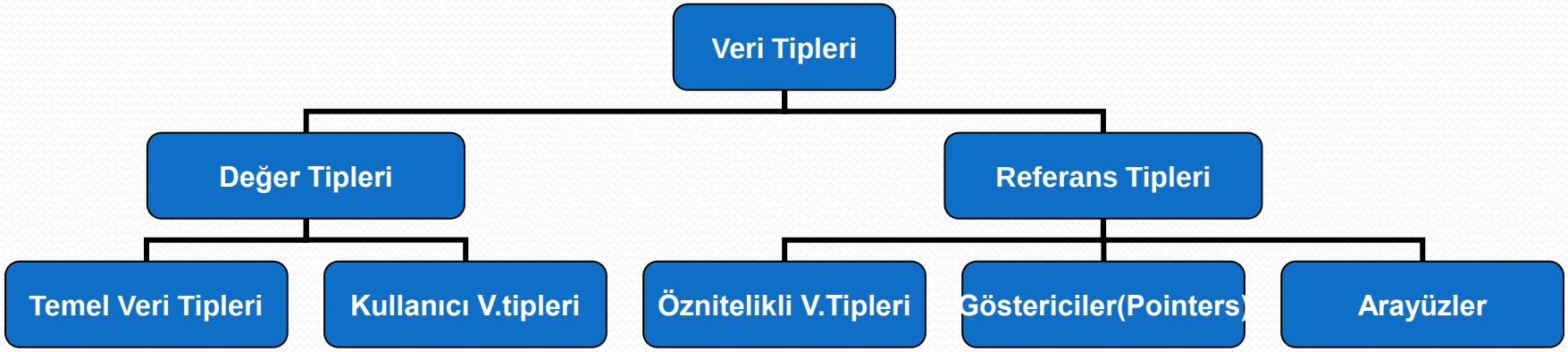
Visual Studio.Net -C#
2. HAFTA
Temel Veri Türleri

C# Dilinde Temel Veri Türleri

- C#'da veri tipleri temel olarak 2'ye ayrılırlar. Bunlar önceden tanımlanmış veri türleri ve kullanıcı tarafından tanımlanmış veri türleridir. Önceden tanımlanmış olan veri türleri de kendi arasında **değer tipi (value type)** ve **referans tipi (reference type)** olarak 2'ye ayrılır.

Veri Türleri Şeması



C# Dilinde Temel Veri Türleri

- Verinin bellekte tutulması 6 bölgeden biri ile olmaktadır Bunlar:

- **Stack Bölgesi:** Program içerisinde bir tamsayı türünden nesnenin çalışma zamanında yüklendiği yer RAM' in stack bölgesidir.

- Tanımlı değişkenlerin tutulduğu bellek alanıdır. Derleyici tarafından değişkenlere yapılacak yer tahsisatı önceden bilinmelidir.

C# Dilinde Temel Veri Türleri

- **Heap Bölgesi:** Bütün C# nesneleri bu bölgede oluşturulur. Stack'ten farklı olarak bu bölgede tahsisatı yapılacak nesnenin derleyici tarafından bilinmesi zorunlu değildir. Bu bölgede bir nesneye alan ayırmak için **new** anahtar sözcüğü kullanılır.

- ***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.)

C# Dilinde Temel Veri Türleri

- **Register Bölgesi:** Registerlar mikroişlemci üzerinde bulunan özel yapılardır Bu yapılarından dolayı diğer bölgelere göre veri transferi daha hızlı bir şekilde yapılabilmektedir.

- **Static Bölge:** Bellekteki herhangi bir bölgeyi temsil eder. Static alanlarda tutulan veriler programın bütün çalışma süresince saklanır. Bir nesneye bu özelliği kazandırmak için static anahtar sözcüğü kullanılır.

- **Sabit Bölge:** Program içerisinde, değerlerin değişmeden sürekli olarak aynı kaldığı bölümdür.

- **RAM Olmayan Bölge:** Bellek bölgesini temsil etmeyen disk alanlarını temsil eder.

Değişken Tanımlama

- Değişken tanımlama aşağıdaki gibidir:
- **<veritürü> <ismi>**

- C#'da bir değişkene herhangi bir değer atamadan onu kullanmak yasaktır. Eğer bir değişkeni kullanmak istiyorsak değişkenlere bir değer verilmesi zorunludur. Bu kural değer ve referans tipleri için de geçerlidir.

- Tanımlamalar ise programın istenilen bir yerinde yapılabilir. Bu konuda herhangi bir kural yoktur.

Değişken Tanımlama

- Değişken isimlendirme ile ilgili temel kuralları aşağıdaki şekilde özetlemek mümkündür:
- C#'da değişken isimlendirmede büyük ve küçük harf duyarlılığı vardır.
- Değişken isimleri nümerik bir karakter ile başlayamaz.
- Değişken isimlerinde boşluk karakteri olamaz.

Değişkenlerin Faaliyet Alanları (Scopes)

- Tanımlanan bir değişkene ancak tanımlandığı blok içerisinden ulaşılabilir. Bu blok aralığına değişkenin faaliyet alanı denir.

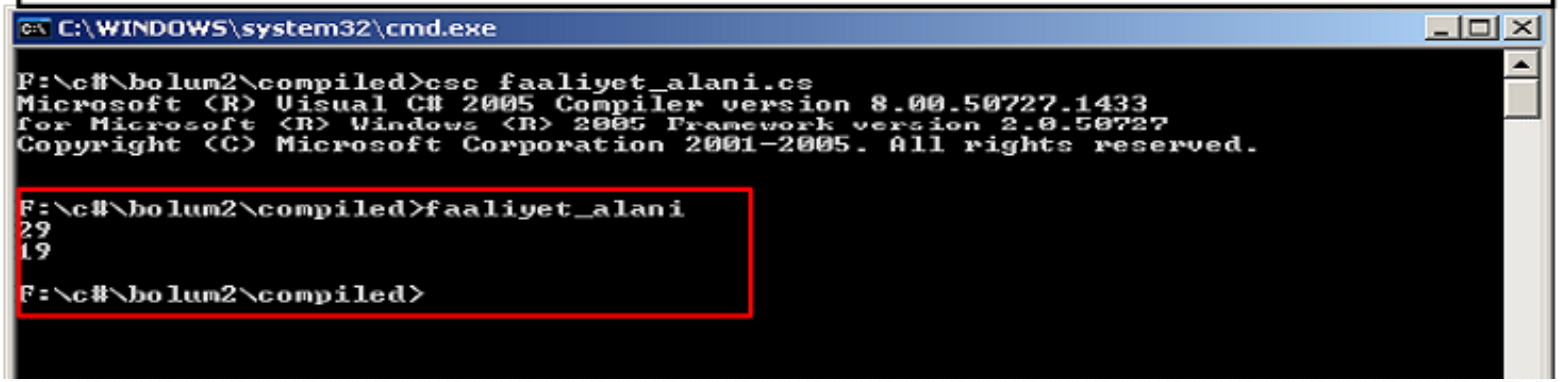
- Bir sınıfın üye elemanı olarak tanımlanmış değişken her zaman sınıfın faaliyet alanı içerisindedir.

- Yerel bir değişken, tanımlandığı blok arasında kaldığı sürece faaliyet alanındadır.

- Döngü bloklarında tanımlanan değişkenler döngünün dışına çıkılmadığı sürece faaliyet alanı içersindedirler.

Değişkenlerin Faaliyet Alanları (Scopes)

```
using System;
public class faaliyet_alani
{  static void Main()
   {
       { int x=29;
         Console.WriteLine(x);
       }
       { int x=19;
        Console.WriteLine(x);
       }
   }
}
```



Değişkenlerin Faaliyet Alanları (Scopes)

- Faaliyet alanı devam eden bir değişkenin tekrar tanımlanması derleme esnasında hataya yol açar.

```
using System;
public class faaliyet_alani
{
   static void Main()
   {
            int x;
            { int x=20;
            }
   }
}
```

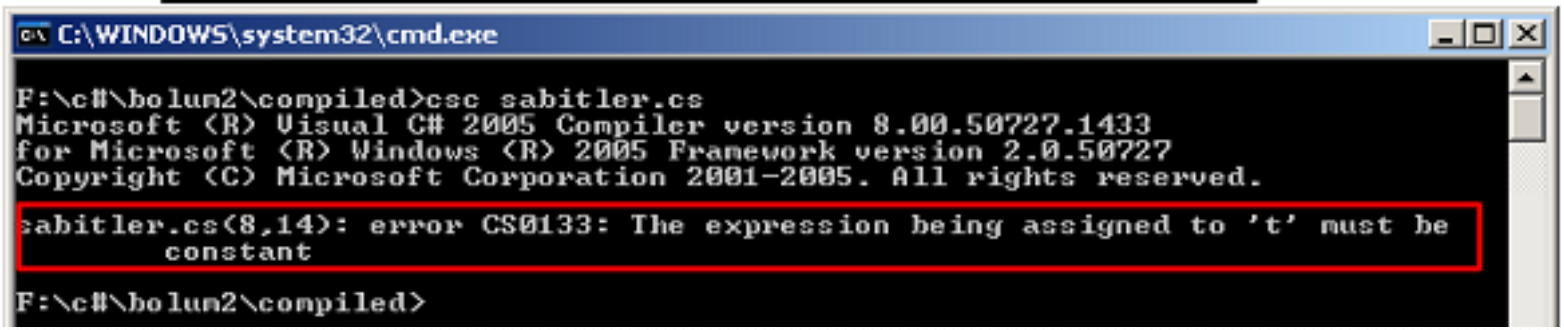
Sabitler

- Program boyunca değerinin değişmeyeceği düşünülen veriler sabit olarak tanımlanırlar. Bu tanımlamayı yapmak için tanımlama satırının başında const anahtar sözcüğünü kullanırız.

- Sabitlere ilk değer verilirken yine sabitler kullanılmalıdır. Değişken tanımlamada olduğu gibi sabitlerde de tanımlandıklarında mutlaka ilk değerleri verilmelidir.
- **const double pi=3.14**

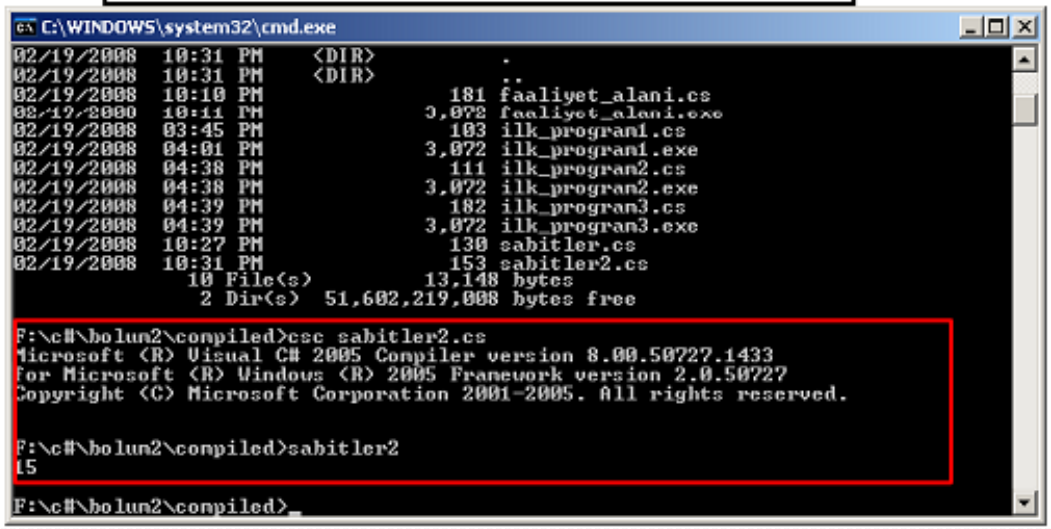
Sabitler

```
using System;
public class faaliyet_alani
{
   static void Main()
   {
            int x=5,y;
            y=10;
            const int t=x+y;
   }
}
```



Sabitler

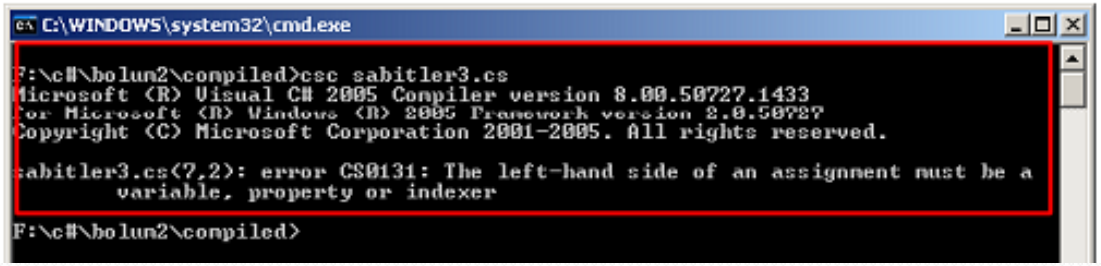
```
using System;
public class faaliyet_alani
{
   static void Main()
   {
            const int x=5,y=10;
            const int t=x+y;
            Console.WriteLine(t);

   }
}
```



Sabitler

```
using System;
public class faaliyet_alani
{   static void Main()
   {
            const int x=5,y=10;
            x+=2;
            const int t=x+y;
            Console.WriteLine(t);
   }
}
```

- Derleme gerçekleşmemektedir.
- Neden?



Sabitler

- Sabitlerle ilgili olarak 3 temel kural vardır: Bunlar:
- Sabitler tanımlandıklarında değerleri atanmalıdır. İlk değer verilmeyen değişkenler sabit olamazlar.
- Sabit ifadelere ancak sabit ifadelerle ilk değer atanabilir.
- Sabit ifadeler kendi yapılarından dolayı static bir nesne oldukları için ayrıca static anahtar sözcüğü kullanılmaz.

Değer ve Referans Tipleri

- Değer tipleri değişkenin değerini direkt bellek bölgesinden alırlar.

- Referans tipleri ise başka bir nesneye referans olarak kullanılırlar. Diğer bir değişler referans tipleri, heap alanında yaratılan nesnelerin adreslerini saklarlar.

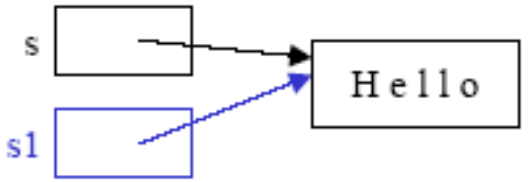
- Değer tipleri yaratıldıklarında stack bölgesinde oluşturulurlar. Referans tipleri ise kullanımı biraz daha sınırlı olan heap bellek bölgesinde saklanırlar.

Değer ve Referans Tipleri

- Temel veri tipleri (int,double, float ...) değer tipi; herhangi bir sınıf türü ise referans tipidir.
- İki değer tipi nesnesi birbirine eşitlenirken değişkenlerde saklanan değerler kopyalanarak eşitlenir ve bu durumda iki yeni bağımsız nesne elde edilmiş olur. Birinin değerini değiştirmek diğerini etkilemez.
- Fakat, iki referans tipi birbirine eşitlendiğinde bu nesnelerde tutulan veriler kopyalanmaz, işlem yapılan nesnelerin heap bölgesindeki adresleridir.
- İki nesne heap bölgesinde aynı yeri gösterdiği için, birinde yapılan değişiklik diğerini de etkileyecektir.

Değer ve Referans Tipleri

	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	int i = 17; int j = i;	string s = "Hello"; string s1 = s;

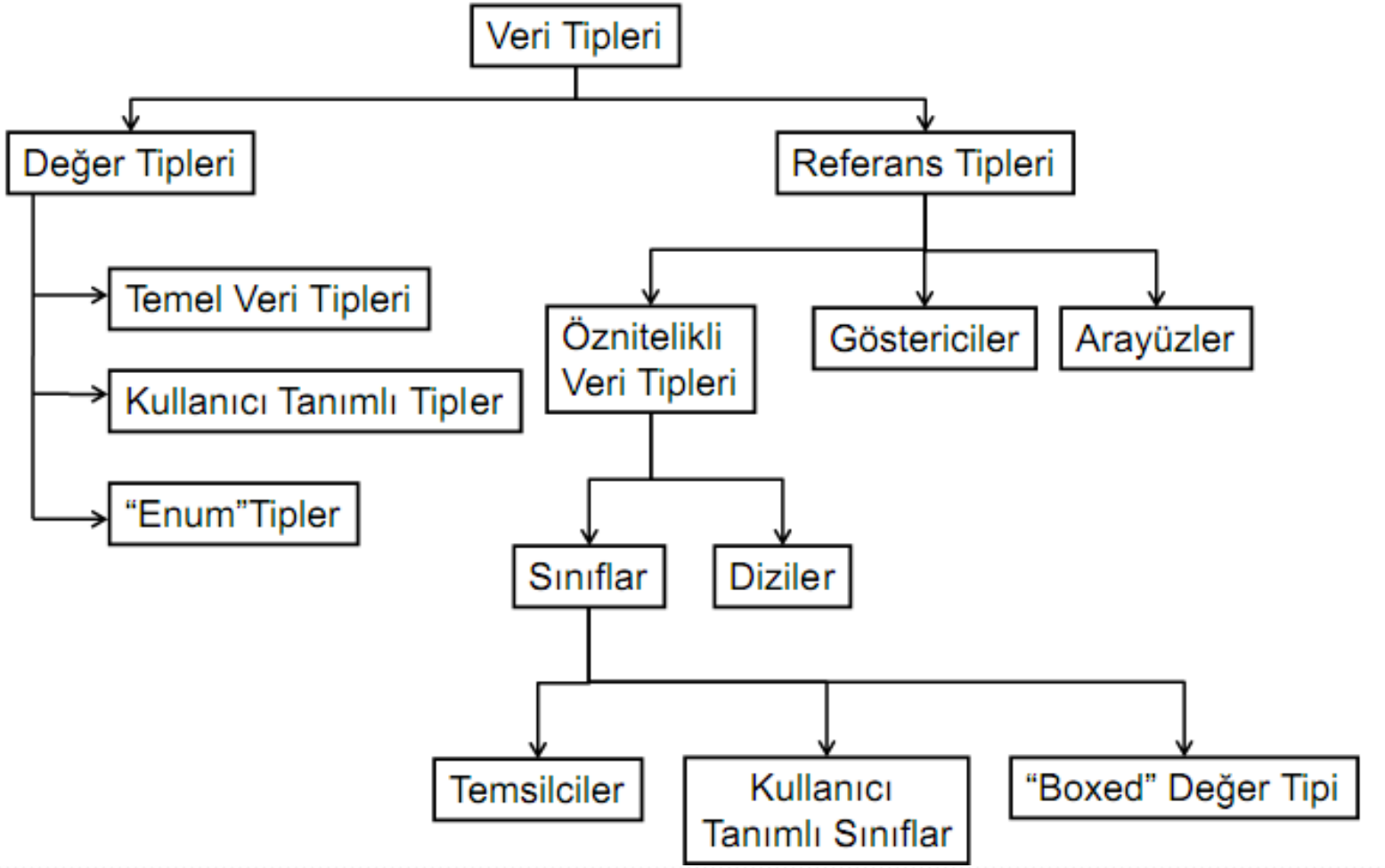


Değer ve Referans Tipleri

- CTS sayesinde, .NET platformu için geliştirilen bütün diller aynı veri tiplerini kullanırlar. Tek değişen veri türlerini tanımlama yöntemi ve sözdizimidir.

- C#'da önceden tanımlanmış temel veri tipleri 15 tanedir. (13 tanesi değer tipi, 2 tanesi ise referans tipi)

Değer ve Referans Tipleri



Değer Tipleri (Value Types)

- Değer tiplerinin tamamı **Object** denilen bir nesneden türemiştir. C#'da her nesne ya da veri tipi aslında **Object** tipidir.

- Değer tiplerinde bir nesnenin değeri direkt olarak saklıdır. Tanımlanan değer tiplerine aşağıdaki şekilde ilk değer ataması yapılabilir.

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

Bu noktada a üzerindeki değişikliklerden b etkilenmeyecektir.

Değer Tipleri (Value Types)

- Değer tiplerine ilk değer verme;

  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

Değer Tipleri (Value Types)

Veri Tipi	Varsayılan Değer
bool	false
byte	0
char	'\0'
decimal	0.0M
double	0.0D
enum	enum sabiti tanımlamasındaki ilk değer
float	0.0F
int	0
long	0L
sbyte	0
short	0
struct	yapı içinde yer alan tüm değer tipleri varsayılan değere, referans tipler ise "null" değere atanır.
uint	0
ulong	0
ushort	0

Değer Tipleri (Value Types)

- Örn: Aşağıdaki programı bilgisayarınızda deneyin.

```
 using System;
public class varsayilan_degerler {
        static void Main() {
        bool a =new bool(); byte a1=new byte();
        char a2=new char(); decimal a3=new decimal();
        double a4=new double();
        float a5=new float();
        Console.WriteLine(a);
        Console.WriteLine(a1);                    Console.WriteLine(a2);
        Console.WriteLine(a3);                    Console.WriteLine(a4);
        Console.WriteLine(a5);
                                }
}
```

C# Tipi	.NET Framework	Tanım	Değer Aralığı
object	System.Object	Tüm CTS türleri için temel sınıf	-
bool	System.Boolean	Mantıksal Doğru/Yanlış	true ya da false
byte	System.Byte	8 bit işaretsiz tamsayı	0 ~ 255
sbyte	System.SByte	8 bit işaretli tamsayı	128 ~ 127
char	System.Char	Karakterleri temsil eder	16 Unicode karakterleri
decimal	System.Decimal	128 bit ondalıklı sayı	$\pm 1,5*10^{-28} \sim \pm 7,9*10^{28}$
double	System.Double	64 bit çift kayan sayı	$\pm 5*10^{-324} \sim \pm 1,7*10^{308}$
float	System.Single	32 bit tek kayan sayı	$\pm 1,5*10^{-45} \sim \pm 3,4*10^{38}$
int	System.Int32	32 bit işaretli tamsayı	-2.147.483.648 ~ 2.147.483.647
uint	System.UInt32	32 bit işaretsiz tamsayı	0 ~ 4.294.967.295
long	System.Int64	64 bit işaretli tamsayı	9.223.372.036.854.775.808 ~ -9.223.372.036.854.775.807
ulong	System.UInt64	64 bit işaretsiz tamsayı	0 ~ 18.446.744.073.709.551.615
short	System.Int16	16 bit işaretli tamsayı	-32.768 ~ 32.767
ushort	System.UInt16	16 bit işaretsiz tamsayı	0 ~ 65.535
string	System.String	Karakter Dizisi	Unicode Karakter Dizisi

Referans Tipleri (Reference Types)

- C# ‘ ta önceden tanımlı iki referans tipi vardır. **Object** ve **String.**

- Object türü C#’ta bütün türlerin türediği sınıftır. Diğer bir deyişle Object türünden bir nesneye herhangi bir veri türünden nesneyi atayabiliriz.

- Object türü özelleştirilerek farklı amaçlara yönelik kullanılabilirler. Object’e eşleştirme (Boxing) işlemi ve tersi, Object’i dönüştürme (Unboxing)

String Türü

- Referans türünden olan stringler, türü Unicode karakterlerden oluşan bir dizi gibi algılanmalıdır.

```
Strings1="Merhaba";
Strings2=".NET";
Strings3=s1+s2;
```

Stringleri arka arkaya eklemek için **+** operatörü kullanılır.

String Türü

- Özel anlamlar içeren karakterleri ifade etmek için \ ifadesini kullanırız (escape). Örn:

- **String path="C:\\WINDOWS\\assembly"**

- String içinde görünen ifadenin aynısını belirtmek için string ifadesinin önüne @ işareti konulur.Örn:

- **String path=@"C:\WINDOWS\assembly"**

Object Veri Türü

- Her nesne object türünden olduğu için bütün değerler ve nesneler object türünden bir değişkene atanabilir.

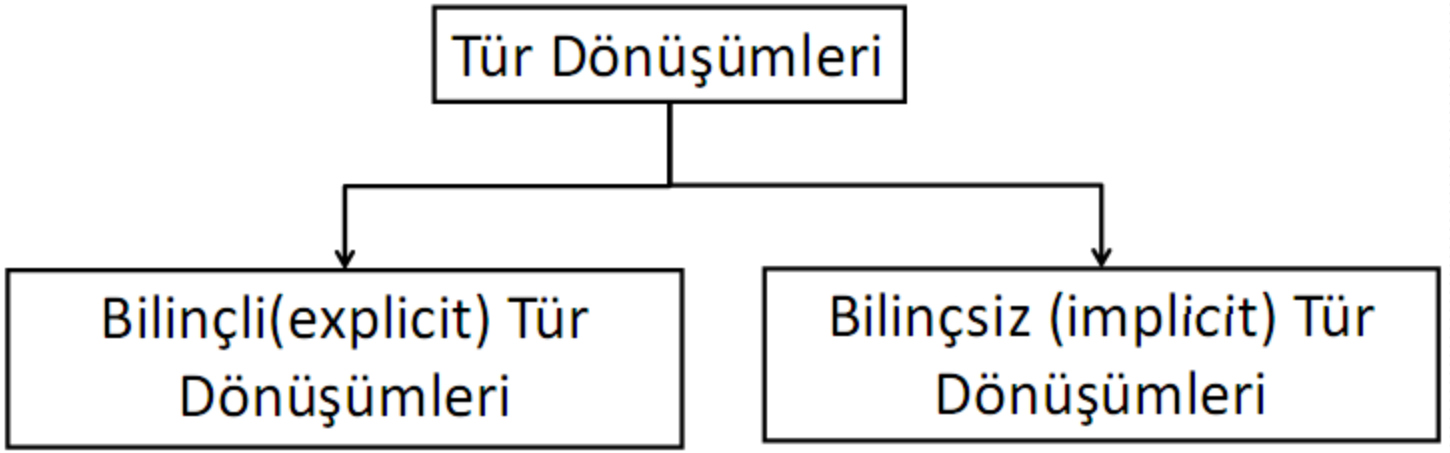
Örnek

Aşağıdaki programı deneyiniz.

```
using System;
public class varsayilan_degerler
{
   static void Main()
   {
            object x;
            x=10;
            Console.WriteLine(x.GetType());
            x="B";
            Console.WriteLine(x.GetType());
            x=8.78F;
            Console.WriteLine(x.GetType());
            x=false;
            Console.WriteLine(x.GetType());
            x=5.489M;
            Console.WriteLine(x.GetType());
   }
}
```

Tür Dönüşümleri

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



Bilinçsiz (Impicit) 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 (Impicit) Tür Dönüşümleri

- Bilinçsiz yapılan tür dönüşümlerinde bir nesnenin türü asla kalıcı olarak değiştirilmez. Bilinçsiz yapılan tür dönüşümleri 2 şekilde gerçekleştirilebilir.
  - Küçük Türün Büyük Türe Dönüştürülmesi
  - Büyük Türün Küçük Türe Dönüştürülmesi

Tür Dönüşümleri

- **Küçük türün büyük türe dönüştürülmesi (Otomotik tip dönüşümü):**
- Küçük tür büyük türe dönüştürülürken fazla bitler sıfır ile doldurulur. Küçük türün yüksek anlamlı bitlerinin sıfırla beslenmesi değişkendeki değeri değiştirmediği için tür dönüşümünde herhangi bir veri kaybı olmaz.

Örn: byte a=12;

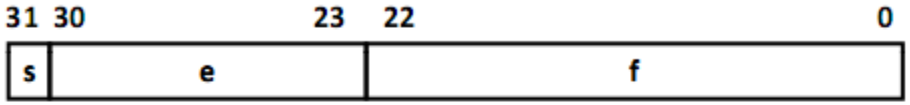
```
int b;
b=a;
```

Tür Dönüşümleri

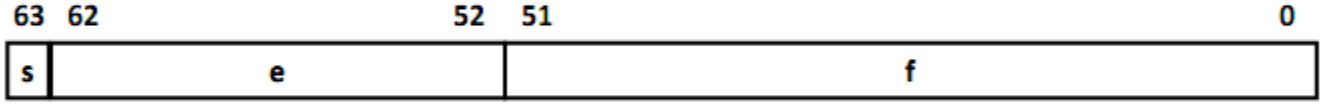
```
using System;
public class Tur_donusumu1
{
  static void Main()
  {
            byte a=20;
            int b;
            b=a;
            Console.WriteLine(b);

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

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

Tür Dönüşümleri

float sayı formatı



double sayı formatı



s : işaret biti

e : üsleri temsil eden bitler

f : noktalı (anlamlı) kısım

Tür Dönüşümleri

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

Tip Dönüşümleri

```
using System;
class Otomatik_tip
{
        public static void Main()       {
                int a;
                float b=32.32f;
                double c;
                c=b;
Console.WriteLine("b'nin değeri="+b+"\nc'nin değeri="+c);
        }
}
```

Tür Dönüşümleri

```
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;
  }
}
```

Tür Dönüşümleri

- Bazı türler arasında tür dönüşümü yapmak 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)

Tür Dönüşümleri

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

- İstenmeyen durum. Ancak "()" cast operatörü ile yapılır.

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) degişken_yada_sabit;

Tür Dönüştürme Operatörü

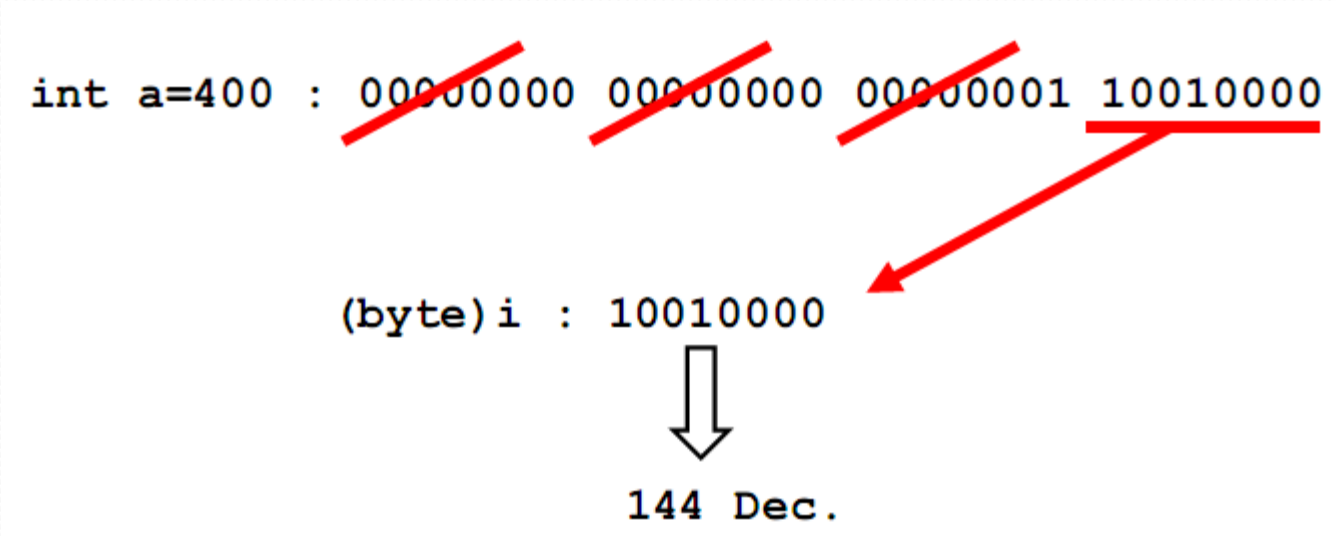
✓ Bilinçsiz yapılan tür dönüşümlerinde büyük türler, küçük türlere dönüştürülemiyordu. Eğer tür dönüştürme operatörü kullanılırsa bu işlem mümkün olur. Örn: Aşağıdaki programı deneyin:

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

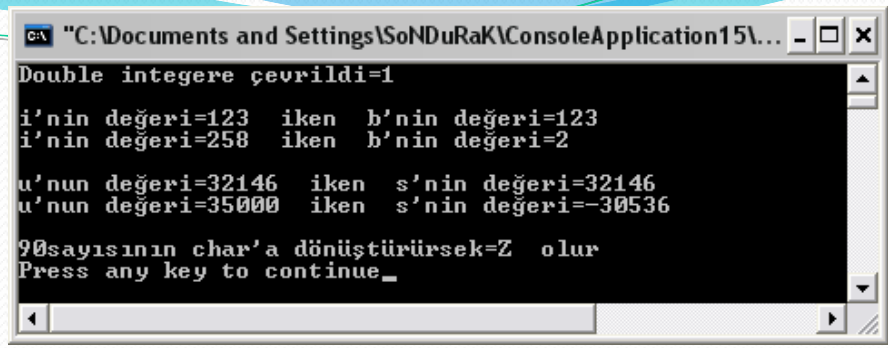
Tür Dönüştürme Operatörü

✓ Programı çalıştırdığımızda ekrana 144 yazdırdı.

Neden?



Tür Dönüşümleri




```
using System;
class Bilinçli_tip {
 public static void Main() {
    double d1,d2; int i;    byte b; char c;  uint u; short s;  d1=5.0;d2=4.0;
    //double int e dönüştü veri kaybı var,virgülden sonrası atılır
    i= (int) (d1/d2);Console.WriteLine("Double integere çevrildi="+i); Console.WriteLine();
   //int'i byte dönüştür, Veri kaybı yok.
    i=123;   b=(byte) i; Console.WriteLine("i'nin değeri="+i+"  iken  b'nin değeri="+b);
   //Veri kaybı var.
    i=258;   b=(byte) i; Console.WriteLine("i'nin değeri="+i+"  iken  b'nin değeri="+b);
   Console.WriteLine();
   //uint'i short'a dönüştür
    u=32146;   s=(short) u;   //Veri kaybı yok.
   Console.WriteLine("u'nun değeri="+u+"  iken  s'nin değeri="+s);
   u=35000;    s=(short) u;  //Veri kaybı var.
   Console.WriteLine("u'nun değeri="+u+"  iken  s'nin değeri="+s);  Console.WriteLine();
    //int'i char'a dönüştür.
   i=90; c=(char) i; Console.WriteLine(i+"sayısının char'a dönüştürürsek="+c+"  olur"); }
}
```

Checked ve Unchecked

- Tür dönüşümlerinde veri kayıplarında programa hata uyarısı verdirebilmek için checked deyimi kullanılır. Checked anahtar sözcüğü ile çalışma zamanında oluşabilecek veri kayıplarının olabileceği durumlarda hata vermesini sağlayabiliriz.

```
// unchecked checked işlemini ters çevirir.
using System;
class turdonusum   {
        static void Main() {
                    int i=256;
                    byte b;
                    checked  //Taşma olduğundan program hata verir.
                    {
                b=(byte) i;
                  }
                   Console.WriteLine(b);
        }
}
```

Checked ve Unchecked

- checked bir blok oluşturduğu için içinde yapılan değişken tanımlamaları dış bloklarda kullanılamaz.

```
using System;
class turdonusum    {
        static void Main() {
                int i=256;
                checked
                {
                 byteb=(byte) i;
                   }
                 Console.WriteLine(b); // Hata verir
        }
}
```

Checked ve Unchecked

- Normal şartlarda yapılan işlemler "**unchecked**"'dir Böyle bir ifadenin konmasının nedeni uzun "checked" blokların oluşturulması istenebilir.

- Bu durumlarda çok fazla blok oluşturmamak için "unchecked" ifadesi kullanılabilir.

Checked ve Unchecked

```
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.

- C# herşey nesne(object) referans türünden türetilmiştir. Temelde bir sınıf vardır. Örneğin object sınıfının ToString() metodu bütün temel veri ve referans türlerinde kullanılır.

ToString() Metodu

- **.ToString()** metodu bütün temel türlerde ya da referans türlerde kullanılabilir. Amacı ise string'e dönüşüm işlemdi yapmaktır.

- string str = 345.59f.ToString()

  - – **56.ToString();**
  - – **12.34F.ToString();**

ToString() Metodu

Örn:

…..

```
Static void Main() {
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

Boxing İşlemi

- Günümüzdeki popüler dillerde referans ve değer tipleri arasında dönüşüm yapılmamaktadır.

- Böyle bir çevrime ihtiyaç duyulduğunda "Boxing" kutulama yapılır. Bu yöntem değer tipindeki verileri "object" nesnesine çevirir.

- 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.

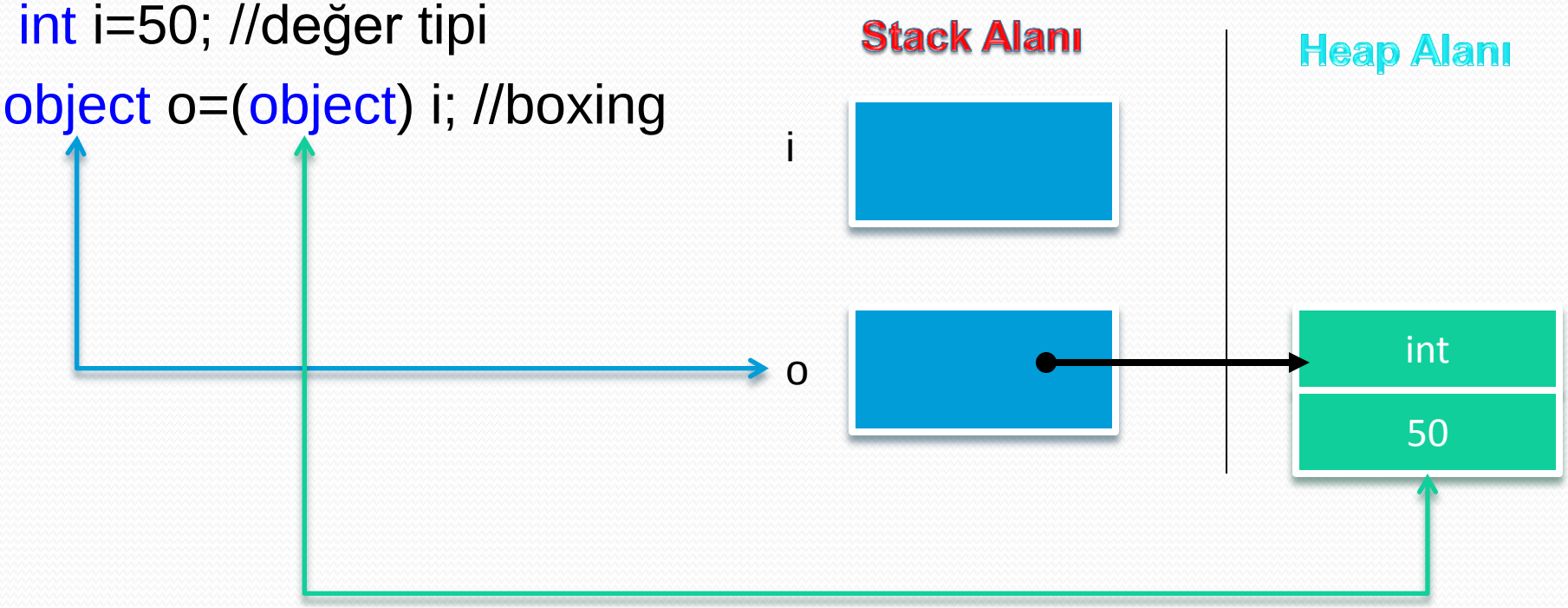
Örn: Bilinçsiz boxing işlemi.

```
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
```



Unboxing İşlemi

- Heap alanındaki nesnelerin değerlerinin bit olarak stack bölgesine kopyalanması işlemine "**unboxing**" adı verilir.

- Boxing işleminin tam tersidir. Bu işlem sonucunda referans türler değer türüne dönüştürülmüş olur. Aşağıdaki koşullara uyularak yapılmalıdır.

  - Unboxing işlemine tabi tutulacak nesnenin daha önceden boxing işlemine tabi tutulmuş olması gerekir.

  - Boxing işlemine tabi tutulmuş olan bu nesnenin unboxing işlemi sırasında doğru türe dönüştürülmesidir

Unboxing İşlemi

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

Örn: int i=50;

object o=i;

int j=**(int)**o;

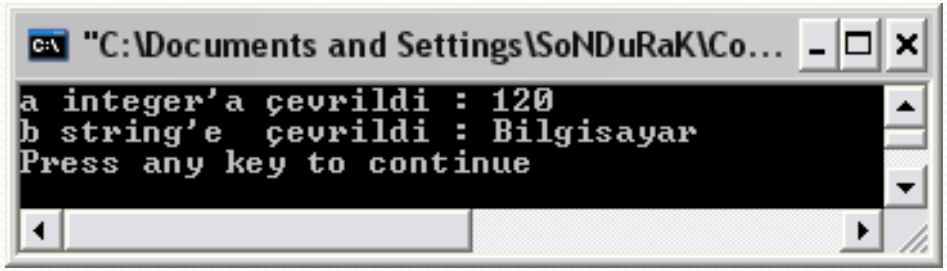
Unboxing İşlemi

- 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);
    }
}
```

- **Örnek:**

```
using System;
class Boxing_UnBoxing_metodu{
        static void Main()   {
                object a=120;          //a'ya Boxing uygulandı
                object b="Bilgisayar"; //b'ye Boxing uygulandı
                int sayı=(int) a;      //a'ya UnBoxing uygulandı
                string str=(string) b;          //b'ye UnBoxing uygulandı
                Console.WriteLine("a integer'a çevrildi : "+sayı);
                Console.WriteLine("b string'e  çevrildi : "+str);
        }
}
```



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.

System.Convert Sınıfı ile Tür 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);
```

System.Convert ile Tür Dönüşümü

Metot	Açıklama
Convert.ToBoolean(str)	str nesnesini bool türüne çevirir.
Convert.ToByte(str)	str nesnesini byte türüne çevirir.
Convert.ToSByte(str)	str nesnesini Signed Byte türüne çevirir.
Convert.ToInt16(str)	str nesnesini short türüne çevirir.
Convert.ToUInt16(str)	str nesnesini ushort türüne çevirir.
Convert.ToInt32(str)	str nesnesini int türüne çevirir.
Convert.ToUInt32(str)	str nesnesini uint türüne çevirir.
Convert.ToInt64 (str)	str nesnesini long türüne çevirir.
Convert.ToUInt64(str)	str nesnesini ulong türüne çevirir.
Convert.ToSingle(str)	str nesnesini float türüne çevirir.
Convert.ToDouble(str)	str nesnesini double türüne çevirir.
Convert.ToDecimal(str)	str nesnesini decimal türüne çevirir.
Convert.ToChar(str)	str nesnesini char türüne çevirir.

System.Convert İle Tür Dönüşümü

```
using System;
Class convert_metodu {
        static void Main()
    {
                int a=50;
                string d="50";
                int b=Convert.ToInt32(a);
                double c=Convert.ToDouble(d);
                Console.WriteLine("b : "+b+"\nc : "+c);
     }
}
```

System.Convert İle Tür Dönüşümü

```
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());
    }
}
```

System.Convert İle Tür Dönüşümü

- 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);
    }
}
```

```
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); }
```

Visual Studio.Net -C#

3. HAFTA

Operatörler, Akış kontrol mekanizmaları, Diziler