NESNE YÖNELİMLİ PROGRAMLAMA(Object Oriented Programming/OOP)

Öğr. Gör. Celil ÖZTÜRK
Marmara Üniversitesi
Teknik Bilimler Meslek Yüksekokulu

İçerik

- ✓ Sınıflar
- ✓ Nesneler
- ✓ Kalıtım(Inheritance),Miras Alma
- ✓ UML Diyagramları
- ✓ This Referans Kelimesi
- ✓ Kapsülleme(Encapsulation)

Nesne Yönelimli Programlama ve Kavramlar

- Java dilinde modül veya altprogramlar, sınıf (class) ve yöntemler'den (method) oluşur. Nesne yönelimli programlamanın (Object Oriented Programming – OOP) temelini de, sınıf ve bu sınıfların örneklendirilmiş hali olan nesneler oluşturur.

SINIF(Class)

Sınıf, birşeyin soyut özellik ve davranışlarını tanımlar. Örneğin “Araba” sınıfının, tüm arabalarda ortak olarak görülen özellik (kapasite, renk, motor gücü vb.) ve davranışları (çalışmak, durmak) içermesi beklenir. Özellikler (Properties) ile davranışları tanımlayan yöntemlere (methods) topluca sınıf üyeleri (Class members) denir. Sınıflar nesne yönelimli programlamanın modülerliğini sağlayan en önemli öğeler konumundadır. Sınıf tanımlamak için aşağıdaki yapı kullanılır.

SINIF(Class)

- Sınıflar, verileri tek tek değişkenlerde tutmak yerine, birbirleri ile ilişkili verileri bir araya getirmek ve bunlara tek bir referans aracılığıyla erişmeyi sağlamaktadır.
- Örnek; Taşıt bir sınıftır. Kişi de bir sınıftır. Çalışan da bir sınıftır. Müşteri de bir sınıftır.
- Ad, soyad, müşteri numarası, çalıştığı departman gibi bilgileri tutmak istediğimiz bir Müşteri sınıfı tanımlanabilir.
- Sınıf kavramı, nesneler hakkında bilgi sağladığı için kullanışlıdır.
- Nesneler, özelliklerini ve davranışlarını(metotlar) Sınıflardan alır.

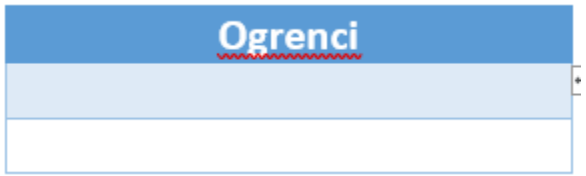
SINIF(Class)

Sınıflar, program içinde kullanılacak değişken ve metot grupları için kullanılan bir şablon oluşturur.

class Sınıf_adı {

……..// Metot ve işlemler

}



Nesne (Object) Kavramı

- Bir sınıf tanımlandıktan sonra aslında yeni bir veri tipi tanımlanmış olunur. Java dilinde oluşturulan bu yeni veri tipine sahip değişkenlere «nesne» adı verilir.
- Nesne daha genel bir sınıfın somutlaşmış örneğidir.
- Benim arabam, Araba sınıfının somutlaşmış bir örneğidir.
- Bir nesne, bulunduğu sınıfın somutlaşmış bir örneğidir.
- Evdeki hayvanımız, aile bireylerimiz, sahip olduğumuz araçlar, sahip olduğumuz Kripto paralar da birer nesnedir.
- Ayrıca olaylar da birer nesne olabilir.
- Nesneler, özelliklerini ve davranışlarını Sınıflardan alırlar.

Nesne (Object) Kavramı

- Nesne yönelimli programlamada, nesneler bir sınıfı temel alarak oluşturulur. Bir sınıfın bir nesne oluşturma işlemine örnekleme (instantiating) denir.
- Sınıflar, temelde yalnızca nesne tanımlamalarıdır. Hafızada yer almaları ancak örneklendirilmeleri ile mümkündür. Bir başka ifade ile sınıflar örneklendirildiklerinde isimleri nesne olur diyebiliriz.
- Nesne yönelimli programlamada her nesne kendi içinde bağımsız bir bütündür. Bir nesnenin diğer nesneler ile ilişkisi mesaj göndermekten ibarettir. Her nesne kendi ile ilgili verileri kendi içinde saklar, bazı verilere dışarıdan erişime izin vermeyebilir.

Nesne (Object) Kavramı

- Bir sınıftan new komutu ile birlikte bir kopya oluşturulduğunda bu kopyaya nesne (object) denir. Bir sınıftan aynı anda, birbirinden bağımsız birçok sayıda nesne oluşturulabilir.
- Nesne tanımlamak için aşağıdaki yapı kullanılır:

Sınıf_adı Nesne_adı= new Sınıf_adı();

Örneğin Araba sınıfından kamyon adlı bir nesne tanımlamak için aşağıdaki yapı kullanılır:

Araba kamyon = new Araba();

Bir nesne değişkeninin o an için hiçbir nesneyi işaret etmediğini belirtmek için null değeri kullanılır.

```
Kamyon=null;
 ...
if (kamyon!=null) kamyon .calis();
```

Erişim Belirteçleri

- Bir sınıf veya nesnenin üyelerine erişim için dört etiket kullanılır.
- Public (Açık): Public üyelere programdaki herhangi bir fonksiyon tarafından erişilebilir. Dolayısıyla public üye veya alanlara farklı paket ve sınıflardan erişilebilir.
- Private (Özel): Bir sınıfın içinde tanımlanan private üyeye sadece o sınıfın içinden erişilebilir. Dolayısıyla private üye veya alanlara, aynı veya farklı paketten, farklı sınıflardan erişilemez.
- Protected (Korumalı): Protected tanımlanmış bir üyeye aynı paket ve sınıf içerisinden erişilebilir fakat paket dışarısından yalnız kalıtım ile oluşturulmuş alt sınıflar erişilebilir.
- Default: Bu üç belirteçten (public, private, protected) hiçbiri kullanılmamışsa default olarak tanımlanmış demektir. Default üyelere sadece tanımlandığı paket içerisinden erişilebilir.

Metot(Yöntem/Davranış)

Bir nesnenin yeteneklerine (kabiliyetlerine) yöntem veya metot adı verilir. Her bir yöntemi nesnenin yapabileceği bir davranışı simgeler. ÖRNEĞİN Kamyon, Araba sınıfına ait bir nesne olarak, o sınıfta tanımlı davranışları sergiler, yani çalışır, durur. Diğer bir ifade ile “çalışmak” Kamyon’un bir yöntemidir.

Bir metot tanımlamak için aşağıdaki yapı kullanılır:

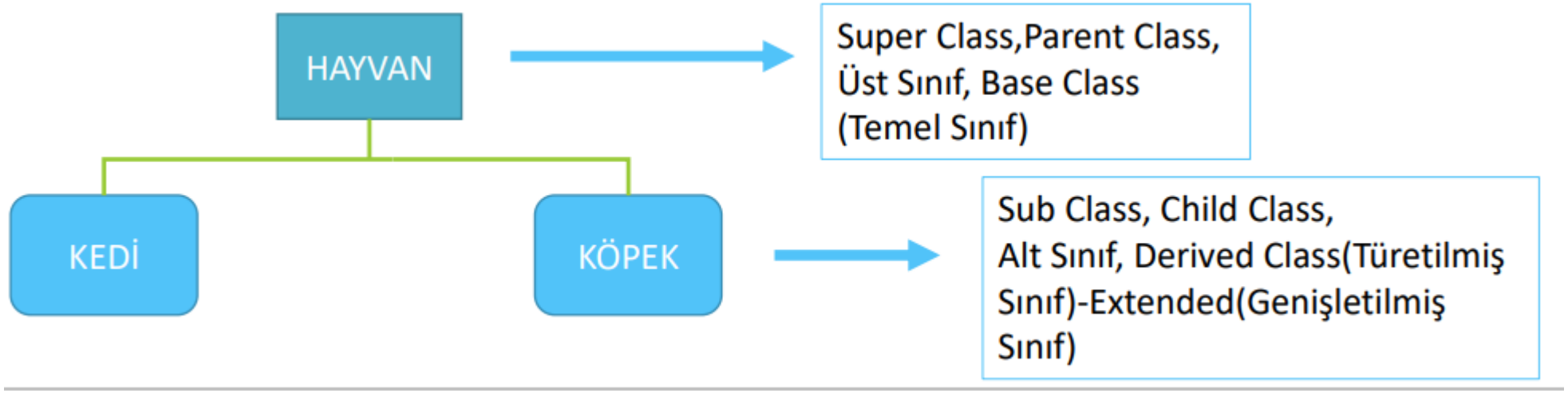
Nesne.Metot_adı();

KALITIM(Inheritance)***

- Miras alma olarak da isimlendirilen kalıtım eğer bir sınıf, bir üst (taban veya ebeveyn de denilebilir) sınıftan türetilirse yeni sınıf (alt veya çocuk sınıf olarak adlandırılabilir) bu üst sınıfın bütün özelliklerine ve metotlarına sahiptir.
- Kalıtım ile özellikleri miras alan sınıf, atasının özelliklerini taşır. Bu durumda miras alınan sınıfa üst sınıf (super-class), miras alan sınıfa ise alt sınıf (sub-class) adı verilir.
- Her alt sınıf ilerde bir üst sınıf olma adayıdır.
- Bir alt-sınıf, üst-sınıfından taşıdığı özelliklere ve işlevlere ek olarak; kendine ait özellikleri ve işlevleri içerebilir. Bir alt-sınıf aynı zamanda, üst-sınıfından taşıdığı işlevleri değiştirebilir.

KALITIM(Inheritance)***

- Hayvan sınıfı, kedi ve köpek sınıfındaki ortak özellikleri görememizi sağlar. Hayvan sınıfı, Kedi ve Köpek türleri ortak olarak ad, cinsiyet değerini alabilir. Aynı şekilde farklı hayvan türleri de bu nitelikleri alabilir.
- Hayvan sınıfı, temel alınarak Kedi ve Köpek sınıfları bu sınıftan türetilmelidir.



KALITIM(Inheritance)***

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.

KALITIM(Inheritance)***

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.

KALITIM(Inheritance)***

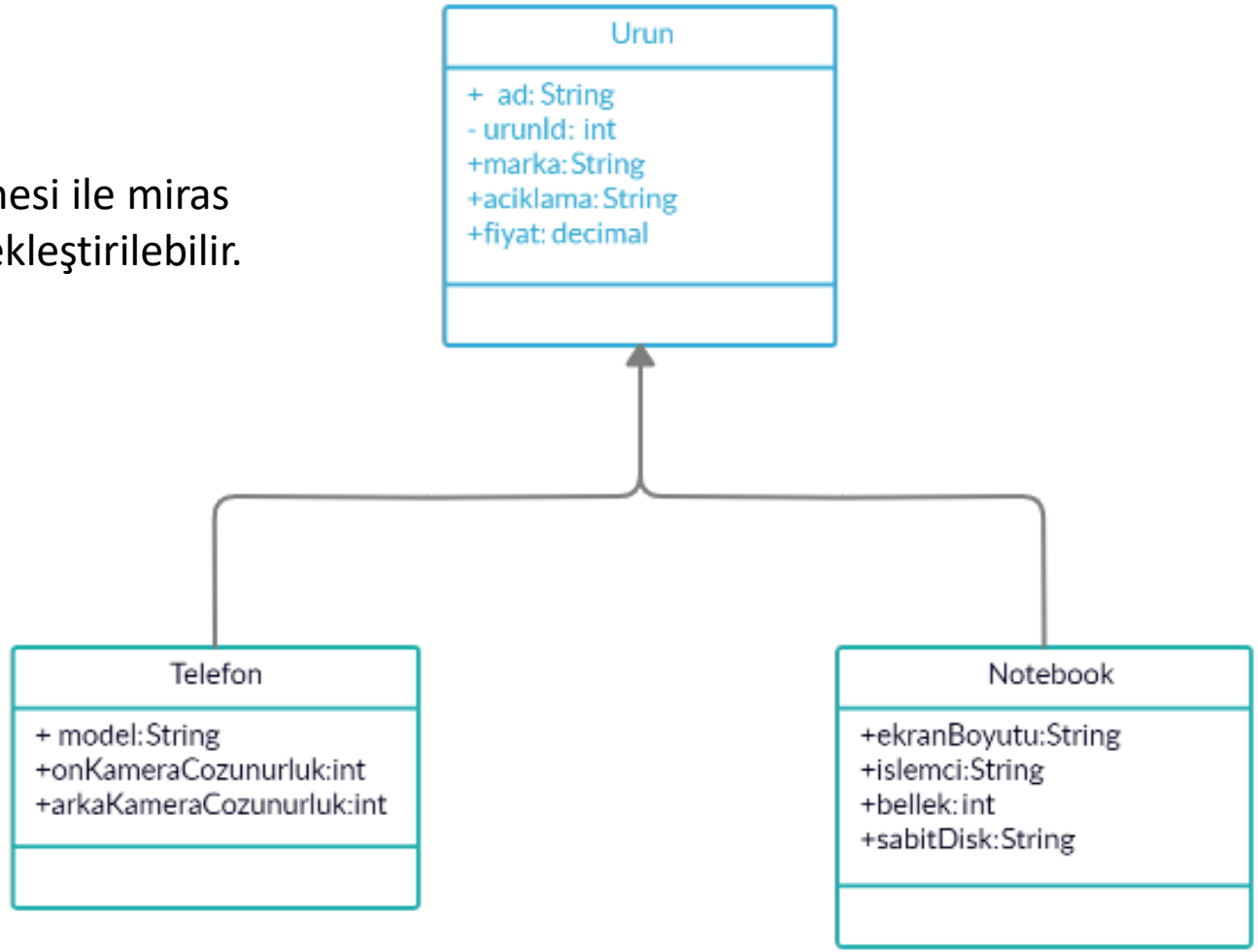
Kalıtım tek yönlü çalışmaktadır:
– Yavru sınıf, ana sınıftan kalıtılarak oluşturulur, ters yönde oluşturulamaz.
– Program içerisinde bir Urun nesnesi oluşturduğunuzda Kitap sınıfının özelliklerine veya metotlarına erişemez.

**-protected erişim belirleyicisine sahip olan özellikler veya metotlar,

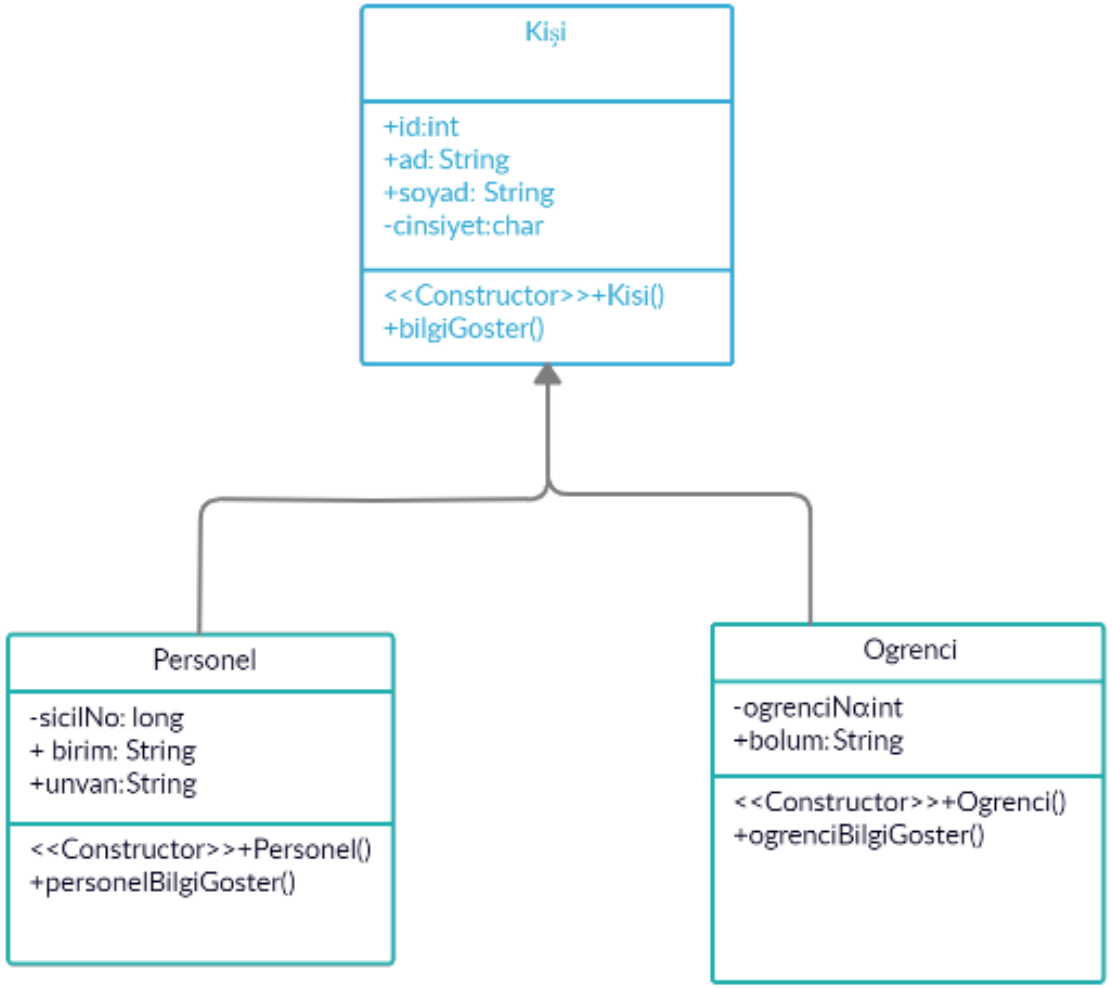
- Tanımlandıkları sınıfın içerisinde ya da
- Tanımlı oldukları sınıflardan kalıtımla oluşturulan sınıflar içerisinden erişilebilirler.
- Bu sınıfların dışında erişilemezler.
- Diğer bir deyişle, protected erişim belirleyicisine sahip üyeler ailenin içerisinde (ana - yavru) erişilebilirler.

KALITIM(Inheritance)***

***Extends kelimesi ile miras alma işlemi gerçekleştirilebilir.



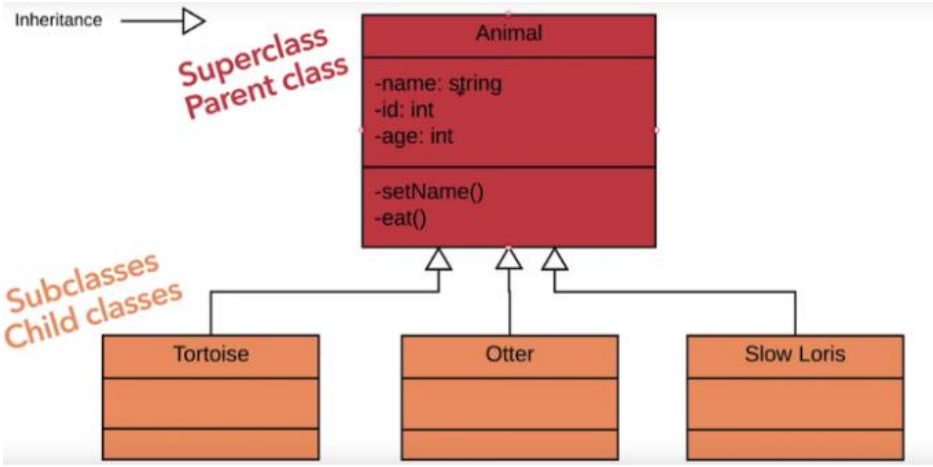
KALITIM(Inheritance)***



KALITIM(Inheritance)***

Java programlama dilinde, sınıfların türetilmesini engellemek için(uninherited) sınıf başlığı «**final**» olarak tanımlanır. C# üzerinde «sealed» anahtar sözcüğüyle kalıtım engellenebilir.

```
final class Ogrenci{
……..//
}
```



UML Diyagramları(Unified Modelling Language)

Sınıf Adı
Özellikler
Yöntemler/Davranışlar

Nesne yönelimli programlama dillerinde analiz ve tasarım sürecinde modelleme için, çoğunlukla diyagramlardan oluşan UML (Unified Modeling Language – Tümleştirilmiş Modelleme Dili) dilinden yararlanılır. Bir sınıfın UML gösterimi şekildeki gibi üç bölümü olan bir kutu biçimindedir. En üstteki bölümde, sınıfın adı bulunur. Ortada sınıfın özellikleri yer alır. Alttaki kısmında ise, yöntem ve işlemlerle ilgili tanımlar bulunur

This

- Bir sınıf yaratıldığında, sınıfın kaynak kodunun bir kopyası bilgisayarın hafızasında (memory) saklanmaktadır.
- Bununla birlikte, er ya da geç bu sınıftan defalarca nesne oluşturulacaktır.
- Her yeni nesne oluşturduğunuzda, nesnenin her üye değişkeni için bellekte saklama alanı sağlanır.
- this referansı ile sınıfın o anki nesnesine ulaşabilirsiniz.

This

- this referansı kodda karışıklığı engellemek için de kullanılabilir.
- Örneğin: Sınıfın bir üyesi veya özelliği ile sınıfın metoduna aktarılan parametrenin aynı isimde olması durumunda karışıklığı önlemek için this referansı kullanılabilir.

This

Burada karışıklıkları önlemek için this referansı kullanılır.

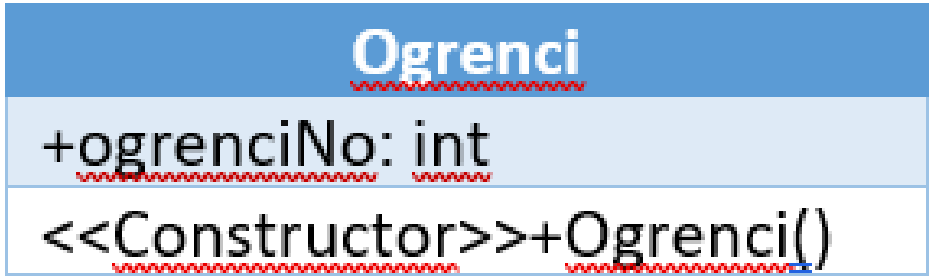
this.fiyat=fiyat

eşitliğinin sol tarafındaki fiyat değişkeni sınıfın özelliğidir. Sağdaki ise metod parametresidir.

```
package thisreferansi;

public class Test {
    private double fiyat;

    public void FiyatHesaplama(double fiyat)
    {
        this.fiyat =fiyat;
    }
}
```

Kurucu Metotlar(Constructor)

- Bir nesnenin oluşmasını sağlayan metotlardır.
- Ogrenci öğrenci = new Ogrenci(); burada Ogrenci() →Java üzerinde Ogrenci sınıfındaki bir metotdur !! ****.
- Constructor metotlar eğer yazılmaz ise derleme esnasında otomatik olarak oluşturulurlar.(Parametre almayan,gövdesi boş bir metot olarak)



***Shortcut **

Alt+Insert →Constructor, Setter, Getter ….

Kapsülleme(Encapsulation)

- Farklı kaynaklarda paketleme veya sarmalama olarak da isimlendirilen kapsülleme, bir sınıfın içeriğinin, onun üyelerini kullananlar tarafından bilinmesine gerek duymadan sadece metodun verdiği hizmetin gösterilmesi işlemidir.
- Her nesne, belli verileri tutar ve belli işlevler görür. Bir sınıf aslında başka sınıfların kullanımı için çeşitli özellik ve metotlar barındıran bir yapıdır. Ancak bir sınıftaki bütün özellik ve metotların dışarıdan bilinmesi veya kullanılması gerekmez. Hatta bazı özellik ve metotlara erişim sınıfın sağlamlığı açısından tehlikeli olabilir.
- Kapsülleme ile bir sınıf, kendi iç bütünlüğünü gizleyebilir ve koruyabilir. Bir sınıfın dışarıdan sadece gereken özellik ve metotlarıyla görülmesi ayrıca basitlik de sağlamaktadır. Kapsülleme, bir sınıfı kullanacak kişinin, ilgilendiği üyelerle sadece sınıfın kendisini ilgilendiren üyeleri bir arada görmesini de engeller.

Kapsülleme(Encapsulation)

- Why Encapsulation ?
- Sınıf özniteliklerinin ve yöntemlerinin daha iyi kontrolü
- Sınıf nitelikleri salt okunur (read-only)(yalnızca getyöntemi kullanıyorsanız) veya salt yazılabilir(write-only) (yalnızca setyöntemi kullanıyorsanız) yapılabilir
- Esnek: programcı, diğer bölümleri etkilemeden kodun bir bölümünü değiştirebilir
- Artan veri güvenliği

Kapsülleme(Encapsulation)

```java
public class Ogrenci {
    private int ogrenciNo;

    public Ogrenci() {

    }

    public int getOgrenciNo() {
        return ogrenciNo;
    }

    public void setOgrenciNo(int ogrenciNo) {
        this.ogrenciNo = ogrenciNo;
    }
}
```

Ogrenci
-ogrenciNo: int
<<Constructor>>+Ogrenci() +getOgrenciNo():int +setOgrenciNo(int):void

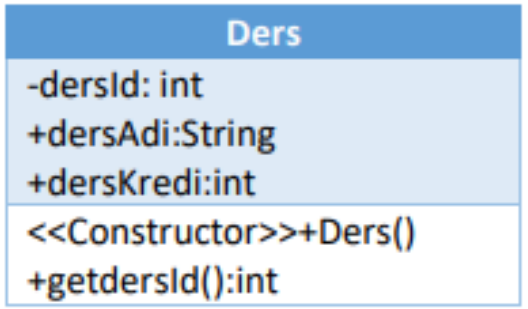
***Shortcut **

Alt+Insert →Constructor, Setter, Getter ….

Kapsülleme(Encapsulation)

```
public class ThisReferansi {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Ogrenci ogrenci1 = new Ogrenci();
        ogrenci1.setOgrenciNo(2020010);
        System.out.println(ogrenci1.getOgrenciNo());
    }
}
```

Kapsülleme(Encapsulation)



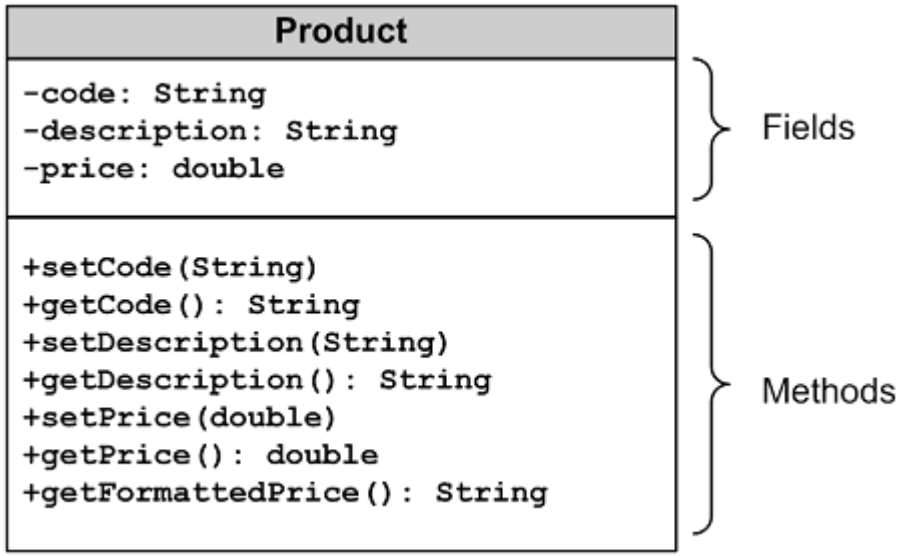
Read Only

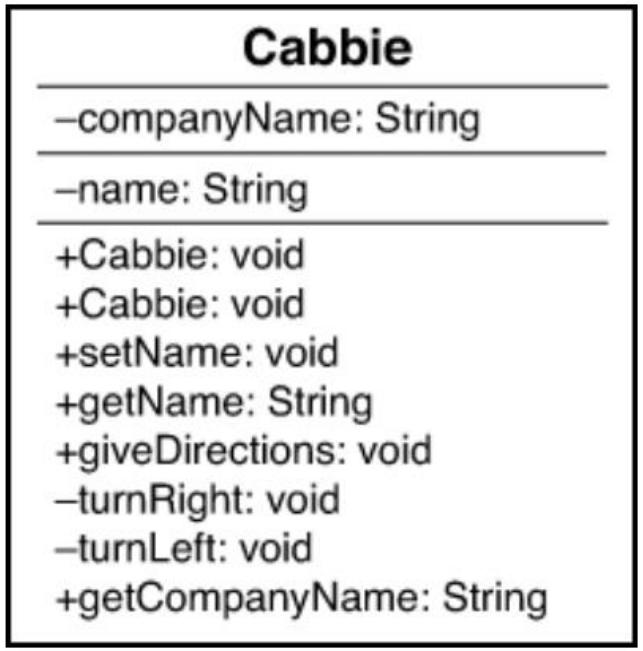
```
public class Ders {
    private int dersId;
    public String dersAdi;
    public int dersKredi;

    public Ders() {
        dersId=(int) (Math.random()*10000); //Rastgele ders numarası oluşturur.
         //Burada belirli bir sıraya göre de numara verilebilir !!
    }

    public int getDersId() {
        return dersId;
    }

}
```

***Shortcut **

Alt+Insert →Constructor, Setter, Getter ….

Kapsülleme(Encapsulation)/Farklı Örnekler





http://www.c-jump.com/bcc/c157c/Week07/W07_0070_class_diagram_example.htm

https://www.informit.com/articles/article.aspx?p=2210836

Kapsülleme(Encapsulation)/Farklı Örnekler

- https://www.coursera.org/lecture/object-oriented-design/1-2-8-encapsulation-in-java-and-uml-oVyNM

- https://www.coursera.org/lecture/object-oriented-design/1-2-10-generalization-with-inheritance-in-java-and-uml-FUucE

Kapsülleme(Encapsulation)/Farklı Örnekler

```
class TV{
 private int boy;
private int en;
private int yukseklik;
public void setUz(int p)
{boy=p;}
public void setEn(int p)
{en=p;}
public void setYuk(int p)
{yukseklik=p;}
public int SesAyari()
{return (boy*en*yukseklik);}
}
```

```
public class TvSesi {
public static void main(String[] args) {
TV obl=new TV();
double ses;
obl.setUz(120);
obl.setEn(100);
obl.setYuk(90);
ses=obl.SesAyari();
System.out.println("Ses Ayarı..:" + ses);}}
```

http://www.c-jump.com/bcc/c157c/Week07/W07_0070_class_diagram_example.htm

https://www.informit.com/articles/article.aspx?p=2210836

Kaynaklar

- Java ve Java Teknolojileri, *Tevfik KIZILÖREN* – Kodlab Yayınları
- Dr Öğr. Üyesi Aysun Zehra ALTIKARDEŞ- Nesne Yönelimli Programlama Notları
- Yrd. Doç. Dr. Deniz KILINÇ Celal Bayar Üniversitesi Nesneye Yönelik Programlama Ders notları
- https://medium.com/@tugrulbayrak/uml-class-diyagramlari-4c3bb7e9cc4c