

Yapıcılar ve Kalıtım

- C#'ta yapıcı metotların türetimiyle ilgili şu kurallar geçerlidir:
 - C#'ta yapıcı metotlar fiziksel olarak türetilmez.
 - Yavru sınıf türünden bir nesne yaratıldığında önce ana sınıfın parametre almayan yapıcı metodu, ardından yavru sınıftaki imzaya uyan yapıcı metot çalıştırılır.
 - Yavru sınıf türünden nesne yaratımında daima yavru sınıfın imzaya uyan bir yapıcı metodu olması gerekir.
 - Yavru sınıf türünden nesne yaratımlarında, ana sınıfın parametre almayan yapıcı metodu yavru sınıfın üye elemanlarıyla işlem yapar.
 - Yavru sınıf türünden nesne yaratımında, yavru sınıftaki ilgili (imzası uygun) yapıcı metoda base takısı eklenmişse ana sınıfın parametre almayan yapıcı metodu çalıştırılmaz.

Yapıcılar ve Kalıtım

```
using System;
class ana
{
    public ana()
    {
        Console.WriteLine("ana sınıfının parametre almayan yapıcı metodu");
    }
}
class yavru:ana
{
    public yavru(int a)
    {
        Console.WriteLine("yavru sınıfının parametre alan yapıcı metodu. alınan parametre: "+a);
    }
}
class esas
{
    static void Main()
    {
        yavru y=new yavru(5);
    }
}
```

- Madde 2:
 - Bu programda ekrana alt alta ana sınıfının parametre almayan yapıcı metodu ve yavru sınıfının parametre alan yapıcı metodu. alınan parametre: 5 yazacaktır.

Not: Constructor içinde yazan Console.WriteLine sadece gösterim amaçlıdır.
Bir sınıf tasarımında sınıf içinde Console.WriteLine önermem... Çünkü ileri de sadece Console'a değil bir çok ortama kod yazacağız. Aynı kodu farklı ortamlardan kullanmak için Sınıfımızı iyi tasarlamalıyız.

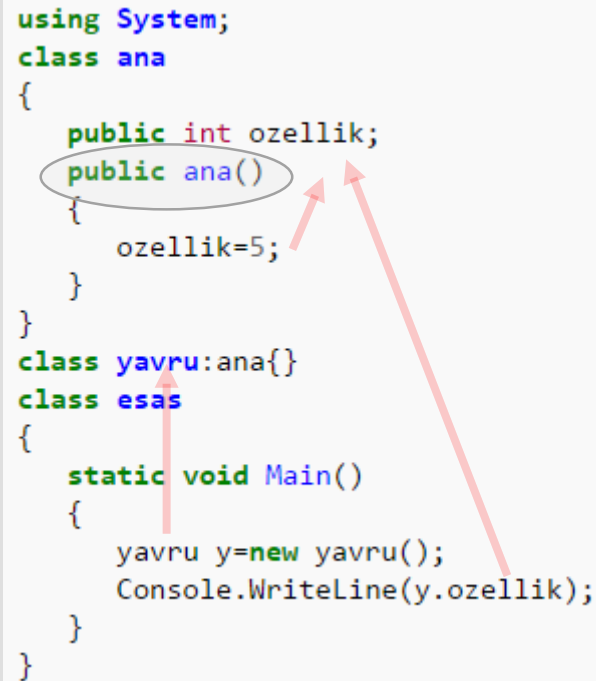
Yapıcılar ve Kalıtım

```
using System;
class ana
{
    public ana()
    {
        Console.WriteLine("ana sınıfının parametre almayan yapıcı metodu");
    }
}
class yavru:ana{}
class esas
{
    static void Main()
    {
        yavru y=new yavru();
    }
}
```

- Bu program hata vermez. Çünkü sınıflar konusunda öğrendiğimiz üzere bir sınıfta hiç yapıcı metot olmadığı durumlarda varsayılan yapıcı metot oluşturulur. Varsayılan yapıcı metot parametre almaz ve hiçbir şey yapmaz. Bu örnekte y nesnesini yaratırken parametre verseydik yukarıda bahsettiğimiz üçüncü kural ihlal edilmiş olacaktı ve dolayısıyla programımız hata verecekti.

Yapıcılar ve Kalıtım

```
using System;
class ana
{
    public int ozellik;
    public ana()
    {
        ozellik=5;
    }
}
class yavru:ana{}
class esas
{
    static void Main()
    {
        yavru y=new yavru();
        Console.WriteLine(y.ozellik);
    }
}
```



- Bu örnekte ekrana 5 yazılacaktır. Aslına bakarsanız dördüncü kuralın tersinin olması imkansız. Çünkü zaten ortada ana tipinden bir nesne yok, o yüzden ana sınıfının parametre almayan yapıcı metodunun kendi sınıfının üye elemanlarıyla çalışması bu örnekte imkansız. Ancak yine de fiziksel olarak bir sınıfta olmayan bir yapıcı metodun o sınıfın üye elemanlarıyla çalışması ilginç.

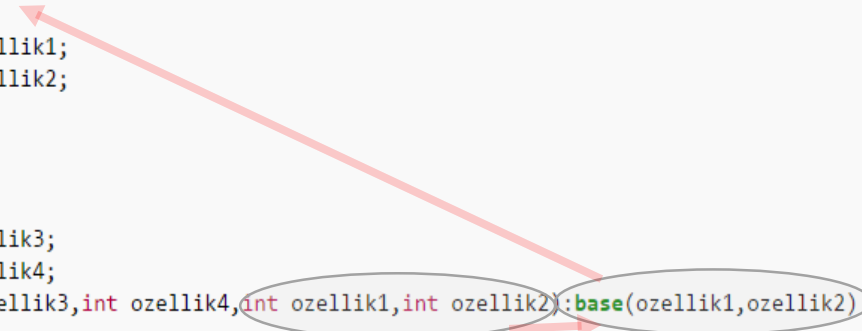
Yapıcılar ve Kalıtım

```
using System;
class ana
{
    public ana(int a){}
}
class yavru:ana{}
class esas
{
    static void Main()
    {
        yavru y=new yavru();
    }
}
```

- **Bu program hata verir.** Çünkü yukarıda saydığımız ikinci kuralı ihlal etmektedir. Çünkü ana sınıfta parametre almayan bir yapıcı metot yoktur. Ana sınıfta bir yapıcı metot tanımlandığı için varsayılan yapıcı metot oluşturulmamıştır.

Yapıcılar ve Kalıtım

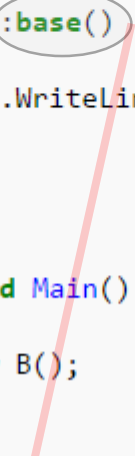
```
using System;
class A
{
    public int Ozellik1;
    public int Ozellik2;
    public A()
    {
        Console.WriteLine("Deneme"),
    }
    public A(int ozellik1,int ozellik2)
    {
        Ozellik1=ozellik1;
        Ozellik2=ozellik2;
    }
}
class B:A
{
    public int Ozellik3;
    public int Ozellik4;
    public B(int ozellik3,int ozellik4,int ozellik1,int ozellik2):base(ozellik1,ozellik2)
    {
        Ozellik3=ozellik3;
        Ozellik4=ozellik4;
    }
}
class esas
{
    static void Main()
    {
        B b=new B(3,4,1,2);
        Console.WriteLine(b.Ozellik1+" "+b.Ozellik2+" "+b.Ozellik3+" "+b.Ozellik4);
    }
}
```



- Bu program ekrana 1 2 3 4 yazar. Bu örnekte base anahtar sözcüğü ana sınıftaki yapıcı metodu temsil etmektedir. Örneğimizde yavru sınıfın yapıcı metodu 4 parametre almakta, bu aldığı parametrelerin ikisini kendi bloğunda kullanmakta kalan iki parametreyi de ana sınıfın imzası uygun yapıcı metoduna göndermektedir. Ana sınıfın imzası uygun yapıcı metodu çalıştığında yavru sınıfın üye elemanlarıyla işlem yapacaktır. Asıl konumuza gelecek olursak bu örnekte yavru sınıfın bir yapıcı metoduna eklenen base takısı ile ana sınıfın bir yapıcı metodunu çalıştırdığımız için yavru sınıftaki base takısı eklenmiş ilgili yapıcı metodu çalıştıracak şekilde yavru sınıf türünden bir nesne yaratıldığında ana sınıfın parametre almayan yapıcı metodu çalıştırılmayacaktır.

Yapıcılar ve Kalıtım

```
using System;
class A
{
    public A()
    {
        Console.WriteLine("A sınıfı");
    }
}
class B:A
{
    public B():base()
    {
        Console.WriteLine("B sınıfı");
    }
}
class Ana
{
    static void Main()
    {
        B b=new B();
    }
}
```



Olmasa da olur.
Default (Geçerli) Durumdur.

- NOT1: base anahtar sözcüğü bu şekilde yalnızca yapıcı metotlarla kullanılabilir. Yani base anahtar sözcüğünü yalnızca yavru sınıftaki yapıcı metoda ekleyebiliriz ve base anahtar sözcüğünün ana sınıfta var olan bir yapıcı metodu belirtmesi gerekir.
- NOT2: Yavru sınıfın yapıcı metoduna base takısı eklenip ana sınıfın bir yapıcı metodu çalıştırıldığı durumlarda önce ana sınıfın ilgili (imzaya uyan) yapıcı metodu, ardından yavru sınıfın ilgili (imzaya uyan) yapıcı metodu çalıştırılır.

Yapıcılar ve Kalıtım

- Bu programda alt alta A sınıfı ve B sınıfı yazılacaktır. Şimdi isterseniz bir de statik yapıcı metotların kalıtımdaki rolüne bakalım.

```
B sınıfının statik yapıcı metodu
A sınıfının statik yapıcı metodu
A sınıfının statik olmayan yapıcı metodu
B sınıfının statik olmayan yapıcı metodu
-----
A sınıfının statik olmayan yapıcı metodu
B sınıfının statik olmayan yapıcı metodu
```

```
using System;
class A
{
    static A()
    {
        Console.WriteLine("A sınıfının statik yapıcı metodu");
    }
    public A()
    {
        Console.WriteLine("A sınıfının statik olmayan yapıcı metodu");
    }
}
class B:A
{
    static B()
    {
        Console.WriteLine("B sınıfının statik yapıcı metodu");
    }
    public B()
    {
        Console.WriteLine("B sınıfının statik olmayan yapıcı metodu");
    }
}
class Ana
{
    static void Main()
    {
        B b1=new B();
        Console.WriteLine("-----");
        B b2=new B();
    }
}
```

Static'ler başlangıçta çalışır.
Daha sonra çalışmaz.

Harici yapıcılar her nesne oluşumda çalışır.