# Nesne Tabanlı Programlama
# Bölüm 5
# Kalıtım (inheritance)

Dr. Öğr. Üyesi Murat TAŞYÜREK (kayubmprogramlama1@gmail.com)

30 Ekim 2023

Kayseri Üniversitesi, Bilgisayar Mühendisliği Bölümü

# Kalıtım ve Kompozisyon (Composition)

- Yazılım geliştirirken önceden yazılmış sınıflar **kalıtım (inheritance) ve kompozisyon (composition)** olmarak üzere iki yöntem ile kullanılır.
- Önceden yazılmış sınıfların yeni yazılacak sınıflar içerisinde direk kullanılması **kompozisyon (composition)** olarak adlandırılır.
- Kompozisyon, bir sınıfın diğerini içermesine izin veren sınıflar arasındaki bir ilişki türüdür.
- Kompozisyona genellikle Has-A (B sınıfı A sınıfını içeririyor manasına gelen) ilişkisi denir. B sınıfın araba A sınıfında motor olduğunu düşünürkesek, bir arabanın motoru vardır.
- Tıpkı kalıtım gibi, kompozisyon da kodun yeniden kullanılmasına izin verir.

# Kompozisyon (Composition)

- Sınıflar arasında bir ilişki oluşturmak (kompozisyon ilişkisi) için, kullanmak istediğimiz sınıfın içinde yeni bir sınıf nesnesi başlatırız.

```
2 references
class ClassA
{
    // class body
}

0 references
class ClassB
{
    ClassA nesne = new ClassA();

    //nesne.... ( ClassA sınıfdan metotların çağrılması);
}
```

# Örnek 1

- Bir arabamızın olduğunu düşünelim.
- Arabamızın motoru vardır ancak başka arabanın da aynı motoru olabilir.
- Nesne tababanlı programlama mantığında düşündüğümüzde az kod yazmak ve yazılan kodu tekrar kullanmak için Araba ve Motor için ayrı ayrı sınıf oluştururuz.
- Araba çalıştığına aslında önce motor çalışmış olur.
- Arabayı durdurduğumuzda önce motor durmuş olur.
- Araba ve motor arasındaki ilişkiyi içeren kodlamayı yapalım.

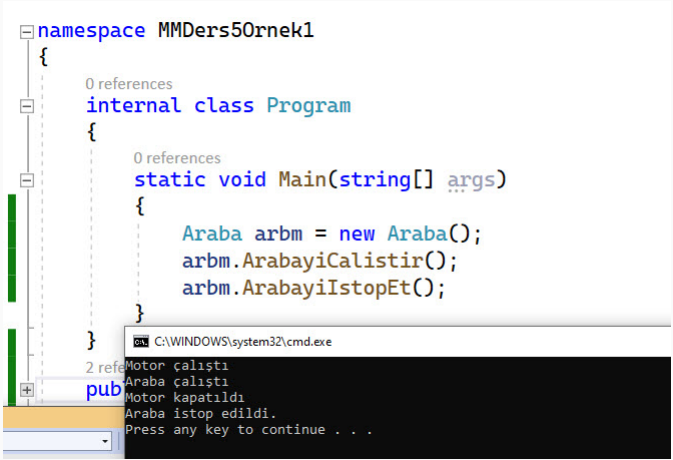
# Araba ve Motor Class

```
public class Motor
{
    1 reference
    public void MotoruCalistir()
    {
        Console.WriteLine("Motor çalıştı");
    }
    1 reference
    public void MotoruKapat()
    {
        Console.WriteLine("Motor kapatıldı");
    }
}
0 references
public class Araba
{
    Motor arabaninMotoru = new Motor();//Compoustion, Araba has Motor
    0 references
    public void ArabayiCalistir()
    {
        arabaninMotoru.MotoruCalistir();
        Console.WriteLine("Araba çalıştı");
    }
    0 references
    public void ArabayiIstopEt()
    {
        arabaninMotoru.MotoruKapat();
        Console.WriteLine("Araba istop edildi.");
    }
}
```

# Kaynak Kodu

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace MMDers5Ornek1
8   {
        0 references
9       internal class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              Araba arbm = new Araba();
14              arbm.ArabayiCalistir();
15              arbm.ArabayiIstopEt();
16          }
17      }
            2 references
18      public class Motor...
            2 references
29      public class Araba...
43  }
```

# Çıktı


namespace MMDers5Ornek1
{
0 references
internal class Program
{
0 references
static void Main(string[] args)
{
Araba arbm = new Araba();
arbm.ArabayiCalistir();
arbm.ArabayiIstopEt();
}
}
C:\WINDOWS\system32\cmd.exe
Motor çalıştı
Araba çalıştı
Motor kapatıldı
Araba istop edildi.
Press any key to continue . . .

# Kalıtım (Inheritance)

- Nesne Yönelimli Programlama dillerinde **kalıtım (inheritance)**, bir sınıfta (class) tanımlanmış değişkenlerin ve/veya metotların (fonksiyon, procedure) yeniden tanımlanmasına gerek olmaksızın yeni bir sınıfa taşınabilmesidir. Bunun için yapılan iş, bir sınıftan bir alt-sınıf (subclass) türetmektir.
- Miras yoluyla (kalıtım), bir kez oluşturulmuş olan sınıfın tekrar tekrar kullanılması olanaklı olur. Böylece, programlar daha kısa olur, programın yazılma zamanı azalır ve gerektiğinde değiştirilmesi ve onarılması kolay olur.
- Kalıtım yoluyla yeni bir sınıf, önceden yazılmış başka bir sınıftan türetilebilir. Üretilen yeni sınıf, türetildiği sınıfın kalıtım yoluyla gelebilen özelliklerine sahip olur.
- Yeni üretilen bir sınıf türetildiği sınıfın özelliklerini değiştirerek kullanabilir. Yeni üretilen sınıfın kendisine ait yeni özellikleri de tanımlanabilir.

# Kalıtım (Inheritance)

- C#'ta kalıtım, mevcut bir sınıftan yeni bir sınıf oluşturmamıza olanak tanır.
- Kalıtılım Nesneye Yönelik Programlamanın (OOP) en temel özelliklerinden birisidir.
- Yeni bir sınıfın oluşturulduğu sınıf, temel sınıf (parent or superclass) olarak bilinir. Yeni sınıfa türetilmiş sınıf (child or subclass) denir.
- Türetilmiş sınıf, temel sınıfın özelliklerini, öznitelikleri ve yöntemlerini devralır. Bu, C#'daki kodun yeniden kullanılabilirliğine yardımcı olur.

# Kalıtım (Inheritance)

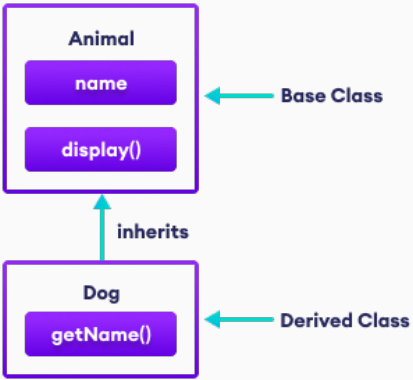
- C#'ta kalıtımı gerçekleştirmek için : sembolünü kullanırız.

```
1 reference
class Animal
{
    // fields and methods
}

// Dog inherits from Animal
0 references
class Dog : Animal
{
    // fields and methods of Animal
    // fields and methods of Dog
}
```

# Kalıtım (Inheritance)

- Burada, Animal temel sınıf. Dog sınıfı miras alan yani türetilmiş (derived class) sınıf. Dog sınıfı artık Animal sınıfının alanlarına ve yöntemlerine erişebilir.



# Animal ve Dog Class

```
// base class
1 reference
class Animal
{
    public string name;

    0 references
    public void display()
    {
        Console.WriteLine("I am an animal");
    }
}
```

```
// derived class of Animal
0 references
class Dog : Animal
{
    0 references
    public void getName()
    {
        Console.WriteLine("My name is " + name);
    }
}
```


# Kaynak Kodu

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        // object of derived class
        Dog dg = new Dog();

        // access field and method of base class
        dg.name = "Kangal";
        dg.display();

        // access method from own class
        dg.getName();
    }
}
// base class
1 reference
class Animal...

// derived class of Animal
2 references
class Dog ...
```

Kalıtım yoluyla elde edilen özellik ve fonksionlar

# Çıktı

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        // object of derived class
        Dog dg = new Dog();

        // access field and method of base class
        dg.name = "Kangal";
        dg.display();

        // access method from own class
        dg.getName();
    }
```

```
C:\WINDOWS\system32\cmd.exe
I am an animal
My name is Kangal
Press any key to continue . . .
```

# Kalıtımın önemi için örnek

- Kalıtımın önemini daha iyi anlamak için bir örnek yapalım.
- Kare, diktörgen ve benzeri normal çok genler ile çalışıyoruz.
- Girdiye göre bu çok genlerin alanların ve çevrelerinin değerlerini bulmak istiyoruz.
- Alan hesaplama formülü tüm normal çokgenler için ortaktır. Bu nedenle Normal Çokgen sınıfı ve çevreyi hesaplamak için AlanHesapla() yöntemini oluşturabiliriz.

# Classes

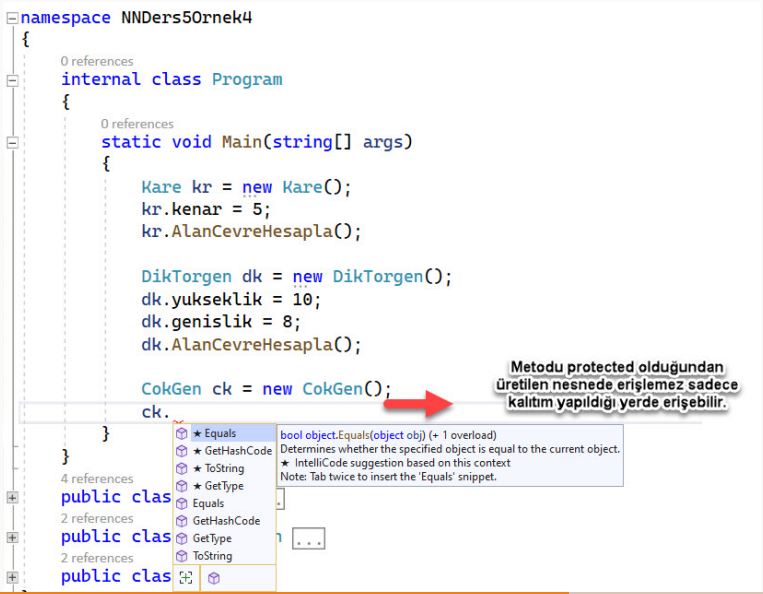
```
public class CokGen
{
    2 references
    protected void AlanHesapla(int width, int lenght)
    {
        Console.WriteLine("Çok genin alanı:" + (width * lenght));
    }
}
2 references
public class DikTorgen : CokGen
{
    public int genislik, yukseklik;
    1 reference
    public void AlanCevreHesapla()
    {
        int cevre = 2 * (genislik + yukseklik);
        Console.WriteLine("Çok genin çevresi :" + cevre);
        AlanHesapla(genislik, yukseklik);
    }
}
2 references
public class Kare : CokGen
{
    public int kenar;
    1 reference
    public void AlanCevreHesapla()
    {
        int cevre = 4 * kenar;
        Console.WriteLine("Çok genin çevresi :" + cevre);
        AlanHesapla(kenar, kenar);
    }
}
```

protected olduğundan sadece alt sınıflardan erişilebilir.

kalıtım

kalıtım

# Kaynak Kodu


namespace NNDers5Ornek4
{
0 references
internal class Program
{
0 references
static void Main(string[] args)
{
Kare kr = new Kare();
kr.kenar = 5;
kr.AlanCevreHesapla();
DikTorgen dk = new DikTorgen();
dk.yukseklik = 10;
dk.genislik = 8;
dk.AlanCevreHesapla();
CokGen ck = new CokGen();
ck.
}
}
4 references
public clas
2 references
public clas
2 references
public clas
Metodu protected olduğundan üretilen nesnede erişemez sadece kalıtım yapıldığı yerde erişebilir.
★ Equals
★ GetHashCode
★ ToString
★ GetType
Equals
GetHashCode
GetType
ToString
bool object.Equals(object obj) (+ 1 overload)
Determines whether the specified object is equal to the current object.
★ IntelliCode suggestion based on this context
Note: Tab twice to insert the 'Equals' snippet.

# Çıktı

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        Kare kr = new Kare();
        kr.kenar = 5;
        kr.AlanCevreHesapla();

        DikTorgen dk = new DikTorgen();
        dk.yukseklik = 10;
        dk.genislik = 8;
        dk.AlanCevreHesapla();

        CokGen ck = new CokGen();
    }
}
```

```
C:\WINDOWS\system32\cmd.exe
Çok genin çevresi :20
Çok genin alanı:25
Çok genin çevresi :36
Çok genin alanı:80
Press any key to continue . . .
```

## Kalıtımın önemi

- Alan hesabının yapıldığı yeri değiştirelim.
- Sonucun bütün sınıfları etkilediğini görelim.
- Program çalıştığından sonuçların değiştiğini göreceksiniz.
- Nesne yönelimli programlama bunun gibi durumlar için önemlidir. Sadece bir yerde değişiklik yaparsınız o sınıftan üretilen sınıflarda veya nesnelerin tamamında güncelleme yapmış olursunuz.

# Kalıtım ve ilk değer alma sırası

- Kalıtım, yeni oluşturulan sınıfın türetildiği sınıfa ait özellikleri alması ve ayrıca kendisine ait özellikleri tanımlayabilmesidir.
- Kalıtım yoluyla sınıflar oluşturulup daha sonra da nesne oluşturulurken öncelikle türetildiği sınıfın nesnesi oluşturulmaya çalışır.
- Bu işlem en son ilk kalıtım alınan sınıfın kurucu fonskiyonuna kadar gidilir. İlk önce o tetiklenir daha sonra ondan türeyen sınıfların metotları tetiklenir.

# Kalıtım ve ilk değer alma sırası

- Bengal kaplanı için bir sınıf olduğunu farz edelim. Bengal kaplanı aslında bir kaplandır ve kaplan da aslında bir hayvandır. Önce hayvan sınıfını tanımlayalım, daha sonra hayvan sınıfından kalıtım yoluyla kaplan sınıfını tanımlayalım ve daha sonra kaplan sınıfından kalıtım yoluyla türeyen bir bengal kaplanı sınıfını tanımlayalım.

- Her üç sınıfında constructor metodu olsun.

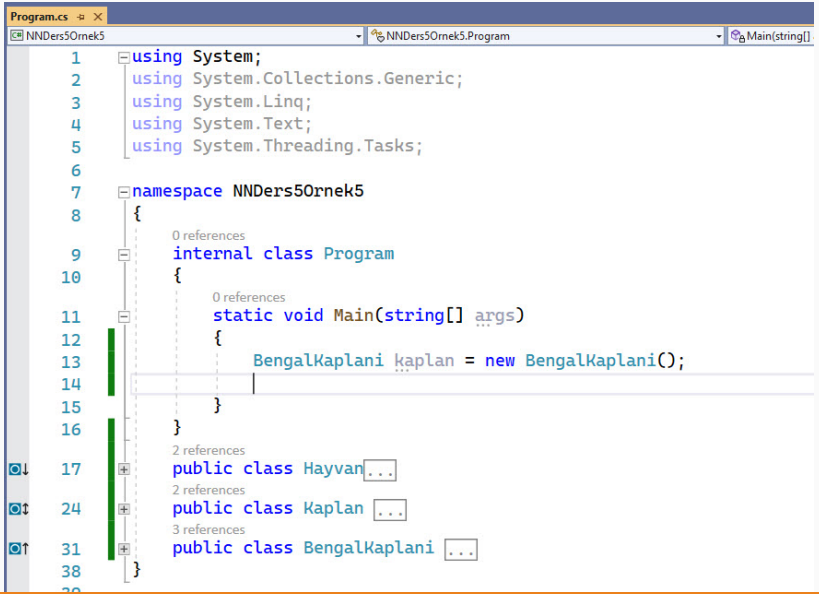
# Classes

```
public class Hayvan
{
    0 references
    public Hayvan()
    {
        Console.WriteLine("Hayvan sınıfı için tetiklenen yapılandırıcı.");
    }
}
2 references
public class Kaplan : Hayvan
{
    0 references
    public Kaplan()
    {
        Console.WriteLine("Kaplan sınıfı için tetiklenen yapılandırıcı.");
    }
}
3 references
public class BengalKaplani : Kaplan
{
    1 reference
    public BengalKaplani()
    {
        Console.WriteLine("Bengal kaplanı sınıfı için tetiklenen yapılandırıcı.");
    }
}
```

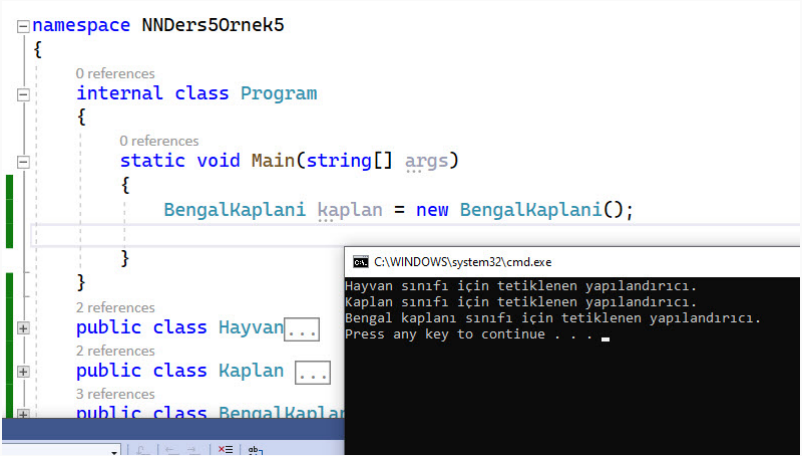

# Kaynak Kodu

```
public class Hayvan
{
    0 references
    public Hayvan()
    {
        Console.WriteLine("Hayvan sınıfı için tetiklenen yapılandırıcı.");
    }
}
2 references
public class Kaplan : Hayvan
{
    0 references
    public Kaplan()
    {
        Console.WriteLine("Kaplan sınıfı için tetiklenen yapılandırıcı.");
    }
}
3 references
public class BengalKaplani : Kaplan
{
    1 reference
    public BengalKaplani()
    {
        Console.WriteLine("Bengal kaplanı sınıfı için tetiklenen yapılandırıcı.");
    }
}
```

# Kurucu fonksiyonların çalışma sırası



# Çıktı


namespace NNDers5Ornek5
{
0 references
internal class Program
{
0 references
static void Main(string[] args)
{
BengalKaplani kaplan = new BengalKaplani();
}
}
2 references
public class Hayvan ...
2 references
public class Kaplan ...
3 references
C:\WINDOWS\system32\cmd.exe
Hayvan sınıfı için tetiklenen yapılandırıcı.
Kaplan sınıfı için tetiklenen yapılandırıcı.
Bengal kaplanı sınıfı için tetiklenen yapılandırıcı.
Press any key to continue . . .

# Ödev - Notebook Sistemi

- 4 Farklı marka için bir laptop kütüphanesinin oluşturulması istenmektedir.
- Her markada İşlemci, RAM, Anakart, DISK ve Ekran bulunmaktadır.
- Ayrıca bu bilgisayarlara sonradan RAM takılabilmektedir.
- Lenova markaların ekran boyutu 17.3 olmaktadır. Sony bilgisayarların işlemcisi en az İ5 olmaktadır. Dell bilgisayarlarda disk SSD olmaktadır. Asus bilgisyarlar ise sadece i7 serisi üretmektedir.
- Bilgisayarın detay özelliklerine girildiğinde RAM'ın veri yolu hızı, İşlemcinin hızı, diskin okuma yazma hızı var ise Ekran kartı markası paylaşımlı olup olmadığı, monitörün boyutu vb. bütün detay özellikler olmalıdır.
- Fiyatlandırma ise laptopta bulunan özelliklere göre yapılmalıdır. Mesala İ3 5. Nesil 2.4 GHZ RAM'ın fiyatı 100$'dır. Bu ram A marka notebook takarsa 110 (%10 kar ile), B markası takarsa 109, C Markası takarsa 113 ve D Marka notebook takarsa 115 olarak satmaktadır.
- Nesne yönelimli programlamada kalıtım mantığıyla ilgili projeyi kodlayınız.
- 6 Kasım 06.59'a kadar gönderilenler 8+2 puan, vize tarihine kadar gönderenler 8 puan (kayubmprogramlama1@gmail.com).