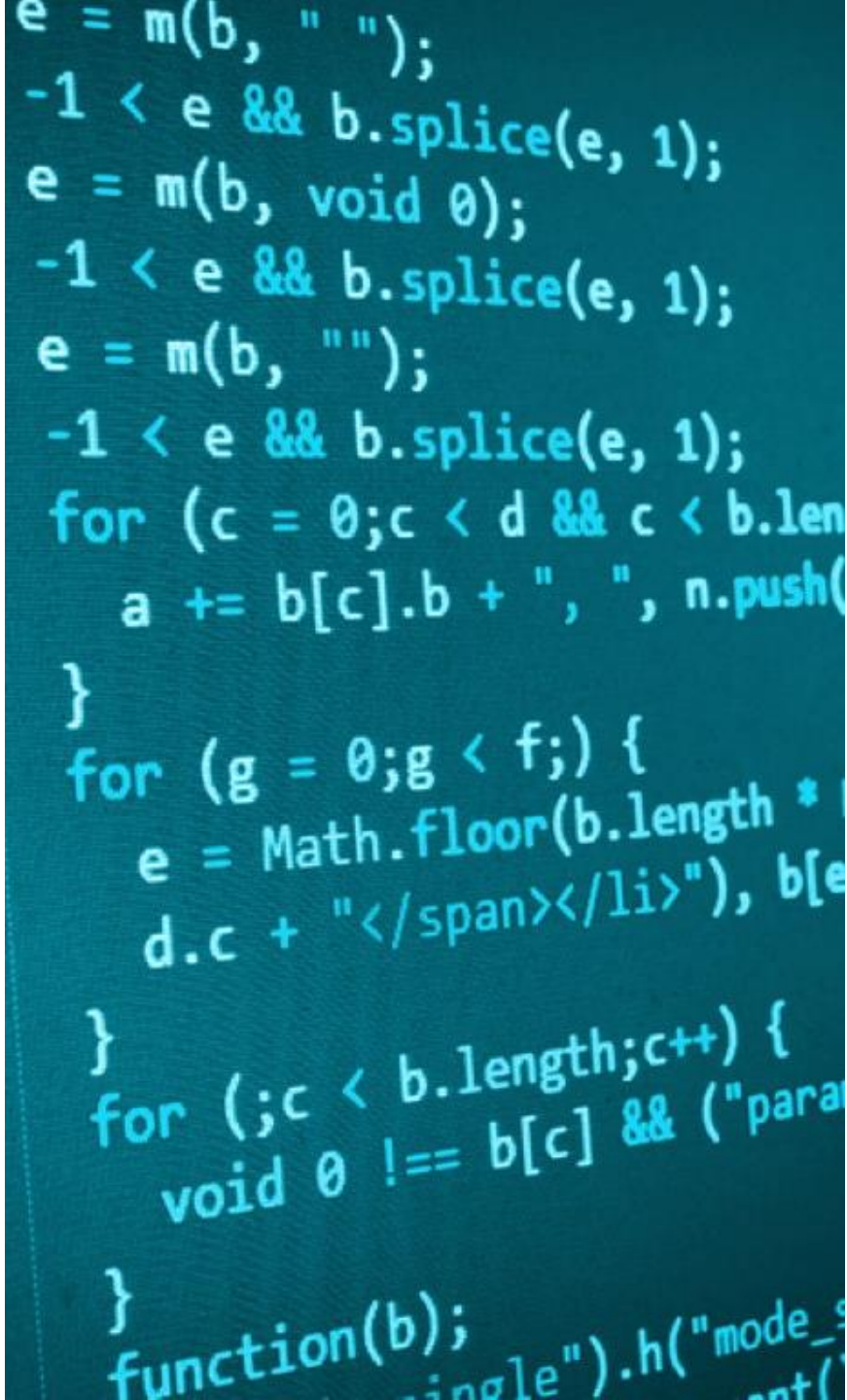
UD5. Introducció a la POO

Verónica Mascarós

Curs 23-24



POO en Java: Elements estàtics

- El comportament "normal" vist fins ara:
  - A les classes definim que aquest objecte tindrà aquests atributs i mètodes, però, per accedir-hi o donar-los valors, necessitem construir objectes d'aquesta classe.
  - Per exemple, un *cotxe* té un color, però a la classe només dius que existirà un color i fins que no construïsques els cotxes, no els assignaràs un color en concret.
  - A la classe *quadrat* definiràs que el càlcul de l'àrea és el "costat elevat a dos", però per calcular l'àrea d'un quadrat necessites tenir un objecte d'aquesta classe i demanar-li que et torne la seua àrea.

POO en Java: Elements estàtics

- Exemple: classe vs. accés

```java
public class Persona {

    private String dni;
    private String nombre;
    private String apellidos;
    private int edad;


    public Persona(String dni, String nombre, String apellidos, int edad) {
        this.dni = dni;
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.edad = edad;
    }


    public String getDni() { return dni; }


    public String getNombre() { return nombre; }


    public String getApellidos() { return apellidos; }


    public int getEdad() { return edad; }


    public void setDni(String dni) { this.dni = dni; }


    public void setNombre(String nombre) { this.nombre = nombre; }
```

```java
Persona persona1 = new Persona("18999548P", "José", "Serrano Márquez", 25);
Persona persona2 = new Persona("20222444L", "María", "Carcelén Sánchez", 17);
```

```java
persona1.setNombre("Juan");
persona1.setEdad(11);
persona2.setNombre("Carmen");
persona2.setEdad(33);
```

```java
String cadena1 = persona1.getNombre() + " " + persona1.getApellidos() + " con DNI " + persona1.getDni();
String cadena2 = persona2.getNombre() + " " + persona2.getApellidos() + " con DNI " + persona2.getDni();
```

```java
if (persona1.getEdad() >= 18) {
    cadena1 += " es mayor de edad";
} else {
    cadena1 += " no es mayor de edad";
}

if (persona2.getEdad() >= 18) {
    cadena2 += " es mayor de edad";
} else {
    cadena2 += " no es mayor de edad";
}
```

POO en Java: Elements estàtics

- Tot i això, els elements estàtics o membres de classe són un poc diferents...
  - Són elements que existeixen dins de la pròpia classe i per accedir-hi no necessitem haver creat cap objecte d'aquesta classe.
  - O siga, en comptes d'accedir-hi a través d'un objecte, accedim a través del nom de la classe.
- Els elements (atributs o mètodes) **static** podem dir que són aquells que **pertanyen a la classe**, en lloc de pertànyer a un objecte particular.

POO en Java: Elements estàtics

- **Atributs estàtics - Variables de classe**:
  - Un atribut **static** és com una variable global de la classe, a la qual tenen accés tots els objectes de la classe.
  - Un objecte de la classe no copia els atributs static → **totes les instàncies comparteixen la mateixa variable**.
  - No és específic de cada objecte. → Només n'hi ha una còpia i el seu valor és compartit per tots els objectes de la classe.
  - Existeix i es pot utilitzar encara que no hi hagen objectes de la classe.
  - Ús d'atributs estàtics: nomClasse.nomAtributEstatic.

POO en Java: Elements estàtics

- **Atributs estàtics - Variables de classe**:
  - Què significa que no és específic de cada objecte?
  - La creació de diverses instàncies de la classe CompteBancari no comporta l'existència de diverses variables totalComptes. Només hi haurà una variable de classe totalComptes, independentment del nº d'instàncies de la classe CompteBancària que es generen. Exemple:

```
public class CuentaBancaria {
    // Atributos o variables miembro
    public double saldo;                  // Variable de instancia
    public static int totalCuentas=0;     // Variable de clase

    // Declaraciones de metodos...
}
```

POO en Java: Elements estàtics

- **Mètodes estàtics**:
  - Un mètode **static** és un mètode global de la classe.
  - Un objecte no fa còpia dels mètodes static.
  - Solen utilitzar-se per accedir a atributs estàtics de la classe.
  - No és necessari instanciar un objecte per poder utilitzar-lo.
  - No poden fer ús de la referència this.
  - Ús de mètodes estàtics: NomClasse.nomMetode(args)

POO en Java: Elements estàtics

- Exemple:

```
public class Punto {
     ...
     //atributo estático de la clase
     private static int cantidadPuntos;
     //cada instancia incrementa este atributo
     public Punto ( ) {
          cantidadPuntos++;
     }
     //método estático que retorna un atributo estático
     public static int getPuntos( )
          return cantidadPuntos;
     }
     ...
}
```

```
public class App {
     public static void main (String[] args) {
          //se crean instancias
          Punto p1 = new Punto ( );
          Punto p2 = new Punto ( );
          Punto p3 = new Punto ( );
          //accedemos al método estático para ver el
          //número de instancias de tipo Punto creadas
          System.out.println(Punto.getPuntos( ));
     }
}
```

POO en Java: Elements estàtics

- La paraula reservada final indica que un valor no pot canviar.
- Se sol combinar amb el modificador static.
- L'identificador d'una variable final per convenció s'ha d'escriure en majúscules.
- Si es defineix com a final:
  - Una classe → No pot tenir classes filles (seguretat i eficiència del compilador).
  - Un mètode → No pot ser redefinit per una subclasse (és com protegir el mètode contra sobreescritura).
  - Un atribut → Ha de ser inicialitzada en declarar-se (si és static) i el seu valor no es pot canviar → Seria una constant.

POO en Java: Elements estàtics

- Exemple:

```
public class Constantes{
    //constantes públicas
    public static final float PI = 3.141592f;
    public static final float E = 2.728281f;

    public static final void mostrarConstantes() {
        System.out.println("PI = " + PI);
        System.out.println("E = " + E);
    }
}

public class App {
    public static void main (String[] args) {
        Constantes.mostrarConstantes();
        System.out.println("PI = " + Constantes.PI);
        System.out.println("E = " + Constantes.E);
    }
}
```

POO en Java: Elements estàtics

- Els valor inicial assignat a una constant (FINAL) no es pot canviar:

```
public class Main {

    private static final int CONSTANT_EXAMPLE = 333;

    public static void main(String[] args) {

        CONSTANT_EXAMPLE = 999;// Error! You can't assign a new value to a final variable!
    }
}
```

- Però no cal inicialitzar una constant immediatament. Això es pot fer més tard. Això si, el valor que se li va assignar inicialment romandrà igual per sempre.

```
public static void main(String[] args) {

    final int CONSTANT_EXAMPLE;

    CONSTANT_EXAMPLE = 999;// This is allowed
}
```

POO en Java: Pas per valor i pas per referència

- A JAVA, el pas de paràmetres als mètodes es realitza per valor.
  - Això implica que quan es passa una variable a un mètode, es passa una còpia del valor en lloc de la variable original.
  - Per tant, qualsevol canvi realitzat al paràmetre dins del mètode NO afecta el valor original de la variable fora del mètode.
- No obstant això, quan es passa un objecte a un mètode, s'està passant la referència de l'objecte, no una còpia de l'objecte en si.
  - Per tant, si passem un objecte com paràmetre, qualsevol canvi realitzat a l'objecte dins del mètode, SI afecta al objecte original fora del mètode.

POO en Java: Pas per valor i pas per referència

- En resum:
- A JAVA els **tipus primitius** (int, char, doouble, boolean,...) sempre es passen per valor als mètodes.
- A JAVA els **objectes** sempre es passen per referència als mètodes.

- Hi ha casos especials, com ara els Strings, ja que aquests són objectes, però són immutables (implica que no es pot modificar un String sense crear una nova instància).