

# Nesneye Dayalı Programlama

Sakarya Üniversitesi  
Bilgisayar ve Bilişim Bilimleri Fakültesi  
Bilgisayar Mühendisliği

Prof. Dr. Ümit Kocabiçak  
Prof. Dr. Cemil Öz

7. HAFTA

## 6. Hafta İçeriği

1. Kalıtım (Inheritance)
2. Soyut sınıf (abstract class)
3. Sealed sınıf(sealed class)
4. Çok biçimlilik ([polymorphism](#))
- 5.

Kalıtım(inheritance), nesne yönelimli programlamanın temel özelliklerinden biridir ve en önemli olanıdır.

Kalıtım, mevcut olan sınıflardan veya temel sınıflardan, türetilmiş sınıf denilen yeni sınıflar oluşturma işlemidir.

Türetilmiş sınıf temel sınıfın tüm özelliklerini taşır ve üye fonksiyonlarını kullanır. Üstelik kendisine ait özellikler ve ilaveler katılabilir.

Temel sınıf ise çocuk sınıflardan etkilenmez.

Kalıtım, Nesne yönelimli programlamanın başlıca parçasıdır. En büyük avantajı ise kodun yeniden kullanılmasına izin vermesidir.

Bir sınıfı yazdıktan ve hatalardan arındırdıktan sonra, bu sınıfa dokunmadan kalıtım özelliği ile değişik durumlara çözüm olarak düzenlenebilir.

- Kalıtım var olan bir sınıftan yeni sınıflar türetme sağlar
- Var olan sınıf ebeveyn(parent), süper veya temel(base class) sınıf olarak isimlendirilir
- Türetilen sınıf çocuk(child) veya alt sınıf (Subclass) olarak isimlendirilir.
- Çocuk sınıf ebeveyn sınıfının özelliklerini miras alır.
  - Temel sınıftaki metotlar ve alanlar
- Çocuk sınıf ebeveyn sınıf metotları ve verileri üzerinde özel haklara sahiptir.
  - Public'de her kes gibi
  - Protected'da erişim sadece çocuk sınıfta
- Çocuk sınıflar kendi özel metodlar ve verilere sahip olabilirler

## Temel Sınıf



## Türetilmiş sınıf



Türetilinen sınıfı gösterir

Türetilmiş sınıf içinde tanımlanmış

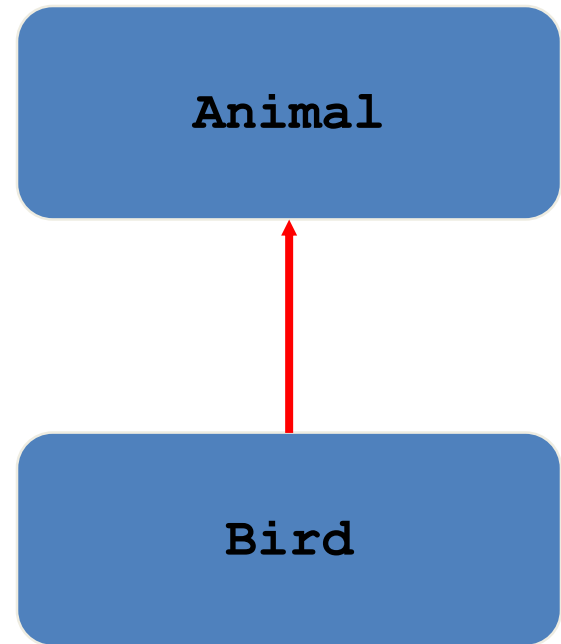
Temel sınıf içinde tanımlanmış

Bu özelliklere türetilmiş sınıftan da erişilebilir

Kalıtım(inheritance)

# Inheritance

- Kalıtım ilişkisi genellikle sınıf diyagramında temel sınıfı işaret eden bir ok ile gösterilir.
- Kalıtım ***is-a** ilişkisi* oluşturmalıdır yani çocuk temel sınıfın daha spesifik sürümü olmalıdır.



# Temel sınıf ve türetilmiş sınıf örnekleri

Temel sınıf	Türetilmiş sınıf
Ogrenci	MasterOgrenci LisansOgrenci PhdOgrenci
GeoNesne	Daire Ucgen Dikdortgen Kare
Personel	Memur OgretimUyesi Mudur
Bankahesabı	TasarrufHesabı MaasHesabı DovizHesabı

```
class Otomobil  
{  
  ...  
}
```

Temel sınıf

temel sınıf özellikleri



```
class SporOtomobil : Otomobil  
{  
  ...  
}
```

iki nokta Temel sınıf

Çocuk sınıf

Burada SporOtomobil sınıfı, Otomobil sınıfından türetilmiştir.



- Çocuk sınıf temel sınıfın metot ve verilerini miras alır. Ancak , erişim durumu bu üyelerin erişim belirteçlerine bağlıdır.
- Private olarak tanımlanan metotlar ve değişkenler çocuk sınıftan erişilemez.
  - Bununla birlikte temel sınıfın private üyeleri çocuk sınıfın bir parçasıdır.
- Public tanımlanan metot ve değişken üyeler erişilebilir. Fakat kapsülleme olayına zarar verebilir.
- Protected, kalıtım sorununu çözer.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Kalitim
{
    class Otomobil
    {
        // temel sınıf
        public string alan1 = "temel sınıf alanı";
        public void Metod1(string deger)
        {
            Console.WriteLine("temel sınıf -- Metod1: {0}", deger);
        }
    }
    class SporOtomobil : Otomobil
    {
        // Türetilmiş sınıf
        public string alan2 = "Türetilmiş sınıf alanı";
        public void Metod2(string deger)
        {
            Console.WriteLine("Türetilmiş sınıf -- Metod2: {0}", deger);
        }
    }
    class Program
    {
        static void Main()
        {
            SporOtomobil so = new SporOtomobil ();
            so.Metod1(so.alan1); // Temel sınıf alanı ile temel sınıf metodu
            so.Metod1(so.alan2); // Türetilmiş sınıf alanı ile temel sınıf metodu
            so.Metod2(so.alan1); // Türetilmiş sınıf metodu ile temel sınıf alanı
            so.Metod2(so.alan2); // Türetilmiş sınıf metodu ile türetilmiş sınıf alanı
            Console.ReadKey();
        }
    }
}

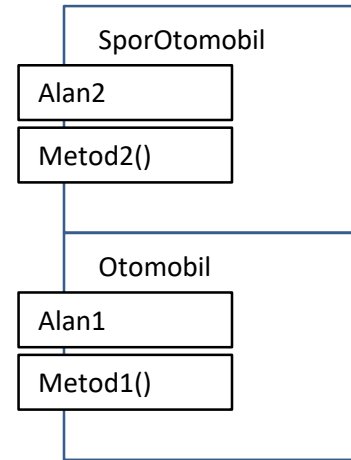
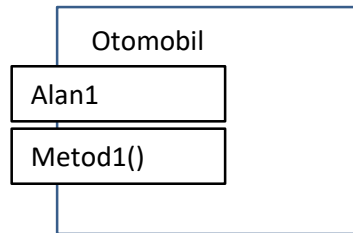
```

file:///C:/Users/cemiloz/Documents/Visual Studio 2015/Projec... — □ ×

```

temel sınıf -- Metod1: temel sınıf alanı
temel sınıf -- Metod1: Türetilmiş sınıf alanı
Türetilmiş sınıf -- Metod2: temel sınıf alanı
Türetilmiş sınıf -- Metod2: Türetilmiş sınıf alanı

```



## **C# tekli kalıtımı destekler**

Bir sınıf sadece bir temel sınıftan miras alır.

## **C++ çoklu kalıtımı destekler**

Bir çocuk sınıf bir den fazla temel sınıftan miras alır

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _01_kalitim
{
    public class anaSınıf
    {
        public anaSınıf()
        {
            Console.WriteLine("ana sınıf kurucusu");
        }
        public void yaz()
        {
            Console.WriteLine("Ben sınıftayım");
        }
    }
    public class ogulSınıf:anaSınıf
    {
        public ogulSınıf()
        {
            Console.WriteLine("Ogul Kurucu");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            ogulSınıf ogul = new ogulSınıf();
            ogul.yaz();
            Console.ReadKey();
        }
    }
}
```

Object sınıfı haricindeki tüm sınıflar object sınıfından türetilmiştir.

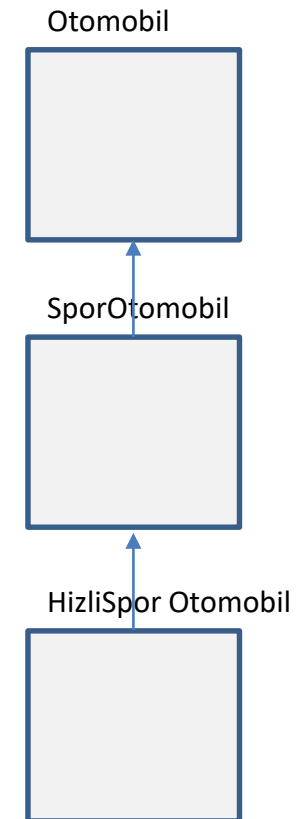
```
class Otomobil  
{  
  
}
```

Object sınıfından kapalı tanımlama

```
class Otomobil : object  
{  
  
}
```

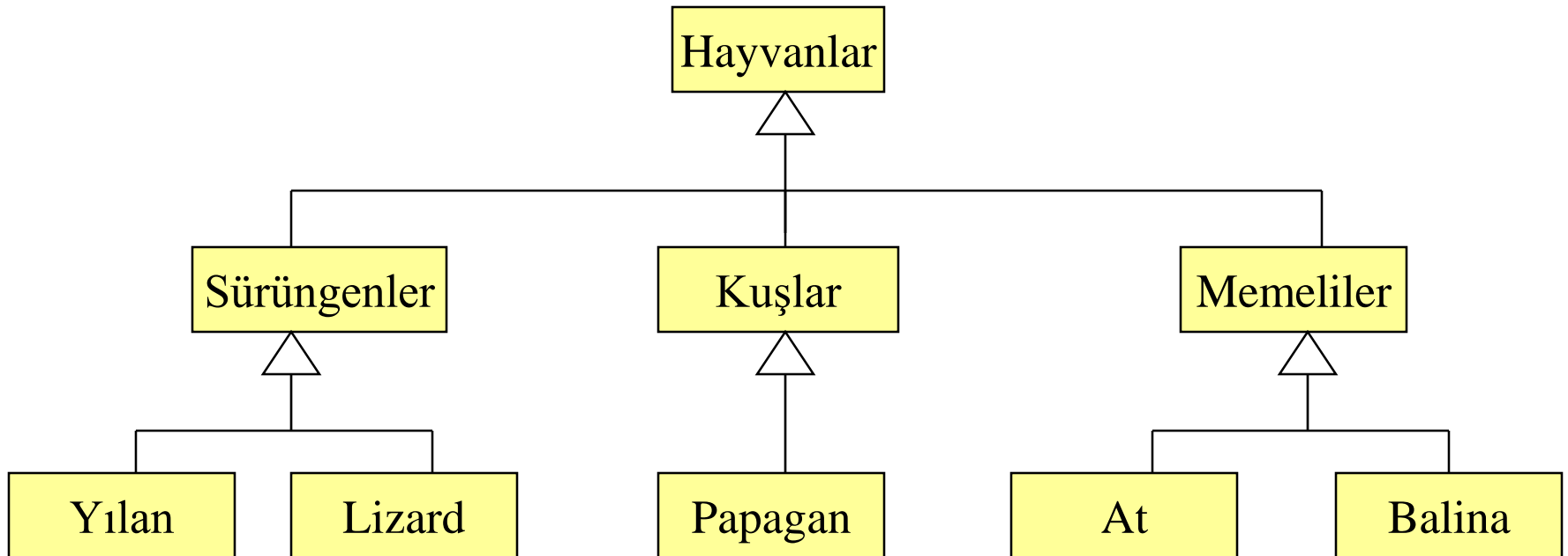
Object sınıfından açık tanımlama

```
class Otomobil
{
    ...
}
class SporOtomobil : Otomobil
{
    ...
}
class HizliSporOtomobil:SporOtomobil
{
    ...
}
```

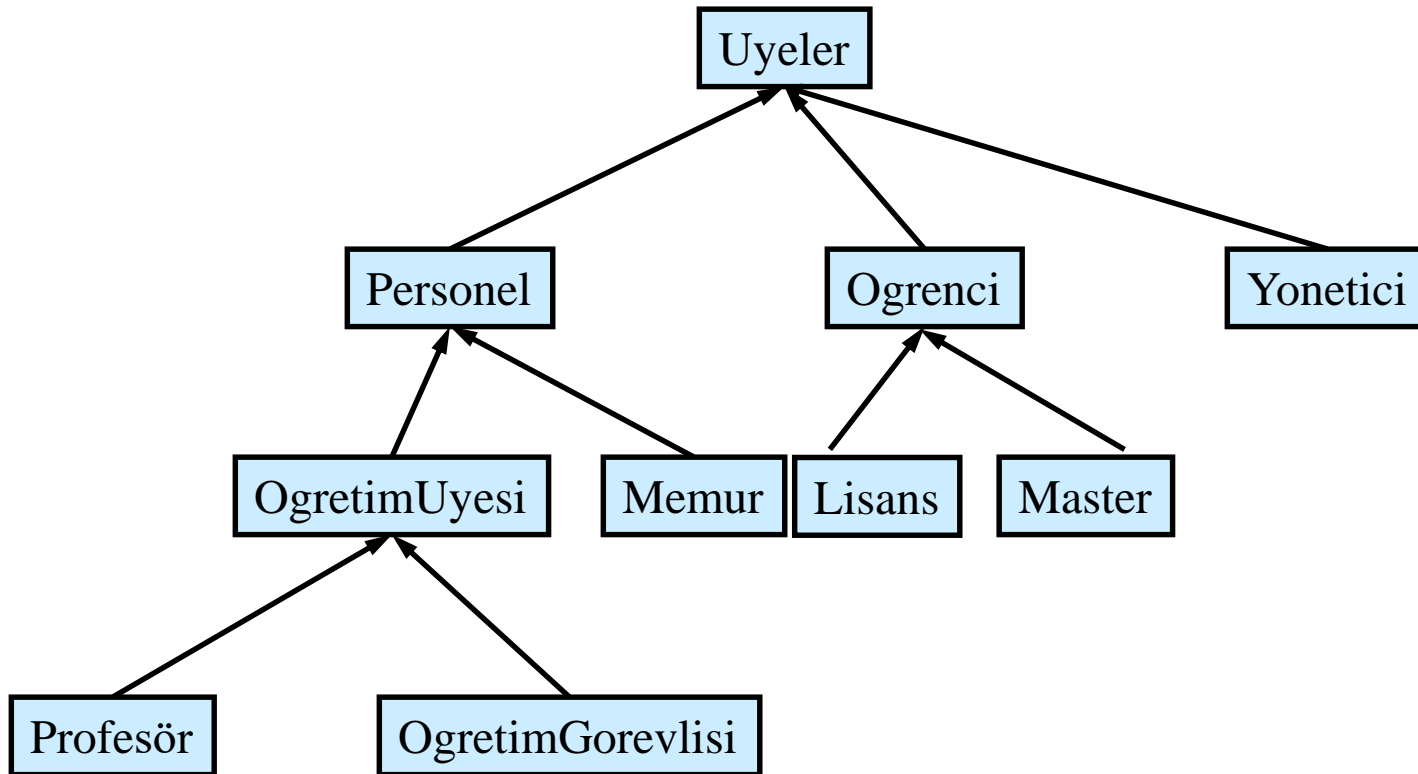


# Class Hierarchies

- Bir temel sınıfın çocuk sınıf bir başka çocuk sınıfın temel sınıfı olabilir.

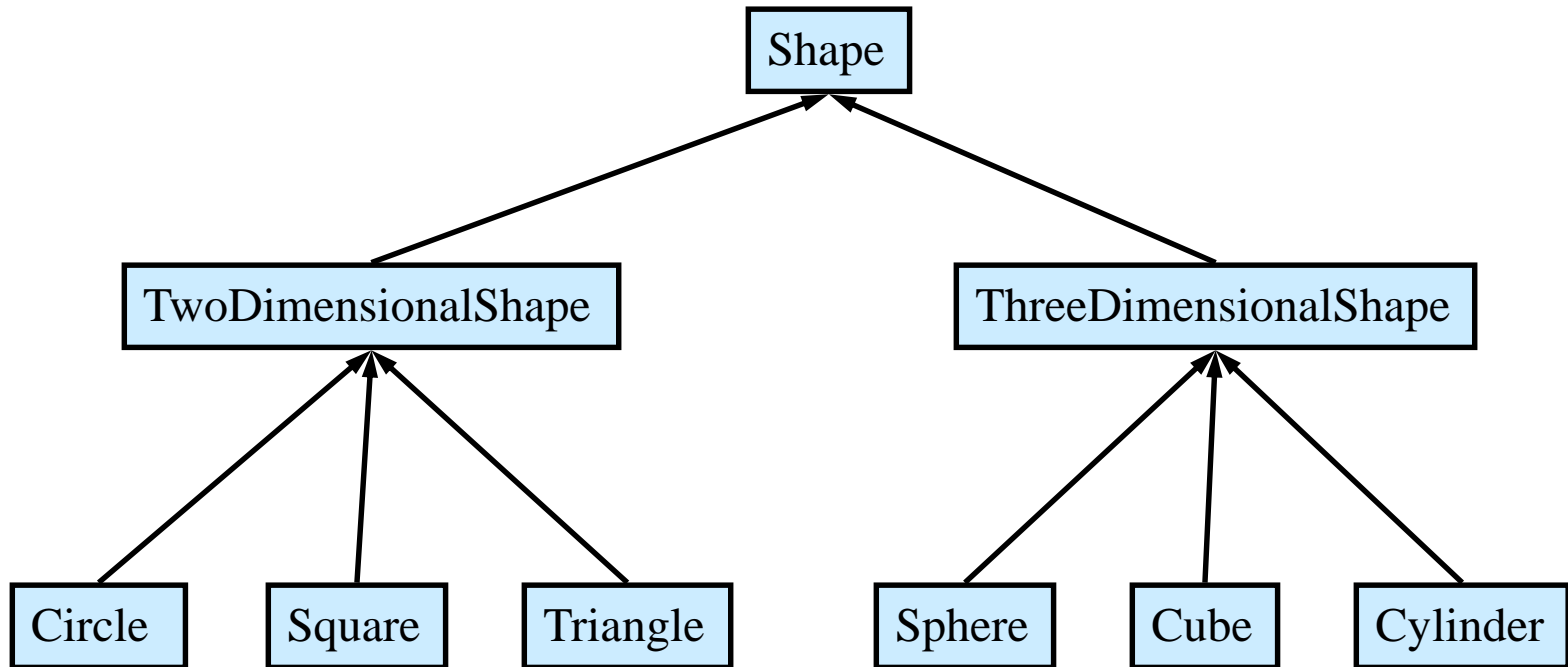


# Class Hierarchies





# Class Hierarchies



Temel sınıfın üyelerini saklama

Temel sınıfın üyeleri otomatik olarak Türetilmiş sınıfın üyeleri haline gelir, bunları silmek mümkün değildir. Fakat bu üyeler gizlenebilir.

```
class Otomobil
{
    // temel sınıf
    string alan1 ;

}
class SporOtomobil : Otomobil
{
    // Türetilmiş sınıf
    // temel sınıf üyesi aynı isim kullanılarak gizlenir
    new string alan1 ;

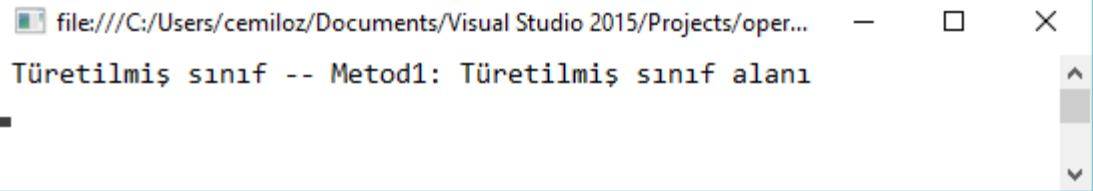
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace Kalitim
```

```
{
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;

    namespace Kalitim
    {
        class Otomobil
        {
            // temel sınıf
            public string alan1 = " temel sınıf alanı";
            public void Metod1(string deger)
            {
                Console.WriteLine(" temel sınıf -- Metod1: {0}", deger);
            }
        }
        class SporOtomobil : Otomobil
        {
            // Türetilmiş sınıf
            new public string alan1 = "Türetilmiş sınıf alanı";
            new public void Metod1(string deger)
            {
                Console.WriteLine(" Türetilmiş sınıf -- Metod1: {0}", deger);
            }
        }
        class Program
        {
            static void Main()
            {
                SporOtomobil so = new SporOtomobil();
                so.Metod1(so.alan1); // maskelenmiş üyeleri kullanma
                Console.ReadKey();
            }
        }
    }
}
```



```
file:///C:/Users/cemiloz/Documents/Visual Studio 2015/Projects/oper...
Türetilmiş sınıf -- Metod1: Türetilmiş sınıf alanı
```

## Base erişim belirteci ile gizlenmiş temel sınıfın üyelerine erişim

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Kalitim
{
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;

    namespace Kalitim
    {
        class Otomobil
        {
            // temel sınıf
            public string alan1 = " temel sınıf alanı";
            public void Metod1(string deger)
            {
                Console.WriteLine(" temel sınıf -- Metod1: {0}", deger);
            }
        }
        class SporOtomobil : Otomobil
        {
            // Türetilmiş sınıf
            new public string alan1 = "Türetilmiş sınıf alanı";
            new public void Metod1(string deger)
            {
                Console.WriteLine(" Türetilmiş sınıf -- alan1: {0}", alan1);
                Console.WriteLine(" maskelenmiş temel sınıf -- alan1: {0}", base.alan1); //base erişim
            }
        }
        class Program
        {
            static void Main()
            {
                SporOtomobil so = new SporOtomobil();
                so.Metod1(so.alan1); // maskelenmiş üyeleri kullanma
                Console.ReadKey();
            }
        }
    }
}
```

file:///C:/Users/cemiloz/Documents/Visual Studio 2015/Project... — □ ×

Türetilmiş sınıf -- alan1: Türetilmiş sınıf alanı  
maskelenmiş temel sınıf -- alan1: temel sınıf alanı

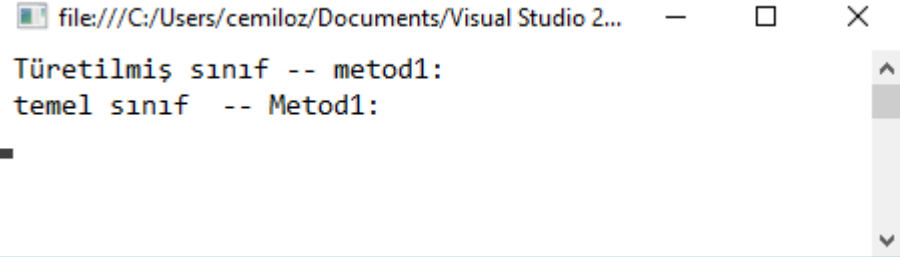
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Kalitim
{
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;

    namespace Kalitim
    {
        class Otomobil
        {
            // temel sınıf
            public void Metod1()
            {
                Console.WriteLine(" temel sınıf -- Metod1: ");
            }
        }

        class SporOtomobil : Otomobil
        {
            // Türetilmiş sınıf
            new public void Metod1()
            {
                Console.WriteLine(" Türetilmiş sınıf -- metod1: ");
            }
        }

        class Program
        {
            static void Main()
            {
                SporOtomobil so = new SporOtomobil();
                // tip dönüşümü ile temel sınıf maskelenmiş elemanına erişim
                Otomobil oto = (Otomobil)so; so.Metod1(); // maskelenmiş üyeleri kullanma, referans ile erişim
                oto.Metod1(); // temel sınıf maskelenmiş metoduna erişim
                Console.ReadKey();
            }
        }
    }
}
```



## Temel sınıfın private üyelerine erişim

```
namespace kalitim1
{
    class Geo2D
    {
        double en; // private üyeler
        double boy; // private üyeler
        public Double En
        {
            get { return en; }
            set { en = value; }
        }
        public double Boy
        {
            get { return boy ; }
            set { boy = value; }
        }

        public void Yazdir()
        {
            Console.WriteLine("En " + En + " ve Boy " + Boy);
        }
    }
    // Dikdortgen, Geo2D den türetilmiştir.
    class Dikdortgen : Geo2D
    {
        public string Ad;
        // Dikdörtgen alanı
        public double Alan()
        {
            return En * Boy ;
        }
        // nesne adını yazdır
        public void AdYazdir()
        {
            Console.WriteLine(" Nesne :" + Ad);
        }
    }
}
```

## Temel sınıfın private üyelerine erişim

```
// Triangle is derived from TwoDShape.
class Ucgen : Geo2D
{
    public string Cesit; // Üçgen çeşitleri

    public double Alan()
    {
        return En * Boy / 2;
    }
    // Üçgenin çeşit bilgisini yazdır
    public void YazdirCesit()
    {
        Console.WriteLine(" Nesne: Üçgen ");
        Console.WriteLine(" Üçgen çeşiti " + Cesit + " dır");
    }
}
```

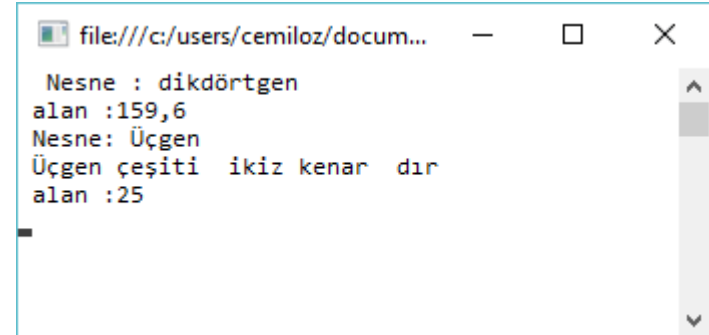
```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace kalitim1
{
    class Program
    {
        static void Main(string[] args)
        {
            Dikdortgen d1 = new Dikdortgen();
            Ucgen U1 = new Ucgen();
            d1.En = 10.5;
            d1.Boy = 15.2;
            d1.Ad = " dikdörtgen ";
            d1.AdYazdir();
            Console.WriteLine(" alan :" + d1.Alan());
            U1.En = 5;
            U1.Boy = 10;
            U1.Cesit = " ikiz kenar ";
            U1.YazdirCesit();
            Console.WriteLine(" alan :" + U1.Alan());

            Console.ReadKey();
        }
    }
}

```



```

file:///c:/users/cemiloz/docum...
Nesne : dikdörtgen
alan :159,6
Nesne: Üçgen
Üçgen çeşiti ikiz kenar dır
alan :25

```



## Virtual ve Override Methods

Türetilmiş sınıfın bir nesnesine temel sınıfın referansını kullanarak , eriştiğinizde, temel sınıfın üyelerine erişilebildiği görüldü

Virtual metotlar, türetilmiş sınıflardan temel sınıfın metotlarına erişimi sağlar.

Eğer aşağıdakiler sağlanır ise türetilmiş sınıftan temel sınıfın metotlarını, referans ile çağırma gerçekleştirilebilir.

- Türetilmiş sınıf ve temel sınıftaki metotlar aynı imzaya ve dönüş tipine sahip olmalıdır.
- Temel sınıftaki metot **virtual** etiketi ile etiketlenmelidir.
- Türetilmiş sınıftaki metot **override** etiketi ile etiketlenmelidir.

```
class Otomobil
{
    // temel sınıf
    virtual public void Metod1()
    {
        Console.WriteLine(" temel sınıf  -- Metod1: ");
    }
}

class SporOtomobil : Otomobil
{
    // Türetilmiş sınıf
    override public void Metod1()
    {
        Console.WriteLine(" Türetilmiş sınıf -- metod1: ");
    }
}
```

Buradaki kod bir önceki ile aynı ama metotlar virtual ve override olarak etiketlenmiş tir. Çıkış farklıdır. Temel sınıfın fonksiyonunu çağırma sürpriz bir şekilde türetilmiş sınıfın metodunu çağırır.

```
class Otomobil
{
    // temel sınıf
    virtual public void Metod1()
    {
        Console.WriteLine(" temel sınıf -- Metod1: ");
    }
}

class SporOtomobil : Otomobil
{
    // Türetilmiş sınıf
    override public void Metod1()
    {
        Console.WriteLine(" Türetilmiş sınıf -- metod1: ");
    }
}

class Program
{
    static void Main()
    {
        SporOtomobil so = new SporOtomobil();
        Otomobil oto = (Otomobil)so;
        so.Metod1(); //
        oto.Metod1();
        Console.ReadKey();
    }
}
```

Türetilmiş sınıf -- metod1:

Türetilmiş sınıf -- metod1:

# Soyut sınıf

Çocuk sınıfların ortak özelliklerini ve işlevlerini taşıyan bir ebeveyn sınıf oluşturmak istersek ve gerçek dünyada ebeveyn sınıfından bir nesne yoksa, temel sınıfı **“soyut sınıf”** olarak tanımlarız.

Örnek: “Memeli” sınıfından direkt bir nesne oluşturulmaz; ancak alt sınıfları tanımlanarak onlardan nesneler oluşturulur.

Soyut sınıfın yöntemlerini, çocuk sınıfları tarafından üzerine yazılmak üzere, sadece şablon olarak tanımlayıp içlerini boş bırakabiliriz veya soyut yöntem (“abstract method”) olarak tanımlayabiliriz.

❑ Çağırın sınıflar için arayüz oluşturur.

❑ çocuk sınıflar üzerine yazarak işlevlerini tanımlar.

# Soyut Sınıf ve Metodların Kullanımı

- Soyut sınıflardan nesne oluşturulamaz.
- Soyut yöntemlere sahip bir sınıfın kendisi de otomatik olarak soyuttur ve öyle tanımlanmak zorundadır.
- Bir soyut sınıfın çocuk sınıfları, ancak temel sınıfın tanımladığı soyut sınıfların üzerine yazdığı ve onlara birer işlev tanımladığı zaman örneklenebilir (“instantiation”).
- Bu durumda çocuk sınıf somut sınıf ( concrete “concrete class ”) olarak adlandırılır.
- Bir soyut sınıf da, soyut yöntemlere ek olarak, somut yöntemler de tanımlayabilir.
- Bir soyut sınıf sadece somut yöntemleri de içerebilir veya sadece somut yöntemleri de içerebilir.
- Eğer bir soyut sınıfın çocuk sınıfı, o sınıfa ait tüm soyut yöntemleri gerçekleştirmezse;  
Çocuk sınıf da soyut tanımlanmak zorundadır.
- Static, final ve private olarak tanımlı yöntemler, üzerine yazılamadıklarından, soyut olarak tanımlanamazlar.

## Soyut sınıf (abstract class)

Soyut sınıflar sadece diğer sınıfların türetildiği temel sınıf olarak kullanılabilir.

- Soyut sınıfların nesnesi( örneği) türetilemez.
- Soyut sınıflar abstract anahtar kelimesi ile tanımlanır.

```
abstrac class Otomobil
{
    ...
}
```

Soyut sınıflar soyut üyeler içerebilir, fakat gerekli değildir. Soyut sınıfın üyeleri soyut üyeler ve normal üyelerin farklı kombinasyonları olabilir.

Soyut sınıf bir başka soyut sınıftan türetilbilir.

```
abstrac class Otomobil
{
    ...
}

abstrac class SporOtomobil: Otomobil
{
    ...
}
```

# Sealed sınıflar

Sealed sınıflar soyut sınıfların aksine kendilerinden sınıf türetilemeyen sınıflardır. Yani temel sınıf olarak kullanılamazlar.

Sealed sınıf sealed anahtarı ile etiketlenir.

Bir sınıf sealed sınıftan türetilmeye çalışılır ise derleme hatası alınır.

```
sealed class Otomobil  
{  
    ...  
}
```

## Kalıtımda Kurucu Metotlar

Türetilmiş sınıf nesnesinin bir kısmı temel sınıfın nesnesidir.

- Nesnenin temel sınıf kısmındaki üyelere ilk değer atamak için türetilmiş sınıf nesnesini oluşturma işlemi sırasında temel sınıf kurucu fonksiyonu çağrılır.
- Kalıtım hiyerarşisinin zincirinde her sınıf, kendi kurucu fonksiyonunu çağırmadan önce temel sınıfının kurucusunu çağırır.



```

class SporOtomobil: Otomobil
{
    int hiz = 280;           // 1. ilk değer atanır
    int vites;               // ilk değer atanır
    public SporOtomobil()    // 3. kurucu metod ifadeleri yürütülür
    {
        ...
    }
}
class Otomobil
{
    public Otomobil()        // 2. Temel sınıf kurucusu çağrılır
    {
        ...
    }
}

```

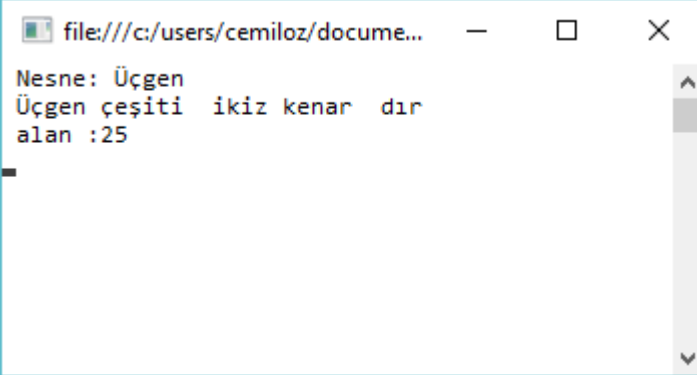
Örneğimizde önce **türetilmiş sınıfın alanlarına** ilk değer ataması gerçekleşecek, daha **sonra temel sınıfın kurucusu** yürütülecek ve son olarak **türetilmiş sınıfın kurucusu** içerisindeki ifadeler yürütülecektir.

# Temel sınıfın kurucusu yok, varsayılan kurucu çağrılır

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace kalitim1
{
    class Geo2D
    {
        double en;
        double boy;
        public Double En
        {
            get { return en; }
            set { en = value; }
        }
        public double Boy
        {
            get { return boy ; }
            set { boy = value; }
        }

        public void Yazdir()
        {
            Console.WriteLine("En " + En + " ve Boy " + Boy);
        }
    }
    // Üçgen Geo2D türetildi
    class Ucgen : Geo2D
    {
        public string Cesit; // Üçgen çeşitleri
        public Ucgen(double _en, double _boy, string _cesit)// kurucu metod
        { En = _en; Boy = _boy; Cesit = _cesit; }
        public double Alan()
        {
            return En * Boy / 2;
        }
        // Üçgenin çeşit bilgisini yazdır
        public void YazdirCesit()
        {
            Console.WriteLine(" Nesne: Üçgen ");
            Console.WriteLine(" Üçgen çeşiti " + Cesit + " dır");
        }
    }
}
```



```
file:///c:/users/cemiloz/docume...
Nesne: Üçgen
Üçgen çeşiti ikiz kenar dır
alan :25
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace kalitim1
{
    class Program
    {
        static void Main(string[] args)
        {
            Ucgen U1 = new Ucgen(5.0,10.0," ikiz kenar ");
            U1.YazdirCesit();
            Console.WriteLine(" alan :" + U1.Alan());

            Console.ReadKey();
        }
    }
}
```

# Base kurucu ile temel sınıfın kurucusunu çağırma

```
namespace kalitim1
{
    class Geo2D
    {
        double en;
        double boy;
        public Geo2D(double _en, double _boy)
        {
            en = _en;
            boy = _boy;
        }
        public Double En
        {
            get { return en; }
            set { en = value; }
        }
        public double Boy
        {
            get { return boy ; }
            set { boy = value; }
        }
    }

    public void Yazdir()
    {
        Console.WriteLine("En " + En + " ve Boy " + Boy);
    }
}

// Triangle is derived from TwoDShape.
class Ucgen : Geo2D
{
    public string Cesit; // Üçgen çeşitleri
    public Ucgen(double _en, double _boy, string _cesit):base(_en,_boy)
    {
        Cesit = _cesit;
    }
    public double Alan()
    {
        return En * Boy / 2;
    }
    // Üçgenin çeşit bilgisini yazdır
    public void YazdirCesit()
    {
        Console.WriteLine(" Nesne: Üçgen ");
        Console.WriteLine(" Üçgen çeşiti " + Cesit +" dır");
    }
}
}
```

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```
namespace kalitim1
```

```
{
```

```
    class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            Ucgen U1 = new Ucgen(5.0,10.0," ikiz kenar ");
```

```
            U1.YazdirCesit();
```

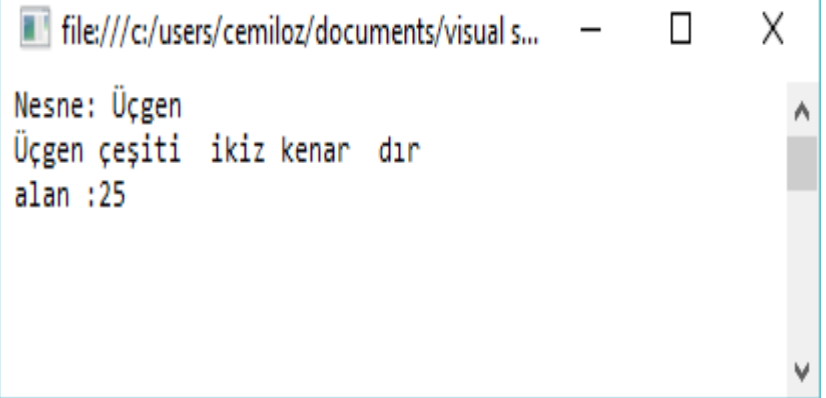
```
            Console.WriteLine(" alan :" + U1.Alan());
```

```
            Console.ReadKey();
```

```
        }
```

```
    }
```

```
}
```



file:///c:/users/cemiloz/documents/visual s... - □ X

Nesne: Üçgen  
Üçgen çeşiti ikiz kenar dır  
alan :25

## Temel sınıfın birden fazla kurucusu olması durumunda

```
namespace kalitim1
{
    class Geo2D
    {
        double en;
        double boy;
        public Geo2D()
        { en = 0.0; boy = 0.0; }
        public Geo2D(double x)
        {
            en = x;
            boy = x;
        }

        public Geo2D(double _en, double _boy)
        {
            En = _en;
            boy = _boy;
        }
        public Double En
        {
            get { return en; }
            set {
                if(value<0)
                    en = 0;
                else
                    en=value;
            }
        }
        public double Boy
        {
            get { return boy ; }
            set { boy = value; }
        }
        public void Yazdir()
        {
            Console.WriteLine("En " + En + " ve Boy " + Boy);
        }
    }
}
```

```

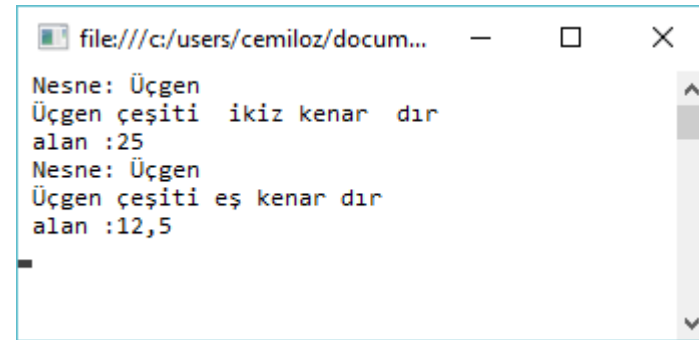
// Triangle is derived from TwoDShape.
class Ucgen : Geo2D
{
    public string Cesit; // Üçgen çeşitleri
    public Ucgen(double x, string _cesit) : base(x)
    {
        Cesit = _cesit;
    }
    public Ucgen(double _en, double _boy, string _cesit):base(_en,_boy)
    {
        Cesit = _cesit;
    }
    public double Alan()
    {
        return En * Boy / 2;
    }
    // Üçgenin çeşit bilgisini yazdır
    public void YazdirCesit()
    {
        Console.WriteLine(" Nesne: Üçgen ");
        Console.WriteLine(" Üçgen çeşiti " + Cesit + " dır");
    }
}
}

```

```

namespace kalitim1
{
    class Program
    {
        static void Main(string[] args)
        {
            Ucgen U1 = new Ucgen(5.0,10.0," ikiz kenar ");
            Ucgen U2 = new Ucgen(5, "eş kenar");
            U1.YazdirCesit();
            Console.WriteLine(" alan :" + U1.Alan());
            U2.YazdirCesit();
            Console.WriteLine(" alan :" + U2.Alan());
            Console.ReadKey();
        }
    }
}

```



```

file:///c:/users/cemiloz/docum...
Nesne: Üçgen
Üçgen çeşiti ikiz kenar dır
alan :25
Nesne: Üçgen
Üçgen çeşiti eş kenar dır
alan :12,5

```



## Çok biçimlilik ( [polymorphism](#) )

- Aynı temel sınıftan türetilmiş olan sınıflarda paylaşılan aynı metodun bu sınıflarda farklı şekillerde uyarlanabilmesidir.
- Nesnenin davranışı çalışma anında belirlendiği için programcılar çok biçimlilik özelliği sayesinde nesnelerin türünü önceden bilmek zorunda kalmazlar

şu soruya cevap verir: **Türeyen sınıflar, temel sınıfta yer alan bir üyeyi nasıl farklı şekillerde uygulayacaklardır?**

- Çok biçimlilik, temel sınıfta yer alan bir üyenin (metot, özellik, indeksleyici ya da olay), türeyen sınıf tarafından nasıl değiştirileceğine dair bir yol sunmaktadır.
- Bir temel sınıf, tanımlayacağı üyenin uygulanışının -yani içerisindeki kodların-, kendisinden türeyen sınıflar tarafından değiştirilebilmesini istiyorsa bu üye virtual anahtar kelimesi ile işaretlenmelidir.
- Türeyen bir sınıf, virtual anahtar kelimesi ile işaretlenmiş bir üyenin uygulanışını kendi sınıfına ait bir iş mantığıyla değiştirmek isteyebilir; ancak zorunda değildir. Üyenin başına override anahtar kelimesi yazılarak yeniden kodlanması ile böyle bir değişiklik mümkün olmaktadır.
- Ezilen (overriden) her üye, ihtiyaç duyulması halinde temel sınıfta yer alan uygulanışı yeniden çağırmakta serbesttir: Kod içerisinde istenen herhangi bir yerde temel sınıfın bir üyesi base anahtar kelimesi ile çağrılabilir.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
public class Çizici
{
    public virtual void Çiz() //temel sınıf metodu
    {
        Console.WriteLine("\t çizici");
    }
}
public class DoğruÇiz : Çizici
{
    public override void Çiz() // türetilen sınıf Çiz metodu
    {
        Console.WriteLine("\t Line");
    }
}
public class DaireÇiz : Çizici
{
    public override void Çiz() // türetilen sınıf Çiz metodu
    {
        Console.WriteLine("\t Circle");
    }
}
public class Program
{
    static void Main(string[] args)
    {
        Çizici[] birÇizici = new Çizici[3];
        birÇizici[0] = new DoğruÇiz(); // türetilen sınıf nesnesi
        birÇizici[1] = new DaireÇiz(); // türetilen sınıf nesnesi
        birÇizici[2] = new Çizici(); // temel sınıf nesnesi
        foreach (Çizici sayac in birÇizici)
        {
            sayac.Çiz(); // çok biçimli çağırısı
        }
        Console.ReadLine();
    }
}
```



file:///c:/users/cemiloz/documents/visual studio 2015

Line  
Circle  
çizici

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

public abstract class Vehicle
{
    public virtual int Wheels()
    {
        return 0;
    }
}

public class Bicycle : Vehicle
{
    public override int Wheels()
    {
        return 2;
    }
}

public class Car : Vehicle
{
    public override int Wheels()
    {
        return 4;
    }
}

public class Truck : Vehicle
{
    public override int Wheels()
    {
        return 18;
    }
}

public class program
{
    static void Main(string[] args)
    {
        Vehicle[] a = new Vehicle[3];
        a[0] = new Bicycle();
        a[1] = new Car();
        a[2] = new Truck();
        foreach (Vehicle sayac in a)
        {
            Console.WriteLine(" {0}", sayac.Wheels()); // çok biçimli çağırısı
        }
        Console.ReadLine();
    }
}
```

file:///c:/users/cemiloz/documents/visual

2  
4  
18

## Bir önceki program collectiolar ile

```
static void Main(string[] args)
{
    List<Vehicle> vehicles = new List<Vehicle>();

    vehicles.Add(new Bicycle());
    vehicles.Add(new Car());
    vehicles.Add(new Truck());

    foreach (Vehicle v in vehicles)
    {
        Console.WriteLine(
            string.Format("A {0} has {1} wheels.",
                v.GetType().Name, v.Wheels()));
    }

    Console.ReadLine();
}
```

 file:///c:/users/cemiloz/documents/visual studio 2015/Projects/cokbicin

```
A Bicycle has 2 wheels.
A Car has 4 wheels.
A Truck has 18 wheels.
```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    public class clsShape
    {
        public int _radius = 5;

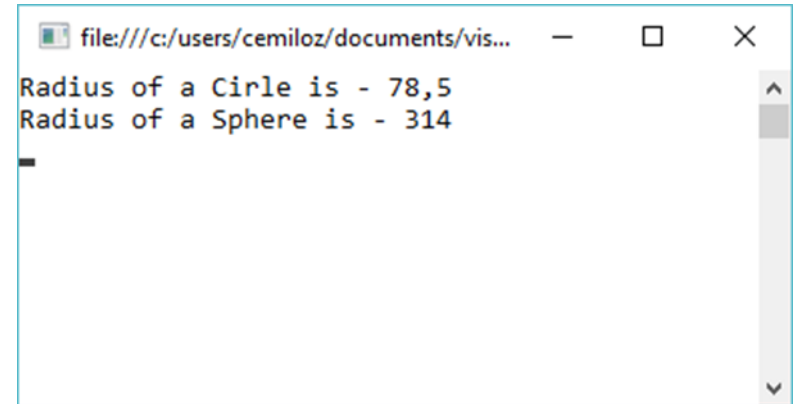
        public virtual double getArea()
        {
            return 0;
        }
    }

    public class clsCircle : clsShape
    {
        public override double getArea()
        {
            return 3.14 * _radius * _radius;
        }
    }

    public class clsSphere : clsShape
    {
        public override double getArea()
        {
            return 4 * 3.14 * _radius * _radius;
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            clsShape objShape1 = new clsCircle();
            clsShape objShape2 = new clsSphere();
            Console.WriteLine("Radius of a Circle is - {0}", objShape1.getArea());
            Console.WriteLine("Radius of a Sphere is - {0}", objShape2.getArea());
            Console.ReadKey();
        }
    }
}

```



```

        public override void Draw()
        {
            // Code to draw a rectangle...
            Console.WriteLine("Drawing a rectangle");
            base.Draw();
        }
    }

    class Triangle : Shape
    {
        public override void Draw()
        {
            // Code to draw a triangle...
            Console.WriteLine("Drawing a triangle");
            base.Draw();
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            // Polymorphism at work #1: a Rectangle,
            Triangle and Circle
            // can all be used wherever a Shape is
            expected. No cast is

```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

class Arac
{
    public void Calistir(string ses)
    {
        Console.WriteLine("Araç çalıştırılıyor: {0}", ses);
    }

    public void MotoruKapat(string ses)
    {
        Console.WriteLine("Araç motoru kapatılıyor: {0}", ses);
    }

    public virtual void Sur()
    {
        Console.WriteLine("Araç Sürülüyor");
    }
}

class Araba : Arac
{
    public void Hizlan()
    {
        Console.WriteLine("Araba Hızlanıyor");
    }

    public void FrenYap()
    {
        Console.WriteLine("Araba Fren Yapıyor");
    }

    public override void Sur()
    {
        Console.WriteLine("Araba Sürülüyor");
    }
}
```

```

class Ucak : Arac
{
    public void Kalk()
    {
        Console.WriteLine("Uçak kalkıyor");
    }

    public void Kon()
    {
        Console.WriteLine("Uçak yere iniliyor");
    }

    public override void Sur()
    {
        Console.WriteLine("Uçuluyor");
    }
}

```

```

class Program
{
    static void Entrance()
    {
        Console.WriteLine("uçak yolculuğu başlıyor:");
        Ucak ucagim = new Ucak();
        ucagim.Calistir("kaptan pilot konuşuyor...");
        ucagim.Kalk();
        ucagim.Sur();
        ucagim.Kon();
        ucagim.MotoruKapat("viiuuuvvv");

        Console.WriteLine("\nAraba yolculuğu başlıyor:");
        Araba arabam = new Araba();
        arabam.Calistir("Brm brm");
        arabam.Hizlan();
        arabam.Sur();
        arabam.FrenYap();
        arabam.MotoruKapat("ptptpttt");

        Console.WriteLine("\nÇok biçimlilik testi");
        Arac v = arabam;
        v.Sur();
        v = ucagim;
        v.Sur();
    }

    static void Main()
    {
        try
        {
            Entrance();
        }
        catch (Exception ex)
        {
            Console.WriteLine("Exception: {0}", ex.Message);
        }
        Console.ReadLine();
    }
}

```

file:///c:/users/cemiloz/documents/visual studio 2015/Projects/c

```

uçak yolculuğu başlıyor:
Araç çalıştırılıyor: kaptan pilot konuşuyor...
Uçak kalkıyor
Uçuluyor
Uçak yere iniliyor
Araç motoru kapatılıyor: viiiuuuvvv

Araba yolculuğu başlıyor:
Araç çalıştırılıyor: Brm brm
Araba Hızlanıyor
Araba Sürülüyor
Araba Fren Yapıyor
Araç motoru kapatılıyor: ptptpttt

Çok biçimlilik testi
Araba Sürülüyor
Uçuluyor

```



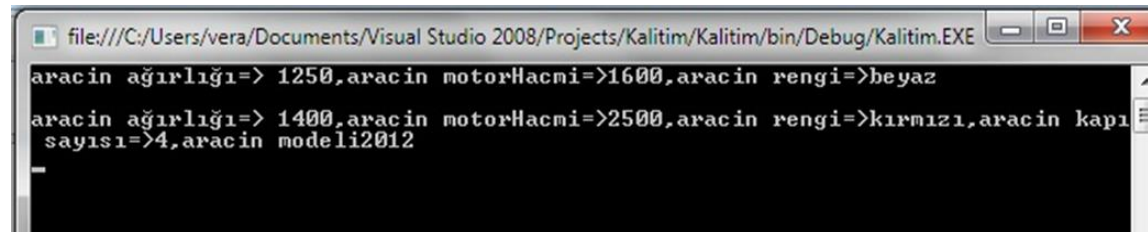
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Kalitim
{
    class otomobil
    {
        public int agirlik; //her yerden erisilebilir erisim seviyesi.
        public int motorHacmi; //her yerden erisilebilir erisim seviyesi.
        public string renk; //her yerden erisilebilir erisim seviyesi.

        public virtual void aracOzellikleri()//virtual anahtar kelimesi
        {
            Console.WriteLine("aracin ağırlığı=> {0},aracin motorHacmi=>{1},aracin rengi=>{2} ", agirlik, motorHacmi, renk);
        }
    }
    class spor : otomobil
    {
        public int kapiSayisi;
        public string model;
        public override void aracOzellikleri()//override anahtar kelimesi
        {
            //base.aracOzellikleri();burayı aktif yaparsak otomobil sinifina ait aracOzellikleri metodu çalışacaktı.
            Console.WriteLine("aracin ağırlığı=> {0},aracin motorHacmi=>{1},aracin rengi=>{2},aracin kapi sayısı=>{3},aracin modeli{4} ",
agirlik, motorHacmi, renk,kapiSayisi ,model );
        }
    }
}
```

class Program

```
{
    static void Main(string[] args)
    {
        otomobil otomobil1 = new otomobil();
        otomobil1.agirlik = 1250;
        otomobil1.renk = "beyaz";
        otomobil1.motorHacmi = 1600;
        otomobil1.aracOzellikleri();
        Console.ReadLine();
        //////////////////////////////////////
        spor spor1 = new spor();
        spor1.agirlik = 1400;
        spor1.renk = "kırmızı";
        spor1.motorHacmi = 2500;
        spor1.kapiSayisi = 4;
        spor1.model = "2012";
        spor1.aracOzellikleri();
        Console.ReadLine();
    }
}
```



```
file:///C:/Users/vera/Documents/Visual Studio 2008/Projects/Kalitim/Kalitim/bin/Debug/Kalitim.EXE
aracin ağırlığı=> 1250,aracin motorHacmi=>1600,aracin rengi=>beyaz
aracin ağırlığı=> 1400,aracin motorHacmi=>2500,aracin rengi=>kırmızı,aracin kapi
sayisi=>4,aracin modeli2012
```

```
using System;
```

```
namespace KalitimBlog
```

```
{
```

```
    class Ev
```

```
    {
```

```
        public string semt;
```

```
        public int alan;
```

```
        public string adres;
```

```
        public void evGoster()
```

```
        {
```

```
            Console.WriteLine("Evin özellikleri ; nSempti: {0}nAlanı: {1}nAdresi:{2}", semt, alan, adres);
```

```
        }
```

```
    }
```

```
//SatilikEv sınıfının base sınıfını Ev olarak tanımlıyoruz
```

```
class SatilikEv : Ev
```

```
{
```

```
    public int fiat;
```

```
    public void fiatGoster()
```

```
    {
```

```
        Console.WriteLine("Fiatı : {0}", fiat);
```

```
    }
```

```
}
```

```
class EvOtomasyon
{
    static void Main(string[] args)
    {
        SatilikEv ev = new SatilikEv();
        ev.adres = "Mehmet Cenk Sokağı No:23";
        ev.alan = 250;
        ev.semt = "Beyoğlu";
        ev.fiat = 1000;
        ev.evGoster();
        ev.fiatGoster();

        Console.WriteLine("n2.Ev Yazdırılıyor..n");

        SatilikEv ev2 = new SatilikEv();
        ev2.adres = "Cennet Mahallesi Bilmem Ne Sokağı No:23";
        ev2.alan = 157;
        ev2.semt = "Beykızı";
        ev2.fiat = 2680;
        ev2.evGoster();
        ev2.fiatGoster();

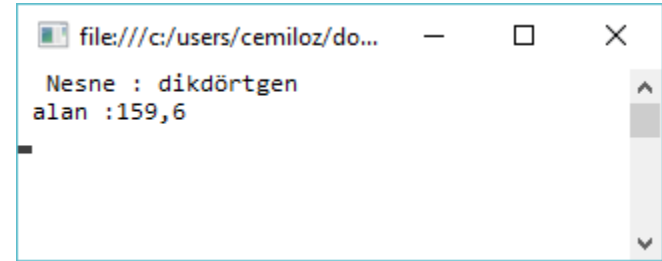
        Console.Read();
    }
}
```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace kalitim1
{
    class Geo2D
    {
        public double En;
        public double Boy;
        public void Yazdir()
        {
            Console.WriteLine("En " + En + " ve Boy " + Boy);
        }
    }
    // Dikdortgen, Geo2D den türetilmiştir.
    class Dikdortgen : Geo2D
    {
        public string Ad;
        // Dikdörtgen alanı
        public double Alan()
        {
            return En * Boy ;
        }
        // nesne adını yazdır
        public void AdYazdir()
        {
            Console.WriteLine(" Nesne :" + Ad);
        }
    }
}

```



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace kalitim1
{
    class Program
    {
        static void Main(string[] args)
        {
            Dikdortgen d1 = new Dikdortgen();
            Dikdortgen d2 = new Dikdortgen();
            d1.En = 10.5;
            d1.Boy = 15.2;
            d1.Ad = " dikdörtgen ";
            d1.AdYazdir();
            Console.WriteLine(" alan :" + d1.Alan());
            d2.En = 10.5;
            d2.Boy = 10.5;
            d2.Ad = " Kare ";
            d2.AdYazdir();
            Console.WriteLine(" alan :" + d2.Alan());

            Console.ReadKey();
        }
    }
}
```

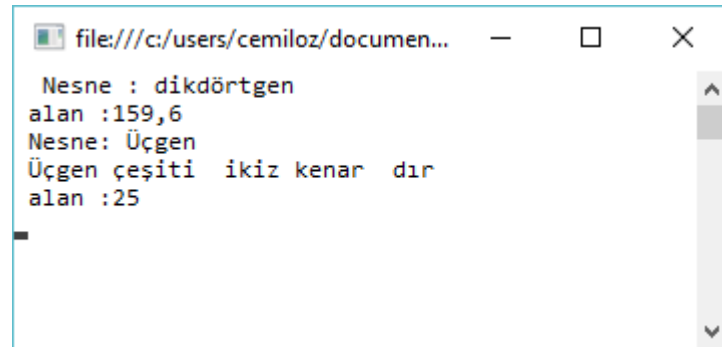
```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace kalitim1
{
    class Geo2D
    {
        public double En;
        public double Boy;
        public void Yazdir()
        {
            Console.WriteLine("En " + En + " ve Boy " + Boy);
        }
    }
    // Dikdortgen, Geo2D den türetilmiştir.
    class Dikdortgen : Geo2D
    {
        public string Ad;
        // Dikdörtgen alanı
        public double Alan()
        {
            return En * Boy ;
        }
        // nesne adını yazdır
        public void AdYazdir()
        {
            Console.WriteLine(" Nesne :" + Ad);
        }
    }
    // Triangle is derived from TwoDShape.
    class Ucgen : Geo2D
    {
        public string Cesit; // Üçgen çeşitleri

        public double Alan()
        {
            return En * Boy / 2;
        }
        // Üçgenin çeşit bilgisini yazdır
        public void YazdirCesit()
        {
            Console.WriteLine(" Nesne: Üçgen ");
            Console.WriteLine(" Üçgen çeşiti " + Cesit + " dır");
        }
    }
}

```



```

file:///c:/users/cemiloz/document...
Nesne : dikdörtgen
alan :159,6
Nesne: Üçgen
Üçgen çeşiti ikiz kenar dır
alan :25

```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace kalitim1
{
    class Program
    {
        static void Main(string[] args)
        {
            Dikdortgen d1 = new Dikdortgen();
            Ucgen U1 = new Ucgen();
            d1.En = 10.5;
            d1.Boy = 15.2;
            d1.Ad = " dikdörtgen ";
            d1.AdYazdir();
            Console.WriteLine(" alan :" + d1.Alan());
            U1.En = 5;
            U1.Boy = 10;
            U1.Cesit = " ikiz kenar ";
            U1.YazdirCesit();
            Console.WriteLine(" alan :" + U1.Alan());

            Console.ReadKey();
        }
    }
}
```



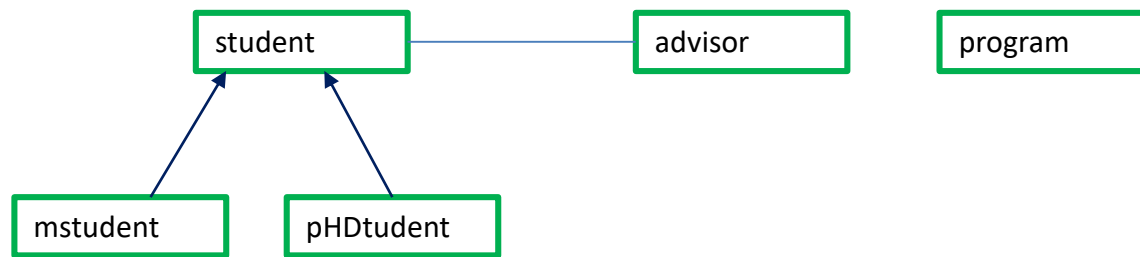
```
using System;
namespace PolymorphismApplication
{
    class Shape
    {
        protected int width, height;
        public Shape( int a=0, int b=0)
        {
            width = a;
            height = b;
        }
        public virtual int area()
        {
            Console.WriteLine("Parent class area :");
            return 0;
        }
    }
    class Rectangle: Shape
    {
        public Rectangle( int a=0, int b=0): base(a, b)
        {

        }
        public override int area ()
        {
            Console.WriteLine("Rectangle class area :");
            return (width * height);
        }
    }
}
```

```
class Triangle: Shape
{
    public Triangle(int a = 0, int b = 0): base(a, b)
    {

    }
    public override int area()
    {
        Console.WriteLine("Triangle class area :");
        return (width * height / 2);
    }
}
class Caller
{
    public void CallArea(Shape sh)
    {
        int a;
        a = sh.area();
        Console.WriteLine("Area: {0}", a);
    }
}
class Tester
{

    static void Main(string[] args)
    {
        Caller c = new Caller();
        Rectangle r = new Rectangle(10, 7);
        Triangle t = new Triangle(10, 5);
        c.CallArea(r);
        c.CallArea(t);
        Console.ReadKey();
    }
}
```



```
Microsoft Visual Studio Debug Console

Student information
Name      :cemil
Lastname   :Öz
Thesis Title :insect bla bla...
advisor information
Title      :Prof. Dr.
Name       :ali
Lastname   :bal
-----
Student information
Name      :cemil
Lastname   :Öz
number of tik : 2
Thesis Title :insect bla bla...
advisor information
Title      :Prof. Dr.
Name       :ali
Lastname   :bal
done!

C:\Users\Cemil OZ\source\repos\kalıtım\bin\Debug\netcoreapp3.1\kalıtım.exe (process 4528) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the co
nsole when debugging stops.
Press any key to close this window . . .
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace kalitim
{
    class advisor
    {
        string name;
        string lastName;
        string title;
        public advisor()
        {
            name="";
            lastName="";
            title="";
        }
        public advisor(string n, string ln,string t)
        {
            name =n;
            lastName = ln;
            title = t;
        }
        public void yazdir()
        {
            Console.WriteLine(" Title           : " + title);
            Console.WriteLine(" Name           : " + name);
            Console.WriteLine(" Lastname       : " + lastName);
        }
    }
}
```

```

using System;
using System.Collections.Generic;
using System.Text;

namespace kalitim
{
    class student
    {
        string name;
        private string lastName;
        public student()
        {
            Name="";
            LastName="";
        }
        public student(string n,string ln )
        {
            Name = n;
            LastName = ln;
        }
        public string LastName
        { get => lastName; set => lastName = value; }
        public string Name
        { get => name; set => name = value; }
        public void yazdir()
        {
            Console.WriteLine(" Name           :" + Name);
            Console.WriteLine(" Lastname       :" + LastName);
        }
    }
}

```

```

class mstudent:student
{
    string thesisTitle;
    advisor a1 = new advisor();
    public mstudent():base()
    { thesisTitle = ""; }
    public mstudent(advisor a,string n, string ln, string tt) : base(n,ln)
    {
        a1 = a;
        thesisTitle = tt;
    }
    public void yazdir()
    {
        Console.BackgroundColor = ConsoleColor.Red;
        Console.WriteLine("  Student information ");
        Console.BackgroundColor = ConsoleColor.White;
        base.yazdir();
        Console.ForegroundColor = ConsoleColor.Blue;
        Console.WriteLine(" Thesis Title      :" + thesisTitle);
        Console.ForegroundColor = ConsoleColor.Black;
        Console.BackgroundColor = ConsoleColor.Red;
        Console.WriteLine("advisor information");
        Console.BackgroundColor = ConsoleColor.White;
        a1.yazdir();
    }
}

```

```

class pHDtudent : student
{
    string thesisTitle;
    advisor a1 = new advisor();
    int nofTik;
    public pHDtudent() : base()
    { thesisTitle = ""; nofTik = 0; }
    public pHDtudent(advisor a, string n, string ln, string tt, int nt) : base(n,
ln)
    {
        a1 = a;
        thesisTitle = tt;
        nofTik = nt;
    }
    public void yazdir()
    {
        Console.BackgroundColor = ConsoleColor.Red;
        Console.WriteLine(" Student information ");
        Console.BackgroundColor = ConsoleColor.White;
        base.yazdir();
        Console.WriteLine(" number of tik      : " + nofTik);
        Console.ForegroundColor = ConsoleColor.Blue;
        Console.WriteLine(" Thesis Title      : " + thesisTitle);
        Console.ForegroundColor = ConsoleColor.Black;
        Console.BackgroundColor = ConsoleColor.Red;
        Console.WriteLine("advisor information");
        Console.BackgroundColor = ConsoleColor.White;
        a1.yazdir();
    }
}
}

```

```
namespace kalitim
{
    class Program
    {
        static void Main(string[] args)
        {
            var ad = new advisor("ali", "bal", "Prof. Dr.");
            var ms1 = new mstudent(ad,"cemil","Öz","insect bla bla...");
            var phd1 = new PhDtudent(ad, "cemil", "Öz", "insect bla bla...", 2);
            ms1.yazdir();
            Console.WriteLine("-----");
            phd1.yazdir();
            Console.WriteLine("done!");
        }
    }
}
```



```

class student
{
    string name;
    private string lastName;
    public student()
    {
        Name="";
        LastName="";
    }
    public student(string n,string ln )
    {
        Name = n;
        LastName = ln;
    }
    public string LastName
    { get => lastName; set => lastName = value; }
    public string Name
    { get => name; set => name = value; }
    public virtual void yazdir()
    {
        Console.WriteLine(" Name           : " + Name);
        Console.WriteLine(" Lastname        : " + LastName);
    }
}

```

```

class mstudent:student
{
    string thesisTitle;
    advisor a1 = new advisor();
    public mstudent():base()
    { thesisTitle = ""; }
    public mstudent(advisor a,string n, string ln, string tt) : base(n,ln)
    {
        a1 = a;
        thesisTitle = tt;
    }
    public override void yazdir()
    {
        Console.BackgroundColor = ConsoleColor.Red;
        Console.WriteLine(" Student information ");
        Console.BackgroundColor = ConsoleColor.White;
        base.yazdir();
        Console.ForegroundColor = ConsoleColor.Blue;
        Console.WriteLine(" Thesis Title      : " + thesisTitle);
        Console.ForegroundColor = ConsoleColor.Black;
        Console.BackgroundColor = ConsoleColor.Red;
        Console.WriteLine("advisor information");
        Console.BackgroundColor = ConsoleColor.White;
        a1.yazdir();
    }
}

```

```

class pHDtudent : student
{
    string thesisTitle;
    advisor a1 = new advisor();
    int nofTik;
    public pHDtudent() : base()
    { thesisTitle = ""; nofTik = 0; }
    public pHDtudent(advisor a, string n, string ln, string tt, int nt) : base(n, ln)
    {
        a1 = a;
        thesisTitle = tt;
        nofTik = nt;
    }
    public override void yazdir()
    {
        Console.BackgroundColor = ConsoleColor.Red;
        Console.WriteLine(" Student information ");
        Console.BackgroundColor = ConsoleColor.White;
        base.yazdir();
        Console.WriteLine(" number of tik      : " + nofTik);
        Console.ForegroundColor = ConsoleColor.Blue;
        Console.WriteLine(" Thesis Title      : " + thesisTitle);
        Console.ForegroundColor = ConsoleColor.Black;
        Console.BackgroundColor = ConsoleColor.Red;
        Console.WriteLine("advisor information");
        Console.BackgroundColor = ConsoleColor.White;
        a1.yazdir();
    }
}
}

```

```

class Program
{
    static void Main(string[] args)
    {
        var ad = new advisor("Ali", "Bal", "Prof. Dr.");
        student[] s = new student[3];
        s[0] = new mstudent(ad, "cemil", "Öz", "insect bla bla...");
        ad = new advisor("Fatma", "Kuralcı", "Dr. Öğr. Üyesi");
        s[1] = new PhDtudent(ad, "Hülya", "Tarafsız", "Computer vision bla bla...", 2);
        ad=new advisor("Nejla", "Herşeyibilen", "Doç. Dr.");
        s[2]= new mstudent(ad, "Cemil", "Öz", "artificial intellegent bla bla...");
        foreach (var st in s)
        {
            st.yazdir();
            Console.WriteLine("-----");
            Console.WriteLine("Press any key to continue");
            Console.ReadKey();
        }
        Console.WriteLine("done!");
    }
}

```

```

ad=new advisor("Nejla", "Herşeyibilen", "Doç. Dr.");
s[2]= new mstudent(ad, "Cemil", "Öz", "artificial intellegent bla bla...");

```

Yerine Aşağıdaki şekilde de tanımlanabilir.

```

s[2]= new mstudent(new advisor("Nejla", "Herşeyibilen", "Doç. Dr."), "Cemil", "Öz", "artificial
intellegent bla bla...");

```