Programació

7.3 Elements estàtics d'una classe

Elements estàtics d'una classe

En Java és possible declarar mètodes i variables que pertanyen a la CLASE en comptes de l'objecte. Això es possible amb el modificador static

Declaració

Els mètodes i atributs estàtics es declaren inserint la paraula "static" immediatament després de l'indicador de visibilitat (public, private o protected).

```
public class ArrayMaterial {

public static int dada = 3;
public static double mitjana(int[] arr) {
      double total = 0.0;
      for (int k=0; k < arr.length; k++) {
         total += arr[k];
      }
      return  total / arr.length;
   }
}
```

Accés

Als mètodes i atributs estàtics es accedeixen usant el nom de la classe a lloc de l'objecte instànciat.

```
double myArray = {1.1, 2.2, 3.3};
...
double average = ArrayMaterial.mitjana(myArray);
int total = ArrayMaterial.dada + 3;
```

Mètodes estàtics o de classe. Per a què?

Els mètodes estàtics o de classe són útils per a mètodes no relacionats amb altres objectes a excepció dels propis paràmetres del mètode. Es a dir, que no afecta a l'objecte que els invoca, ni tampoc cal que un objecte els invoque.

Un bon exemple de la utilitat dels mètodes estàtics es troba a la llibreria estàndard de Java trucada Math.

```
public class Math {
    public static double abs(double d) {...}
    public static int abs(int k) {...}
    public static double cos(double d) {...}
    public static double pow(double b, double exp) {...}
    public static double random() {...}
    public static int round(float f) {...}
    public static long round(double d) {...}
    public static double sin(double d) {...}
    public static double sqrt(double d) {...}
    public static double tan(double d) {...}
    . . .
}
```

Restriccions dels mètodes estàtics

Al moment que hem definit que un mètode estàtic pertany a la classe i NO l'objecte es produeixen un seguit de limitacions.

Al cos d'un mètode estàtic NO ES POT REFERENCIAR un atribut ni un mètode de la classe que no siga estàtic.

ENCARA QUE...

Al cos d'un mètode estàtic SÍ QUE ES PODEN INSTANCIAR OBJECTES.

```java
public void mostrarInformacio(){
    System.out.println("Nom: " + nom);
    System.out.println("DNI: " + dni);
    System.out.println("Saldo: " + saldo + " euros");
}

// Mètode main
Run | Debug
public static void main(String[] args) {
    this.mostrarInformacio();
```

Exemple

```java
public class run {
   public static void main(String[] args) {
      Driver driver = new Driver();
   }
}
```

Atributs estàtics o de classe

Qualsevol atribut de la classe pot ser declarat com a estàtic incloent la paraula “static” immediatament abans de la declaració del seu tipus.

```
public class StaticStuff{

   public static double staticDouble;
   public static String staticString;
   . . .
}
```

Què és diferent en un atribut estàtic?

1)Un atribut estàtic pot ser referenciat usant el nom de la seua classe (o el nom d’un objecte, encara que esta manera no és una pràctica recomanable; de fet els entorns de desenvolupament mostraran un “Warning”).

2)Instanciant un segon objecte del mateix tipus no incrementa el número de variables estàtiques en memòria. *(Veure exemple de la diapositiva següent)*

Atributs estàtics. Exemple

Exemple

```
StaticStuff s1, s2;
s1 = new StaticStuff();
s2 = new StaticStuff();
s1.staticDouble = 3.7;
System.out.println( s1.staticDouble );
System.out.println( s2.staticDouble );
s1.staticString = “abc”;
s2.staticString = xyz;
System.out.println( s1.staticString );
System.out.println( s2.staticString );
```

L’eixida del programa per pantalla serà:

```
3.7
3.7
xyz
xyz
```

Atributs estàtics. Per a què?

Els atributs estàtics són útils per a situacions on les dades necessiten ser compartides per múltiples objectes del mateix tipus (de la mateixa classe).

Un bon exemple de la utilitat dels atributs de tipus estàtic es troba a la llibreria estàndard de Java anomenada Color.

```
public class Color {
   public static final Color black = new Color(0,0,0);
   public static final Color blue = new Color(0,0,255);
   public static final Color cyan = new Color(0,255,255);
   public static final Color darkGray = new Color(64,64,64);
   . . .
}
```

També a la llibreria Math: `Math.PI;`

Exercici Bombeta.

- Volem modelar una casa amb moltes bombetes, de manera que cada bombeta es puga encendre o apagar de manera individual.
- Per a això farem una classe Bombeta amb una variable privada que indique si està encesa o apagada, així com un mètode que ens diga l'estat d'una bombeta concreta.
- A més volem posar un interruptor general de manera que si este s'apaga, totes les bombetes quedaran apagades.
- Quan l'interruptor general s'active (per defecte està activat) les bombetes tornen a estar enceses o apagades segons el seu propi estat.
- Cada bombeta s'encén i s'apaga individualment, però només respon que està encesa si el seu interruptor particular està activat i a més hi ha llum general.