BİL 201
NESNE YÖNELİMLİ PROGRAMLAMA (NYP)

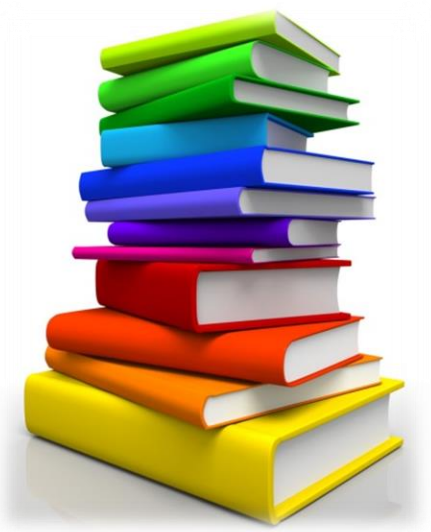
DERS # 9

Öğretim Üyesi: Doç. Dr. Deniz Kılınç

BAKIRÇAY ÜNİVERSİTESİ

BÖLÜM 9 – Soyut Sınıflar (Abstract Classes)

- Bu bölümde aşağıdaki konular anlatılacaktır.
  - Soyut Sınıfların Oluşturulması
  - Soyut Sınıfların Kullanımı



Soyut Sınıflar

- Kalıtım kavramını anladıktan sonra, sınıfların yaratılması daha da kolaylaşır.
- Yavru (Child) bir sınıf oluşturduğunuzda, genel özellikleri **miras alınır** ve sonrasında sınıfa gereken yeni, spesifik özellikler **eklenir/kazandırılır**.

  Örneğin;

  Ressam ile Sair sınıfları Sanatci sınıflarının daha da özelleşmiş halidir.

  Bu sınıflar, Sanatci sınıfının erişilebilen **tüm özelliklerini ve metotlarını** barındırmakla beraber **spesifik** özellikleri de barındırırlar.

Soyut Sınıflar (devam…)

- **Temel sınıfları** düşünmenin bir yolu, yavru sınıfların ortak özelliklerinin ***tümünü düşünmek*** olabilir.
- Genişletilmiş sınıf incelendiğinde, ebeveyn sınıfının daha genel olduğu **gözlenir**.
- Bazen yavru sınıfların daha genel bir örneğini oluşturmak için ebeveyn sınıf oluşturulur.
  - Örneğin, önceden bir Sanatci yaratmamış olabilirsiniz; herbir Sanatci bir Ressam'ın , Sair'in ya da Muzisyen'in daha genel halidir.

Soyut Sınıflar (devam…)

*Sonradan genişletilmek üzere yaratılan fakat kendisinden **nesne oluşturulmayan** sınıflara soyut sınıf (abstract class) denir.*

- Soyut sınıfları tanımlarken **abstract** anahtar kelimesi kullanılır.
- Soyut sınıflar da normal sınıflar gibi özellikler ve metotlar içermektedir. Normal sınıflardan farklı olarak bu sınıflardan new operatörünü kullanarak **nesne yaratılamaz**.
- Bunun yerine, soyut sınıflar bir **ana sınıf sağlar**.
- Soyut sınıflar genellikle soyut metotlar (abstract methods) içerirler.
- Soyut bir metot **hiçbir metot ifadesi içermez**; bu sınıftan türetilen sınıflar, bu metotları ezmelidir.

Soyut Sınıflar (devam…)

- Soyut bir metodun başlığında, isteğe bağlı erişim belirleyicisi, `abstract` anahtar kelimesi, istenilen metodun tipi ve adı bulunur:

```
public abstract void Hesapla();
```

- Soyut metotlarda kod gövdesi **bulunmaz**. Bu metotların gövdeleri **kalıtılan sınıfta tanımlanır**.
- Soyut bir sınıftan miras alınarak, yeni bir sınıf yaratıldığında, yeni sınıfta `override` anahtar kelimesini kullanarak **soyut metotların** gövdeleri oluşturulmalıdır.

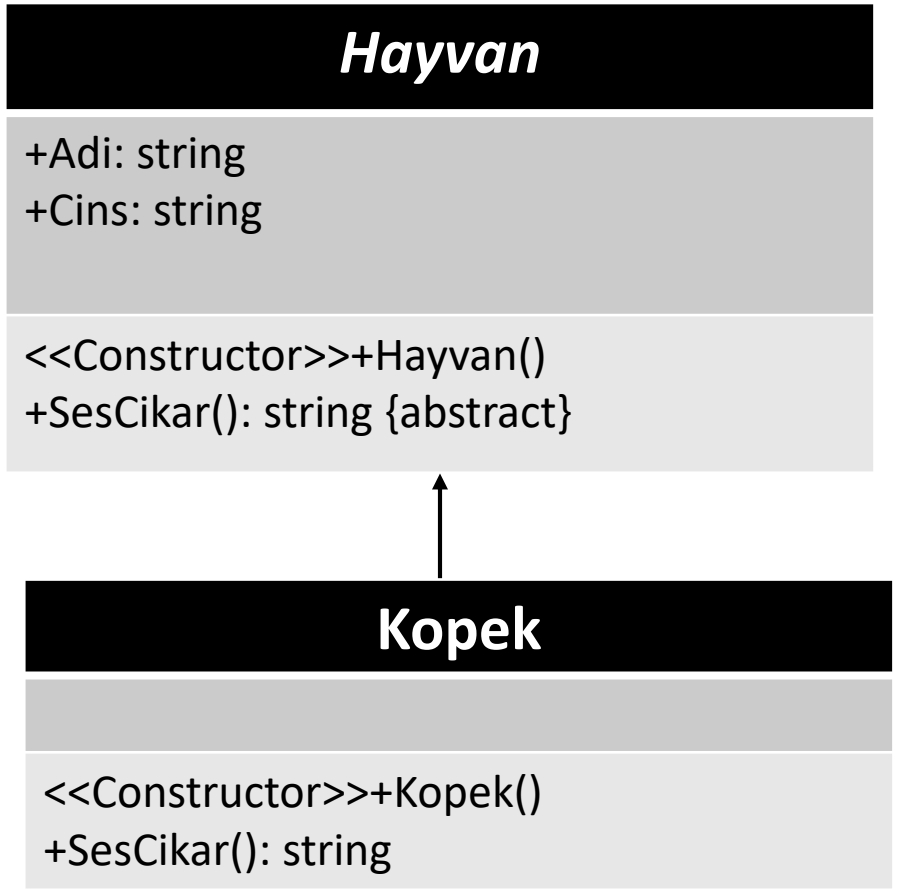
Soyut Sınıflar (devam...)

`virtual` anahtar kelimesi ile tanımlanan metodun yavru sınıfta **ezilmesi zorunlu değildir**.

Fakat `abstract` tanımlanmış **metot ezilmelidir**.

Aslında esas **çok biçimlilik (polymorphism)** soyut metotlar ve ara yüzler (interface) aracılığı ile gerçekleştirilmektedir.

Örnek1: Hayvan Soyut Sınıfının Yaratılması



Örnek1: Hayvan Soyut Sınıfının Yaratılması

```
public abstract class Hayvan
{
    1 reference
    public string Adi { get; set; }
    0 references
    public string Cins { get; set; }
    0 references
    public Hayvan()
    {
        //Todo
    }
    2 references
    public abstract string SesCikar();
}
```

```
2 references
public class Kopek : Hayvan
{
    2 references
    public override string SesCikar()
    {
        return "Havhav";
    }
}
```

Örnek1: Hayvan Soyut Sınıfının Yaratılması

- Form üzerinde bir **Kopek nesnesi** yaratalım.
- Kopek bilgilerini ekranda göstermek için form üzerinde **KopekBilgisiGoster()** isimli bir metot oluşturalım.

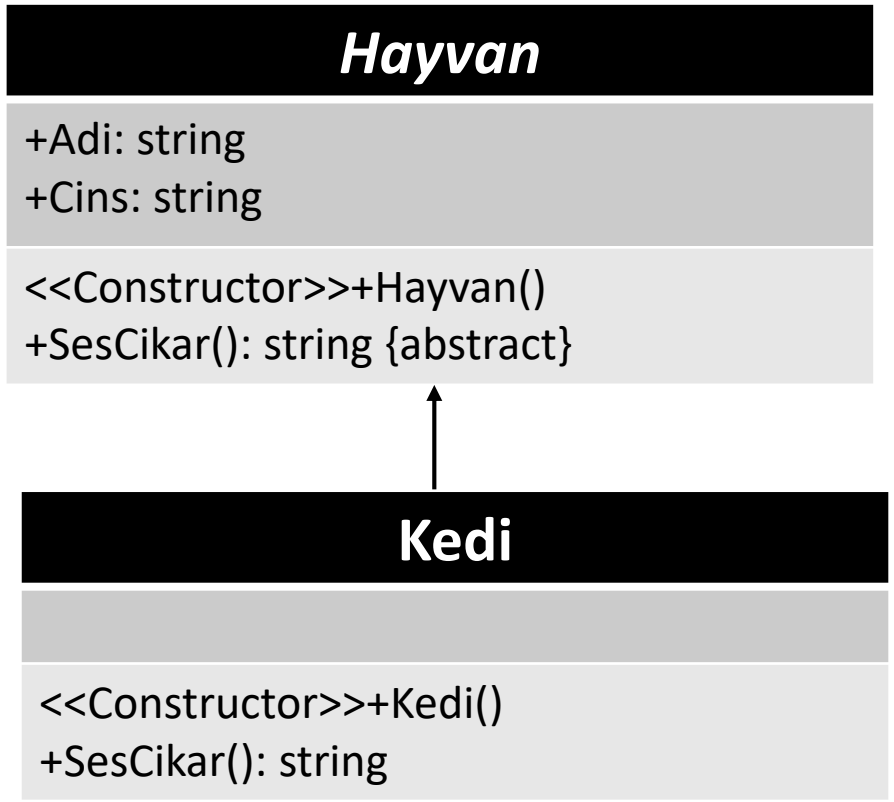
```
1 reference
private void KopekBilgisiGoster(Kopek kopek)
{
    MessageBox.Show("Köpek adı: " + kopek.Adi + Environment.NewLine +
                    "Ses çıkar: " + kopek.SesCikar());
}

1 reference
private void Form1_Load(object sender, EventArgs e)
{
    Kopek kopek = new Kopek();
    kopek.Adi = "Zeytin";
    KopekBilgisiGoster(kopek);
}
```

Soyut Sınıflar (devam…)

- Soyut bir metot **tanımlamadan da** soyut bir sınıf oluşturulabilir fakat **soyut sınıf oluşturulmadan** soyut bir metot oluşturulamaz.
- Child sınıfında override edilmesi gereken metot kendi tanımına sahipse, bu metot **virtual** olarak tanımlanır. Eğer kendine ait bir tanım/gövde yoksa **abstract** olarak tanımlanır.
- Örnekteki; Kopek sınıfında SesCikar() metodu **ezilmek zorundadır**. Bu metodun içerisinde herhangi bir işlem yapılmayabilir fakat bu metot **mutlaka olmalıdır**. Bu metodu ezerken override anahtar kelimesi kullanılır.

Örnek1: Hayvan Soyut Sınıfının Yaratılması



Örnek1: Hayvan Soyut Sınıfının Yaratılması

```
public class Kedi : Hayvan
{
    4 references
    public override string SesCikar()
    {
        return "Miyavvvv";
    }
}
```

Örnek1: Hayvan Soyut Sınıfının Yaratılması

- Form üzerinde bir **Kedi nesnesi** yaratalım.
- Kopek bilgilerini ekranda göstermek için form üzerinde **KediBilgisiGoster()** isimli bir metot oluşturalım.

```
1 reference
private void KediBilgisiGoster(Kedi kedi)
{
    MessageBox.Show("Kedi adı: " + kedi.Adi + Environment.NewLine +
                    "Ses çıkar: " + kedi.SesCikar());
}
```

```
Kedi kedi = new Kedi();
kedi.Adi = "Boncuk";
KediBilgisiGoster(kedi);
```

Örnek1: Hayvan Soyut Sınıfının Yaratılması

Dikkat: İki metot arasında **ne fark var**? Daha iyi bir çözümü olan var mı?

```
1 reference
private void KopekBilgisiGoster(Kopek kopek)
{
    MessageBox.Show("Köpek adı: " + kopek.Adi + Environment.NewLine +
                    "Ses çıkar: " + kopek.SesCikar());
}

1 reference
private void KediBilgisiGoster(Kedi kedi)
{
    MessageBox.Show("Kedi adı: " + kedi.Adi + Environment.NewLine +
                    "Ses çıkar: " + kedi.SesCikar());
}
```

Örnek1: Hayvan Soyut Sınıfının Yaratılması

- Kedi ve Kopek sınıfından yaratılmış nesneler Hayvan parametresi alan **HayvanBilgisiGoster()** metoduna aktarılabilirler.
- Bu yönteme teknik olarak **upcast** de denir.
- Ortaya konulan yazılım tasarım prensibine **L**SP (Liskov Substitution Principle - Liskov Yerine Geçme/Değiştirme) prensibi denilir (Barbara Liskov).

Örnek1: Hayvan Soyut Sınıfının Yaratılması

```
2 references
private void HayvanBilgisiGoster(Hayvan hayvan)
{
    MessageBox.Show("Hayvan adı: " + hayvan.Adi + Environment.NewLine +
                    "Ses çıkar: " + hayvan.SesCikar());
}
1 reference
private void Form1_Load(object sender, EventArgs e)
{
    Kopek kopek = new Kopek();
    kopek.Adi = "Zeytin";
    HayvanBilgisiGoster(kopek);


    Kedi kedi = new Kedi();
    kedi.Adi = "Boncuk";
    HayvanBilgisiGoster(kedi);
}
```

Senaryo: Çalışan Maaş Hesapla

DeepBlue şirketinde 3 tip çalışan bulunmaktadır: Standart, Muhasebeci ve Satışçı. Her çalışan İsim ve Maaş bilgisine sahiptir. Tüm çalışanlar baz olarak, aylık çalıştıkları 21 iş günü karşılığı asgari ücretin 2 katı maaş almaktadır. Muhasebeciler o ay yaptıkları her fazla mesai için gün başına (Maaş/21) tutarında mesai ücreti almaktadırlar. Satışçılar ise fazla mesai ücreti almayıp, o ay gerçekleştirdikleri satışın %10'u kadar satış primi almaktadırlar.

- Veli Gümüş standart bir çalışandır.
- Ahmet Demir Muhasebeci olup, bir ay boyunca 3 gün fazla mesai yapmıştır.
- Mehmet Çelik Satışçı olup, bir ay boyunca 1000 TL'lik satış yapmıştır.
- Asgari ücret 2000 TL olduğuna göre her üç personelin bir aylık maaşlarını hesaplayınız.
- DeepBlue firmasına Ahmet Demir, Mehmet Çelik ve Veli Gümüş isimli kullanıcıları ekleyelim. Firmanın o ayki çalışanlarına ödemesi gereken toplam maaşı bulalım.
- **Not:** Çok biçimlilik ve soyut sınıf kullanınız.

Soyut Sınıflar Özet

- Soyut sınıflar, birden fazla child (yavru) sınıf için ortak özelliklerin ve/veya metotların ihtiyaç olduğu durumlarda ortak kısımları toparlama amacı ile kullanılabilirler.
- Soyut sınıf kullanımları önemlidir, ancak zorunlu değildir.
- Soyut sınıftan nesne türetilemez.
- Soyut sınıflar private olamazlar.
- Soyut sınıflar sealed (mühürlü) olamazlar.

Yararlanılan Kaynaklar

- Sefer Algan , HER YÖNÜYLE C# , Pusula Yayıncılık, İstanbul, 2003
- Volkan Aktaş, HER YÖNÜYLE C# 5.0 , Kodlab Yayıncılık, İstanbul, 2013
- Milli Eğitim Bakanlığı "Nesne Tabanlı Programlama", 2012

İyi Çalışmalar...

Doç. Dr. Deniz Kılınç

deniz.kilinc@bakircay.edu.tr

drdenizkilinc@gmail.com

www.denizkilinc.com