4. Desenvolupament de classes

2024-2025

Continguts

- ✓ Concepte de classe.
- ✓ Estructura i membres d'una classe.
- ✓ Encapsulació.
- ✓ Llibreries.
- ✓ Tipus de variables Wrapper.*

1. Què és una classe ?

“**Les clases en Java (Java Class) son plantilles per a la creació d’objectes**”

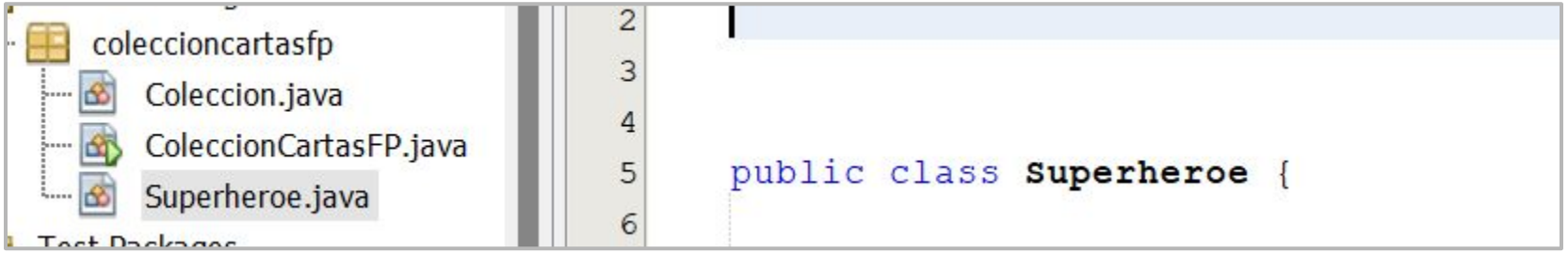
El llenguatge Java ens proporciona un **conjunt de classes ja creades** que es poden emprar directament per construir els nostres programes.

Un bon exemple que en vist durant el curs és la classe *Scanner*.

Però gairebé sempre és necessari generar classes noves, d’acord amb les necessitats de cada programa concret.

2. Estructura i membres d'una classe

Important: El nom del fitxer ha de ser igual que el nom de la classe java.



Com declarar una classe ?

Suposant que els [ ] són valors opcionals:

```
[ ] class NomClasse [ ] {

        [atributs]
        [mètodes]

}
```

Recorda que un constructor és considerat un mètode.

Atributs

Els atributs són declaracions de variables de tipus primitius o de referències a objectes d'altres classes, seguint la sintaxi següent:

```
[modificadors] <tipus> <etiqueta>  [= <valorInicial>] ;
```

- Normalment emprarem el modificador "private", excepte en casos especials, com la declaració de constants. En aquest cas, solem emprar "public static final".
- Podem observar com la definició d'una dada pot estar acompanyada d'una inicialització explícita [= <valorInicial>] opcionalment.

Mètodes

Els mètodes representen funcionalitats i operacions. La seva sintaxis:

```
[modificadors] <tipusRetorn> <nomMetode> ([llistaDeParametres]) {
        <declaracioDeVariablesLocals>
        <codiDelMetode>
}
```

- Normalment emprarem el modificador “public”.
- Els constructors NO tendran cap <tipusRetorn>.

Mètodes (paràmetres)

Els paràmetres són una sèrie de valors. Li hem d'especificar de quin tipus són i li hem de posar una etiqueta.

```
public void nomMetode(String x, int n) {
```

Quan cridam el mètode hem de donar valor a cada paràmetre respectant l'ordre de la declaració: *a.nomMetode("Joan", 21);*

Mètodes accessors

Totes les classes amb atributs privats acostumen a proporcionar uns mètodes de lectura (get) i escriptura (set). Això permet el seu ús i manipulació.

```
private int nom;

public int getNom(){
    return nom;
}
```

```
private String nom;

public void setNom(String n){
    nom = n;
}
```

Constructors

Els constructors són els mètodes que s'utilitzen per **crear els objectes** de la classe.

Sempre tenen el **mateix nom que la classe**. *(case sensitive)*

Cada classe, com a **mínim,** ha de tenir **un constructor**.

Els constructors permeten:

- Recollir valors (pas de paràmetres en el moment de construcció).
- Gestionar errors que puguin aparèixer en la fase d'inicialització.
- Aplicar processos, en els quals poden intervenir tot tipus de sentències (condicionals i repetitives).

Tipus de Constructors

Constructor buid
(sense paràmetres)

```
/* Constructor vacio */
public Superheroe(){

}
```

Constructor amb paràmetres

```
/* COnstructor con parámetros base */
public Superheroe(String nombre, int fuerza, int vida){
    this.nombre = nombre;
    this.fuerza = fuerza;
    this.vida = vida;
    this.rareza = "comun";
}
```

Emprar un constructor des del "main"

Podem emprar un constructor des de qualsevol altre mètode.

Com a exemple inicialitzarem el superheroi de l'exemple anterior des del "main".

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

    //Empram el constructor buid
    Superheroe spiderman = new Superheroe();

    //Empram el constructor amb paràmetres
    Superheroe thor = new Superheroe("Thor", 4, 8);
```

3. Encapsulació

- L'encapsulació és un principi fonamental de la programació orientada a objectes. Consisteix a **ocultar l'estat intern de l'objecte** i forçar a que tota la interacció es faci a través dels mètodes de l'objecte.

  (mètode de la caixa negra)

- Qualsevol ha de poder emprar una classe sense saber l'estructura interna.

- L'encapsulació també ens permet controlar que l'estat d'un objecte és consistent. *Exemple: a la classe Persona que no fiquin un DNI no vàlid.*

Modificadors d'accés

- **public** : Dona accés a tothom. Tant de lectura com escriptura.
- **private** : Prohibeix l'accés a tothom menys pels mètodes de la pròpia classe. És útil per definir mètodes intern que no volen ser exposats a altres classes.
- **protected** : Es comporta com a public per a les subclasses i les classes del mateix paquet (package) i com a private per a la resta de classes.
- **(sense modificador) :** Es comporta com a public per a les classes del mateix paquet (package) i com a private per a la resta de classes.

Modificadors d'accés

	Classe	Paquet	Subclasse	Tots els altres
public	**SI**	**SI**	**SI**	**SI**
protected	**SI**	**SI**	**SI**	**NO**
Sense modificador	**SI**	**SI**	**NO**	**NO**
private	**SI**	**NO**	**NO**	**NO**

Overloading (sobrecàrrega)

- La sobrecàrrega de mètodes és la funcionalitat que permet tenir **mètodes diferents amb un mateix nom**.
- La **llista de paràmetres** ha de ser suficientment **diferent** per permetre una determinació inequívoca del mètode que es crida.
- El compilador només pot **distingir** el mètode que es crida **a partir del nombre i tipus dels paràmetres** indicats en la crida.
- Es molt habitual que les classes tenguin **més d'un constructor i tots han de tenir el mateix nom**.

Paraula reservada: **static**

- Indica en comptes de pertànyer a una instància prèviament declarada, pertany a un tipus en sí mateix. No s'ha d'instanciar.

- S'utilitza per **variables o mètodes comuns a tots els objectes** de la classe.

- Un cas típic és l'ús d'un mètode per comptar tots els objectes d'una classe, també s'utilitza per aquells mètodes on no hi ha cap necessitat de crear l'objecte.

- Exemple per emprar un mètode o variable estàtica : *Math.max(5, 7);*

Paraula reservada: **this**

La paraula reservada “this” serveix per fer referència a l’objecte actual, sobre el qual s’està executant el mètode.

- Habitualment emprar “this” resulta redundant. Ja que es sap de quin mètode o variable s’està fent referència.

- Un cas típic on sí que resulta necessari és si li passem un paràmetre a un mètode o constructor amb el mateix nom que una altra variable de la classe.

  En aquest cas, la paraula "this" ens serveix per tenir la referència de quina és la variable de la classe.

Exemple de l'ús del **this**

```java
public class Usuario {

    private String nombre;
    private String email;
    private String contraseña;
    private boolean esEncriptada;

    public Usuario(String nombre, String email, String contraseña) {
        this.nombre = nombre;
        this.email = email;
        this.contraseña = contraseña;
        this.esEncriptada = false;
    }
```

4. Llibreries

- Java proporciona un mecanisme, anomenat package, per poder agrupar classes. Una llibreria es correspon amb un paquet. Aquest, tendrà un conjunt de classes.

- Podem emprar aquestes classes si importam el paquet al nostre projecte.

```
1  package introducciofp;
2
3  import java.util.ArrayList;
4  import java.util.Arrays;
5  import java.util.List;
6
7  //Podem importar totes les classes d'un paquet:
8  // import java.util.*;
9
```

5. Variables de tipus “Wrapper”

- Les classes “envoltorio” o “Wrapper”, ens permeten tractar els tipus primitius com si fossin objectes. Això ens serà d’ajuda en alguns casos on estem obligats a emprar Objectes i no tipus primitius.
- Hi ha 8 classes Wrapper. Una per cada tipus primitiu.
- Al ser un Objecte, totes elles començaran per majúscula.

Tipus de variables Wrapper

Wrapper	Tipus primitiu
Integer	int
Long	long
Float	float
Double	double
Character	char
Boolean	boolean
Byte	byte
Short	short

Mètodes propis dels Wrapper

Hi ha una serie de mètodes que hem estat emprant.

Per exemple:

```
int numero;
numero = Integer.parseInt("-124");
```

Ens permet convertir un String a un int

Exercici amb Wrapper

Desenvolupa un programa sobre el que, declara 8 variables, una de cada tipus de variable primitiu.

Instancia i inicialitza cada variable.

Mitjançant l'us de wrappers, passa les variables numériques a unes noves de tipus String i les variables de caràcter a noves variables numèriques.

Imprimeix les noves variables.