ERİŞİM BELİRLEYİCİLER (ACCESS MODİFİERS)

Bir java deyimi (belli bir iş yapan kod veya kodlardan oluşan blok) bir değişkeni çağırabiliyorsa (kullanabiliyorsa), o kod sözkonusu değişkene erişebiliyor, denir. Benzer olarak, bir java kodu bir metodu çağırabiliyorsa, o kod sözkonusu metoda erişebiliyor, diyoruz. **Bir sınıfın içindeki kodlar, başka bir sınıfın içindeki değişkenlere ve metotlara erişebiliyorsa, o sınıf sözkonusu sınıfa erişebiliyor, diyoruz.** Paketler için de benzer tanım geçerlidir.

Bir Java öğesi (değişken, metot, sınıf, paket) tanımlanırken, o öğeye kimlerin erişebileceğini belirtme olanağı vardır. Bunlara Erişim belirleyiciler (Access modifiers, access levels) denir. Java terimleriyle söylersek, erişim belirtkeleri sistemin güvenliğini sağlar. Dört tane erişim belirtkesi vardır:
friendly, private, protected, public.

Erişim Belirtkesi	*İzinler*
public	Bütün sınıflar erişebilir
private	Alt-sınıf dahil başka hiçbir sınıf erişemez
protected	Alt-sınıflar ve aynı pakettekiler erişebilir
<default> friendly	Aynı pakettekiler erişebilir

1. public

public bir değişkeni, bir metodu ya da bir sınıfı niteleyebilir. Nitelediği öğeler herkese açık olur. Başka pakette olsa bile, program içindeki, her kod onlara erişebilir. public damgalı bir sınıfın değişkenlerine ve metotlarına kendi alt-sınıfları ve dışarıdaki başka sınıflar kısıtsız erişebilir. public damgalı değişkenler ve metotlar için de kısıtsız erişim vardır. Uygulama programlarında main() metodunun public damgalı olmasının nedeni budur. Örnekler:

a. public sınıf bildirimi

```
public class class_adı {
        Class gövdesi
}
```

b. public değişken bildirimi

```
public long  r ;
```

c. public metot bildirimi

```
public  double toplam(int m, double  d){
return m + d ;
}
```

2. private

Bazı değişken, metot ya da sınıflara başka sınıftaki kodların erişmesini engellemek isteyebiliriz. Bunun için private nitelemesini kullanırız. private erişim belirleyicisi, public belirtkesinin karşıtı gibidir. private damgalı öğelere yalnız aynı sınıftaki kodlar erişebilir, başka sınıftaki kodlar erişemez. Kendi alt-sınıfları bile erişemez. Bir alt-sınıf, atasının public ve ön-tanımlı öğelerine erişebilir, ama private öğelerine erişemez. Onlara erişebilmesi için, super class interface-fonksiyonu kullanılır. Bunun nasıl olduğunu ileride göreceğiz. Sözdizimi şöyledir:

a. private sınıf bildirimi

```
private class class_adı {
        Class gövdesi
}
```

b. private değişken bildirimi

```
private long  r ;
```

c. private metot bildirimi

```
private  double toplam(int m, double  d){
return m + d ;
}
```

3. protected

Bir sınıf içindeki değişken ve metotlara alt-sınıfların erişebilmesini, ama paket içindeki ya da program içindeki başka kodların erişmesini engellemek isteyebiliriz. Bunun için sözkonusu öğeyi, protected damgası ile nitelemek yetecektir. Bu demektir ki, alt-sınıf, üst-sınıfın protected damgalı öğelerine sanki public öğelermiş gibi erişir. Görüldüğü gibi, protected belirtkesi, ön-tanımlı belirtke ile private belirtkesinin işlevleri arasında bir işleve sahiptir. Alt sınıflara erişme yetkisi verdiği için, kalıtım (inheritance) olgusunda önemli rol oynar. Sözdizimi şöyledir:

a. protected sınıf bildirimi

```
protected class class_adı {
        Class gövdesi
}
```

b. protected değişken bildirimi

```
protected long  r ;
```

c. protected metot bildirimi

```
protected  double toplam(int m, double  d){
return m + d ;
}
```

1. Friendly

Bir öğenin önüne hiçbir erişim belirtkesinin konmadığı durumdur. Erişim belirtkesi konmamışsa ön-tanımlı (default) belirtke etkin olur. Buna, bazı kaynaklarda dostça erişim (frendly access) denir.

a. paket bildirimi

Paketler yalnızca ön-tanımlı erişime sahiptir, başka erişim belirtkesi almazlar. Paket içindeki her sınıf pakette olan her değişken ve metoda erişebilir. Ama başka paketlerdeki sınıflar erişemez.

```
  package paket_adı {
        Paket gövdesi  }
```

b. erişim belirtecsiz (default) sınıf bildirimi

Sınıf bidiriminde, sınıfın önüne hiçbir erişim belirtkesi konmazsa, o sınıf içindeki değişken ve metotlara o sınıfı içeren paketteki bütün kodlar erişebilir. Kuruluş yapısı şöyledir:

```
class class_adı {
   Class gövdesi
}
```

c. erişim belirteçsiz (default) metot bildirimi

Metot bildiriminde hiçbir erişim belirtkesi konulmazsa, default belirke geçerlidir. Bu halde, class'ın kendisi, alt-sınıflar ve aynı paket içindeki diğer sınıflar erişebilir. Örnek:

```
int topla (int m, int n) {
  return m+n ;
}
```

d. erişim belirteçsiz (default) değişken bildirimi

Değişken bildiriminde hiçbir erişim belirtkesi konulmazsa, ön-tanımlı (default) belirke geçerlidir. Bu halde, sınıfın kendisi, alt-sınıfları ve aynı paket içindeki diğer sınıflar erişebilir. Örnek:

```
float kesir;
```

Örnek-1:

```
class Test {
  int a;              // friendly erişim
  public int b;       // public erişim
  private int c;      // private erişim
      //  c  ye erişen metot
  void setc(int i) { // c  ye değer atar
    c = i;
  }
  int getc() {        // c  nin değeri
    return c;
  }
}
public class erisimbelirleyiciler {
    public static void main(String[] args) {

    Test ob = new Test();
    // a ile b ye direkt erişilebilir
    ob.a = 10;
    ob.b = 20;
    //  ob.c = 100;          // Hata!
    ob.setc(100); // geçerli
    System.out.println("a, b, ve c: " + ob.a + " " +
                       ob.b + " " + ob.getc());
  }
}
```

Örnek-2:

```
class Stack {
  private int stk[] = new int[10];
  private int top;
  Stack() {    top = -1;  }
  void push(int item) {
    if(top==9)
      System.out.println("Stack doldu.");
    else
      stk[++top] = item; }
  int pop() {
    if(top < 0) {
      System.out.println("Stack dolmadı.");
      return 0;    }
    else
      return stk[top--];
  }
}
public class erisim2 {
  public static void main(String args[]) {
    Stack st1 = new Stack();
    Stack st2 = new Stack();
    for(int i=0; i<10; i++) st1.push(i);
    for(int i=10; i<20; i++) st2.push(i);
    System.out.println("Stack in st1:");
    for(int i=0; i<10; i++)
      System.out.println(st1.pop());
    System.out.println("Stack in st2:");
    for(int i=0; i<10; i++)
      System.out.println(st2.pop());
  }}
```

PAKETLER (PACKAGES)

Paketler sınıfları düzenlemek için kullanılabilir. Bunu yapmak için, programdaki ilk yorumusz ve boş olmayan ifade olarak aşağıdaki satırı eklemeniz gerekir:
Pacage paketadi;
Bir sınıf package deyimi olmadan tanımlanırsa, varsayılan pakete yerleştirildiği söylenir.

```
package  p1;

public class C1 {
  public int x;
  int y;
  private int z;

  public void m1() {
  }
  void m2() {
  }
  private void m3() {
  }
}
```

```
package  p1;

public class C2 {
  void aMethod() {
    C1 c1 = new C1();
    can access c1.x;
    can access c1.y;
    cannot access o.z;

    can invoke c1.m1();
    can invoke c1.m2();
    cannot invoke c1.m3();
  }
}
```

```
package p2;

public class C3 {
  void aMethod() {
    C1 c1 = new C1();
    can access c1.x;
    cannot access c1.y;
    cannot access c1.z;

    can invoke c1.m1();
    cannot invoke c1.m2();
    cannot invoke c1.m3();
  }
}
```