CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UNITAT 8
PROGRAMACIÓ ORIENTADA A OBJECTES I
EXERCICIS

PROGRAMACIÓ
CFGS DAW

Autors:
Carlos Cacho y Raquel Torres
Revisat per:
Lionel Tarazon - lionel.tarazon@ceedcv.es
Fco. Javier Valero – franciscojavier.valero@ceedcv.es
José Manuel Martí - josemanuel.marti@ceedcv.es

2021/2022

Llicència

Aquests exercicis estan pensats per a començar **creant classes molt senzilles** (apartat A) **que després aniràs millorant i ampliant** (apartats B, C…) de manera que practiques i aprengues a poc a poc els aspectes fonamentals de la POO. **El més important és que entengues què està passant**. Si no ho tens clar "juga" amb el codi, fes proves "a vore què succeeix si...", revisa la teoria, etc. Si encara així no ho entens, pregunta en el fòrum (còpia-pega el codi si pertoca).

APARTAT A – CLASSES SIMPLES AMB ATRIBUTS

En cada exercici deus crear un programa amb dues classes: una classe principal (pots anomenar-la per exemple UD8_ProgramaPunt, segons l'exercici) que només contindrà la funció main, a més d'una **altra classe** (amb els seus atributs i mètodes) que utilitzarás des del main de la classe principal per a fer proves sobre el seu funcionament.

En aquest apartat les classes només contindran atributs (variables) i farem algunes proves senzilles amb elles per a entendre com s'instàncien objectes i s'accedeix als seus atributs. Ara com ara no utilitzes cap modificador en els atributs (public, private, static, final...).

Exercici A1 – Punt

Crea un programa amb una classe anomenada Punt que representarà un punt de dues dimensions en un pla. Sols contindrá dos atributs sencers anomenats **x** e **y** (coordenades).

En el main de la classe principal instància 3 objectes Punt amb les coordenades (5,0), (10,10) i (-3, 7). Mostra per pantalla les seues coordenades (utilitza un println per a cada punt). Modifica totes les coordenades (prova diferents operadors com = + - += *=...) i torna a imprimir-les per pantalla.

Exercici A2 – Persona

Crea un programa amb una classe anomenada Persona que representarà les dades principals d'una persona: **dni**, **nom**, **cognoms** i **edat**.

En el main de la classe principal instància dos objectes de la classe Persona. Després, demana per teclat les dades de totes dues persones (guarda'ls en els objectes). Finalment, imprimeix dos missatges per pantalla (un per objecte) amb un missatge de l'estil "Ferrandis Lujan Garcia amb DNI … és / no és major d'edat".

Exercici A3 – Rectangle

Crea un programa amb una classe anomenada Rectangle que representarà un rectangle mitjançant dues coordenades (x1,y1) i (x2,y2) en un pla, per la qual cosa la classe haurà de tindre quatre atributs sencers: **x1**, **y1**, **x2**, **y2**.

En el main de la classe principal instància 2 objectes Rectangle en (0,0)(5,5) i (7,9)(2,3). Mostra per pantalla les seues coordenades, perímetres (suma de costats) i àrees (ample x alt). Modifica totes les coordenades com consideres i torna a imprimir coordenades, perímetres i àrees.

Exercici A4 – Artícle

Crea un programa amb una classe anomenada Artícle amb els següents atributs: **nom**, **preu** (sense IVA), **iva** (sempre serà 21) i **cuantsQueden** (representa quants queden en el magatzem).

En el main de la classe principal instància un objecte de la classe article. Assigna-li valors a tots els seus atributs (els que vulgues) i mostra per pantalla un missatge de l'estil "Pijama - Preu:10€ - IVA:21% - PVP:12,1€" (el PVP és el preu de venda al públic, és a dir, el preu amb IVA). Després, canvia el preu i torna a imprimir el missatge.

APARTAT B – CONSTRUCTORS

El constructor és el mètode que s'executa quan s'instància un objecte. Si no està definit Java executarà un constructor per defecte que crearà l'objecte i inicialitzarà totes les seues variables a zero, però esta no és una bona pràctica. És millor definir nosaltres on constructor que controle què ha de succeir quan es cree l'objecte.

En aquest apartat tens que modificar els programes de l'apartat anterior (o fes una còpia del projecte si el prefereixes) **i fer els canvis indicats**.

Exercici B1 – Punt

Afig a la classe Punt un constructor amb paràmetres que copie les coordenades passades com a argument als atributs de l'objecte. Així:

```
public Punt(int x, int y){
    this.x = x;
    this.y = y;
}
```

Copiem els valors passats com a argument als atributs de l'objecte. Tingues en compte que **int x** i **int y** són variables locals del mètode, NO són els atributs del objecte. Per a fer referència als atributs de l'objecte cal utilitzar **this**.

Fixa't que ja no serà possible fer Punt p = new Punt(). Ara serà obligatori fer per exemple Punt p = new Punt(2, 7). En l'apartat A havies de recordar-te d'asignar valors a x i y després de crear un punt, la qual cosa no és una bona idea en projectes grans amb centenars d'objectes (és molt fàcil equivocar-se). Ara és impossible equivocar-se perquè Java no et deixarà. Hem assegurat que tots els punts sempre tindran coordenades.

Corregeix el main i utilitza el constructor amb paràmetres per a instanciar els objectes, passant-li com a argument els valors desitjats.

Exercici B2 – Persona

Afig a Persona el constructor de baix i corregeix el main per a utilitzarlo:

```
public Persona(String dni, String nom, String cognoms, int edat) {
    this.dni = dni;
    this.nom = nom;
    this.cognoms = cognoms;
    this.edat = edat;
}
```

Tingues en compte que no és obligatori que els paràmetres del constructor es criden igual que els atributs de l'objecte (en tal cas no seria necessari utilitzar **this**). Podríem fer-ho així:

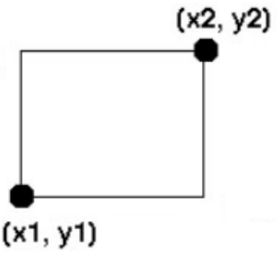
```
public Persona(String id, String nom, String cog, int e) {
        dni = id;
        nom = nom;
        cognoms = cog;
        edat = e;
}
```

Tampoc és obligatori passar al constructor tots els atributs de la classe. Podriem decidir per exemple que en el nostre programari totes les persones han de tindre nom, cognoms i edat, però no és obligatori el DNI (nounats i xiquets). Aquest constructor també seria vàlid:

```
public Persona(String nom, String cog, int e) {
        nom = nom;
        cognoms = cog;
        edat = e;
}
```

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).

Exercici B3 – Rectangle



En el nostre programari necessitem assegurar-nos que la coordenada (x1,y1) represente la cantonada inferior esquerra i la (x2,y2) la superior dreta del rectangle, com en el dibuix.

Afig a Rectangle un constructor amb els 4 paràmetres. Inclou un if que comprove els valors **(*)**. Si són vàlids guardarà els paràmetres en l'objecte.

Si no ho són mostrarà un missatge de l'estil "ERROR al instanciar Rectangle..." utilitzant System.err.println(…). No podrem evitar que s'instancie l'objecte però almenys avisarem per pantalla.

Corregeix el main per a utilitzar aquest constructor. Hauria de mostrar un missatge d'error.

(*) Pista: És suficient amb un if ((condició) && (condició))

Exercici B4 – Article

Afig un constructor amb 4 paràmetres que assigne valors a nom, preu, iva i cuantsQueden. Aquest constructor haurà de mostrar un missatge d'error si algun dels valors nom, preu, iva o cuantsQueden no són vàlids. Què coindicions creus que podrien determinar si son vàlids o no? Raona-ho i implementa el codi.

Corregeix el main i prova de crear diversos articles. Introdueix alguns amb valors incorrectes per a comprovar si avisa de l'error.

APARTAT C – GETTERS I SETTERS

Un pilar fonamental de la programació orientada a objectes (POO) és el encapsulament:

> *"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, eliminant efectes secundaris i interaccions. D'aquesta manera l'usuari de la classe pot obviar la implementació dels mètodes i propietats per a concentrar-se només en com usar-los.*
>
> *D'altra banda s'evita que l'usuari puga canviar el seu estat de maneres imprevistes i incontrolades."* Font: Wikipedia

Per això, una pràctica molt habitual en POO consisteix en **ocultar tots els atributs** (fer-los **private**) perquè no es puguen modificar directament des de fora de la *clase .En el seu lloc afegirem **mètodes getters (get = agafar) i setters (set = fixar)** visibles (**public**) que permeten llegir i modificar aquests atributs des de fora de la classe. La clau està en el fet que en tractar-se de mètodes podrem incloure el codi necessari per a controlar **l'accés als atributs i protegir-les d'usos incorrectes**.

En aquest apartat tens que modificar els programes de l'apartat anterior (o fes una còpia del projecte si el prefereixes) **i fer els canvis indicats**.

Exercici C1 – Punt

Modifica els atributs de Punt perquè siguen **private**. Fixa't que des del main ja no et deixarà utilitzar ni modificar els atributs x e y dels objectes.

Afegirem els **getteres**: **int getX()** i **int getY()** que retornaran els valors de x e i respectivament. És una forma indirecta de llegir els seus valors.

Afegirem també els **setters**: **void setX(int x)** i **void setY(int y)** que copiaran el valor passat com a paràmetre als atributs de la classe.

Tant getters com a setters han de ser **public**.

Corregeix el main per a utilitzar els getters i setters. Prova a instanciar diversos objectes, mostrar els seus valors per pantalla, modificar-los, etc.

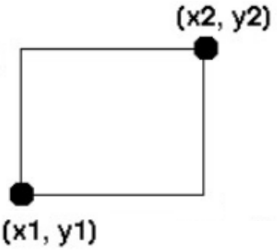
```
public class Punto {

    private int x;
    private int y;

    public Punto(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX(){
        return x;
    }

    public int getY(){
        return y;
    }

    public void setX(int x) {
        this.x = x;
    }

    public void setY(int y) {
        this.x = y;
    }
}
```

Exercici C2 – Persona

Aplica l'encapsulament bàsic a la classe Persona: Declara tots els seus atributs com private i crea tots els getters i setters necessaris (un get i un set per atribut).

Corregeix el main per a utilitzar els getters i setters. Prova a instanciar diversos objectes, mostrar els seus valors per pantalla, modificar-los, etc.

Exercici C3 – Rectangle



Aplica l'encapsulament bàsic a la classe Rectangle: Declara tots els seus atributs com private i crea tots els getters i setters necessaris (un get i un set per atribut).

Recordes la condició explicada en B3? Tindràs que programar els setters de manera que comproven el valor passat com a argument abans de guardar-lo en l'objecte. Si no fora correcte es mostrarà un missatge d'error (i NO es guardarà el valor).

Corregeix el main per a utilitzar els getters i setters. Prova a instanciar diversos objectes, mostrar els seus valors, modificar-los, etc. Prova diversos valors erronis per a comprovar si funciona.

Exercici C4 – Article

Aplica l'encapsulament bàsic a la classe Article: Declara tots els seus atributs com private i crea tots els getters i setters necessaris (un get i un set per atribut).

Programa els setters perquè comproven els valors i els guarden en l'objecte sol si són correctes. En cas contrari mostra un missatge d'error.

APARTAT D – AFEGINT MÉTODES ÚTILS

Una classe ben dissenyada deuria incloure mètodes que realitzen operacions amb la informació dels objectes. D'aqueixa manera la classe disposarà de funcionalitats útils tant per a nosaltres com per a altres programadors. Això és una pràctica habitual i molt recomanable.

Per exemple, la classe Scanner té mètodes com getInt(), getDouble, getLine(), etc. que algú ha programat i podrem utilitzar quan els necessitem sense haver de programar-ho nosaltres ni preocupar-nos per com funcionen internament. El mateix succeeix amb la classe String (charAt, substring, toCharArray, etc.), la classe Math (random, min, max, abs, etc.), la classe Arrays, etc.

Totes estes són classes que Java incorpora per defecte (n'hi ha milers). Existeixen moltes altres que permeten fer tot tipus de coses com crear interfícies gràfiques amb botons, treballar amb imatges, llegir i escriure en arxius, reproduir música, enviar o rebre dades a través de la xarxa, etc.

En aquesta unitat estem aprenent a dissenyar i programar les nostres pròpies classes, els fonaments de la Programació Orientada a Objectes, necessari en qualsevol projecte programari d'una certa envergadura.

Recorda que els mètodes poden ser **private** (només poden utilitzar-se des de dins de la classe) o **public** (poden utilitzar-se des de fora, formen part de l'interfície de la classe).

En aquest apartat tens que modificar els programes de l'apartat anterior (o fes una còpia del projecte si ho prefereixes) **i anyadir-los els mètodes que s'indiquen.**

Exercici D1 – Punt

Afig a la classe Punt els següents mètodes públics:

- **public void imprimeix()** // Imprimeix per pantalla les coordenades. Exemple: "(7, -5)"
- **public void setXY(int x, int y)** // Modifica totes dues coordenades. És com un setter doble.
- **public void desplaça(int dx, int dy)** // Desplaça el punt la quantitat (dx,dy) indicada. Exemple: Si el punt (1,1) es desplaça (2,5) llavors estarà en (3,6).
- **public int distancia(Punt p)** // Calcula i retorna la distància entre el propi objecte (this) i un altre objecte (Punt p) que es passa com a paràmetre: distància entre dues coordenades.

Prova d'utilitzar aquests mètodes des del main per a comprovar el seu funcionament.

Exercici D2 – Persona

Afig a la classe Persona els següents mètodes públics:

- **public void imprimeix()** // Imprimeix la informació de l'objecte: "DNI:… Nom:… etc."
- **public boolean esMajorEdat()** // Retorna true si és major d'edat (false si no).
- **public boolean esJubilat()** // Retorna true si té 65 anys o més (false si no).
- **public int diferenciaEdat(Persona p)** // Retorna la diferència d'edat entre 'this' i p.

Prova d'utilitzar aquests mètodes des del main per a comprovar el seu funcionament.

Exercici D3 – Rectangle

Afig a la classe Rectngle mètodes públics amb les següents funcionalitats:

- Mètode per a imprimir la informació del rectangle per pantalla.
- Mètodes setters dobles i quàdruples: **setX1Y1**, **set X2Y2** i **setAll(…)**.
- Mètodes **getPerimetre** i **getArea** que calculen i retornen el perímetre i àrea de l'objecte.

Prova d'utilitzar aquests mètodes des del main per a comprovar el seu funcionament.

Exercici D4 – Article

Afig a la classe Article mètodes públics amb les següents funcionalitats:

- Mètode per a imprimir la informació de l'article per pantalla.
- Mètode **getPVP** que retorne el preu de venda al públic (PVP) amb iva inclòs.
- Mètode **getPVPDescompte** que retorne el PVP amb un descompte passat com a argument.
- Mètode **vendre** que actualitza els atributs de l'objecte després de vendre una quantitat 'x' (si és possible). Retornarà true si ha sigut possible (false en cas contrari).
- Mètode **emmagatzema** que actualitza els atributs de l'objete després d'emmagatzemar una quantitat 'x' (si és possible). Retornarà true si ha sigut possible (false en cas contrari).

APARTAT E – STATIC I FINAL

Els modificadors **static** i **final** són opcionals, poden utilitzar-se tant en atributs com en mètodes i poden combinar-se tots dos:

- **static**: L'atribut o mètode pertany a la classe (no a l'objecte). Per això pot utilitzar-se sense instanciar cap objecte, des del nom de la classe: NomClasse.atribut o NomClasse.metode(...). Com el valor s'emmagatzema en la classe NO pren valors diferents en cada objecte (com sí que succeeix amb els atributs 'normals' no static).

  Per exemple:

  - L'atribut salariMinim d'Empleat (ss comú per a tots, pot canviar).
  - Mètodes útils com Arrays.fill(…) o String.valueOf(…) que podem utilitzar directament des de les classes Arrays i String sense instanciar un objecte.

    És important saber que d*esde un mètode *static no es poden utilitzar atributs ni mètodes no *static. Al contrari sí que és possible.
- **final**: Un atribut final no es pot modificar. Pot tindre valors diferents en cada objecte, però ha de fixar-se el seu valor en el constructor. Un mètode final no es pot redefinir en una sub-classe heretada (veurem herència en la següent unitat).

  Per exemple:

  - El DNI de la classe Persona. No pot canviar i cada objecte té el seu.
  - La combinació de static i final combina totes dues característiques.

  Per exemple:

  - L'atribut Math.PI és static i final. Pertany a la classe y no pot canviar.

Exercici E1– Punt

Necessitem un mètode que ens permeta crear un objecte Punt amb coordenades aleatòries. Aquesta funcionalitat no depén de cap objecte concret per la qual cosa serà estàtica. Haurà de crear un nou Punt (utilitza el constructor) amb x e i entre -100 i 100, i després retornar-lo (amb return).

- public static Punt creaPuntoAleatorio()

Prova'l en el main per a comprovar que funciona. Crea diversos punts aleatoris amb Punt.creaPuntAleatori() i imprimeix el seu valor per pantalla.

Exercici E2 – Persona

El DNI d'una persona no pot variar. Afig el modificador final a l'atribut dni i assegura't que es guarde el seu valor en el constructor. Lleva el mètode setDNI(…) que de totes maneres ja no es podrà utilitzar perquè Java no et deixarà modificar l'atribut dni.

La majoria d'edat als 18 anys és un valor comú a totes les persones i no pot variar. Crea un nou atribut anomenat majoriaEdad que siga static i final. Hauràs d'inicialitzar-lo a 18 en la declaració. Utilitza-ho en el mètode que comprova si una persona és major d'edat.

Crea un mètode **static boolean validarDNI(String dni)** que retorne true si dni és vàlid (té 8 números i una lletra). Si no, retornarà false. Utilitza'l en el constructor per a comprovar el dni (si no és vàlid, mostra un missatge d'error i no guardes els valors).

Realitza algunes proves en el main per a comprovar el funcionament dels canvis realitzats. També pots utilitzar Persona.validarDNI(…) per exemple per a comprovar si uns DNI introduïts per teclat són vàlids o no (sense necessitat de crear cap objecte).

Exercici E3 – Rectangle

Necessitem fer alguns canvis perquè totes les coordenades estiguen entre (0,0) i (100,100). Afig a la classe Rectangle dos atributs anomenats min i max. Aquests valors són comuns a tots els objectes i no poden variar. Pensa que modificaions necessites afegir a min i max.

Utilitza min i max en el constructor i en els setters per a comprovar els valors (com de costum, si no són correctes mostra un missatge d'error i aplicacions els canvis).

També necessitem un mètode no constructor per a crear rectangles aleatoris. Implementa-ho.

Realitza proves en el main per a comprovar el seu funcionament.

Exercici E4– Article

A Espanya existeixen tres tipus d'IVA segons la mena de producte:

- L'IVA general (21%): per a la majoria de productes a la venda.
- L'IVA reduït (10%): hostaleria, transport, habitatge, etc.
- L'IVA super reduït (4%): aliments bàsics, llibres, medicaments, etc.

Aquests tres tipus d'IVA no poden variar i a cada article se li aplicarà un dels tres.

Raona quins canvis seria necessari realitzar a la classe Article i implementa'ls.