Nesne Yönelimli Programlama

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Not: Bu dersin sunumları, “Java Programlama Dili ve Yazılım Tasarımı, Altuğ B. Altıntaş, Papatya Yayıncılık, 2016” kitabı kullanılarak hazırlanmıştır.

Konular

- Paket
- Varsayılan Paket
- Paket Oluşturma
- Çakışma
- Erişim Belirleyiciler
- Kapsülleme

Paket

- Erişimde iki taraf bulunur; **kütüphaneyi kullananlar (clients)** ve **kütüphaneyi oluşturanlar.**
- Java programlama dili dört adet erişim belirleyicisi sunmaktadır:
  - `friendly`
  - `public`
  - `private`
  - `protected`
- **Paketler kütüphaneyi oluşturan elemanlardır.**
- **Bir paket içerisinde çok sayıda sınıf olabilir.**

Paket

- Aşağıdaki gösterimde `BufferedReader` sınıf ismi `java.io` paketindedir (`java.io` Java ile gelen standart bir pakettir).

```
import java.io.BufferedReader;
```

- **Başka paketlerin içerisinde de `BufferedReader` sınıf ismi tekrar kullanılabilir.**
- Yukarıda `java.io` **paketinin içerisinde bulunan** `BufferedReader` **sınıfının kullanılacağı ifade edilmiştir.**
- Paketin içerisindeki tek bir sınıfı kullanmak yerine, **ilgili paketin içerisindeki tüm sınıfları kullanmak mümkündür.**

```
import java.io.*;
```

Paket

```
package PaketOrnekler;

import java.io.*;

public class PaketOrnekler{
  public static void main(String args[]) throws IOException{
      BufferedReader sf = new BufferedReader(
              new InputStreamReader(System.in));
      System.out.print("Ders Adını Giriniz: ");
      String bilgi = sf.readLine();
      System.out.println("Ders Adı: " + bilgi);
  }
}
```

```
Ders Adını Giriniz: Nesne Yönelimli Programlama
Ders Adı: Nesne Yönelimli Programlama
```

5

Konular

- Paket
- Varsayılan Paket
- Paket Oluşturma
- Çakışma
- Erişim Belirleyiciler
- Kapsülleme

Varsayılan Paket

- **.java** uzantılı fiziksel dosya **derlendiği zaman** buna tam karşılık gelen **.class** uzantılı fiziksel **dosya elde edilir.**
- .java dosyasında **birden fazla sınıf tanımlanmış ise**, tanımlanan **her sınıf için** ayrı ayrı fiziksel **.class dosyaları üretilir.**
- **.java uzantılı dosya** ile **public sınıfın ismi birebir aynı olmalıdır.**

Varsayılan Paket

- Aşağıdaki dosya derlendiğinde isimleri **Test1.class** ve **Test2.class** olan **iki adet .class** uzantılı dosya elde edilir.
- Test1.java dosyasının en üstüne herhangi bir paket bildirimi yapılmadığından **Java bu sınıfları varsayılan paket (default package) olarak tanımlar.**

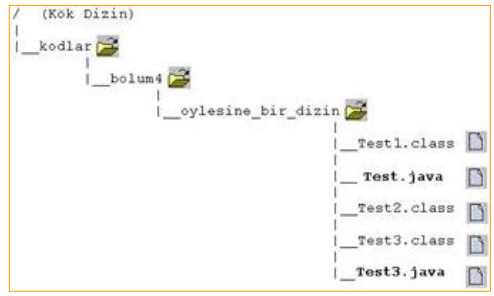
```
public class Test1 {
    public void kos() {

    }
}

class Test2 {
    public void kos() {

    }
}
```

Varsayılan Paket

- Benzer şekilde **aşağıdaki dosya aynı dizine Test3.java adıyla kayıt edilebilir.**

```
public class Test3 {
    public void kos() {

    }
}
```

- Compile işleminden sonra dizin aşağıdaki gibi olur.



9

Konular

- Paket
- Varsayılan Paket
- Paket Oluşturma
- Çakışma
- Erişim Belirleyiciler
- Kapsülleme

Paket Oluşturma

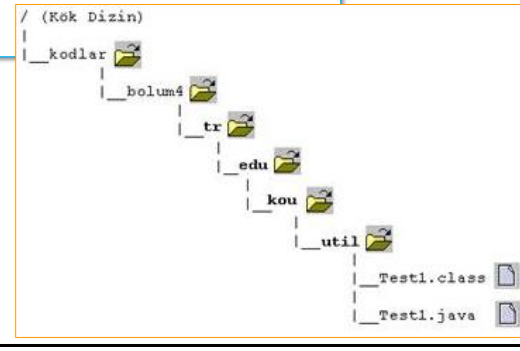
- Paketlerde **aynı amaca yönelik sınıflar bir çatı altında toplanır.**
- Aşağıdaki Test1.java dosyası herhangi bir dizine yerleştirilemez, **tr.edu.kou.util paketine ait bir sınıftır.**
- **Test1.java dosyasının** bu paket ismiyle **aynı dizin yapısına kayıt edilmesi gerekir.**

```
package tr.edu.kou.util;

public class Test1 {

    public Test1() {
        System.out.println("tr.edu.kou.util.Test1 nesnesi olusturuluyor");
    }

    public static void main(String args[]) {
        Test1 pb = new Test1();
    }
}
```

Paket Oluşturma

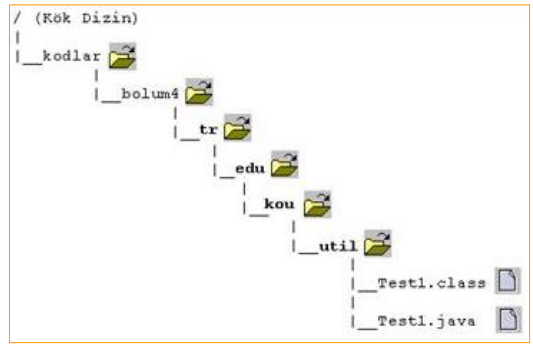
- Test1.java dosyası içerisinde belirtilen **Test1 sınıfının ismi** artık **tr.edu.kou.util.Test1'dir.**

```
package tr.edu.kou.util;

public class Test1 {

    public Test1() {
        System.out.println("tr.edu.kou.util.Test1 nesnesi olusturuluyor");
    }

    public static void main(String args[]) {
        Test1 pb = new Test1();
    }
}
```



Paket Oluşturma

- Paket isimleri için kullanılan yapı **İnternet alan isim sistemiyle (Internet DNS) aynıdır.**
- İnternet alan adı sistemi, **www.kou.edu.tr adresinin dünya üzerinde tek olacağını garantiler.**
- Aynı mantık, paket isimlerine de uygulanarak, **paket içerisindeki sınıf isimlerinin çakışması engellenir.**



13

Konular

- Paket
- Varsayılan Paket
- Paket Oluşturma
- Çakışma
- Erişim Belirleyiciler
- Kapsülleme

Çakışma

- **Ayrı paket içerisinde aynı isimdeki sınıflar** uygulamada kullanılırsa karışıklık olur.
- Aynı adlara sahip **sınıflar farklı paketlerde** ise **açık yolunu yazmak gerekir.**
- **`tr.edu.kou.util`** paketinin içerisine kendi **`ArrayList`** sınıfımızı oluşturalım.

```
package tr.edu.kou.util;

public class ArrayList {
    public ArrayList() {
        System.out.println("tr.edu.kou.util.ArrayList nesnesi olusturuluyor");
    }
}
```

- **Aynı programda import edilirse `java.util.*` içindeki `ArrayList` ile çakışma** olacaktır.

15

Çakışma

```
import java.util.*;
import tr.edu.kou.util.*;

public class Cakisma {
    public static void main(String args[]) {
        System.out.println("Baslagic..");
        ArrayList al = new ArrayList();
        System.out.println("Bitis..");
    }
}
```

```
Cakisma.java:8: reference to ArrayList is ambiguous, both class tr.edu.kou.util.
ArrayList in tr.edu.kou.util and class java.util.ArrayList in java.util match
ArrayList al = new ArrayList();
^
Cakisma.java:8: reference to ArrayList is ambiguous, both class tr.edu.kou.util.


ArrayList in tr.edu.kou.util and class java.util.ArrayList in java.util match
ArrayList al = new ArrayList();
^
2 errors
```

Çakışma

- Aynı adlara sahip sınıflar farklı paketlerde ise **açık yolunu yazmak gerekir.**

```
import java.util.*;
import tr.edu.kou.util.*;

public class Cakisma2 {
    public static void main(String args[]) {
        System.out.println("Baslagic..");
        tr.edu.kou.util.ArrayList al = new tr.edu.kou.util.ArrayList();
        System.out.println("Bitis..");
    }
}
```

17

Konular

- Paket
- Varsayılan Paket
- Paket Oluşturma
- Çakışma
- Erişim Belirleyiciler
- Kapsülleme

Erişim Belirleyiciler

- Java dilinde 4 tür erişim belirleyici vardır:
  - `friendly`
  - `public`
  - `protected`
  - `private`
- Bu **erişim belirleyiciler global alanlar** (statik veya değil) ve **yordamlar** (statik veya değil) için kullanılabilir.
- Ayrıca, **sınıflar için** (dahili sınıflar hariç –inner class) sadece **public** ve **friendly** erişim belirleyicileri kullanılabilir.

19

Erişim Belirleyiciler

`friendly`

- `friendly` erişim belirleyicisi **global alanlara** (statik veya değil), **yordamlara** (statik veya değil) ve **sınıflara atanabilir.**
- `friendly` türünde **erişim belirleyicisine sahip** olan global **alanlar** (statik veya değil) içerisinde bulundukları **paketin diğer sınıfları tarafından erişilebilirler.**
- Fakat, **diğer paketlerin içerisindeki sınıflar tarafından erişilemezler.**
- Diğer paketlerin içerisindeki sınıflara karşı `private` erişim belirleyici etkisi oluşturmuş olurlar.

20

Erişim Belirleyiciler

`friendly`

- **`friendly` yordamlara**, yalnızca **paketin kendi içerisindeki diğer sınıflar tarafından erişilebilir.**
- **`friendly` yordamlara** diğer paketlerin içerisindeki sınıflar tarafından erişilemezler.
- Aynı şekilde, **sınıflara `friendly` erişim belirleyicisi atanabilir.**
- **`friendly`** erişim belirleyicisine sahip **sınıfa**, **aynı paket içerisindeki diğer sınıflar erişilebilir.**
- Ancak, **diğer paketlerin içerisindeki sınıflar erişemezler.**

Erişim Belirleyiciler

`friendly`

- Bir global alan veya sınıf **`friendly`** yapılmak isteniyorsa önüne **hiç bir erişim belirleyicisi konulmaz (default).**
- **`tr\edu\kou\`** dizini altına yeni bir dizin oluşturup, ismini **`gerekli`** verelim.
- Yani **`tr\edu\kou\gerekli`** paketini oluşturmuş olduk.
- Bunun içerisine adları **`Robot`** ve **`Profesor`** olan 2 adet **`friendly`** sınıf yazalım.

```
package tr.edu.kou.gerekli;

class Robot {

    int calisma_sure = 0;
    String renk = "beyaz";
    int motor_gucu = 120;

    Robot() {
        System.out.println("Robot olusturuluyor");
    }
}
```

```
package tr.edu.kou.gerekli;

class Profesor {
    public void kullan() {
        Robot upuaut = new Robot(); // sorunsuz
    }
}
```

Erişim Belirleyiciler

`friendly`

- **`Asistan`** sınıfı **`tr.edu.kou.util`** paketi altında tanımlandığı için **`tr.edu.kou.gerekli`** paketi altında tanımlı olan **`Robot`** sınıfına hiç bir şekilde erişemez.

```
package tr.edu.kou.gerekli;

class Robot {

    int calisma_sure = 0;
    String renk = "beyaz";
    int motor_gucu = 120;

    Robot() {
        System.out.println("Robot olusturuluyor");
    }
}
```

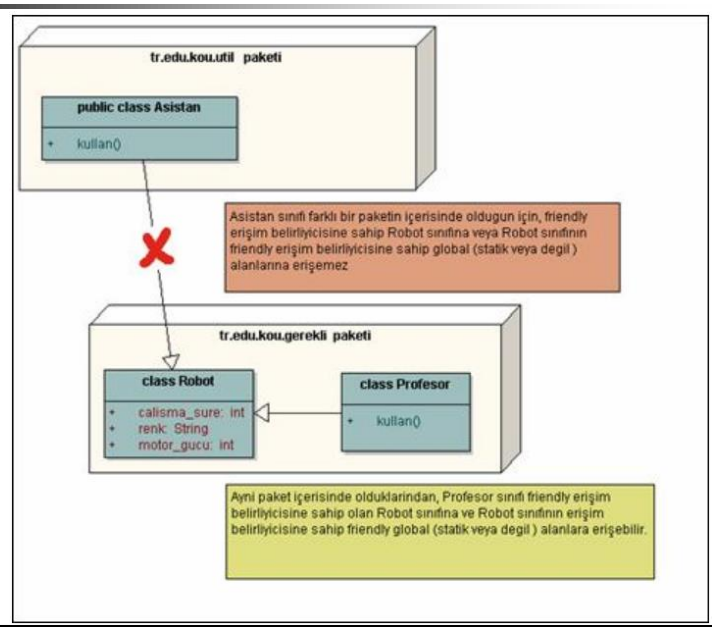
```
package tr.edu.kou.util;

import tr.edu.kou.gerekli.*;

public class Asistan {
    public void arastir() {
        System.out.println("Asistan arastiriyor");
    }

    public void kullan() {
        // Robot upuaut = new Robot(); Hata! erişemez
    }
}
```

```
package tr.edu.kou.gerekli;

class Profesor {
    public void kullan() {
        Robot upuaut = new Robot(); // sorunsuz
    }
}
```

23

Erişim Belirleyiciler

`friendly`



24

Erişim Belirleyiciler

`public`

- **`public`** erişim belirleyicisine sahip olan **sınıflara**, **global alanlara** ve **yordamlara** herkes tarafından erişilebilir.
- **`public`** erişim belirleyicisine sahip olan **global alanlar** veya **yordamlar herhangi bir yerden doğrudan çağırılabilir.**

Erişim Belirleyiciler

`public`

- Örnekte, **`tr.edu.kou.util`** paketindeki **`Makine`** sınıfının 2 adet global alanı (**`devir_sayisi`** ve **`model`**) bulunmaktadır.

```
package tr.edu.kou.util;

public class Makine {

    int devir_sayisi;
    public String model = "2002 model";

    public int degerAl() {
        return devir_sayisi;
    }

    public void degerAta(int deger) {
        this.devir_sayisi = deger;
        calis();
    }

    void calis() {
        for (int i = 0; i < devir_sayisi; i++) {
            System.out.println("Calisiyor, devir_sayisi = " + i);
        }
    }
}
```

Erişim Belirleyiciler

public

```
package tr.edu.kou.util;

public class Makine {

    int devir_sayisi;
    public String model = "2002 model";

    public int degerAl() {
        return devir_sayisi;
    }

    public void degerAta(int deger) {
        this.devir_sayisi = deger;
        calis();
    }

    void calis() {
        for (int i = 0; i < devir_sayisi; i++) {
            System.out.println("Calisiyor, devir_sayisi = " + i);
        }
    }
}
```

- **`int`** türündeki **`devir_sayisi`** alanı **`friendly`** erişim belirleyicisine sahiptir.
- Sadece **`tr.edu.kou.util`** paketinin içerisindeki diğer sınıflar tarafından erişilebilir.
- Diğer **`String`** tipindeki **`model`** alanı ise her yerden erişilebilir (**`public`** erişim belirleyicisine sahiptir).
- **`degerAl()`** yordamı **`public`** erişim belirleyicisine sahiptir yani her yerden erişilebilir.
- Aynı şekilde **`degerAta(int deger)`** yordamı da her yerden erişilebilir.
- **`calis()`** yordamı **`friendly`** belirleyicisine sahiptir, sadece **`tr.edu.kou.util`** paketinin içerisindeki sınıflar erişebilir.

27

Erişim Belirleyiciler

public

- Aşağıdaki örnekte **`tr.edu.kou.util`** paketinin altındaki tüm sınıfların kullanılacağı belirtilmiştir.
- **`Ustabasi`** sınıfının yapılandırıcısında **`public`** olan **`Makine`** sınıfına ait bir nesne oluşturulabilir.
- Oluşturulan nesnenin **`friendly`** erişime sahip olan **`devir_sayisi`** alanına ve **`calis()`** yordamına erişilemez.
- **`Ustabasi`** sınıfı **`tr.edu.kou.util`** paketinde değildir.

```
import tr.edu.kou.util.*;

public class UstaBasi {                                    ;

    public UstaBasi() {
        Makine m = new Makine();
        // int devir_sayisi = m.devir_sayisi ; ! Hata ! erişemez
        m.degerAta(6);
        int devir_sayisi = m.degerAl();
        String model = m.model;
        // m.calis() ; ! Hata ! erişemez
    }
}
```

28

Erişim Belirleyiciler

`private`

- **`private`** olan global alanlara ve yordamlara **(sınıflar private olamazlar, dahili sınıflar-inner class hariç) aynı paket içerisinden veya farklı paketlerden erişilemez.**
- **`private`** olan **global alanlara** ve **yordamlara ait olduğu sınıfın içinden erişilebilir.**

Erişim Belirleyiciler

`private`

- **`Kahve`** sınıfının yapılandırıcısı **`private`** olarak tanımlanmıştır.
- Başka bir sınıf, **`Kahve`** sınıfının yapılandırıcısını çağıramaz.
- **`private`** yapılandırıcı aynı sınıftaki yordamlar tarafından çağırılabilir (**`siparisGarson()`**).

```
package tr.edu.kou.gerekli;

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;
    }
}
```

Erişim Belirleyiciler

`private`

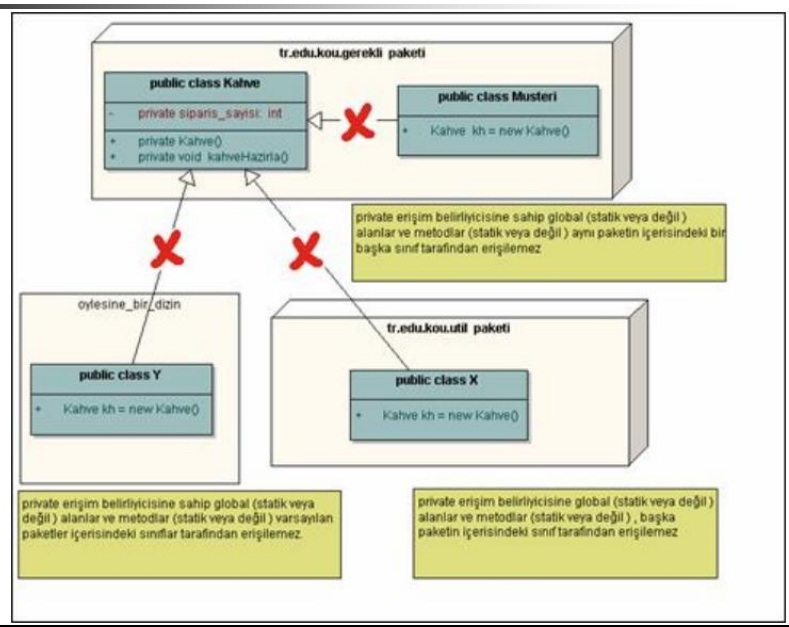
- **`private`** olarak tanımlanmış **global alanlara** ve **yordamlara** aynı paket içerisinde olsa bile kesinlikle erişilemez.

```
package tr.edu.kou.gerekli;

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);
      }
}
```

31

Erişim Belirleyiciler

`private`



Erişim Belirleyiciler

`protected`

- **Sadece global alanlar** ve **yordamlar** **`protected`** erişim belirleyicisine sahip olabilirler.
- **Sınıflar** **`protected`** **erişim belirleyicisine sahip olmazlar** (dahili sınıflar-inner class hariç).
- Ancak, **sınıflar** **`friendly`** **veya** **`public`** **erişim belirleyicisine sahip olabilirler.**
- **`protected`** erişim belirleyicisi kalıtım (inheritance) konusu ile ilişkilidir.
- Kalıtım konusu hakkında kısaca, bir sınıftan başka sınıfların türetilmesi denilebilir.

Erişim Belirleyiciler

`protected`

```
class Kedi extends Hayvan
```

- Yukarıda şu ifade edilmiştir:
  - **Her Kedi bir Hayvandır.**
  - **Hayvan sınıfından Kedi türetilmiştir.**
  - Bizim oluşturacağımız **her Kedi nesnesi bir Hayvan olacaktır.**
  - Her türetilen yeni kedi, **kendisine özgü özellikleri** de taşıyacaktır.

Erişim Belirleyiciler

protected

- **Hayvan** sınıfından türetilen **Kedi** sınıfı **tr.edu.kou.gerekli** paketi içerisine yerleştirilmiş olsun.

```
package tr.edu.kou.util;

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

```
javac Kedi.java
```

```
java tr.edu.kou.gerekli.Kedi
```

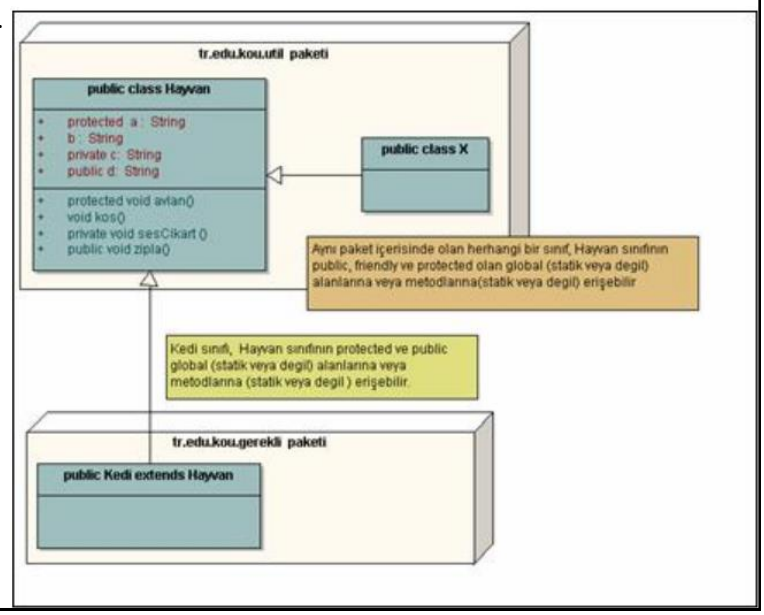
```
Kedi olusturuluyor
Hayvan.a
Hayvan.d
```

```
package tr.edu.kou.gerekli;

import tr.edu.kou.util.*;

public class Kedi extends Hayvan {
    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();
    }
}
```

35

Erişim Belirleyiciler

protected



Erişim Belirleyiciler

Erişim belirleyiciler ve erişim hakları özet.

ERİŞİM BELİRLEYİCİ	AYNI SINIFTAN	AYNI PAKETTEKİ TÜM SINIFLARDAN	FARKLI PAKETTEKİ KALITILMIŞ SINIFLARDAN	FARKLI PAKETTEKİ TÜM SINIFLARDAN
PUBLIC	EVET	EVET	EVET	EVET
PROTECTED	EVET	EVET	EVET	HAYIR
DEFAULT	EVET	EVET	HAYIR	HAYIR
PRIVATE	EVET	HAYIR	HAYIR	HAYIR

37

Konular

- Paket
- Varsayılan Paket
- Paket Oluşturma
- Çakışma
- Erişim Belirleyiciler
- Kapsülleme

Kapsülleme

- **Kapsülleme (encapsulation),** nesneye yönelik programlama özelliklerinden birisidir.
- Dışarıdaki **başka bir uygulama** nesnelerle sadece **arabirimler (public) sayesinde iletişim kurar.**
- Ancak, **arka planda işi yapan esas kısım gizlenir.**
- Olaylara bu açıdan bakıldığında, **nesneler iki kısma bölünmelidir:**
  - **Interface (arabirim):** Nesnenin dünya ile iletişim kurabilmesi için gerekli kısımlardır.
  - **Implementation:** İşlevleri gerçekleştiren kısımlardır.

39

Kapsülleme

- **`Makine2`** nesnesine **`get()`** ve **`set()`** yordamları erişebilir.
- Geriye kalan global alanlara veya **`calis()`** yordamına ulaşım söz konusu değildir.
- **Nesne iki kısımdan oluşturulmuştur.**
- 1) interface
  - **`get()`**
  - **`set()`**

  2) implementation
  - **`calis()`**

```
package tr.edu.kou.util;

public class Makine2 {
    private int alinan = 0;
    private int geridondurulen = 0;

    public int get() {
        return geridondurulen;
    }

    public void set(int i) {
        alinan = i;
        calis();
    }

    private void calis() {
        for (int j = 0; j < alinan; j++) {
            System.out.println("Sonuc = " + j);
        }
        geridondurulen = alinan * 2;
    }
}
```

Kapsülleme

- Başka bir paket içerisinde olan uygulama, **`tr.edu.kou.util.Makine2`** sınıfının sadece iki yordamına erişebilir, **`get()`** ve **`set()`**

```
package tr.edu.kou.gerekli;

import tr.edu.kou.util.*;

public class GetSet {
    public static void main(String args[]) {
        Makine2 m2 = new Makine2();
        m2.set(5);
        int deger = m2.get();
        // m2.calis() ; // Hata !
        // m2.alinan ; // Hata !
        // m2.geridondurulen; // Hata !
        System.out.println("Deger =" + deger);
    }
}
```

41

Kapsülleme

- **Sınıflar için erişim tablosu**
  (Sınıflar **`protected`** veya **`private`** olamazlar).

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

- Örneğin bir A sınıfı olsun:
  - **`public A`** sınıfına aynı paketin içerisindeki başka bir sınıf erişebilir.
  - **`public A`** sınıfına ayrı paketin içerisindeki başka bir sınıf erişebilir.
  - **`public A`** sınıfına ayrı paketten erişebildiğinden buradan yeni sınıflar türetilebilir.
  - **`friendly A`** sınıfına aynı paketin içerisindeki başka bir sınıf erişebilir.
  - **`friendly A`** sınıfına ayrı paketin içerisindeki başka bir sınıf erişemez.
  - **`friendly A`**'ya ayrı paketten erişilemediğinden, buradan yeni sınıflar türetilemez.

42

Kapsülleme

- **Statik veya statik olmayan yordamlar için erişim tablosu**
  (Yordamlar **`public`**, **`protected`**, **`friendly`** ve **`private`** olabilir).

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

- Örneğin, **`public X`** sınıfının içerisinde **`f()`** yordamı olsun:
  - **`public f()`** yordamı, aynı paket içerisinden erişilebilir.
  - **`protected f()`** yordamı, hem aynı paket içerisinden hem de X sınıfından türetilmiş ayrı paketteki bir sınıf tarafından erişilebilir.
  - **`friendly f()`** yordamı, yalnızca aynı paket içerisinden erişilebilir.
  - **`private f()`** yordamına, yalnızca kendi sınıfı içerisinden erişilebilir. Başka bir sınıfın bu yordama erişmesi mümkün değildir.

43

Kapsülleme

- **Statik veya statik olmayan global alanlar için erişim tablosu**
  (Global alanlar **`public`**, **`protected`**, **`friendly`** ve **`private`** olabilir).

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

- Örneğin **`public X`** sınıfının içerisindeki **`String`** sınıfı tipindeki uzunluk adında bir alanımız olsun:
  - **`public`** uzunluk alanı, aynı paket içerisinden erişilebilir.
  - **`protected`** uzunluk alanı, hem aynı paket içerisinden, hem de X sınıfından türetilmiş ayrı paketteki bir sınıf tarafından erişilebilir.
  - **`friendly`** uzunluk alanı, yalnızca aynı paket içerisinden erişilebilir.
  - **`private`** uzunluk alanı, yalnızca kendi sınıfı içerisinden erişilebilir. Başka bir sınıfın bu alana erişmesi mümkün değildir.

44

Araştırma ödevi

- Procedural programming, structured programming, object based programming, object oriented programming, event-driven programming hakkında araştırma ödevi hazırlayınız.
  - Ödev bir veya birkaç kaynaktan olduğu gibi alınarak hazırlanmayacaktır. Metin hazırlayanın kendisi tarafından oluşturulacaktır.
  - Ödevi hazırlarken kullanılan kaynakların tümü ödevin sonunda listelenecektir.
  - Ödevlerde öğrenci numarası ve adı soyadı yazılı olan bir kapak sayfası olacaktır.