Unió Europea
Fons Social Europeu
L'FSE inverteix en el teu futur

GENERALITAT VALENCIANA

I.E.S. Maria Enríquez
Gandia

UF08 - POO I

- Teoria -

PROGRAMACIÓ
CFGS DAM

Autor:
Àngel Olmos Giner

segons el material de Carlos Cacho i Raquel Torres
(Programació) i Sergio Badal (Entorns de Desenvolupament)

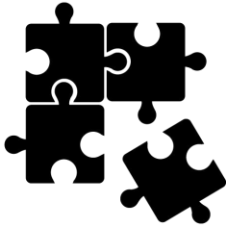
a.olmosginer@edu.gva.es
2022/2023

POO - I

ÍNDEX DE CONTINGUTS

1. INTRODUCCIÓ

INTRODUCCIÓ



Programació Orientada a Objectes (POO)

Dividir el problema en **objectes**

Cada objecte funciona de forma totalment independent

1. INTRODUCCIÓ

INTRODUCCIÓ

Una **classe** descriu un grup d'objectes que contenen una informació similar (atributs) i un comportament comú (mètodes).

Un **objecte** és un element del programa que posseeix les seues pròpies dades i el seu propi funcionament.

Abans de poder utilitzar un objecte, s'ha de definir la seua classe. La classe és la definició d'una mena d'objecte.

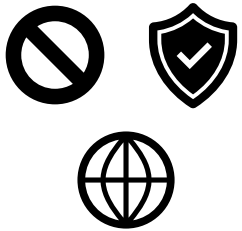
1. INTRODUCCIÓ

PROPIETATS DE LA POO

ENCAPSULAMENT

Una classe es compon tant de variables (**atributs**) com de funcions i procediments (**mètodes**). De fet no es poden definir variables ni mètodes fora d'una classe (no hi ha variables globals)

OCULTACIÓ



Hi ha una zona oculta al definir la classes (zona **privada / protegida**) que només és utilitzada per aquesta classe i per alguna classe relacionada.
Hi ha una zona **pública** (anomenada interfície de classe) que pot ser utilitzada per qualsevol altra classe

1. INTRODUCCIÓ

PROPIETATS DE LA POO

POLIMORFISME

Cada mètode d'una classe **pot tindre diverses definicions diferents**.

Per exemple, en el cas del joc parxís:

- *partida.iniciar(4)* comença una partida per a quatre jugadors
- *partida.iniciar("roig", "blau")* comença una partida de dos jugadors per als colors roig i blau

Aquestes són dues formes diferents d'emprar el mètode *iniciar*, que és polimòrfic.

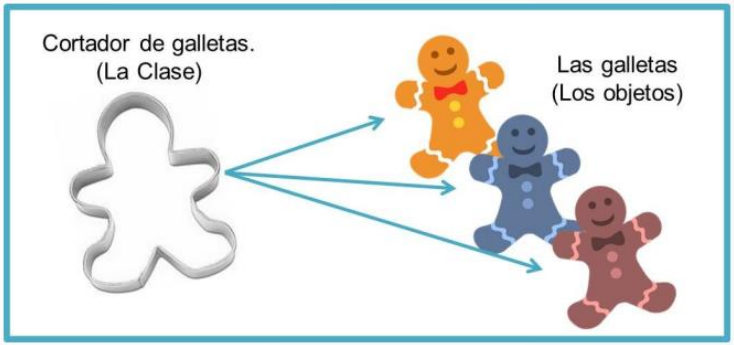
HERÈNCIA

Una classe pot **heretar** propietats (atributs i mètodes) **d'una altra classe** → POO - II

POO - I

ÍNDEX DE CONTINGUTS

2. FONAMENTS D'UNA CLASSE



Una **classe és com un motle**. A partir d'ella es poden crear objectes
Descriu un grup d'objectes que contenen una **informació similar** (atributs), un **comportament comú** (mètodes) i les mateixes **relacions amb altres objectes**

Abans de poder utilitzar un objecte s'ha de definir la classe a la qual pertany. La definició inclou:

- Atributs: Les variables membre de la classe. Poden ser *public, private o protected*
- Mètodes: Les funciones membre de la classe. Són les accions o operacions que pot realitzar la classe. Igual que els atributs poden ser *public, private o protected*

2. FONAMENTS D'UNA CLASSE

```
[accés] class nomDeClase {
        [accés] [static] tipus atribut1;
        [accés] [static] tipus atribut2;
        //…més atributs...

        [accés] [static] tipus métode1(llistaDeParàmetres) {
                //...codi del mètode...
        }
        [accés] [static] tipus mètode2(llistaDeParàmetres) {
                //...codi del mètode…
        }
        //… més mètodes...
}
```

2. FONAMENTS D'UNA CLASSE

- Cada classe **es crea en un arxiu Java diferent** (amb el mateix nom de la classe), i s'utilitzen fora de la classe
- Per exemple, podríem tindre:
  - un arxiu *Persona.java* (amb la classe *Persona* de l'exemple anterior)
  - a més d'un arxiu *Programa.java* que només tindrà la funció principal *main* des de la que podrem crear objectes de tipus *Persona*
- **Els atributs d'una classe es diuen variables d'instància** perquè cada objecte de la classe conté les seues pròpies variables
- La paraula opcional *static* serveix per a fer que el mètode o l'atribut a la qual precedeix es puga utilitzar de manera genèrica
- És a dir, **es pot fer ús d'una cridada *static* sense necessitat de crear una instància** per a accedir, per exemple, a algun mètode de la classe → ??

2. FONAMENTS D'UNA CLASSE

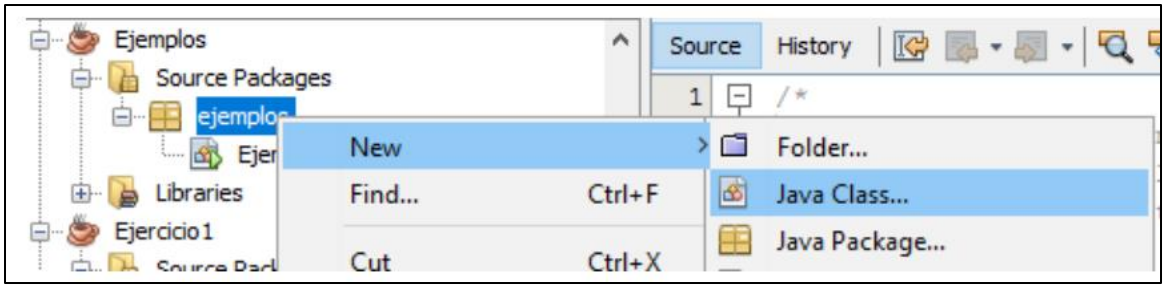
```
public class Persona {
    String nom;
    int edat;

    // Estableix el nom de la persona
    void setNom(String n) {
        nom = n;
    }

    // Estableix l'edat de la persona
    void setEdat(int e) {
        edat = e;
    }

    // Retorna el nom de la persona
    String getNom() {
        return nom;
    }

    // Retorna l'edat de la persona
    int getEdat() {
        return edat;
    }

    // Mostra el seu nom per pantalla
    void imprimeNom() {
        System.out.println(nom);
    }

    // Retorna true si és major d'edat, false en cas contrari
    boolean esMajorEdat() {
        return (edat >= 18)
    }
}
```

La classe *Persona* ens servirà per a crear tants objectes *Persona* com necessitem, cadascun amb el seu *nom* i *edat*

2. FONAMENTS D'UNA CLASSE

```
public class Persona {
        String nom;
        int edat;

        // Retorna l'edat de la persona
        int getEdat() {
                return edat;
        }

        // Retorna el nom de la persona
        String getNom() {
                return nom;
        }
}
```

DIY



La classe *Persona* ens servirà per a crear tants objectes *Persona* com necessitem, cadascun amb el seu *nom* i *edat*

POO - I

ÍNDEX DE CONTINGUTS

3. OBJECTES

INSTANCIACIÓ D'UN OBJECTE

- Quan creem una **classe**, definim un **nou tipus de dades** que pot utilitzar-se **per a crear** (instanciar) **objectes** d'eixa classe
- La instanciació es fa de la següent manera:
  - En primer lloc, cal declarar una variable del tipus de la classe
  - En segon lloc, es necessitarà una còpia física de l'objecte i assignar-la a la variable
- Això es fa utilitzant **l'operador *new*** que assigna dinàmicament memòria a un objecte i **retorna una referència** → recordeu algun exemple ja utilitzat?
- Aquesta referència s'emmagatzema en una variable

DIY

```
Persona p1;
p1 = new Persona();
```

Forma reduïda??

```
Persona p1 = new Persona();
```

3. OBJECTES

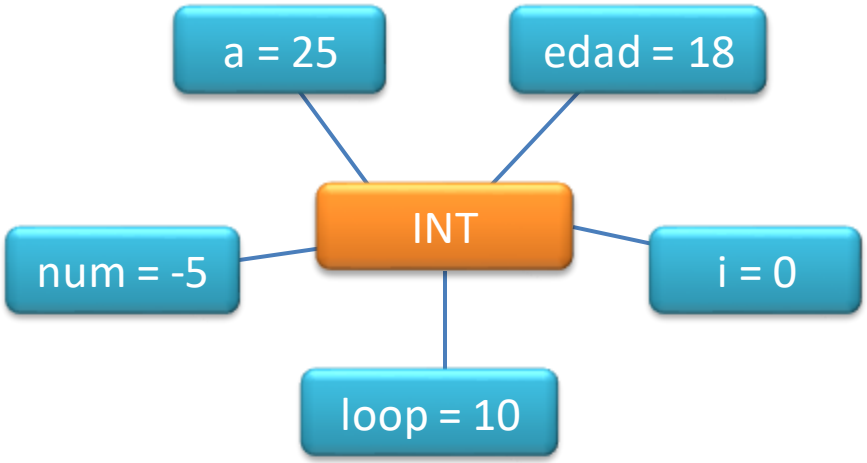
INSTANCIACIÓ D'UN OBJECTE

- Els **objectes interactuen els uns amb els altres**, en contraposició a la visió tradicional en la qual un programa és una col·lecció de subrutines (funcions o procediments), o simplement una llista d'instruccions
- Cada objecte és capaç de **rebre missatges, processar dades i enviar missatges** a altres objectes
- Les dades de cada objecte són **individuals i independent dels altres objectes**

3. OBJECTES

INSTANCIACIÓ D'UN OBJECTE

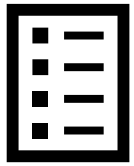
- Els conceptes de **classe i objectes** són anàlegs als de **tipus de dades i variable**
- Definida una classe podem crear objectes d'aqueixa classe, igual que disposant d'un determinat tipus de dada (per exemple el tipus enter), podem definir variables d'aquest tipus



3. OBJECTES

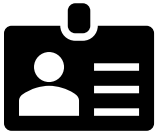
CARACTERÍSTIQUES D'UN OBJECTE

- **Estat**: definit pels valors de les propietats (atributs) i per les relacions que pot tindre amb altres objectes



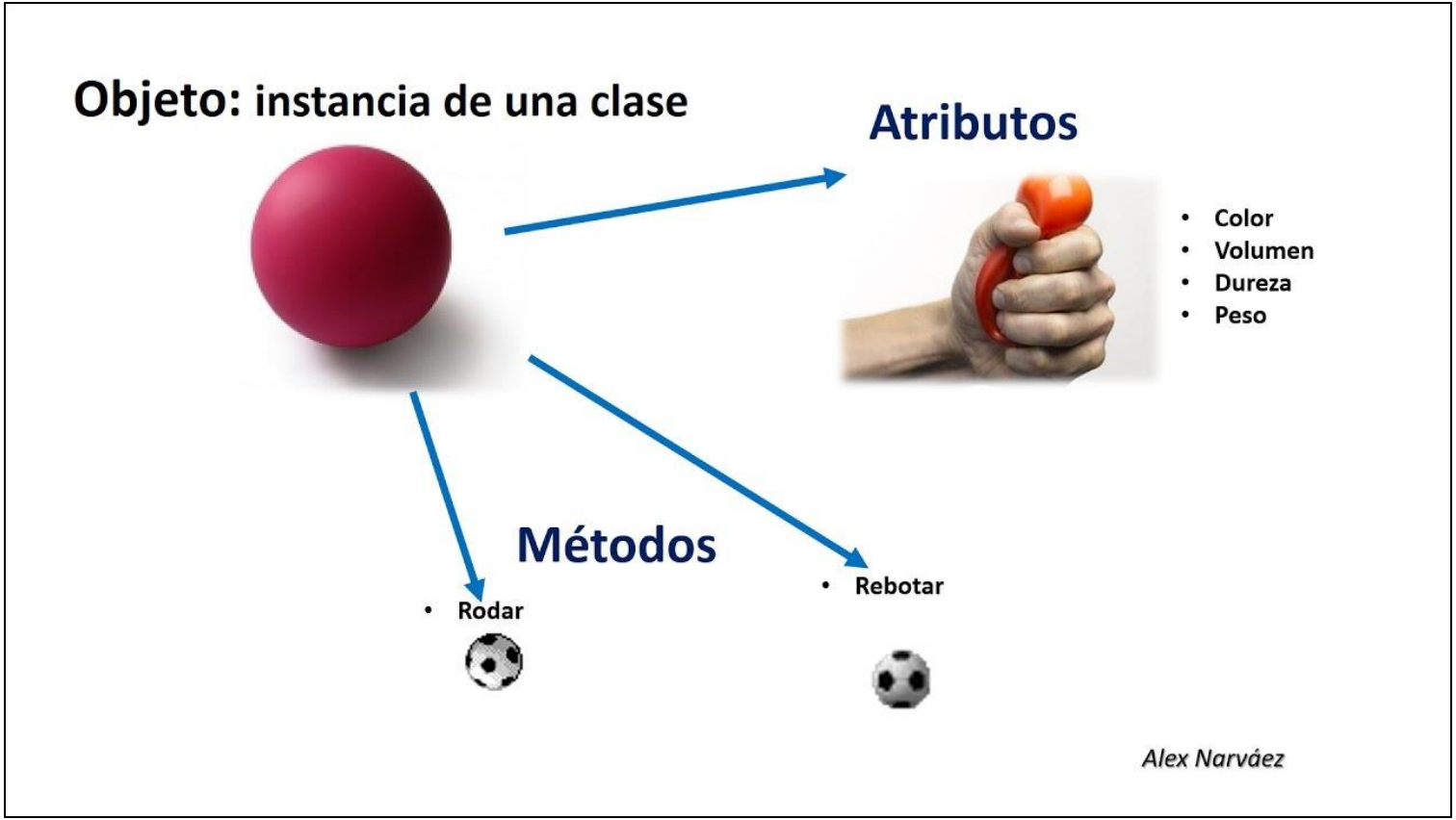
- **Comportament**: indica tot el que objecte pot fer, i es defineix com el conjunt d'operacions (funcions/mètodes) del model

- **Identitat**: significa que cada objecte és únic, encara que tinga el mateix estat que uns altres



3. OBJECTES

CARACTERÍSTIQUES D'UN OBJECTE

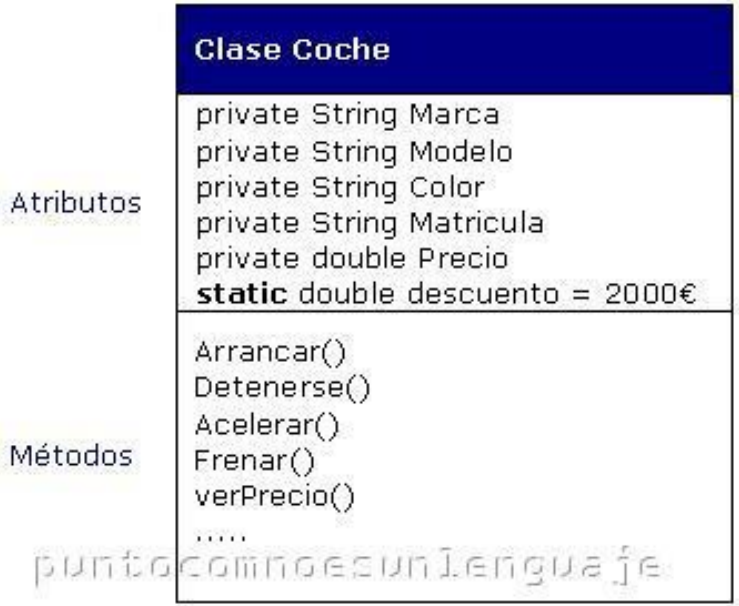


3. OBJECTES

CARACTERÍSTIQUES D'UN OBJECTE

3. OBJECTES

CARACTERÍSTIQUES D'UN OBJECTE



3. OBJECTES

ASSIGNACIÓ DE VARIABLES DE REFERÈNCIA

Què fa aquest codi?

```
Persona p1 = new Persona();
Persona p2 = p1;
```

- Podríem pensar que a *p2* se li assigna una còpia de l'objecte *p1*, però no és així
- El que succeeix és que la referència *p1* es copia en *p2*, per la qual cosa *p2* permetrà accedir al mateix objecte referenciat per *p1*
- Per tant, qualsevol canvi que es faça a l'objecte referenciat a través de *p2* afectarà l'objecte *p1*

3. OBJECTES

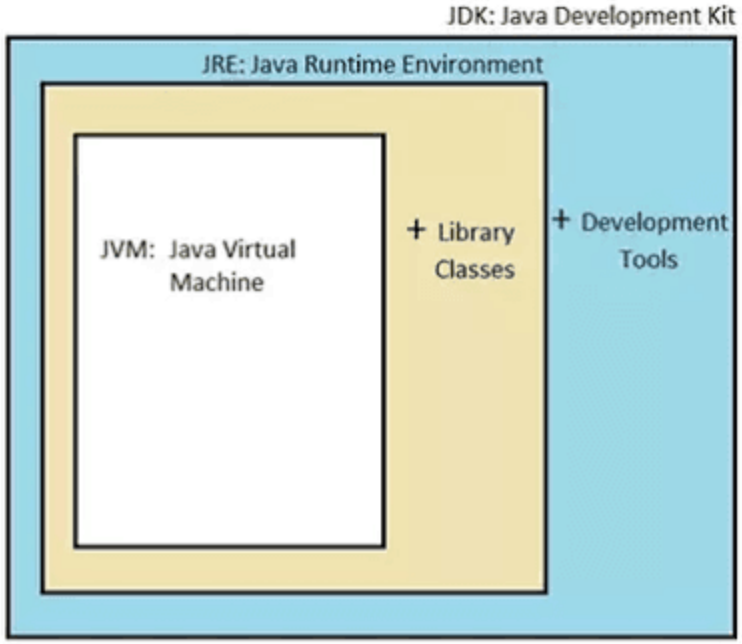
RECOL·LECTOR DE FEM

- *Garbage collector*: s'encarrega de **gestionar els objectes que es deixen d'usar i d'eliminar-los de memòria**
- Aquest procés és **automàtic i impredictible** i treballa en un fil (*thread*) de baixa prioritat
- Actua quan detecta que un objecte fa massa temps que no s'utilitza
- Aquesta eliminació **depèn de la màquina virtual de Java** (JVM)
- En quasi totes les JVM la recol·lecció es realitza **en un determinat lapse de temps**
- Recordeu què era la JVM?

3. OBJECTES

RECOL·LECTOR DE FEM

- **JVM** – Java Virtual Machine: **responsable de ejecutar el programa** Java línea por línea → AKA intérprete
- **JRE** – Java Runtime Environment: proporciona un **entorno para ejecutar** el programa Java. Lo utilizan aquellos que solo desean ejecutar los programas → los usuarios finales
- **JDK –** Java Development Kit: kit que proporciona el entorno para desarrollar y ejecutar el programa Java



JDK = JRE + Development Tools
JRE = JVM + Library Classes

Diferencias entre JDK, JRE y JVM: https://javadesdecero.es/fundamentos/diferencias-jdk-jre-jvm/

POO - I

ÍNDEX DE CONTINGUTS

4. VISIBILITAT

Els membres d'una classe (atributs i mètodes) poden definir-se com:

- ***public***: Es pot utilitzar des de qualsevol classe
- ***private***: Només pot utilitzar-ho la mateixa classe
- ***protected***: Pot utilitzar-ho la mateixa classe i també les subclasses heretades (POO - II)

Els principis de la POO diuen que:

- per a mantenir l'encapsulació en els objectes, hem d'aplicar l'especificador ***public*** a les funcions que formen la interfície pública
- i denegar l'accés a les dades usades per aquestes funcions mitjançant l'especificador ***private***

En un **paquet**, que és un agrupament lògic de classes en un mateix directori, els atributs i els mètodes d'aquestes classes són **públics** per defecte per a la resta de classes existents en el mateix paquet, i **privats** per a qualsevol classe que es trobe fora (llevat que s'especifique el contrari).

4. VISIBILITAT
Accés a atributs

- Per tal de poder accedir als valors dels atributs d'un objecte (lectura / escriptura), cal utilitzar la següent nomenclatura:

  nom_objecte.nom_atribut

- Si per exemple volem guardar el valor d'un atribut en una variable, faríem:

  [tipus] nom_variable = nom_objecte.nom_atribut;

- Si volem escriure un determinat atribut d'un objecte, faríem ...:

  nom_objecte.nom_atribut = valor_a_escriure;

4. VISIBILITAT
Accés a atributs

```
public class Persona {
    String nom;
    int edat;

    // Estableix el nom de la persona
    void setNom(String n) {
        nom = n;
    }

    // Estableix l'edat de la persona
    void setEdat(int e) {
        edat = e;
    }

    // Retorna el nom de la persona
    String getNom() {
        return nom;
    }

    // Retorna l'edat de la persona
    int getEdat() {
        return edat;
    }

    // Mostra el seu nom per pantalla
    void imprimeNom() {
        System.out.println(nom);
    }

    // Retorna true si és major d'edat, false en cas contrari
    boolean es
        re
    }
}
```

```
Persona Thomas = new Persona();
Thomas.edat = 84;
int edat = Thomas.edat;
Thomas.nom = "Thomas Alva Edison";
System.out.println(Thomas.nom + " va morir
a l'edat de " + edat + " anys");
```

DIY

4. VISIBILITAT
Accés a atributs



Fes el bloc d'Exercicis A

POO - I

ÍNDEX DE CONTINGUTS

5. MÈTODES

Creació

Qualsevol tipus vàlid, **incloent els tipus de classes que crea el programador** o *void*

public, *protected* o *private*

```
[visibilitat] tipus nom_del_mètode([llista de paràmetres]) {
        // cos del mètode
        [return element_retornat]
}
```

Parells tipus i identificador separats per comes

- La majoria de les vegades és convenient crear mètodes per a accedir als atributs de la classe de forma que s'oculta o abstrau l'estructura interna de les dades → ???
- També solen definir-se mètodes per a ser utilitzats internament per la classe

5. MÈTODES

Creació

```
public class Alumne {
    // atributs
    private String nom;
    private String cognom;
    private int edat;

    // mètodes d'accés
    public void setNom(String inNom){
        nom = inNom;
    }
    public void setCognom(String inCognom){
        cognom = inCognom;
    }
    public void setEdat(int inEdat){
        edat = inEdat;
    }
    public String toString(){
        return (nom + " té " + edat + " anys");
    }
}
```

DIY

.. i després instància un objecte amb nom “Roger1rC”

5. MÈTODES
Ús de mètodes

Per tal d'utilitzar els mètodes d'un objecte o d'una classe estàtica, escriurem:

[tipus retornat] nom_objecte.nom_mètode([param1, param2, ...])

```
public class Alumne {
    // atributs
    private String nom;
    private String cognom;
    private int edat;

    // mètodes d'accés
    public void setNom(String inNom){
        nom = inNom;
    }
    public void setCognom(String inCognom){
        cognom = inCognom;
    }
    public void setEdat(int inEdat){
        edat = inEdat;
    }
    public String toString(){
        return (nom + " té " + edat + " anys");
    }
}
```

Roger1rC.setNombre("Roger");
Roger1rC.setEdad(19);
System.out.println(**Roger1rC**.toString());

DIY

5. MÈTODES
Ús de mètodes

Exemple 1: Implementa la classe *Article* (una altra, aquesta no és la de l'exercici del butlletí). Aquesta classe representa cada objecte amb els següents atributs:

- *cod* (cadena caràcters)
- *titol* (cadena caràcters)
- *format* (cadena caràcters)
- *preu_lloguer* (nombre real)

També defineix tres mètodes que permeten calcular, respectivament:

- el preu de lloguer d'un dia: retorna el valor del lloguer de l'article
- de dos dies: lloguer de dos dies fent un descompte del 20%
- una setmana: lloguer per a 5 dies (sense descompte)

Instància 2 objectes de la classe *Article*, dona valor als seus atributs i mostra per pantalla el valor dels 3 tipus de lloguers de cada objecte.

5. MÈTODES

Ús de mètodes

```java
public class Article {
    String cod;
    String titol;
    String format;
    double preu_lloguer;

    double preu1(){
        return preu_lloguer;
    }
    double preu2(){
        return preu_lloguer * 1.8;
    }
    double preu_setmana(){
        return preu_lloguer * 5;
    }
}
```

```java
Article article1 = new Article();
Article article2 = new Article();

article1.cod = "001";
article1.titol = "Títol 1";
article1.format = "DVD";
article1.preu_lloguer = 2.50;

article2.cod = "002";
article2.titol = "Títol 2";
article2.format = "DVD";
article2.preu_lloguer = 3.40;

System.out.println("Lloguer art: " + article1.cod + ", 1 dia:" + article1.preu1());
System.out.println("Lloguer art: " + article1.cod + ", 2 dies:" + article1.preu2());
System.out.println("Lloguer art: " + article1.cod + ", 1 setmana:" + article1.preu_setmana());

System.out.println("Lloguer art: " + article2.cod + ", 1 dia:" + article2.preu1());
System.out.println("Lloguer art: " + article2.cod + ", 2 dies:" + article2.preu2());
System.out.println("Lloguer art: " + article2.cod + ", 1 setmana:" + article2.preu_setmana());
```

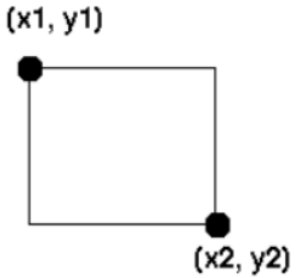
5. MÈTODES
Ús de mètodes

Exemple 2: En aquest exemple anem a implementar la classe *Quadrat* (en realitat és un rectangle, però l'anomenem quadrat per a diferenciar-ho de la classe *Rectangle* dels exercicis), que representa "quadrats" mitjançant dues coordenades 2D, i defineix tres mètodes que permeten calcular, respectivament:

- la diagonal = $(Lx^2 + Ly^2)^{1/2}$ --> Lx, Ly: longitud horitzontal / vertical
- el perímetre = 4 * (diagonal / $2^{1/2}$)
- l'àrea = 0.5 * $diagonal^2$

Crea 2 "quadrats" de diferents coordenades i després visualitza el seu perímetre i la seua àrea.

NOTA: El criteri de representació pren les coordenades horitzontals (x) creixents d'esquerra a dreta, i les verticals (y) creixents de dalt a baix



5. MÈTODES
Ús de mètodes

Exemple 2

```java
public class Quadrat {
    double x1, x2, y1, y2;

    double diagonal(){
        return Math.sqrt(Math.pow(x2-x1,2)+Math.pow(y2-y1,2));
    }
    double perimetre(){
        return 4 * (diagonal() / Math.sqrt(2));
    }
    double area(){
        return 0.5 * Math.pow(diagonal(),2);
    }
}
```

```java
Quadrat Q1 = new Quadrat();
Quadrat Q2 = new Quadrat();

Q1.x1 = 2; Q1.y1 = 2; Q1.x2 = 4; Q1.y2 = 4;
Q2.x1 = 1; Q2.y1 = 1; Q2.x2 = 5; Q2.y2 = 5;

System.out.println("El perímetre de Q1 és: " + Q1.perimetre());
System.out.println("L'àrea de Q1 és: " + Q1.area());
System.out.println("El perímetre de Q2 és: " + Q2.perimetre());
System.out.println("L'àrea de Q2 és: " + Q2.area());
```

POO - I

ÍNDEX DE CONTINGUTS

6. CONSTRUCTORS

És una tasca bastant pesada inicialitzar totes les variables d'una classe cada vegada que es crea una instància (un objecte)

Com que el procés és tan comú, Java permet que els objectes s'inicialitzen quan es creen → mitjançant un **constructor**

```
Article article1 = new Article();
Article article2 = new Article();

article1.cod = "001";
article1.titol = "Títol 1";
article1.format = "DVD";
article1.preu_lloguer = 2.50;

article2.cod = "002";
article2.titol = "Títol 2";
article2.format = "DVD";
article2.preu_lloguer = 3.40;
```

Un **constructor** és un mètode especial que no retorna mai un valor, sempre retorna una referència a una instància de la classe i és anomenat automàticament en crear un objecte d'una classe, és a dir en usar la instrucció *new*.

6. CONSTRUCTORS

- Un *constructor* no és més que un mètode que té el mateix nom que la classe
- Constructors per defecte: quan no es desitja passar els paràmetres d'inicialització en construir la instància de la classe (*new nom_classe()*), s'inicialitzen les dades assignant-los els següents valors per defecte:
  - zero '0' a les variables numèriques
  - *null* a totes les referències a objectes
- Una vegada es definisca un constructor per a la classe, es deixa d'utilitzar el constructor per defecte
- Una classe pot tindre tants constructors com vulgues sempre que tinguen diferent número i/o tipus de paràmetres (perquè no hi haja ambigüitat en com utilitzar-ho) → POLIMORFISME

Existeix la referència ***this*** que apunta a l'objecte instanciat. S'utilitza dins de les funcions per a accedir als atributs i mètodes de propi l'objecte:

```
this.edat = 25;
this.distancia(x,y);
```

6. CONSTRUCTORS

Exemple 3: En aquest exemple veurem com crear un constructor de classe. Per a això ens basarem en la classe *Article* creada anteriorment i li afegirem el constructor

```
public Article(String cod, String titol, String format, double preu_lloguer){
    // Amb 'this' el que fem és accedir als atributs de l'objecte de la classe
    this.cod = cod;
    this.titol = titol;
    this.format = format;
    this.preu_lloguer = preu_lloguer;
    // "this.cod" és l'atribut de l'objecte, mentre de "cod" és el valor que
    // l'usuari passa quan instància l'objecte. El nom dels atributs del
    // constructor poder ser qualsevol, no cal que tinguen el mateix nom que
    // els atributs de la classe
}
```

Crea ara 2 objectes *Article* utilitzant el constructor i mostra per pantalla el valor dels seus atributs.

6. CONSTRUCTORS

Exemple 3: En aquest exemple veurem com crear un constructor de classe. Per a això ens basarem en la classe *Article* creada anteriorment i li afegirem el constructor

```
Article article1 = new Article("003","Titanic","VHS",1.25);
Article article2 = new Article("004","Star Wars - Episodio I","BlueRay",9.90);

System.out.println("Lloguer art: " + article1.titol + ", 1 dia: " + article1.preu1());
System.out.println("Lloguer art: " + article1.titol + ", 2 dies: " + article1.preu2());
System.out.println("Lloguer art: " + article1.titol + ", 1 setmana: " + article1.preu_setmana());

System.out.println("Lloguer art: " + article2.titol + ", 1 dia: " + article2.preu1());
System.out.println("Lloguer art: " + article2.titol + ", 2 dies: " + article2.preu2());
System.out.println("Lloguer art: " + article2.titol + ", 1 setmana: " + article2.preu_setmana());
```

Crea ara 2 objectes *Article* utilitzant el constructor i mostra per pantalla el valor dels seus atributs.

6. CONSTRUCTORS

Fes el bloc d'Exercicis B

POO - I

ÍNDEX DE CONTINGUTS

7. CONSTANTS DE CLASSE I D'OBJECTE

RECORDATORI:

- Atributs i mètodes d'una classe es diuen ...
- Els atributs d'un objecte es diuen perquè cada instància de la classe (cada objecte) conté els seus atributs propis → les dades de cada objecte són individuals i independent dels altres objectes
- Els atributs i mètodes *static* d'una classe (es poden *utilitzar de manera genèrica*) es diuen

- Els membres constants es defineixen a Java a través de la paraula reservada ...

7. CONSTANTS DE CLASSE I D'OBJECTE

Identifica:

1. *membres d'una classe*
2. *variables d'instància*
3. *atributs i mètodes de classe*

7. CONSTANTS DE CLASSE I D'OBJECTE

En Java hi ha 4 possibilitats per a definir atributs constants:

- Atributs ***static***: prenen valors comuns a tots els objectes i es poden modificar. L'atribut pertany a la classe (no a l'objecte). Com el valor s'emmagatzema en la classe NO pren valors diferents en cada objecte. Poden utilitzar-se encara que no existisca cap objecte instanciat. D'es d'un mètode *static* no es poden utilitzar atributs ni mètodes *no-static*. Al contrari sí que és possible

- Atributs ***final***: són valors constants, però poden ser diferents en cada objecte. El seu valor s'inicialitza en la fase de construcció de l'objecte i ja no es pot modificar

- Atributs ***static final***: combinen les característiques de *static* i *final* (comú i no modificable)

- Resta d'atributs (sense *static* ni final): atributs modificables i diferents per a cada objecte

7. CONSTANTS DE CLASSE I D'OBJECTE

Exemple 4: Per a il·lustrar l'ús dels qualificadors *final* i *static* dins de la nostra classe *Article*, definirem tres nous atributs:

- La constant *IVA,* comú a tots els objectes i accessible sense instanciar un objecte
- Un atribut privat que comptabilitzarà el nombre d'instàncies i accessible per a tots els objectes de tipus *Article*
- Un atribut constant String, diferent per a cada objecte, que permeta identificar-los

Fes els canvis que calga a la classe *Article* i modifica el programa principal per a fer ús d'aquests nous atributs on es vegen els canvis

7. CONSTANTS DE CLASSE I D'OBJECTE

Exemple 4

```
public static final double IVA = 0.16;
private static int num = 0;
final String identificador;

// Constructor de la classe
public Article(String cod, String titol, String format, double preu_lloguer, String id){
    // Amb 'this' el que fem és accedir als atributs de l'objecte de la classe
    this.cod = cod;
    this.titol = titol;
    this.format = format;
    this.preu_lloguer = preu_lloguer;
    this.identificador = id;
```

```
Article article1 = new Article("003","Titanic","VHS",1.25,"TIT");
Article article2 = new Article("004","Star Wars - Episodio I","BlueRay",9.90,"SW-I");

System.out.println("L'IVA de tots els articles és: " + Article.IVA);

System.out.println("Lloguer art: " + article1.identificador + ", 1 dia: " + article2.preu1());
System.out.println("Lloguer art: " + article2.identificador + ", 2 dies: " + article2.preu2());
```

POO - I

ÍNDEX DE CONTINGUTS

8. GETTERS I SETTERS

Un pilar fonamental de la programació orientada a objectes (POO) és l'**encapsulament:**

private

*"Es denomina encapsulament a **l'ocultació de l'estat**, és a dir, de les dades membre d'un objecte de manera **que només es puga canviar mitjançant les operacions definides** per a aqueix objecte. Cada objecte està aïllat de l'exterior. **L'aïllament protegeix les dades** associades a un objecte contra la seua modificació per qui no tinga dret a accedir a ells [...] D'altra banda **s'evita que l'usuari puga canviar el seu estat de maneres imprevistes** i incontrolades."* Font: Wikipedia

mètodes

8. GETTERS I SETTERS

PASSOS:

1. **Ocultar tots els atributs** (fer-los ***private***) perquè no es puguen modificar directament des de fora de la classe
2. Afegir **mètodes *getters* (*get* = agafar) i *setters* (*set* = fixar)** visibles (***public***) que permeten llegir i modificar aquests atributs des de fora de la classe
3. Al tractar-se de mètodes, podrem incloure el codi necessari per a controlar **l'accés als atributs i protegir-les d'usos incorrectes**

8. GETTERS I SETTERS

Exemple 5: modifica la classe *Article* per a encapsular els seus atributs.
Crea *getters* per a:

- cod
- titol
- format
- preu_lloguer

i un sol *setter* que modifique els 4 atributs a la vegada.

Una vegada creats, modifica els objectes *Article* amb el *setter* i mostra el resultat dels canvis amb els *getters*.

8. GETTERS I SETTERS

Exemple 5

```
// Getters
public String getCod(){
    return this.cod;
}
public String getTitol(){
    return this.titol;
}
public String getFormat(){
    this.format = format;
    this.preu_lloguer = preu_lloguer;
}
```

```
public class Article {
    private String cod;
    private String titol;
    private String format;
    private double preu_lloguer
```

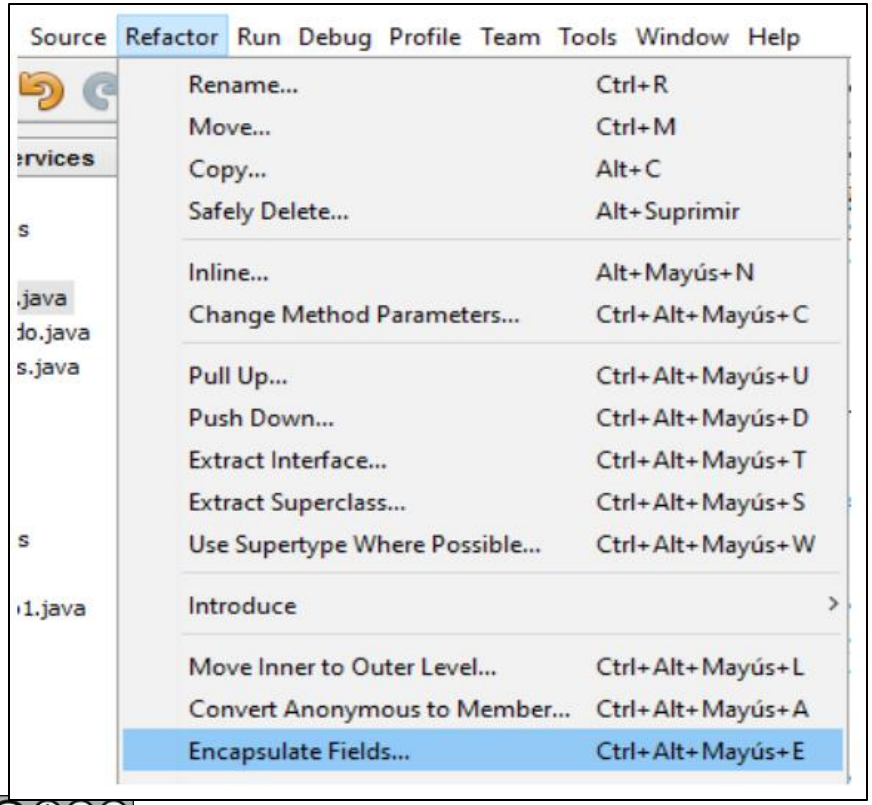
```
Article article5 = new Article("005","Rain Man","VHS",6.25,"RAMA");
Article article6 = new Article("006","El gran Lebowski","BETA",4.90,"EGLE");

System.out.println("Lloguer art: " + article5.getTitol() + ", 1 setmana: " + article5.preu_setmana());
System.out.println("Lloguer art: " + article6.getTitol() + ", 1 dia: " + article6.preu1());

article6.modificarTots("006","El gran Lebowski - VOSE","DVD",3.5);
System.out.println("Lloguer art: " + article6.getTitol() + ", 1 dia: " + article6.preu1());
```

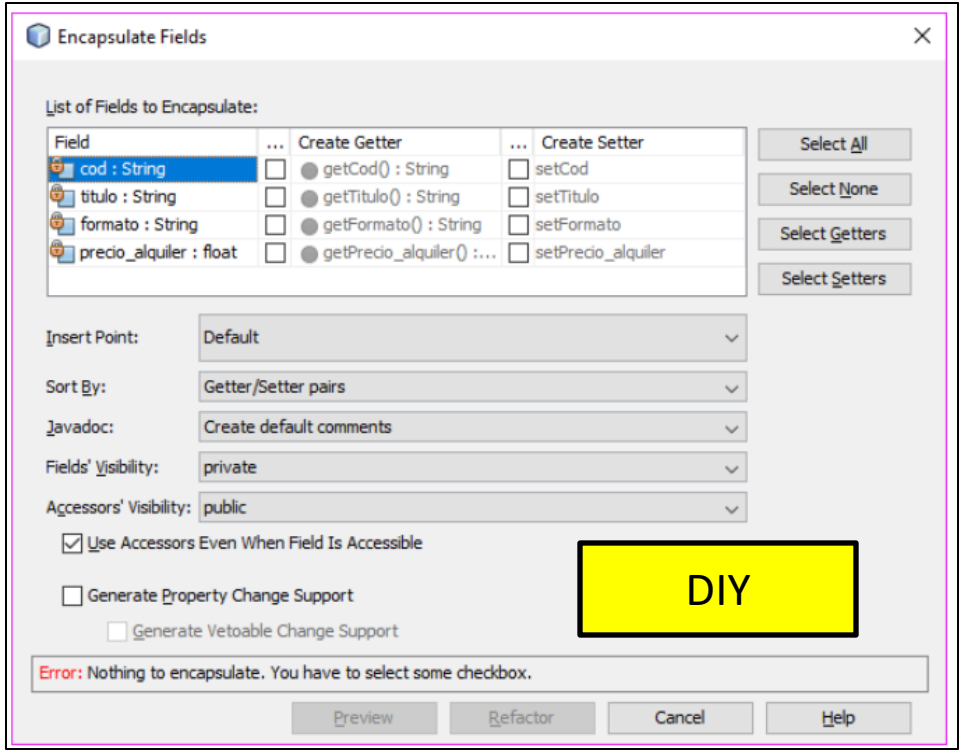
8. GETTERS I SETTERS

Fes el bloc d'Exercicis C

8. GETTERS I SETTERS

Una altra manera de crear els mètodes *getters* i *setters* és utilitzant el *refactor* de Netbeans





POO - I

ÍNDEX DE CONTINGUTS

9. ARRAYS D'OBJECTES

```
NomDeLaClasse [] Objectes;
NomDeLaClasse Objectes[];
```

```
Objectes = new NomDeLaClasse[n]; // n és el número de referències a Objectes
```

```
NomDeLaClasse [] Objectes = new NomDeLaClasse[n];
NomDeLaClasse Objectes[] = new NomDeLaClasse[n];
```

Per a crear els objectes cal **cridar** explícitamente al **constructor per cada element creat**:

```
Objecte[i] = new NomDeLaClasse();
```

9. ARRAYS D'OBJECTES

Exemple 6: Crearem un array que continga 10 objectes de tipus *Article*. Els instanciarem tots amb noms genèrics, consecutius i preus aleatoris [0,10[.
Primer creem un vector de *Articles* anomenat *elsMeusArticles* (és un array de referències a objectes *Article*).
Després instanciarem els 10 objectes, guardant cadascun dels objectes referenciats.
La classe *Article* no es veuria modificada.

Finalment mostra per pantalla els 3 preus de lloguer de cada articles (amb només 2 decimals) i utilitzant un bucle for-each

9. ARRAYS D'OBJECTES

Exemple 6

```
Article [] elsMeusArticles = new Article[10];

// Creem l'objecte per a cadascun dels elements de l'array
for (int cont = 0; cont < elsMeusArticles.length; cont++) {
    elsMeusArticles[cont] = new Article("00" + cont,"Article " + cont,"DVD",Math.random()*10,"ID" + cont);
}

// Recorrem l'array i mostrem la informacio amb un bucle for
for (int cont = 0; cont < elsMeusArticles.length; cont++) {
    System.out.println("Codi d'article: " + elsMeusArticles[cont].getCod());
    System.out.printf("Lloguer 1 dia: %.2f %n", elsMeusArticles[cont].preu1());
    System.out.printf("Lloguer 2 dies: %.2f %n", elsMeusArticles[cont].preu2());
    System.out.printf("Lloguer 1 setmana: %.2f %n", elsMeusArticles[cont].preu_setmana());
}

// Recorrem l'array i mostrem la informacio amb un bucle for-each
for (Article art : elsMeusArticles) {
    System.out.println("Codi d'article: " + art.getCod());
    System.out.printf("Lloguer 1 dia: %.2f %n", art.preu1());
    System.out.printf("Lloguer 2 dies: %.2f %n", art.preu2());
    System.out.printf("Lloguer 1 setmana: %.2f %n", art.preu_setmana());
}
```

9. ARRAYS D'OBJECTES

Fes el bloc d'Exercicis D i E

POO - I

Autor:
Àngel Olmos Giner
segons el material de Carlos Cacho i Raquel Torres