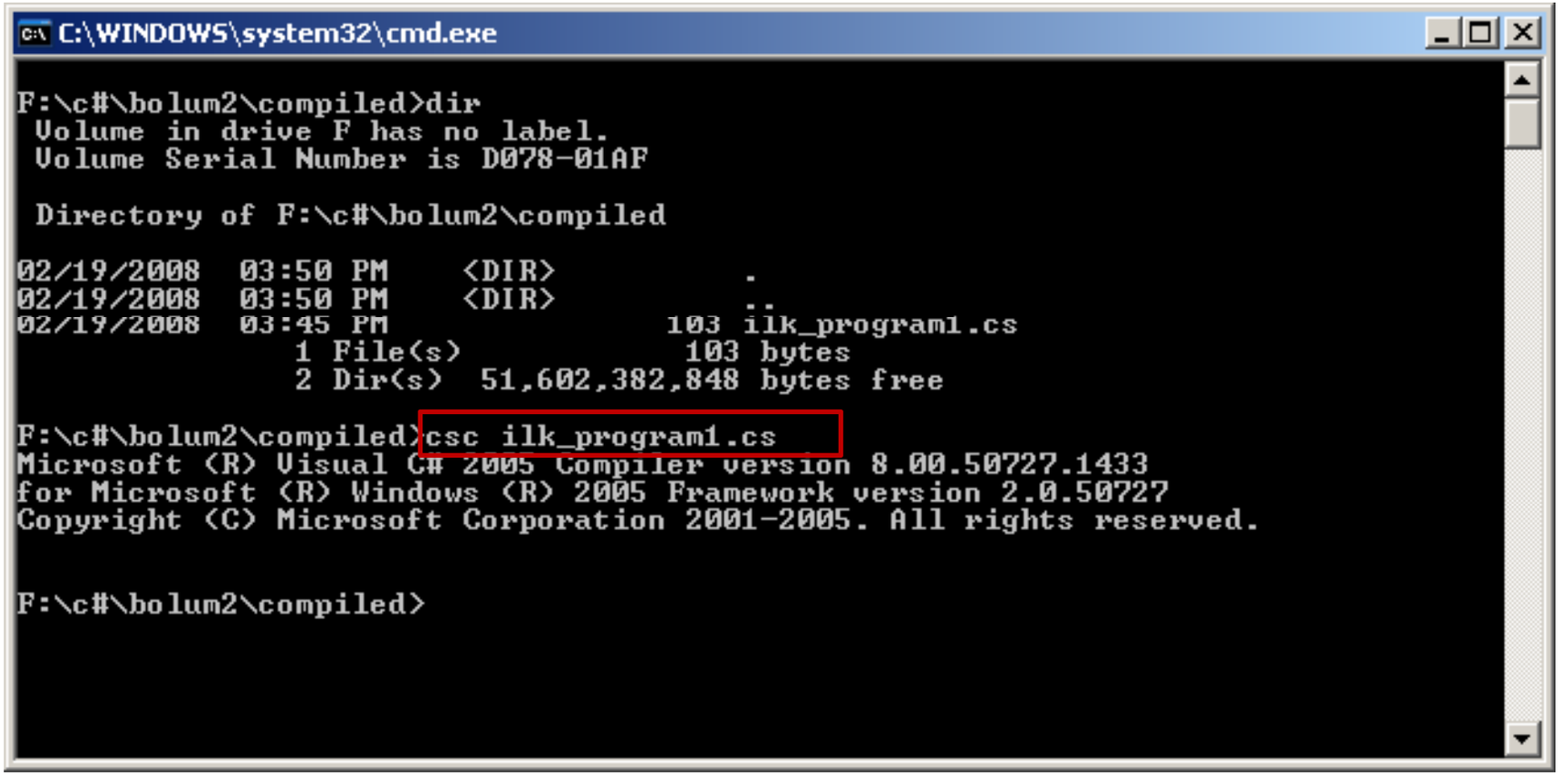
Temel Veri Türleri

İlk C# Programı

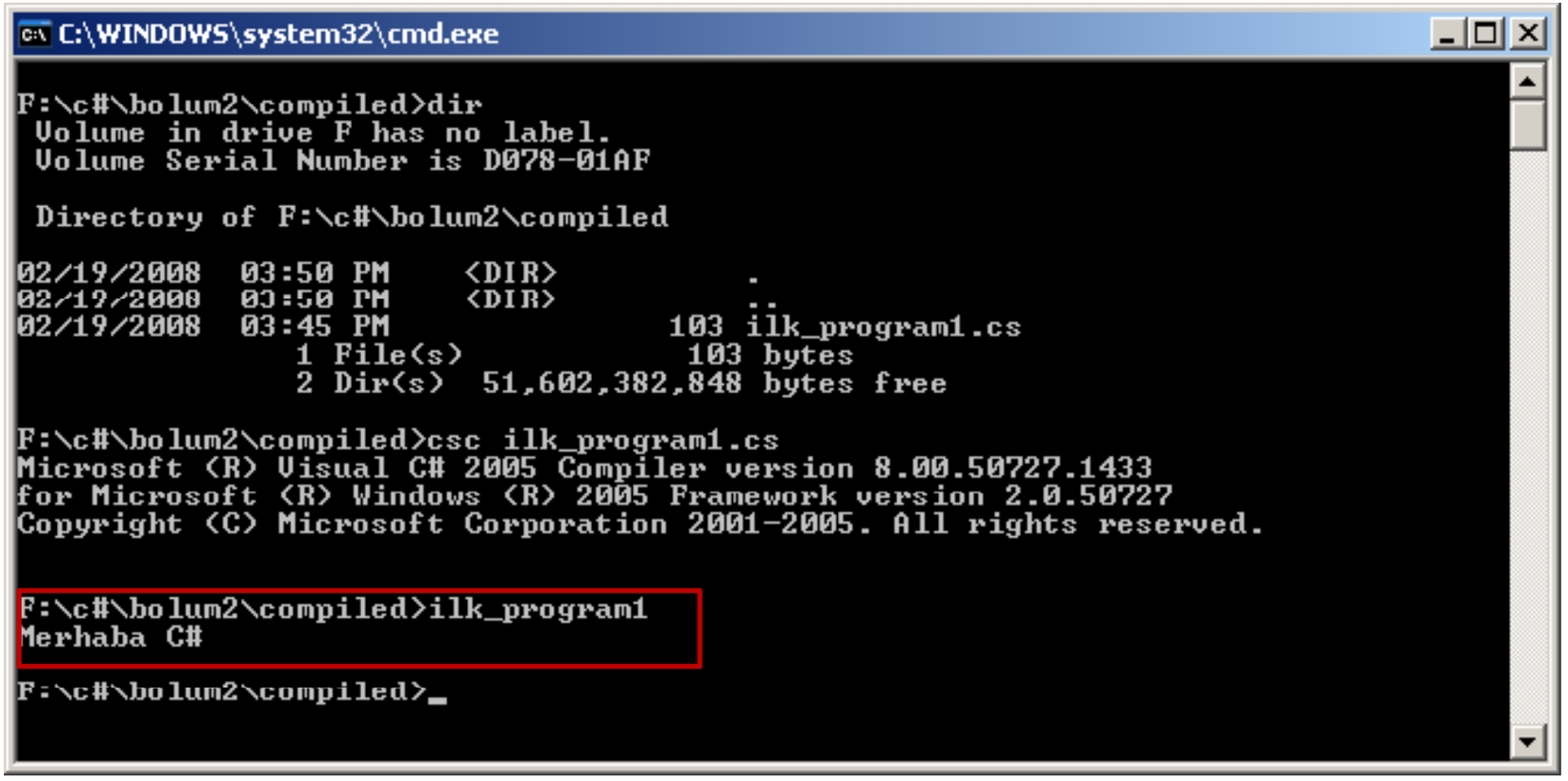
```
class ilk_program1
{ static void Main()
    {
            System.Console.WriteLine("Merhaba C#");
    }
}
```

C# dilinde yazılmış kaynak kod dosyalarının uzantıları .cs'dır. Kaynak kodu içeren dosyanın cs uzantılı olması zorunlu değildir. Ancak bu şekildeki adlandırma yaygındır.



Yazdığımız kaynak program csc yardımıyla derlenir. Csc'nin bir çok farklı parametresi csc/? yazılarak görüntülenebilir.

Artık çalıştırılabilir bir dosyamız aynı klasör içerisinde oluşturulmuştur.



```
class ilk_program1
{ static void Main()
    {
            System.Console.WriteLine("Merhaba C#");
    }
}
```

C# dili daha önce de denildiği gibi %100 nesne yönelimli bir dildir. Nesne olmayan hiçbir şey yoktur. C ve C++ dillerinde programın çalışması main işlevinden başlar ancak main işlevi hiçbir zaman bir sınıf içerisinde olmamıştır.

C# dilinde herşey sınıflarla temsil edildiği için main işlevi de bizim belirlediğimiz bir sınıfın işlevi olmak zorundadır.

- Bütün C# programları en az bir sınıf içermelidir. Sınıf bildirimi içinde olmayan programlar derlenmez.

- Main() işlevi bizim için C ve C++ dillerinde olduğu gibi programımızın başlangıç noktasıdır.

- C'de diğer bazı dillerde olduğu gibi kaynak koddaki bütün satırlar ";" ile sonlandırılır. (Bazı durumlar hariç.)

- Sınıflar ve işlevler açılan ve kapanan küme parantezler { } içerisine yazılırlar.

- C# dilinde birçok kavram sınıf dediğimiz nesneler üzerine kurulmuştur. Her sınıfın iş yapan çeşitli elemanları vardır. İş yapan bu elemanlara metot ya da işlev denilmektedir.

.NET' i meydana getiren sınıf kütüphanesi hiyerarşik bir yapı sunmaktadır. Sınıflar isim alanı (namespace) dediğimiz kavramla erişilmesi kolay bir hale gelmiştir.

```
using System;

namespace Ornekalan1
{
   public class Program1
   {
      static void Main(string[] args)
      {

      }
   }

   public class Program2
   {
      public void foksiyon1()
      {

      }
      public void fonksiyon2()
      {

      }
   }

}

namespace Ornekalan2
{
   public class Program3
   {

   }

}
```

```
using System;
class ilk_program2
{ static void Main()
    {
            Console.WriteLine("Merhaba C#");
    }
}
```

"using System" deyimi ile System isim alanındaki bütün sınıflara doğrudan erişim hakkına sahip oluruz.

```
using System;
class ilk_program2
{  static void Main()
    {
            Console.WriteLine("Bir tusa basin...");
            Console.ReadLine();
            Console.WriteLine("Bir tusa basin...");
    }
}
```

ReadLine metodu da WriteLine gibi kullanılır ancak metodun parantezlerine herhangi bir şey yazılmaz.

C# Dilindeki 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.

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.
- 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.
- 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:

  <veri türü> <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.

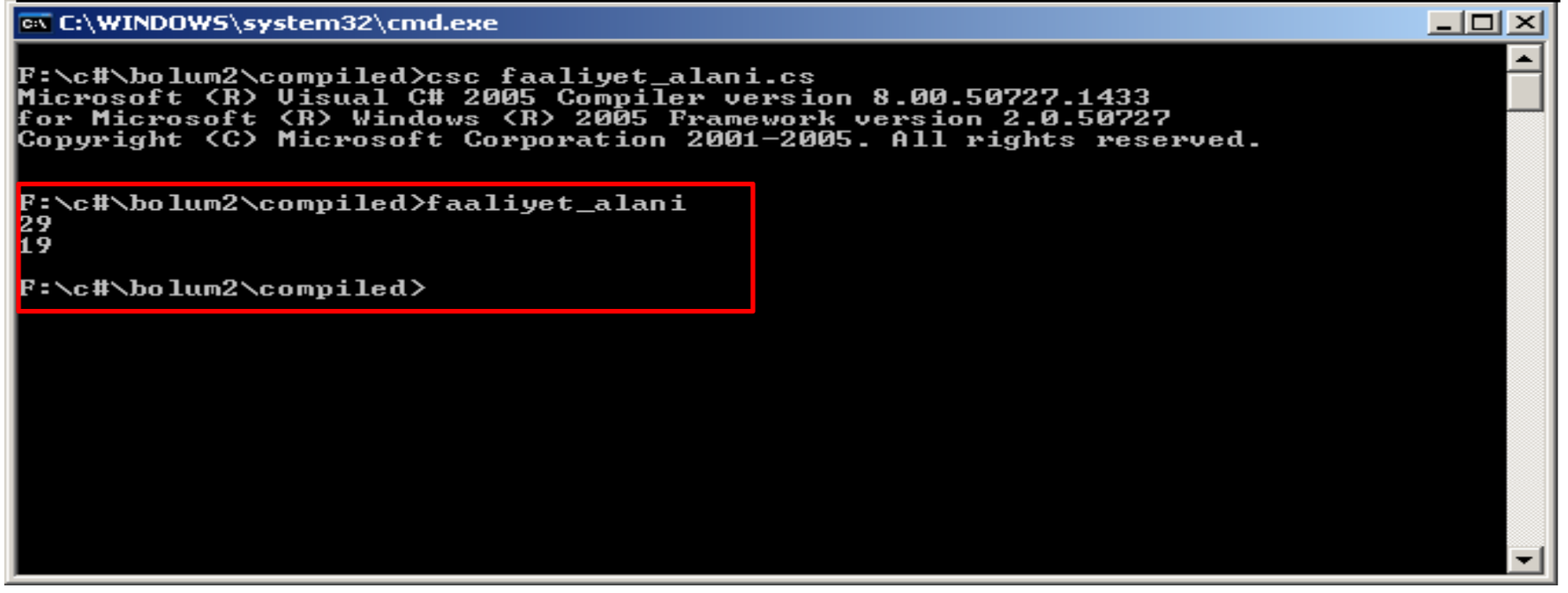
Değişken isimlendirme ile ilgili temel kuralları aşağıdaki şekilde özetlemek mümkündür:

- 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.

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



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

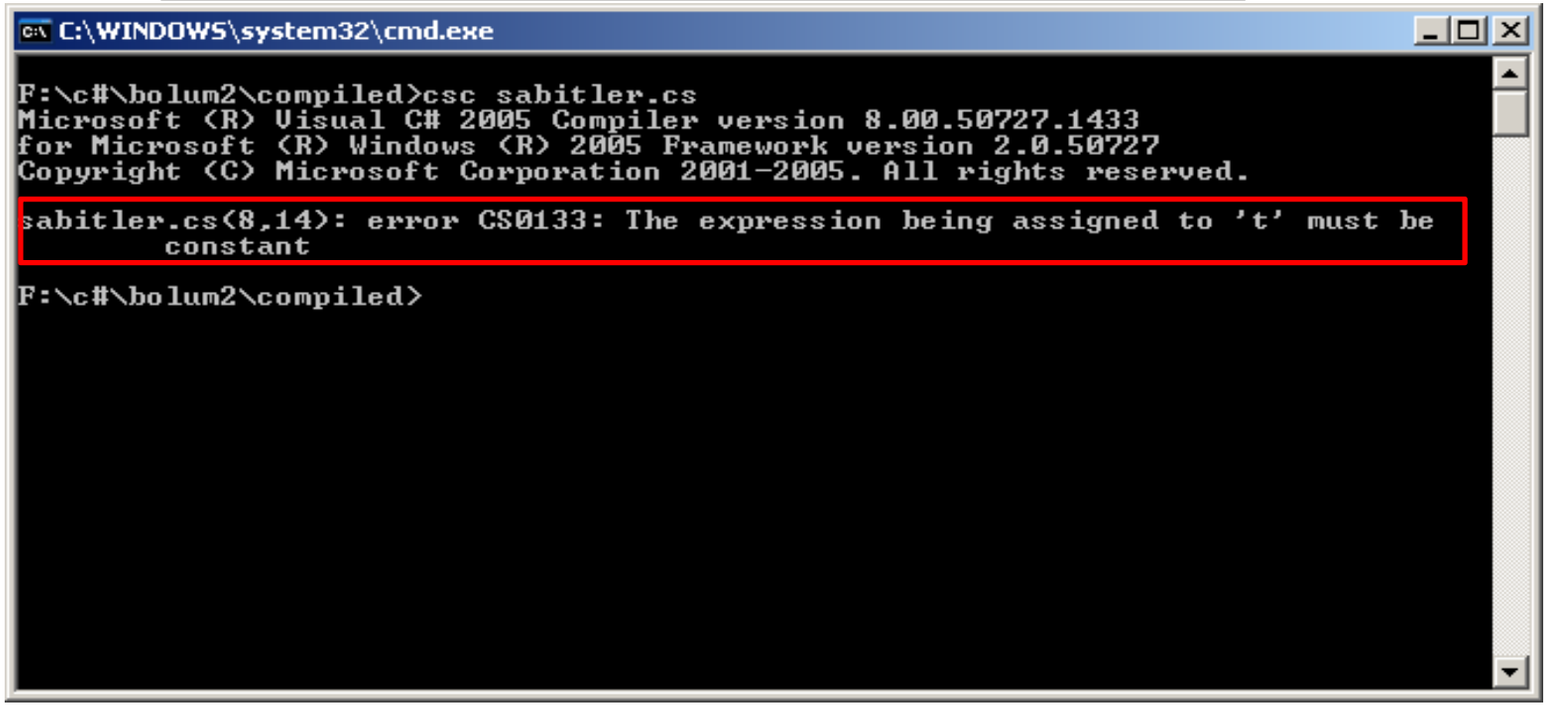
```
using System;
public class faaliyet_alani
{
   static void Main()
   {
            int x;
            { int x=20;
            }
   }
}
```

Sabitler

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

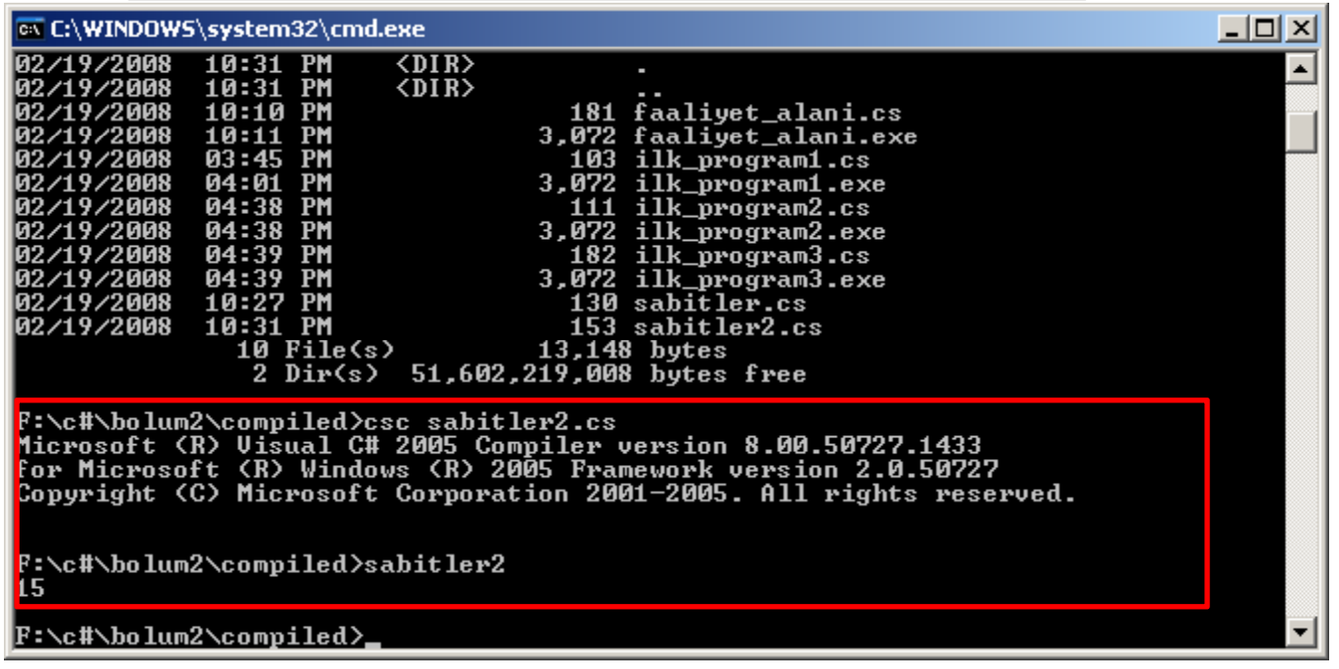
- Sabitlere ilk değer verilirken yine sabitler kullanılmalıdır. Değişken tanımlamada olduğu gibi sabitlerde de tanımlandıklarında mutlaka ilk değerleri verilmelidir.

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



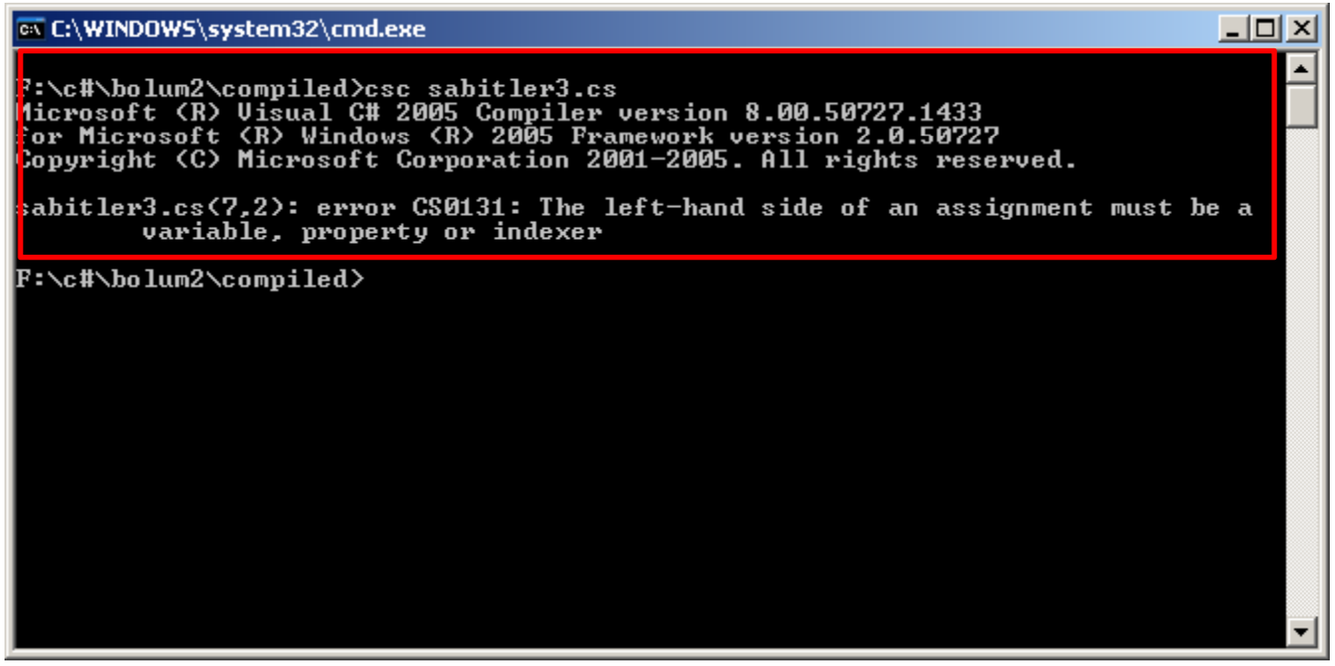
```
using System;
public class faaliyet_alani
{
  static void Main()
  {
        const int x=5,y=10;
        const int t=x+y;
        Console.WriteLine(t);

  }
}
```



```
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?



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.

- 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.

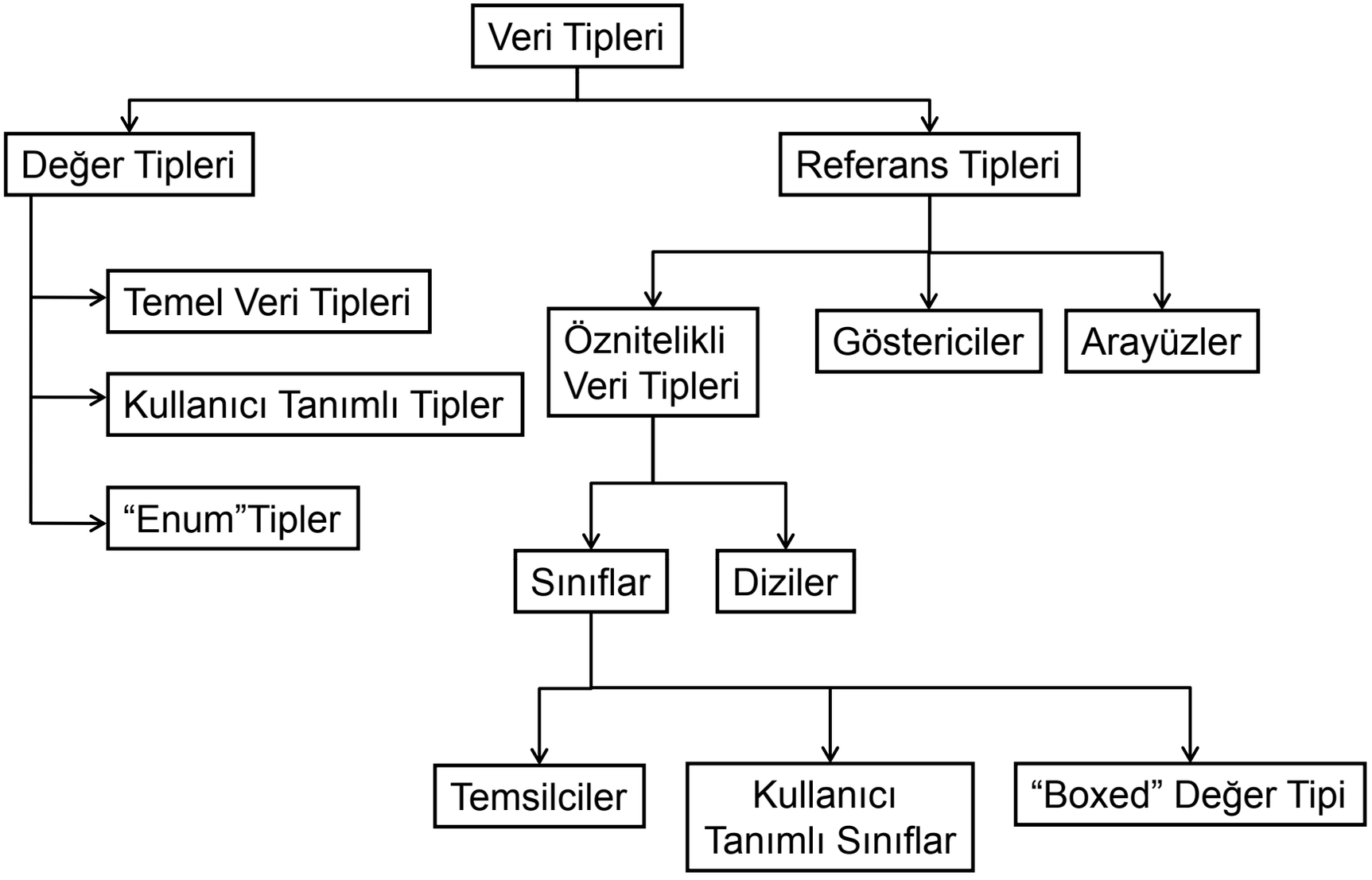
Fakat, iki referans tipi birbirine eşitlendiğinde bu nesnelerde tutulan veriler kopyalanmaz, işlem yapılan nesnelerin heap bölgesindeki adresleridir.

İki nesne heap bölgesinde aynı yeri gösterdiği için, birinde yapılan değişiklik diğerini de etkileyecektir.

CTS (Common Type System)

- 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)


Veri Tipleri
Değer Tipleri
Referans Tipleri
Temel Veri Tipleri
Kullanıcı Tanımlı Tipler
“Enum”Tipler
Öznitelikli Veri Tipleri
Göstericiler
Arayüzler
Sınıflar
Diziler
Temsilciler
Kullanıcı Tanımlı Sınıflar
“Boxed” Değer Tipi

Değer Tipleri

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

- Tanımlanan değer tiplerine aşağıdaki şekilde ilk değer atması yapılabilir. Örn:

```
int a;

a=new int();     veya          a=0;
```

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**

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

- C#'da önceden tanımlanmış 2 tane temel referans tipi vardır. Bunlar string ve object türleridir.
- Object türü C#'da bütün türlerin türediği bir sınıf yapısıdır. Kullanıcı tarafından sonradan tanımlanacak olan bütün veri tipleri aslında Object türünden türemiş olacaktır.
- Diğer bir deyişle Object türünden bir nesneye herhangi bir veri türünden nesneyi atayabiliriz. Çünkü C#'da bütün nesneler birer Object'tir.

String Türü

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

```
String s1="Merhaba";
String s2=".NET";
String s3=s1+s2;
```

Stringleri arka arkaya eklemek için + operatörü kullanılır.

- 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.

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());
   }
}
```