BİL 201
NESNE YÖNELİMLİ PROGRAMLAMA (NYP)

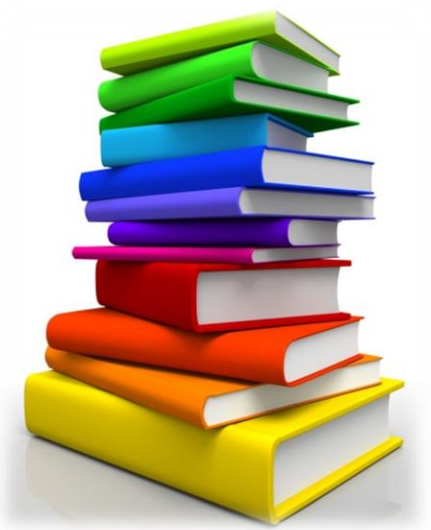
DERS # 8

Öğretim Üyesi: Doç. Dr. Deniz Kılınç

BAKIRÇAY ÜNİVERSİTESİ

BÖLÜM 8 – Metot Ezme ve Çok biçimlilik

- Bu bölümde aşağıdaki konular anlatılacaktır.
  - Temel Sınıfların Metotlarını Ezme (**Override**)
  - Çok biçimlilik (**Polymorphism**)



Temel Sınıfın Metotlarını Ezme

- Önceden var olan bir sınıftan miras alınarak, genişletilmiş yeni bir sınıf oluşturduğunuzda, yeni sınıf içerisinde temel sınıfın *tüm özellik ve metotları* tanımlanmış olur.
- Bazen parent (ebeveyn) sınıfın üyeleri, özellikleri ve metotları tam olarak child (yavru) sınıftan yaratılan nesneler için ***uygun olmayabilir***.

Farklı işlemler yapan fakat aynı isimdeki özellik veya metotların kullanımına ***çok biçimlilik*** *(polymorphism) denmektedir.*

Temel Sınıfın Metotlarını Ezme (devam…)

Çok biçimlilik, “birçok form içeren” anlamına gelmektedir.

Aynı isimde *olmalarına rağmen* farklı işlemlerin yer aldığı metotlar için kullanılır.

Temel Sınıfın Metotlarını Ezme (devam…)

- Günlük hayatta çok biçimliliğe **örnek** gösterebilecek çeşitli olaylar bulunmaktadır:
  - Tüm müzik aletleri için "çalmak" eylemi kullanılmasına rağmen, *bir gitarın bir davuldan* farklı bir biçimde çalınıyor olması. (Cal() metotları aynı)
  - Tüm araçlar için "sürmek" eylemi kullanılmasına rağmen, bir *otomobilin* kullanışının bir *bisiklet* kullanımından farklı olması,
  - Tüm okulların "mezun olma koşulları"na sahip olması fakat bir *lise* mezuniyeti ile *ilkokul* mezuniyeti koşulları arasında farklılık olması çok biçimliliğe örnek gösterilebilir.

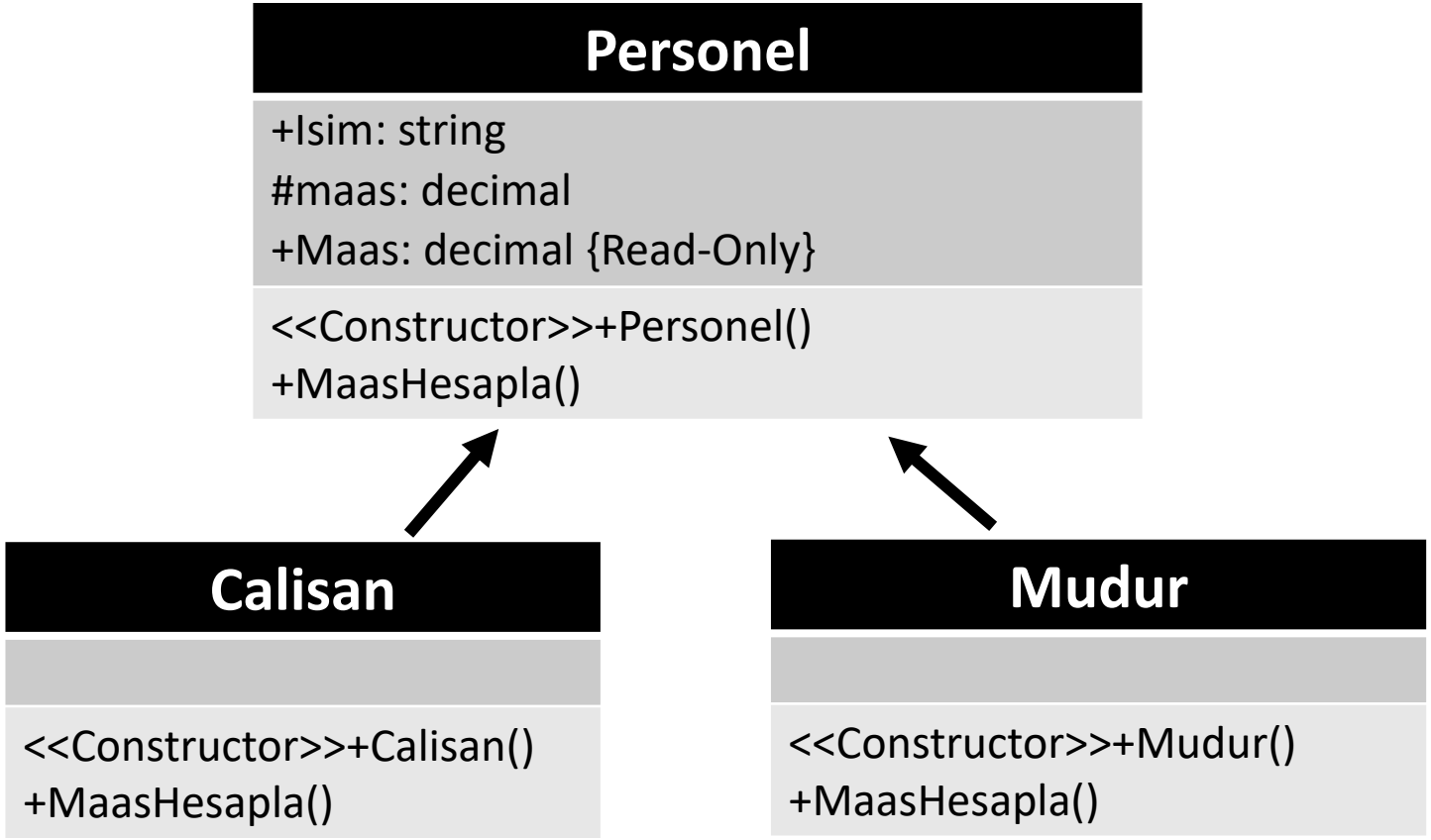
Temel Sınıfın Metotlarını Ezme (devam…)

- Bir **virtual** metot (ya da property) child sınıftaki aynı isme sahip *bir metot tarafından* başına **override** anahtar kelimesi eklenerek ezilmesine olanak sağlar.

```
public class Parent
{
        public virtual void Metot1()
        { …
        }
}

public class Child: Parent
{
        public override void Metot1()
        { …
        }
}
```

Örnek 1: Maaş Hesapla



Örnek 1: Maaş Hesapla

Senaryo 1:

- Personel maaş hesaplama ile ilgili ***Personel Temel sınıfında*** herhangi özel bir hesaplama yapılmıyor.
- Çalışan maaşları 4000 TL'dir.
- Müdür maaşları 7000 TL'dir.
- Çok biçimlilik kullanarak bu durumu gerçekleştirelim.
- Form üzerinde nesneleri oluşturarak test ve **DEBUG** işlemi gerçekleştirelim.

Örnek 1: Maaş Hesapla

```
public class Personel
{
    0 references
    public string Isim { get; set; }
    protected Decimal maas;
    3 references
    public Decimal Maas {
        get {
            return maas;
        }
    }
    1 reference
    public Personel()
    {
        //Todo
    }
    5 references
    public virtual void MaasHesapla()
    {
        //Todo
    }
}
```

Senaryo 1

```
public class Calisan : Personel
{
    1 reference
    public Calisan()
    {
        //Todo
    }
    5 references
    public override void MaasHesapla()
    {
        maas = 4000;
    }
}
```

```
public class Mudur : Personel
{
    1 reference
    public Mudur()
    {
        //Todo
    }
    5 references
    public override void MaasHesapla()
    {
        maas = 7000;
    }
}
```

Örnek 1: Maaş Hesapla

Senaryo 1:

- **Calisan** MaasHesapla() metodu çağırıldığında Temel Sınıf olan **Personel'in** MaasHesapla() metodu çağrılıyor mu?

- **Cevap: Hayır**
- **İhtiyaç olsaydı?**

```
Personel personel = new Personel();
personel.MaasHesapla();
MessageBox.Show(personel.Maas.ToString());

Calisan calisan = new Calisan();
calisan.MaasHesapla();
MessageBox.Show(calisan.Maas.ToString());

Mudur mudur = new Mudur();
mudur.MaasHesapla();
MessageBox.Show(mudur.Maas.ToString());
```

Yavru sınıf, temel sınıfın metotlarına base anahtar sözcüğü ile erişebilir.

Örnek 1: Maaş Hesapla

Senaryo 2:

- Personel maaş hesaplama ile ilgili ***Personel Temel sınıfında*** özel bir hesaplama yapılıyor.
  - **Maaş** = **ASGARIUCRET** (2000)+ **AILEGECIMINDIRIMI (500)**;
- Çalışan maaşları 1.5 kat.
- Müdür maaşları 3.5 kat.
- Çok biçimlilik kullanarak bu durumu implemente edelim.
- Form üzerinde nesneleri oluşturarak test ve **DEBUG** işlemi gerçekleştirelim.

Örnek 1: Maaş Hesapla

```
public class Personel
{
    private const Decimal ASGARIUCRET = 2000;
    private const Decimal AILEGECIMINDIRIMI = 500;
    0 references
    public string Isim { get; set; }
    protected Decimal maas;
    3 references
    public Decimal Maas {
        get {
            return maas;
        }
    }
    1 reference
    public Personel()
    {
        //Todo
    }
    7 references
    public virtual void MaasHesapla()
    {
        maas = ASGARIUCRET + AILEGECIMINDIRIMI;
    }
}
```

Senaryo 2

```
public class Calisan : Personel
{
    1 reference
    public Calisan()
    {
        //Todo
    }
    7 references
    public override void MaasHesapla()
    {
        base.MaasHesapla();
        maas *= 1.5M;
    }
}
```

```
public class Mudur : Personel
{
    1 reference
    public Mudur()
    {
        //Todo
    }
    7 references
    public override void MaasHesapla()
    {
        base.MaasHesapla();
        maas *= 4M;
    }
}
```

Örnek 2: Öğrenci ve Burslu Öğrenci

Senaryo:

- Öğrenciler; *isim*, *kredi sayısı* ve *toplam ders ücreti* bilgilerine sahiptirler.
- Öğrencilerin toplam ders ücretleri *read-only* olup, *birim ders ücreti* ile aldıkları kredi sayısı çarpılarak *DersUcretiHesapla()* isimli bir fonksiyon aracılığı ile hesaplanmaktadır.

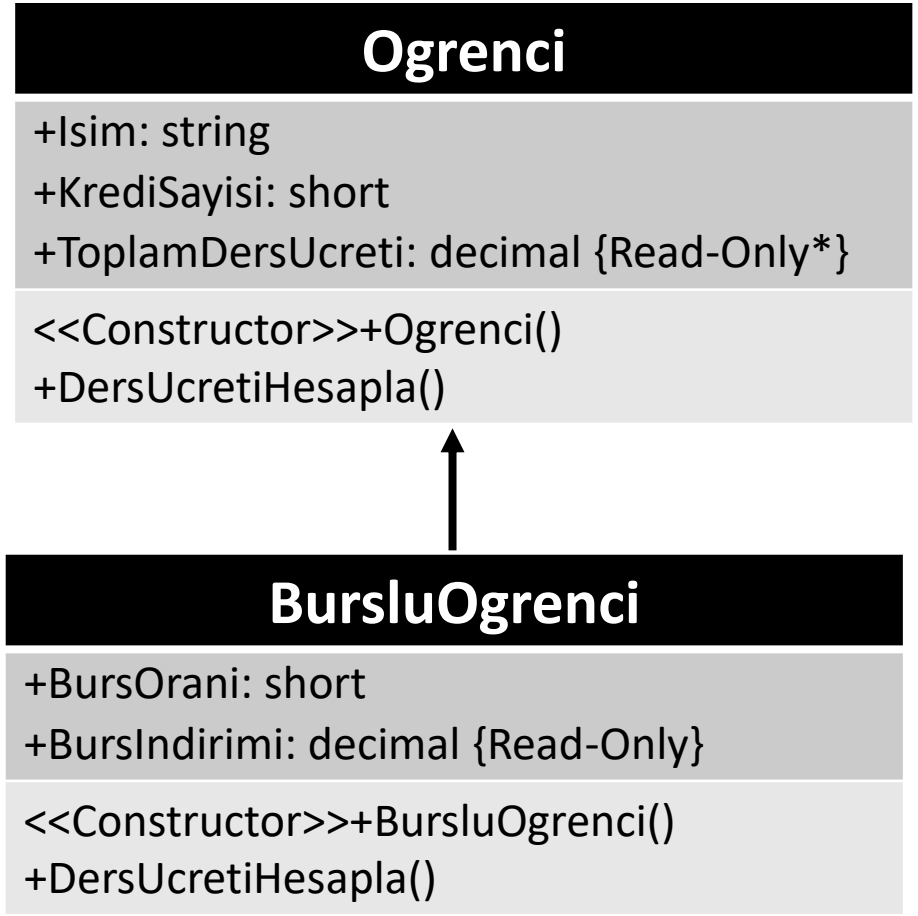
  BIRIMDERSUCRETI = 60

- Burslu öğrenci de bir öğrencidir.
- Burslu öğrenciler, toplam ders ücreti üzerinden burs oranı kadar burs indirimi alırlar. Burs indirimi *read-only* olup aşağıdaki gibi hesaplanır.

  ```
  BursIndirimi = (ToplamDersUcreti * BursOrani) / 100
  ```

- Çok biçimlilik kullanarak bu durumu implemente edelim.

Örnek 2: Öğrenci ve Burslu Öğrenci



Örnek 2: Öğrenci ve Burslu Öğrenci

```
public class Ogrenci
{
    private const Decimal BIRIMDERSUCRETI = 60;
    0 references
    public string Isim { get; set; }
    3 references
    public short KrediSayisi { get; set; }
    5 references
    public Decimal ToplamDersUcreti { get; protected set; }
    1 reference
    public Ogrenci()
    {
        //Todo
    }
    4 references
    public virtual void DersUcretiHesapla()
    {
        ToplamDersUcreti = BIRIMDERSUCRETI * KrediSayisi;
    }
}
```

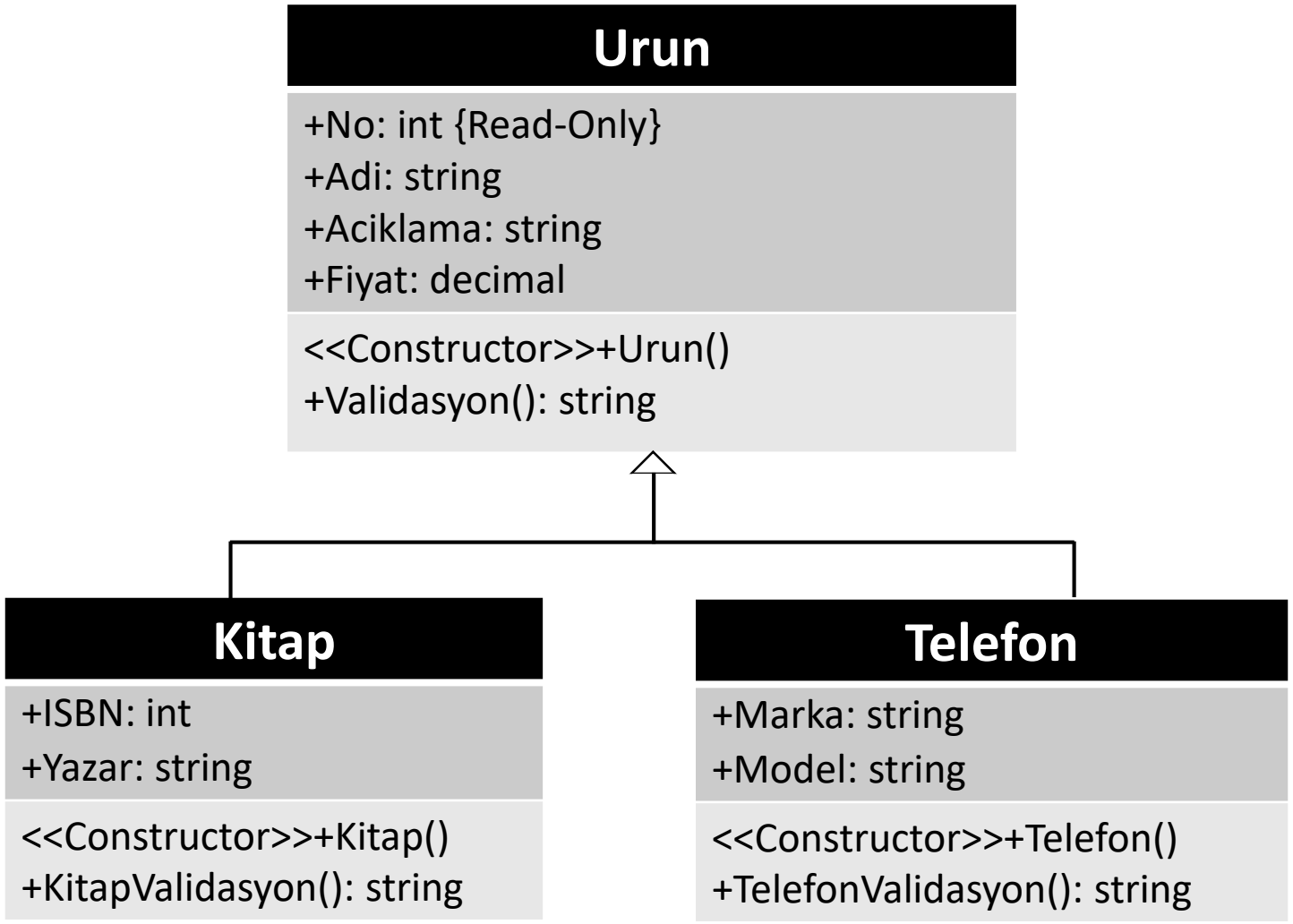
```
public class BursluOgrenci : Ogrenci
{
    2 references
    public short BursOrani { get; set; }
    3 references
    public decimal BursIndirimi { get; private set; }
    1 reference
    public BursluOgrenci()
    {
        //Todo
    }
    4 references
    public override void DersUcretiHesapla()
    {
        base.DersUcretiHesapla();
        BursIndirimi = (ToplamDersUcreti * BursOrani)/100;
        ToplamDersUcreti -= BursIndirimi;
    }
}
```

Örnek 2: Öğrenci ve Burslu Öğrenci

```
private void Form1_Load(object sender, EventArgs e)
{
    Ogrenci ogrenci = new Ogrenci();
    ogrenci.KrediSayisi = 23;
    ogrenci.DersUcretiHesapla();
    MessageBox.Show("Öğrenci toplam Ders ücreti: " + ogrenci.ToplamDersUcreti);

    BursluOgrenci bursluOgrenci = new BursluOgrenci
    {
        KrediSayisi = 23,
        BursOrani = 20
    };
    bursluOgrenci.DersUcretiHesapla();
    MessageBox.Show("Burslu Öğrenci" + Environment.NewLine +
                    "Burs İndirimi: " + bursluOgrenci.BursIndirimi + Environment.NewLine +
                    "ToplamDers Ücreti: " + bursluOgrenci.ToplamDersUcreti);
}
```

Örnek 3: İlk Kalıtım Örneği - Hatırla !



Örnek 3: İlk Kalıtım Örneği - Hatırla !

```
public string Validasyon()
{
    string hataMesaji = "";
    if (this.Adi == "")
        hataMesaji += "Ad özelliği boş olamaz.";
    if (this.Fiyat == 0)
        hataMesaji += "Fiyat özelliği 0 olamaz.";
    return hataMesaji;
}
```

```
public string TelefonValidasyon()
{
    string hataMesaji = "";
    if (this.Marka == "")
        hataMesaji += "Marka özelliği boş olamaz.";
    if (this.Model == "")
        hataMesaji += "Model özelliği boş olamaz.";
    return this.Validasyon() + " / " + hataMesaji;
}
```

```
public string KitapValidasyon()
{
    string hataMesaji = "";
    if (this.ISBN == 0)
        hataMesaji += "ISBN özelliği boş olamaz.";
    if (this.Yazar == "")
        hataMesaji += "Yazar özelliği boş olamaz.";
    return this.Validasyon() + " / " + hataMesaji;
}
```

Örnek 3: İlk Kalıtım Örneği - Hatırla !

- Eski tasarım *tam olarak* doğru değil.
- KitapValidasyon() ve TelefonValidasyon() **isimli** metotları *oluşturmaya artık gerek yok*.
- Bunun yerine, Temel Sınıftaki Validasyon() metodu virtual yapılarak yavru sınıflarda ezilebilir ve NYP (OOP) çerçevesinde daha **doğru bir tasarım gerçekleştirilir**.

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