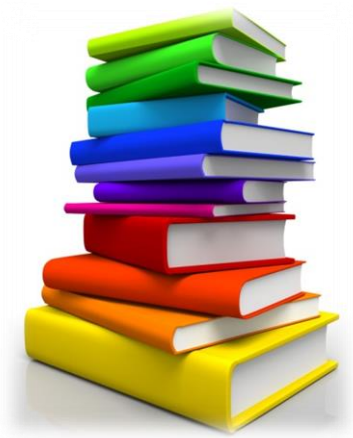
YZM 2105
Nesneye Yönelik Programlama

Yrd. Doç. Dr. Deniz KILINÇ
Celal Bayar Üniversitesi
Hasan Ferdi Turgutlu Teknoloji Fakültesi
Yazılım Mühendisliği

BÖLÜM - 6

Kalıtım (Inheritance) - I



Bu bölümde;

- Kalıtım Kavramı,
- Kalıtım Terimleri,
- Sınıfların Genişletilmesi,
- protected Erişim Belirleyicisi,
- Temel Sınıfların Metotlarını Ezme
- Çok biçimlilik

ile ilgili konular anlatılacaktır.

Kalıtım, Miras (Inheritance) Kavramı

Kalıtım (Inheritance) Kavramı

- Sınıfları anlamak gerçek hayatta nesneleri **düzenlemenize** yardımcı olur.
- Kalıtımı anlamak onları daha net bir şekilde **organize** etmenizi sağlar.
- Eğer Braford'u hiç duymadıysanız zihninizde canlandırmanız **mümkün değildir**.

Kalıtım (Inheritance) Kavramı

? Braford ?

Hayvan

Memeli

Kalıtım (Inheritance) Kavramı

? Braford ?

İnek

Kalıtım (Inheritance) Kavramı (devam...)

- Bu fikir onun memeli olduğunu öğrenince daha da büyür ve onun bir inek olduğunu öğrenince bu fikir zihninizde net bir hal alır.
- Braford'un bir inek olduğunu öğrendiğinizde, onun birçok inekte ortak olan özelliklere sahip olduğunu anlarsınız.
- Bir Braford'u ayırt edebilmek için sadece ona ilişkin rengi, büyüklüğü, işaretleri gibi ufak detayları öğrenmeniz gerekir.

Kalıtım (Inheritance) Kavramı (devam...)

- Halbuki Braford'un özelliklerinin çoğu, şu sınıfların **hiyerarşik yapısından** gelir:

  Hayvan → Memeli → İnek

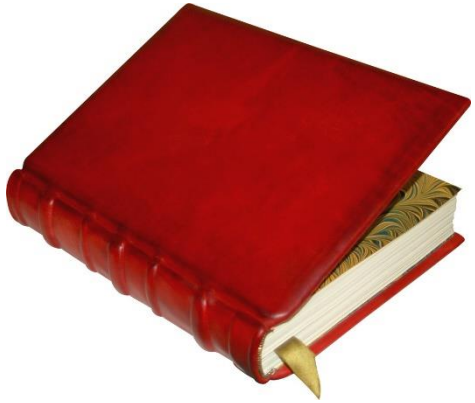
- Tüm **"object-oriented"** programlama dilleri kalıtımlardan *aynı sebepler* için *faydalanmaktadır*:
  - Programlarda kullanılan nesneleri düzenlemek,
  - Kalıtımla bildiklerinizi kullanarak yeni nesneleri yaratmak
  - Kod **reusability'sini** (tekrar kullanılabilirlik) arttırmak.

Kalıtım (Inheritance) Kavramı (devam...)

- Kalıtım, sizin genel bir kategori hakkındaki bildiklerinizi daha **spesifik bir kategoriye** uygulamanıza olanak sağlayan prensiptir.
- Kalıtım terimi kullanıldığında, genetik kalıtımı düşünebilirsiniz.
  - **Kan grubu** veya **göz rengi** kalıtılmış genlerin ürünüdür.
  - **Yürüyüşünüzün** babaannenizle aynı olması, ki bu yürüyüş size **babanızdan kalıtılmış** denebilir.

Kalıtım (Inheritance) Kavramı (devam...)

- Farklı tipte *ürünler* satan Ürün Satış uygulaması geliştirmek istediğimizi varsayalım.



Bu ürünlerin **sınıflarını** **oluşturabilir miyiz**?

Kalıtım (Inheritance) Kavramı (devam...)

Telefon ve Kitap ürünlerinin **özellikleri** nelerdir?

Telefon
+No: int +Adi: string +Marka: string +Model: string +Aciklama: string +Fiyat: decimal

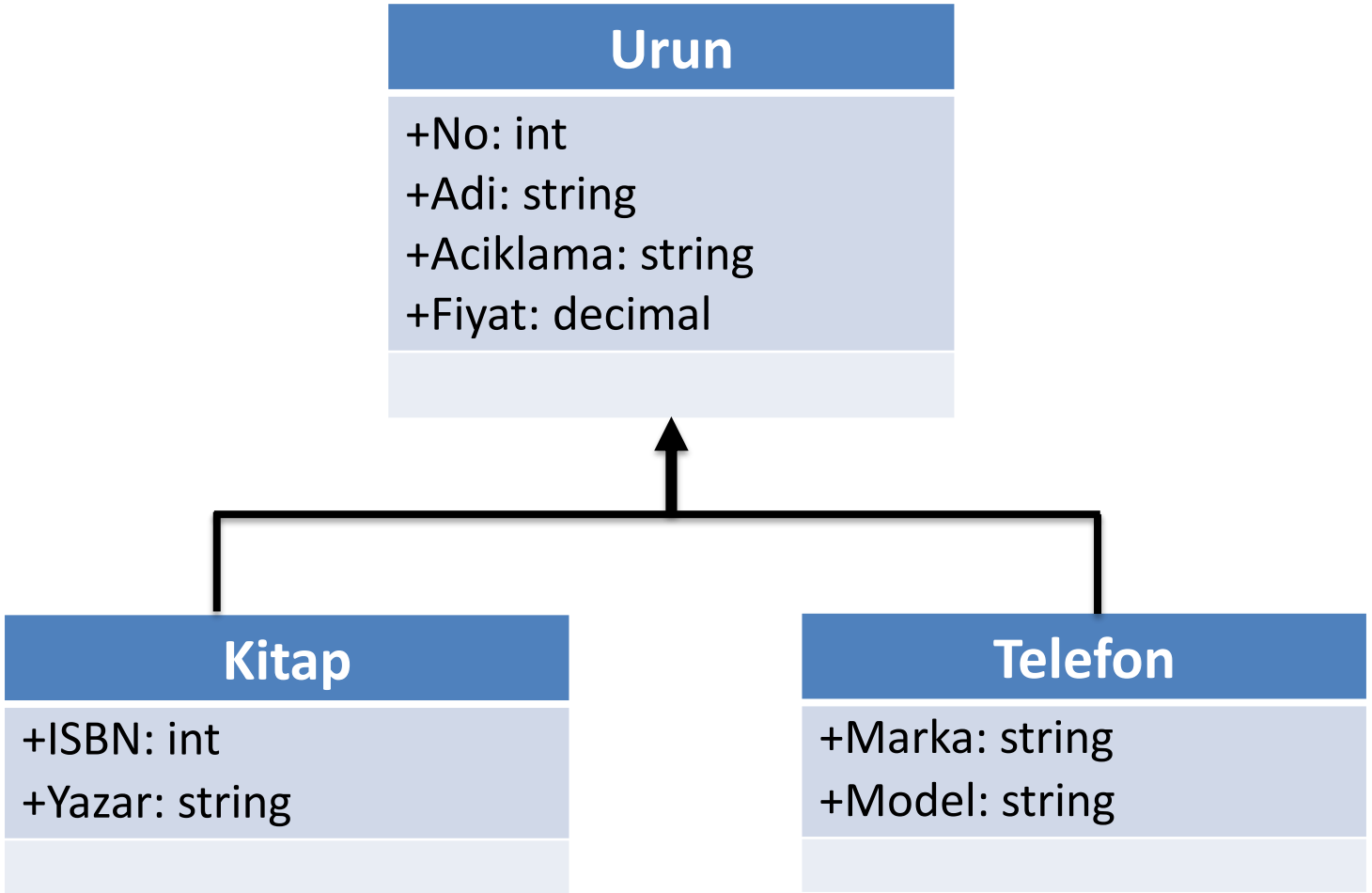
Kitap
+No: int +ISBN: int +Adi: string +Yazar: string +Aciklama: string +Fiyat: decimal

Kalıtım (Inheritance) Kavramı (devam...)

Telefon ve Kitap ürünlerinin **ortak özellikleri** nelerdir?

UrunOrtakOzellikler
+No: int +Adi: string +Aciklama: string +Fiyat: decimal

Örnek1: İlk Kalıtım Örneği



Kalıtımda Kullanılan Terimler

- Kalıtım için temel alınan sınıflara, Urun sınıfı gibi, temel sınıflar (ing.: **base classes**) denir.
- Temel sınıftan kalıtılarak oluşturulmuş sınıfa, Kitap gibi,
  - kalıtılmış sınıf (ing.: **derived class**) veya
  - genişletilmiş sınıf (ing.: **extended class**) denir.

Kalıtımda Kullanılan Terimler (Devam...)

- Ayrıca superclass ve subclass terimleri de temel sınıf ve kalıtılmış sınıflar için kullanılmaktadır.
  - Kitap sınıfı Urun superclass'ının subclass'ıdır.
- Buna benzer bir kullanım ayrıca ana (ing.: **parent**) ve yavru (ing.: **child**) sınıf kavramları da kullanılmaktadır.
  - Kitap sınıfı Urun ana sınıfının yavru sınıfıdır.

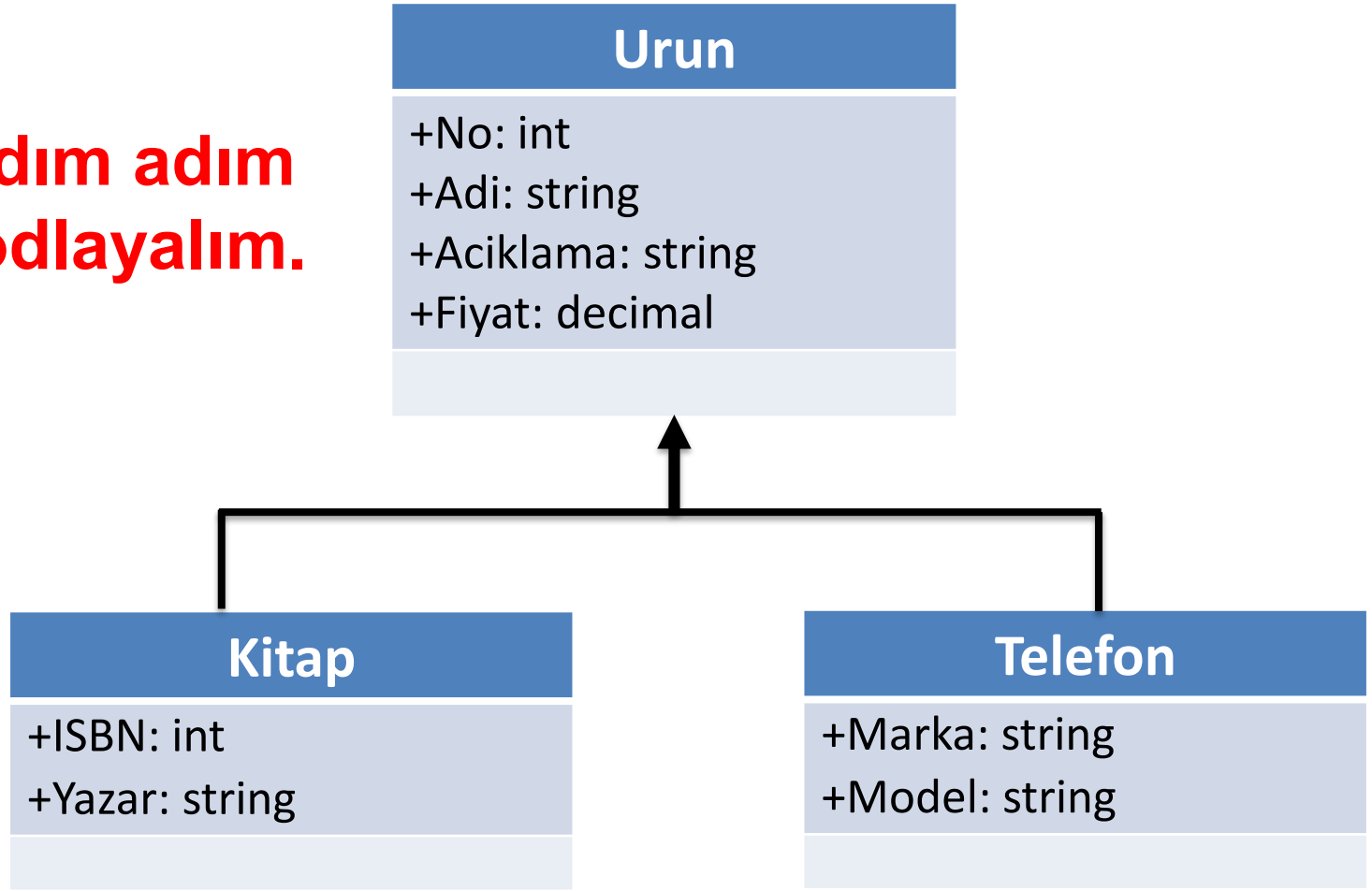
Sınıfların Genişletilmesi

- Başka ***bir sınıftan kalıtım ile yeni genişletilmiş bir sınıf yaratmak için*** (O sınıfın bir yavru sınıfını oluşturmak için) sınıf başlığında yavru sınıfın adı, iki nokta üst üste, ana sınıf adı yazılarak tanımlanır.

```
class [yavruSinifAdi] : [anaSinifAdi]
{
}
```

Örnek1: İlk Kalıtım Örneği

Adım adım kodlayalım.



Örnek1: İlk Kalıtım Örneği (devam…)

Adım 1

- Üç sınıfı da ayrı ayrı yaratalım.
  - Urun
  - Kitap
  - Telefon
- Form üzerinde üç sınıftan birer tane **nesne** *oluşturalım*.

Örnek1: İlk Kalıtım Örneği (devam…)

Adım 1

```
public class Urun
{
    0 references
    public int No { get; set; }
    0 references
    public string Adi { get; set; }
    0 references
    public string Aciklama { get; set; }
    0 references
    public decimal Fiyat { get; set; }
}
```

```
public class Kitap
{
    0 references
    public int ISBN { get; set; }
    0 references
    public string Yazar { get; set; }
}
```

```
public class Telefon
{
    0 references
    public string Marka { get; set; }
    0 references
    public string Model { get; set; }
}
```

Örnek1: İlk Kalıtım Örneği (devam…)

Adım 1

```
1 reference
private void btnTest_Click(object sender, EventArgs e)
{
    Urun o = new Urun();
    o.No = 1;
    o.Adi = "U1";

    Kitap k = new Kitap();
    k.ISBN = 23232;
    k.Yazar = "XXX";

    Telefon t = new Telefon();
    t.Marka = "Samsung";
    t.Model = "Galaxy xxx";
}
```

Örnek1: İlk Kalıtım Örneği (devam…)

Adım 2

- Kalıtım / Miras işlemini gerçekleştirelim.
  - Kitap ve Telefon sınıflarını Urun sınıfından miras alalım.
- Form üzerinde oluşturulan nesnelerin özelliklerini **gözlemleyin**.
  - Her nesne kaç özelliğe sahip?

Örnek1: İlk Kalıtım Örneği (devam…)

Adım 2

```
public class Telefon : Urun
{
    1 reference
    public string Marka { get; set; }
    1 reference
    public string Model { get; set; }
}
```

```
public class Kitap : Urun
{
    1 reference
    public int ISBN { get; set; }
    1 reference
    public string Yazar { get; set; }
}
```

Örnek1: İlk Kalıtım Örneği (devam…)

Adım 2

```
private void btnTest_Click(object sender, EventArgs e)
{
    Urun o = new Urun();
    o.No = 1;
    o.Adi = "U1";

    Kitap k = new Kitap();
    k.ISBN = 23232;
    k.Yazar = "XXX";
    k.No = 11;
    k.Adi = "Sindrella";


    Telefon t = new Telefon();
    t.Marka = "Samsung";
    t.Model = "Galaxy xxx";
    t.No = 22;
    t.Adi = "S6 Edgde";
}
```

Adım 2 Açıklama

- Kitap ve Telefon sınıflarında oluşturulan her bir nesne otomatik olarak Urun sınıfının erişim belirleyicisi **public** olan tüm *özelliklerini* içermektedir.
- Kalıtım **tek yönlü** çalışmaktadır:
  - Yavru sınıf, ana sınıftan kalıtılarak oluşturulur, *ters yönde* **oluşturulamaz**.
  - Program içerisinde bir **Urun nesnesi** oluşturduğunuzda **Kitap sınıfının** özelliklerine veya metotlarına erişemez.

Örnek1: İlk Kalıtım Örneği (devam…)

Adım 3

- Urun sınıfında No özelliğini **read-only** yapalım.
- Urun sınıfına bir tane **Constructor** ekleyelim ve burada No özelliğinin **random** olarak dolmasını sağlayalım.
- Form'da her nesneye ait **No** özellik değerini gösterelim.

Urun
+No: int {Read-Only} +Adi: string +Aciklama: string +Fiyat: decimal
<<Constructor>>+Urun()

Örnek1: İlk Kalıtım Örneği (devam…)

Adım 3

```
public class Urun
{
    4 references
    public int No { get; private set; }
    3 references
    public string Adi { get; set; }
    0 references
    public string Aciklama { get; set; }
    0 references
    public decimal Fiyat { get; set; }
    1 reference
    public Urun()
    {
        Random r = new Random();
        this.No = r.Next(0, 100000);
    }
}
```

```
private void btnTest_Click(object sender,
{
    Urun o = new Urun();
    o.Adi = "U1";
    MessageBox.Show(o.No.ToString());

    Kitap k = new Kitap();
    k.ISBN = 23232;
    k.Yazar = "XXX";
    k.Adi = "Sindrella";
    MessageBox.Show(k.No.ToString());

    Telefon t = new Telefon();
    t.Marka = "Samsung";
    t.Model = "Galaxy xxx";
    t.Adi = "S6 Edgde";
    MessageBox.Show(t.No.ToString());
}
```

Adım 3 Açıklama

- *Ana sınıfta* **No özelliği** read-only yapılınca *yavru sınıflarda* da bu özellik read-only oldu.
- Sadece Urun sınıfından oluşturulan nesne mi No özellik değeri **aldı**? (Cevap: **Hayır**)
- Kitap ve Telefon sınıflarından oluşturulan nesneler de birer No **özellik değeri** aldılar.
- **Kalıtım** sadece özelliklerin değil aynı zamanda **metotlar ve kurucuların** da *ana sınıftan* (Urun) miras alınarak yavru sınıflara (Kitap, Telefon) aktarılmasını sağlar.

Örnek1: İlk Kalıtım Örneği (devam…)

Adım 4

- Kitap ve Telefon sınıflarına da **Constructor** ekleyelim.
- Nesneler form üzerinde oluşturulduğunda **SIRAYLA** hangi **Constructların** çalıştığını **DEBUG** işlemi yaparak gözlemleyelim.

Kitap
+ISBN: int +Yazar: string
<<Constructor>>+Kitap()

Telefon
+Marka: string +Model: string
<<Constructor>>+Telefon()

Örnek1: İlk Kalıtım Örneği (devam…)

Adım 4

```
Kitap k = new Kitap();
k.ISBN = 23232;
```

Kitap nesnesi yaratılma aşaması.

```
public Kitap()
{
}
```

Kitap Constructor'ın ilk satırına düşer ancak **tamamlamadan** Urun sınıfının **Constructor'ına** gider.

```
public Urun()
{
    Random r = new Random();
    this.No = r.Next(0, 100000);
}
```

Urun sınıfının **Constructor'ı tamamlanır.**

```
public Kitap()
{
}
```

Kitap sınıfının **Constructor'ı tamamlanır.**

Örnek1: İlk Kalıtım Örneği (devam…)

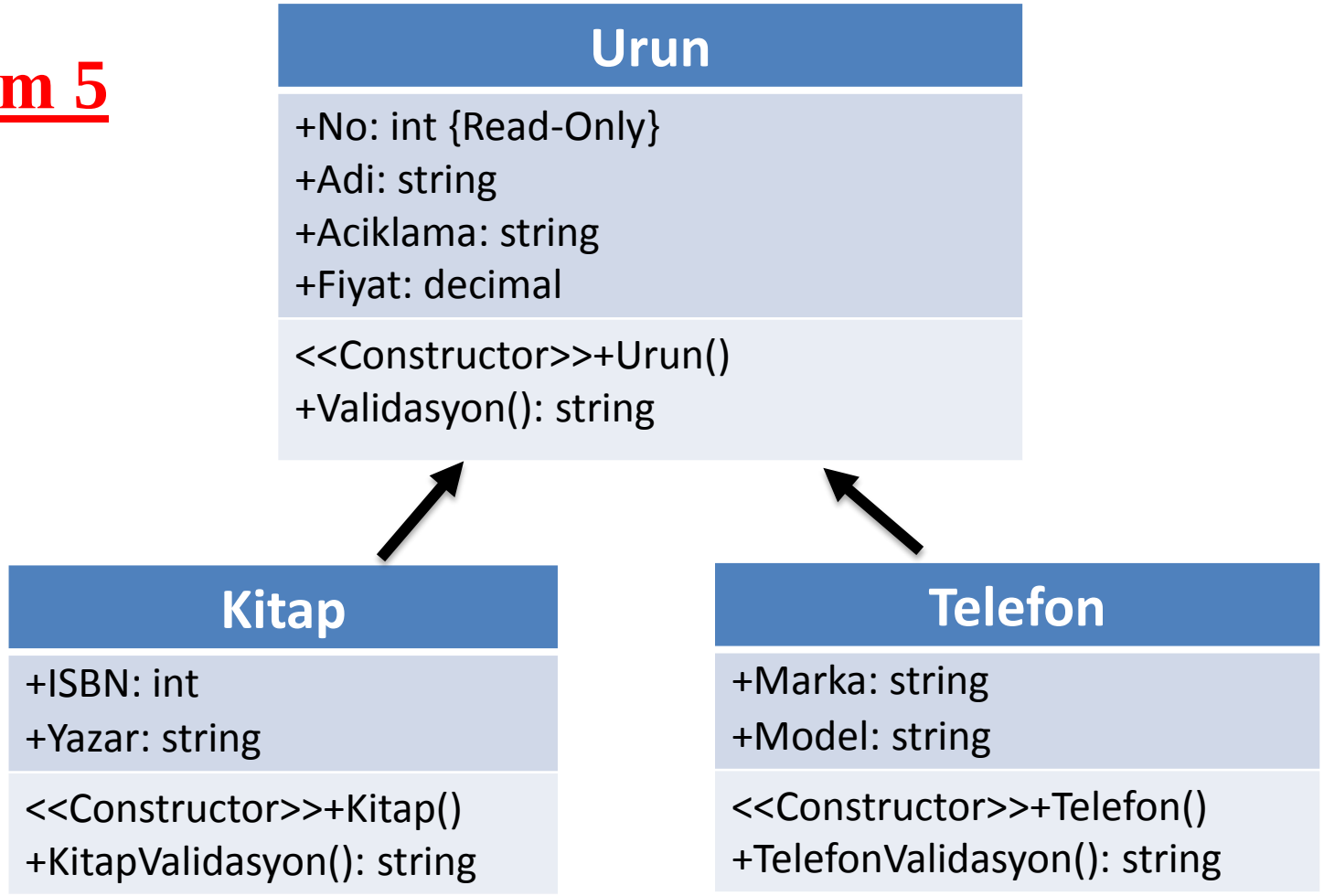
Adım 5

- Bir ürünün **Adı** ve **Fiyatı** boş geçilemez.
  - Bu ürün Kitapsa **ISBN** ve **Yazar** adı,
  - Telefon ise **Model** ve **Marka** özellikleri

  de ayrıca boş olamaz.
- Ürün sınıfına *Validasyon()* isimli bir metot ekleyelim.
- Kitap sınıfına *KitapValidasyon()* isimli bir metot ekleyelim.
  - Temel sınıftan **Validasyon()** metodunu da çağırsın.
- Telefon sınıfına *TelefonValidasyon()* isimli bir metot ekleyelim.
  - Temel sınıftan **Validasyon()** metodunu da çağırsın.

Örnek1: İlk Kalıtım Örneği (devam…)

Adım 5



Örnek1: İlk Kalıtım Örneği (devam…)

Adım 5

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

```
public string KitapValidasyon()
{
    string hataMesaji = "";
    if (this.ISBN == 0)
        hataMesaji += "ISBN özelliği boş olamaz." + Environment.NewLine;
    if ((this.Yazar == "") || (this.Yazar == null))
        hataMesaji += "Yazar özelliği boş olamaz." + Environment.NewLine;
    //Üst sınıfın Validasyon metodu çağrılıyor
    return hataMesaji + this.Validasyon();
}
```

Örnek1: İlk Kalıtım Örneği (devam…)

Adım 5

```
public string TelefonValidasyon()
{
    string hataMesaji = "";
    if ((this.Marka == "") || (this.Marka == null))
        hataMesaji += "Marka özelliği boş olamaz." + Environment.NewLine;
    if ((this.Model == "") || (this.Model == null))
        hataMesaji += "Model özelliği boş olamaz." + Environment.NewLine;
    //Üst sınıfın Validasyon metodu çağrılıyor
    return hataMesaji + this.Validasyon();
}
```

```
Urun o = new Urun();
MessageBox.Show(o.Validasyon());

Kitap k = new Kitap();
MessageBox.Show(k.KitapValidasyon());

Telefon t = new Telefon();
MessageBox.Show(t.TelefonValidasyon());
```

Örnek1: İlk Kalıtım Örneği (devam…)

Adım 6

- Ürün sınıfındaki *Validasyon()* metodunun erişim belirleyicisini **public'ten protected'a** *çekersek* ne olur?
  - **Soru1:** ***Validasyon()*** metodu hala Form üzerinden çağrılabilir mi?
  - **Soru2:** *KitapValidasyon()* bu metodu çağırabilir mi?
  - **Soru3:** *TelefonValidasyon()* bu metodu çağırabilir mi?

Değiştirip Görelim !!!

Örnek1: İlk Kalıtım Örneği (devam…)

Adım 6

- **Soru1:** *Hayır*
- **Soru2:** *Evet*
- **Soru3:** *Evet*

```
3 references
protected string Validasyon()
{
    string hataMesaji = "";
    if ((this.Adi == "") || (this.Adi == null))
        hataMesaji += "Ad özelliği boş olamaz." + Environment.NewLine;
    if (this.Fiyat == 0)
        hataMesaji += "Fiyat özelliği 0 olamaz." + Environment.NewLine;
    return hataMesaji;
}
```

Adım 6 Açıklama

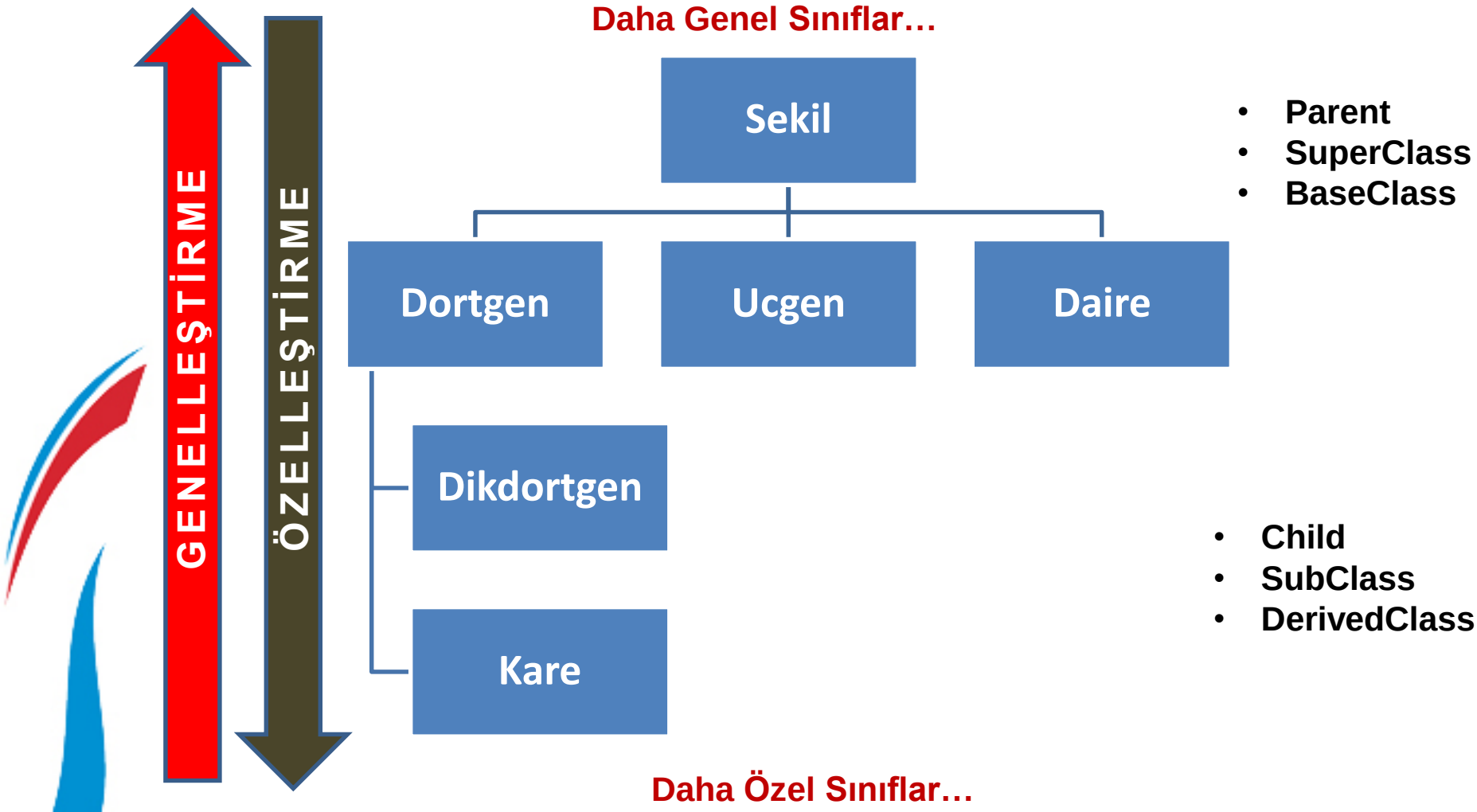
- protected erişim belirleyicisine sahip olan özellikler veya metotlar,
  - Tanımlandıkları sınıfın içerisinde ya da
  - Tanımlı oldukları sınıflardan kalıtımla oluşturulan sınıflar içerisinden erişilebilirler.
- Bu sınıfların **dışında** erişilemezler.
- Diğer bir deyişle, protected erişim belirleyicisine sahip **üyeler** ailenin içerisinde (ana - yavru) *erişilebilirler*.

Sınıfların Genişletilmesi (devam...)

- Sınıftan türetilerek yeni bir sınıf oluşturulmasını engellemek için sınıf başlığı `sealed` anahtar sözcüğüyle tanımlanır. Hazır olarak gelen String sınıfı sealed sınıflara örnek olarak verilebilir.

```
sealed class sinif
{
}
```

Sınıfların Genişletilmesi (devam...)



- Parent
- SuperClass
- BaseClass

- Child
- SubClass
- DerivedClass

Kalıtım Avantajları

- Kalıtımı kullanabilme kabiliyeti, programı
  - daha kolay ve az kod yazma,
  - daha kolay anlama ve
  - daha az hata ile karşılaşmayı sağlamaktadır.
- Kalıtımı kullanarak, düzgün bir şekilde, hızlıca yeni sınıflar yaratılabilir.

Yararlanılan Kaynaklar

- Sefer Algan, HER YÖNÜYLE C# , Pusula Yayıncılık, İstanbul, 2003
- Milli Eğitim Bakanlığı, «Nesne Tabanlı Programlama», 2012
- Joyce Farrel, An Introduction to Object - Oriented Programming, Cengage Learning, 2011
- http://www.AlgoritmaveProgramlama.com

İYİ ÇALIŞMALAR…

Yrd. Doç. Dr. Deniz KILINÇ

deniz.kilinc@cbu.edu.tr