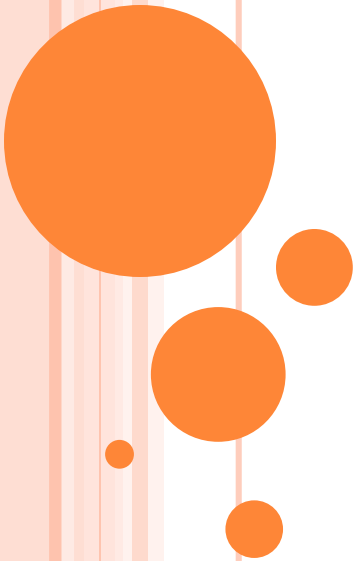
PAKET ERİŞİMLERİ SINIFLARIN YENİDEN KULLANIMI



Paket (*Package*)

- Paketler kütüphaneleri oluşturan elemanlardır.
- import java.io.BufferedReader;
- *BufferedReader* sınıf isminin *java.io* paketinde tek olduğunu anlıyoruz. Fakat, başka paketlerin içerisinde *BufferedReader* sınıf ismi kullanılabilir.

- Paketin içerisindeki tek bir sınıfı kullanmak yerine ilgili paketin içerisindeki tüm sınıfları tek seferde kullanmak için:
  - import java.io.* ;

VARSAYILAN PAKET (*DEFAULT PACKAGE*)

```
public class Test1 {
 public void kos() {
  }
}

class Test2 {
   public void kos()
    {
  }
}
```

Test1.java

Dosya derlendiğinde adları *Test1.class* ve *Test2.class* olan 2 adet fiziksel *.class* dosyası elde edilir.

Test1.java dosyanın en üstüne herhangi bir paket ibaresi yerleştirilmediğinden dolayı Java bu sınıfları varsayılan paket (*default package*) olarak algılayacaktır.

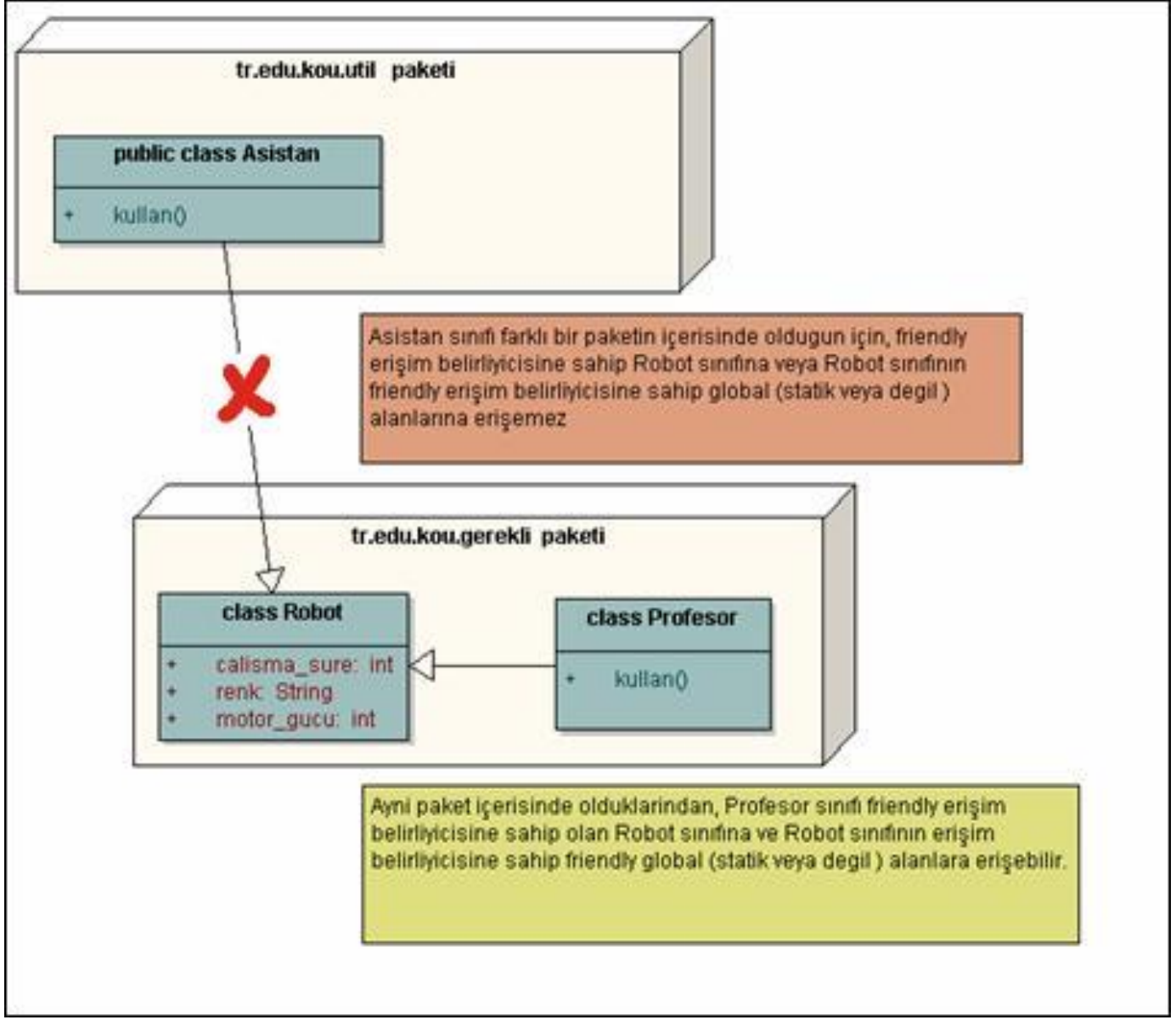
Paket Oluşturma

- Paket oluşturmanın temel amaçlarından birisi, aynı amaca yönelik iş yapan sınıfları bir çatı altında toplamaktır;
- Yazılan sınıflar daha derli toplu olur, aranılan sınıflar daha kolay bulunabilir.

```
package tr.edu.turgutozal.yazmuh.yazm209;
 public class Lab1 {
...
```

ERİŞİM BELİRLEYİCİLER

- Java dilinde 4 tür erişim belirleyicisi vardır: friendly, public, protected, private.
- friendly
  - global alanlara (statik veya değil), yordamlara (statik veya değil) ve sınıflara atanabilir.
  - friendly global alanlar (statik veya değil) içerisinde bulundukları paketin diğer sınıfları tarafından erişilebilirler.
  - Fakat, diğer paketlerin içerisindeki sınıflar tarafından erişilemezler. Yani, diğer paketlerin içerisindeki sınıflara karşı private erişim belirleyici etkisi oluşturmuş olurlar.



Bir global alan veya sınıf friendly yapılmak isteniyorsa önüne hiç bir erişim belirleyicisi konulmaz.

PUBLİC (HERKESE AÇIK)

- public erişim belirleyicisi sahip olabilen sınıflar, global alanlar ve yordamlar herkes tarafından erişilebilir.
- Bu erişim belirleyicisine sahip olan global alanlar veya yordamlar herhangi bir yerden doğrudan çağrılabildiklerinden dolayı dış dünya ile arasındaki arabirim rolünü üstlenirler. ()

PRİVATE (ÖZEL)

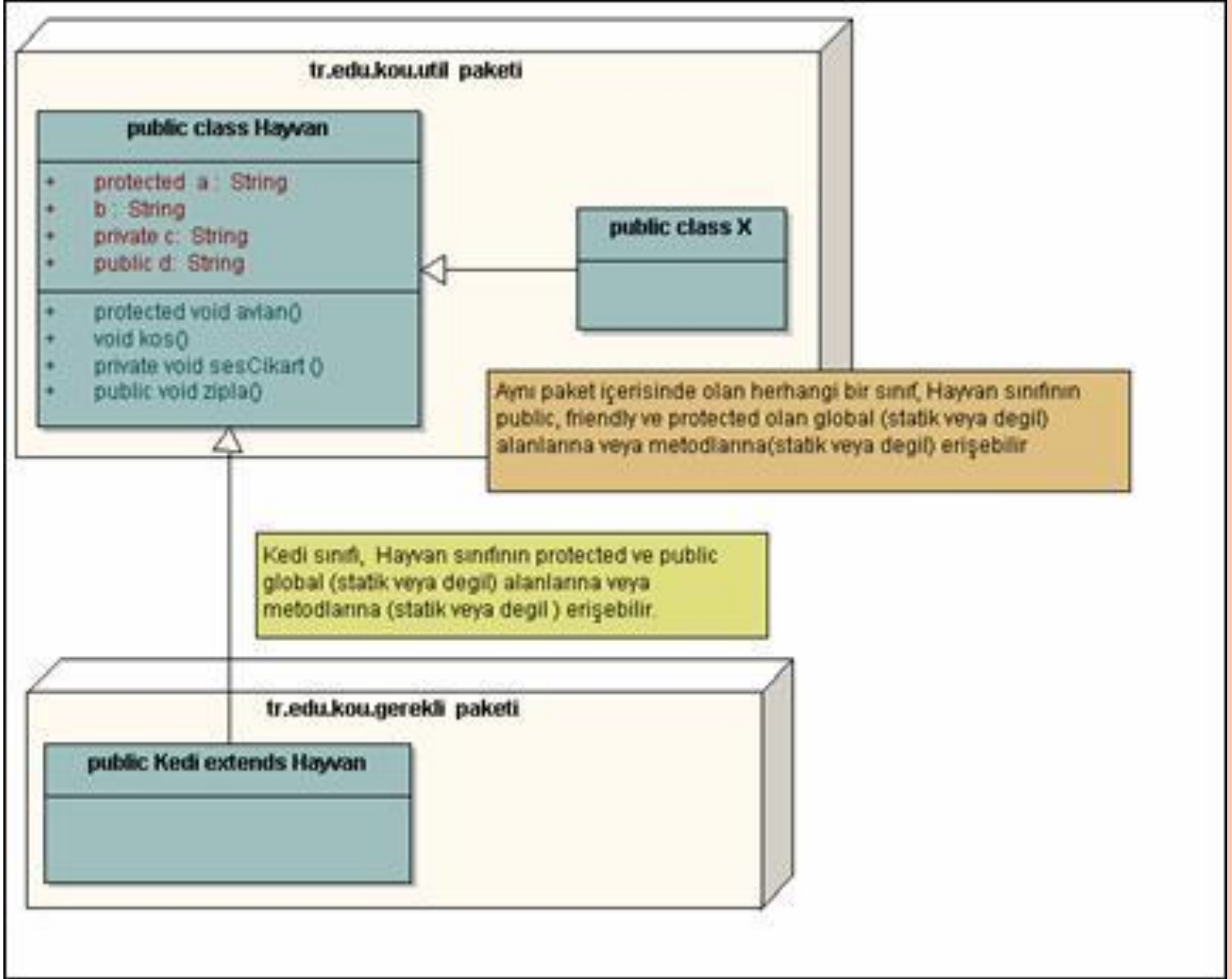
- private olan global alanlara veya yordamlara (sınıflar private olamazlar; dahili sınıfar-*inner class* hariç) aynı paket içerisinden veya farklı paketlerden erişilemez.
- Ancak ait olduğu sınıfın içinden erişilebilir.
- private belirleyicisine sahip olan yordamların içerisinde devamlı değişebilecek /geliştirilebilecek olan kodlar yazılmalıdır.

```
class Kahve {
    private int siparis_sayisi;

    private Kahve() {
    }

    private void kahveHazirla() {
      System.out.println(siparis_sayisi
       + " adet kahve hazirlandi");
    }


    public static Kahve siparisGarson(int sayi) {
     Kahve kahve = new Kahve(); //dikkat
     kahve.siparis_sayisi = sayi ;
     kahve.kahveHazirla();
     return kahve;
    }
}
```

```
public class Musteri {

        public static void main(String args[]) {
            // Kahve kh = new Kahve() ;   //  Hata !
            // kh.kahveHazirla() ;        //  Hata !
              // kh.siparis_sayisi = 5 ;     //  Hata !
            Kahve kh = Kahve.siparisGarson(5);
        }
}
```

PROTECTED (KORUMALI ERİŞİM)

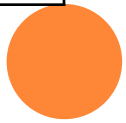
- Sadece global alanlar ve yordamlar protected erişim belirleyicisine sahip olabilirler.
- Sınıflar protected erişim belirleyicisine sahip olmazlar (dahili sınıflar-inner class hariç);
- protected erişim belirleyicisine sahip alanlara sadece aynı paketin içerisindeki sınıflar erişebilir.


tr.edu.kou.util paketi
public class Hayvan
+ protected a : String
+ b : String
+ private c: String
+ public d: String
+ protected void avlan()
+ void kos()
+ private void sesCikart ()
+ public void zipla()
public class X
Aynı paket içerisinde olan herhangi bir sınıf, Hayvan sınıfının public, friendly ve protected olan global (statik veya degil) alanlarına veya metodlarına(statik veya degil) erişebilir
Kedi sınıfı, Hayvan sınıfının protected ve public global (statik veya degil) alanlarına veya metodlarına (statik veya degil) erişebilir.
tr.edu.kou.gerekli paketi
public Kedi extends Hayvan

KAPSÜLLENME (*ENCAPSULATİON*)

- Nesneye yönelik programlama özelliklerinden birisi kapsüllenmedir;
- Dışarıdaki başka bir uygulamanın bizim nesnemiz ile sadece arabirimler (public) sayesinde iletişim kurması gerektiğini,
- ancak, arka planda işi yapan esas kısmın gizlenmesi gerektiğini söyler.

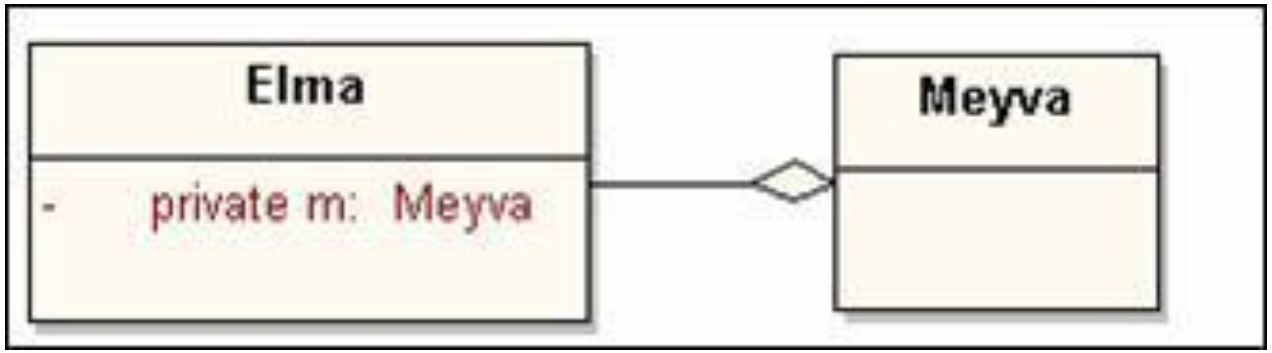
	Aynı Paket	Ayrı Paket	Ayrı paket-türetilmiş
`public`	erişebilir	erişebilir	erişebilir
`protected`	-	-	-
`friendly`	erişebilir	erişemez	erişemez
`private`	-	-	-



SINIFLARIN TEKRAR KULLANILMASI

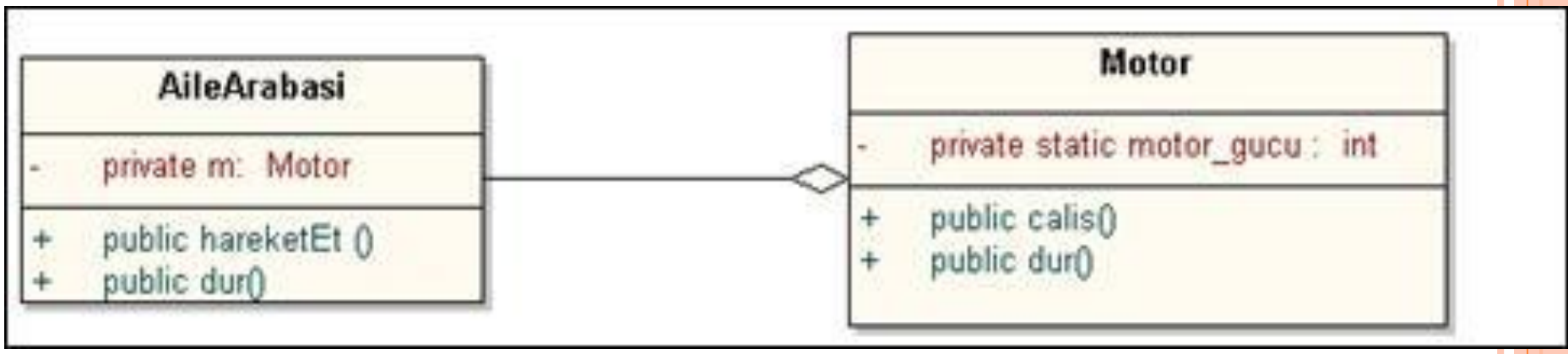
- **Komposizyon**

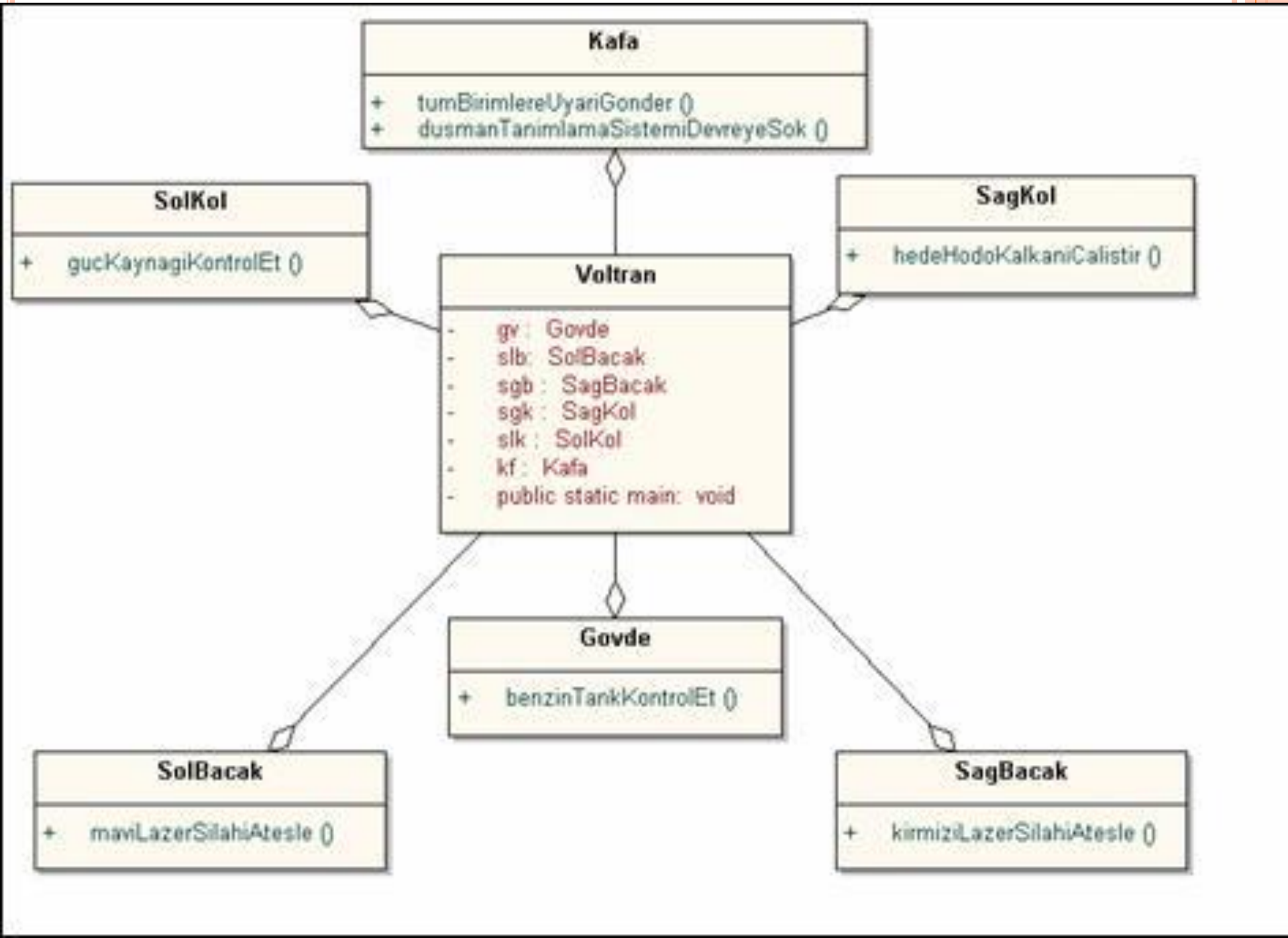
```
class Meyva {
  //...
}
class Elma {
  private Meyva m = new Meyva();
  //...
}
```



```
public class Motor {
  private static int motor_gucu = 3600;

  public void calis() {
   System.out.println("Motor Calisiyor") ;
  }

  public void dur() {
   System.out.println("Motor Durdu") ;
  }
}
```

```
public class AileArabasi {
  private Motor m = new Motor();
  public void hareketEt() {
    m.calis();
    System.out.println("Aile Arabasi Calisti");
  }
  public void dur() {
    m.dur();
    System.out.println("Aile Arabasi Durdu");
  }
  public static void main(String args[]) {
    AileArabasi aa = new AileArabasi() ;
    Aa.hareketEt();
    Aa.dur();
  }
}
```


AileArabasi
- private m: Motor
+ public hareketEt ()
+ public dur()
Motor
- private static motor_gucu : int
+ public calis()
+ public dur()


Kafa
+ tumBirimlereUyariGonder ()
+ dusmanTanimlamaSistemiDevreyeSok ()
SolKol
+ gucKaynagiKontrolEt ()
SagKol
+ hedeHodoKalkaniCalistir ()
Voltran
- gv : Govde
- slb: SolBacak
- sgb : SagBacak
- sgk : SagKol
- slk : SolKol
- kf : Kafa
- public static main: void
Govde
+ benzinTankKontrolEt ()
SolBacak
+ maviLazerSilahiAtesle ()
SagBacak
+ kirmiziLazerSilahiAtesle ()

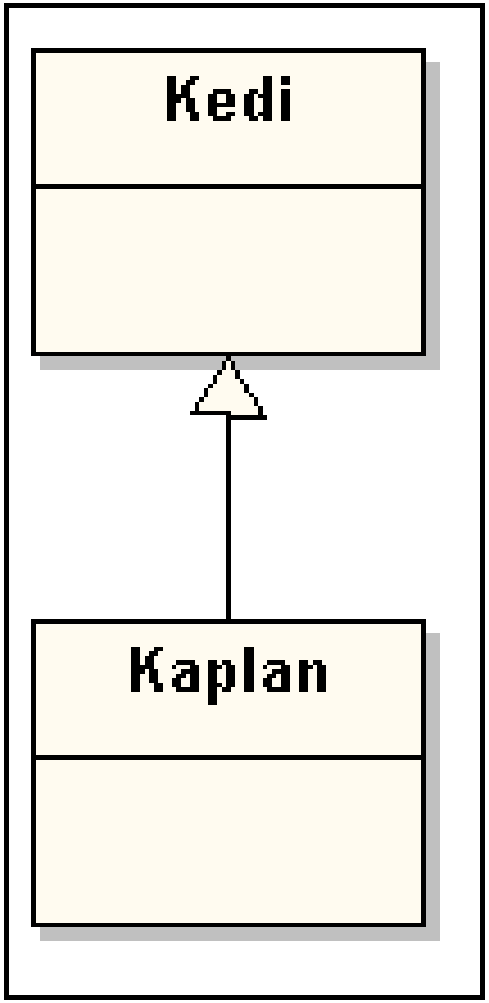
KALITIM

- Kalıtım bir sınıftan diğer bir sınıfın türemesidir.
- Yeni türeyen sınıf, türetilen sınıfın global alanlarına ve yordamlarına (statik veya değil) otomatik olarak sahip olur (*private* olanlar hariç).

```
class Kedi {
  //..
}

class Kaplan extends Kedi {
 //..
}
```

```
public class Hayvan {
    protected String a = "Hayvan.a";
    String b = "Hayvan.b"; //friendly
    private String c = "Hayvan.c";
    public String d = "Hayvan.d"; ;
}

public class Kedi extends Hayvan { // Türeyen
  public Kedi() {
    System.out.println("Kedi olusturuluyor");
    System.out.println(a);
    // System.out.println(b); // ! Hata ! erisemez ?
    // System.out.println(c); // ! Hata ! erisemez ?
    System.out.println(d);
  }

public static void main(String args[]) {
    Kedi k = new Kedi();
  }
}
```

Kedi olusturuluyor
Hayvan.a
Hayvan.d


Kedi
Kaplan

```
class Kedi {
    protected int ayakSayisi = 4 ;
    public void yakalaAv() {
            System.out.println("Kedi sinifi Av yakaladi");
    }
    public static void main(String args[]) {
            Kedi kd= new Kedi() ;
            kd.yakalaAv() ;
    }
}
class Kaplan extends Kedi {
    public static void main(String args[] ) {
         Kaplan kp = new Kaplan();
         kp.yakalaAv();
        System.out.println("Ayak Sayisi = "+kp.ayakSayisi);}
     }
}
```

Gizli Kalıtım

- Oluşturduğumuz her yeni sınıf otomatik ve gizli olarak *Object* sınıfından türer.
- *Object* sınıfı Java programlama dili içerisinde kullanılan tüm sınıfların tepesinde bulunur.

```
public class YeniBirSinif extends Object {
}
```

```
 public class YeniBirSinif {
  public static void main(String[] args) {
  YeniBirSinif ybs1 = new YeniBirSinif();
     YeniBirSinif ybs2 = new YeniBirSinif();
     System.out.println("YeniBirSinif.toString()" + ybs1 ) ;
     System.out.println("YeniBirSinif.toString()" + ybs2 ) ;
     System.out.println("ybs1.equals(ybs2)"+ybs1.equals(ybs2)) ;
     // ....
  }
}
```

```
YeniBirSinif.toString() YeniBirSinif@82f0db
YeniBirSinif.toString() YeniBirSinif@92d342
ybs1.equals(ybs2) false
```

```
public class Person
{
    private String name;
    public Person()
    {
        name = "No name yet.";
    }
    public Person(String initialName)
    {
        name = initialName;
    }
    public void setName(String newName)
    {
        name = newName;
    }
    public String getName()
    {
        return name;
    }
   public void writeOutput()
   {
        System.out.println("Name: " + name);
   }
    public boolean sameName(Person otherPerson)
    {
        return (this.name.equalsIgnoreCase(otherPerson.name));
    }
}
```

Extended Classes from Person Class