
}
```

Checked ve Unchecked

-  Büyük veri türleri küçük türlere çevrilirken veri kayıpları meydana gelebiliyordu. Bazı durumlarda bu veri kayıpları olmuyorsa çevrim işlemi yapılması aksi halde hata oluşması istenebilir.
-  Bu durumda “**checked**” anahtar sözcüğü kullanılabilir. Bu sözcük veri kaybı oluşması durumunda derleyicinin hata vermesini sağlar.



Aşağıdaki kod parçasını bilgisayarınızda deneyiniz:

```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        int i=256;
        byte b;
        checked
        { b=(byte)i;
        }
        Console.WriteLine(b);
    }
}
```



Checked anahtar sözcüğü ile çalışma zamanında oluşabilecek veri kayıplarının olabileceği durumlarda hata vermesini sağlayabiliriz.

```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        int i=256;
        checked
        { byte b=(byte)i;
        }
        Console.WriteLine(b);
    }
}
```



Not: checked bir blok oluşturduğu için içinde yapılan değişken tanımlamaları dış bloklarda kullanılamaz.



Normal şartlarda yapılan işlemler “unchecked”dir. Böyle bir ifadenin konmasının nedeni uzun “checked” blokları içerisinde bazen “unchecked” blokların oluşturulması istenebilir. Bu durumlarda çok fazla blok oluşturmamak için “unchecked” ifadesi kullanılabilir.

```
using System;
class TurDonusumu11
{
    static void Main()
    {
        int i1 = 255;
        int i2 = 500;
        byte b, c;

        checked
        {
            b = (byte)i1;
            Console.WriteLine(b);

            unchecked
            {
                c = (byte)i2;
            }
            Console.WriteLine(c);
        }
    }
}
```

Referans ve Değer Türleri Arasındaki Dönüşüm

Object Türü ve ToString() Metodu

- ✓ Temel veri türleri de dahil olmak üzere bütün veri tipleri object denilen bir referans türünden türemiştir. Türeme, kalıtım yolu ile olduğu için var olan özellikler her zaman korunur.
- ✓ Sınıf kütüphanesinin ve temel veri türlerinin atası olan object nesnesinin özellikleri ve iş yapan metotları bütün türlerde mevcuttur.

ToString() Metodu

✓ .ToString() metodu bütün temel türlerde ya da referans türlerde kullanılabilir. Amacı ise string'e dönüşüm işlemi yapmaktır.

- `56.ToString()` ;
- `12.34F.ToString()` ;

✓ Yukarıdaki örnekler verilen sabit ifadelerin string'e dönüştürülmesi için kullanılmıştır.



Aşağıdaki kod parçasını bilgisayarınızda deneyiniz:

```
using System;

class TurDonusumu12
{
    static void Main()
    {
        int x = 10;
        int y = 20;

        string a = x.ToString();
        string b = y.ToString();

        Console.WriteLine(x + y);
        Console.WriteLine(a + b);
    }
}
```

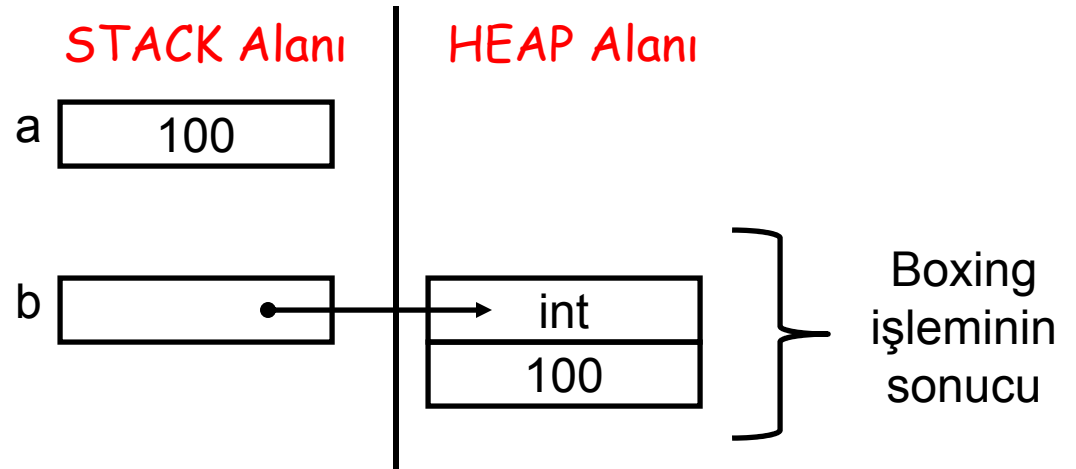
Boxing İşlemi

- ✓ Bir çok modern dilde değer tipleri ile referans tipler arasında dönüşüm söz konusu değil iken C# dilinde bu mümkündür.
- ✓ “Boxing” (kutulama) olarak ifade edilen yöntem değer tipindeki verileri “object” nesnesine çevirir.



Object nesnelere de referans tip oldukları için “**heap**” bellek alanında tutulurlar. Bu yüzden object nesnesine çevrilen değer tipleri “**stack**” alanından “**heap**” alanına kopyalanır. “**stack**” alanında object türündeki değişken, “**heap**” alanını gösterecek şekilde ayarlanır. Bu işleme “**boxing**” adı verilir.

```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        int a=100;
        object b=a;
    }
}
```







Boxing işlemi bilinçsiz olarak yapılabileceği gibi tür dönüştürme operatörleri kullanılarak bilinçli olarak da yapılabilir. **Örn:**


```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        int a=100;
        object b=a;
    }
}
```

```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        int a=100;
        object b=(object) a;
    }
}
```

Unboxing

-  Heap alanındaki nesnelerin değerlerinin bit olarak stack bölgesine kopyalanması işlemine “unboxing” adı verilir. Bu işlem sonucunda referans türler değer türüne dönüştürülmüş olur.

-  Unboxing işleminin çalışma zamanında hata vermemesi için gerekli iki önemli koşul vardır:
 - Unboxing uygulanacak nesnenin daha önceden boxing işlemiyle dönüştürülmüş olması
 - Boxing’e tabi tutulmuş bu nesnenin unboxing ile doğru türe dönüştürülmesidir.

-  Unboxing işlemi bilinçsiz bir biçimde yapılmaz, mutlaka tür dönüşüm operatörü kullanılmalıdır.



Aşağıdaki kod parçasını bilgisayarınızda deneyiniz:

```
using System;

class TurDonusumu13
{
    static void Main()
    {
        int i = 10;
        object o = i;

        int j = (int)o;

        Console.WriteLine(i);
        Console.WriteLine(j);
    }
}
```




Aşağıdaki kod parçasını bilgisayarınızda deneyiniz:

```
using System;

class TurDonusumu14
{
    static void Main()
    {
        int i = 10;
        object o = i;

        long l = (long)o;

        Console.WriteLine(i);
        Console.WriteLine(l);
    }
}
```

System.Convert Sınıfı ile Tür Dönüşümü



.NET sınıf kütüphanesinde yer alan “Convert” sınıfı string değerleri ve temel veri türlerini birbirine çevirmek için kullanılır. Her bir veri türü için ayrı bir çevrim fonksiyonuna sahiptir.

<code>Convert.ToBoolean()</code>	<code>Convert.ToInt64()</code>
<code>Convert.ToByte()</code>	<code>Convert.ToUInt64()</code>
<code>Convert.ToSByte()</code>	<code>Convert.ToSingle()</code>
<code>Convert.ToInt16()</code>	<code>Convert.ToDouble()</code>
<code>Convert.ToUInt16()</code>	<code>Convert.ToDecimal()</code>
<code>Convert.ToInt32()</code>	<code>Convert.ToChar()</code>
<code>Convert.ToUInt32()</code>	

```
using System;

class TurDonusumu15
{
    static void Main()
    {
        string s1, s2;
        int i1, i2, t;

        Console.Write("1.Sayıyı Girin:");
        s1 = Console.ReadLine();

        Console.Write("2.Sayıyı Girin:");
        s2 = Console.ReadLine();

        i1 = Convert.ToInt32(s1);
        i2 = Convert.ToInt32(s2);

        t = i1 + i2;

        Console.WriteLine("Toplam = " + t.ToString());
    }
}
```



Dönüşüm işleminin sonucunda anlamlı bir sonuç elde edilemeyeceği durumlarda hata meydana gelir.

```
using System;

class TurDonusumu16
{
    static void Main()
    {
        char a = 'a';
        bool b = Convert.ToBoolean(a);
    }
}
```