

## 5.6 İsim Saklama

Hem ana hem de alt sınıfta aynı isimli üye eleman varsa ana sınıftaki üye eleman gizlenir. Buna bilinçsiz saklama (gizleme) denir ve ana sınıftaki üyeye direk olarak ulaşılamaz.

### Örnek:

```
using System;
namespace Kalitim
{
    class Sinif1
    {
        public int sayi = 100;
    }
    class Sinif2 : Sinif1
    {
        public int sayi = 200;
    }

    class Program
    {
        static void Main(string[] args)
        {
            Sinif2 y = new Sinif2();
            Console.WriteLine(y.sayi);
            Console.ReadKey();
        }
    }
}
```

### Çıktısı:

200

Bilinçsiz saklama hata üretmez ama uyarı üretir:

```
C:\Users\yemre\source\repos\operatorAsiriYukleme\Program.cs(10,20,10,24): warning CS0108: 'Sinif2.sayi' hides inherited member 'Sinif1.sayi'. Use the new keyword if hiding was intended.
```

Böyle bir durumda isim saklamayı açıkça belirtmek gerekir. İsim saklamayı açıkça belirtmek için `new` anahtar sözcüğünü kullanırız. Yukarıdaki programda sınıfları şu şekilde yazarsak derleyici uyarı vermez:

```
class Sinif1
{
```

```

        public int sayi = 100;
    }
    class Sinif2 : Sinif1
    {
        public new int sayi = 200;
    }

```

Gizlediğimiz üye elemana erişmek içinse **base** anahtar sözcüğünü kullanırız:

```

using System;
namespace Kalitim
{
    class Sinif1
    {
        public int sayi = 100;
    }
    class Sinif2 : Sinif1
    {
        public new int sayi = 200;
        internal int ogren()
        {
            return base.sayi;
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Sinif2 y = new Sinif2();
            Console.WriteLine(y.ogren());
            Console.ReadKey();
        }
    }
}

```

**Çıktısı:**

100

## 5.7 Türetilen Sınıfta Aynı İsimde Metot Tanımlama

Ana sınıf ve alt sınıfta aynı isimde metotlar varsa ana sınıftaki metot (bilinçsiz) saklanır.

**Örnek**

```

using System;
namespace Kalitim

```

```

{
    class Sinif1
    {
        public void topla()
        {
            Console.WriteLine("Sinif1 fonksiyonu çalıştı.");
        }
    }
    class Sinif2 : Sinif1
    {
        public void topla()
        {
            Console.WriteLine("Sinif2 fonksiyonu çalıştı.");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Sinif1 abc = new Sinif1();
            abc.topla();

            Sinif2 def = new Sinif2();
            def.topla();

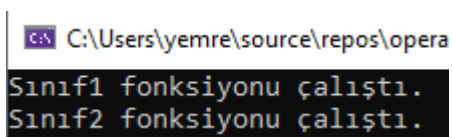
            Console.Read();
        }
    }
}

```

Derleyici aşağıdaki uyarıyı verir:

'Sinif2.topla()' hides inherited member 'Sinif1.topla()'. Use the new keyword if hiding was intended.

**Çıktısı:**



```

C:\Users\yemre\source\repos\opera
Sinif1 fonksiyonu çalıştı.
Sinif2 fonksiyonu çalıştı.

```

Yukarıdaki kodda uyarı almamak için (metot saklamayı bilinçli yapmak için) alt sınıf metod tanımlamasında new kullanılır:

```

class Sinif1
{
    public void topla()
    {

```

```

        Console.WriteLine("Sınıf1 fonksiyonu çalıştı.");
    }
}
class Sinif2 : Sinif1
{
    public new void topla()
    {
        Console.WriteLine("Sınıf2 fonksiyonu çalıştı.");
    }
}

```

## 5.8 Sanal Metotlar (Virtual)

Sanal metot, temel sınıf içinde bildirilmiş ve türemiş sınıf içinde de tekrar bildirilen metotlardır. Sanal metotlar kullanılarak çok biçimlilik uygulanmış olur.

Temel sınıfta bir sanal metot bildirildiğinde bu temel sınıftan türeyen sınıflardaki metotlar temel sınıftaki sanal metodu devre dışı bırakabilirler (`override`).

Sanal metotları bildirmek için `virtual` anahtar sözcüğü kullanılır.

Türeyen sınıfta ise, temel sınıftaki sanal metotları devre dışı bırakmak için `override` anahtar sözcüğü kullanılır. Eğer `override` edilirse alt sınıf içindeki metot, `override` edilmezse ana sınıf içindeki `virtual` metot çalışır.

Virtual metotlar `private` olarak tanımlanamazlar, `public`, `protected`, `internal` şeklinde bildirilebilirler.

### Örnek:

```

class Personel
{
    protected double tabanMaas, maas;
    public virtual double maasHesapla()
    {
        maas = tabanMaas;
        return maas;
    }
}
class YetkiliPersonel : Personel
{
    double satisTutari;
    public override double maasHesapla()
    {

```

```
        maas = tabanMaas + satisTutari * 0.1;
        return maas;
    }
}
```