Programació Orientada a Objectes

1.- Introducció

Java és 100% Orientat a Objectes, però encara no sabem el que és un objecte. Este tema teòric és fonamental, ja que entendre estos conceptes donen sentit a este curs.

La Programació Orientada a Objectes (POO) és un paradigma de la programació diferent al mètode clàssic o estructurat, perquè utilitza objectes, les seues propietats i el seu comportament per a resoldre problemes i generar programes i aplicacions.

Els objectes són totalment independents entre si, esta característica dels objectes fa que els problemes siguen més senzills, ja permetre dividir el problema, també augmenta la modularitat dels programes i la seua reutilització.

La **POO utilitza tècniques noves com el polimorfisme, l'encapsulament, l'herència**, etc. La majoria de llenguatges d'última generació permeten a més de la programació clàssica la POO per tant pot entendre's la POO com una evolució de la programació clàssica.

Per tant:

- Un **objecte** és un element del programa que posseeix les seues **pròpies dades i el seu propi funcionament.**
- Una **classe** descriu un **grup d'objectes** que contenen una **informació similar** (atributs) i un **comportament comú** (mètodes).

Abans de poder utilitzar un objecte, s'ha de definir la seua classe. Per tant, definir una classe és indicar com funciona un determinat tipus d'objectes, a partir d'esta definició podrem crear objectes de la classe.

Les propietats de la POO són les següents:

- **Encapsulament**. Una classe es compon tant de variables (**atributs**) com de funcions i procediments (**mètodes**). De fet no es poden definir variables (ni funcions) fora d'una classe (és a dir no hi ha variables globals).
- **Ocultació**. Hi ha una zona oculta al definir la classe (**zona privada**) que només és utilitzada per aqueixa classe i per alguna classe relacionada. Hi ha una **zona pública** (anomenada també interfície de la classe) que pot ser utilitzada per qualsevol part del codi.
- **Polimorfisme**. Cada mètode d'una classe pot tindre diverses definicions diferents. Per tant es pot donar la mateixa ordre a diversos objectes per a que responguen de manera diferent.
- **Herència**. Una classe pot heretar propietats (atributs i mètodes) d'una altra.

2.- Classes, Objectes i Instàncies

El primer concepte i més important de la POO és la distinció entre **classe** i **objecte**.

Una classe és una plantilla. Defineix de manera **genèrica** com seran els objectes d'un determinat tipus. A partir de la classe podem crear **Objectes**

Per exemple, imaginem que en un programa o joc volem definir una classe per a representar persones.

A esta classe l'anomenem **Persona** i té una sèrie **d'atributs** com **Nom**, **Cognoms** o **Edat** (que normalment són **propietats**), i una sèrie de **comportaments** que poden tindre, com **Parlar()**, **Caminar()** o **Menjar()** i que s'implementen com a **mètodes de la classe** (funcions).

Persona
Atributs: Nom Cognoms Edat
Mètodes: Parlar() Caminar() Menjar()

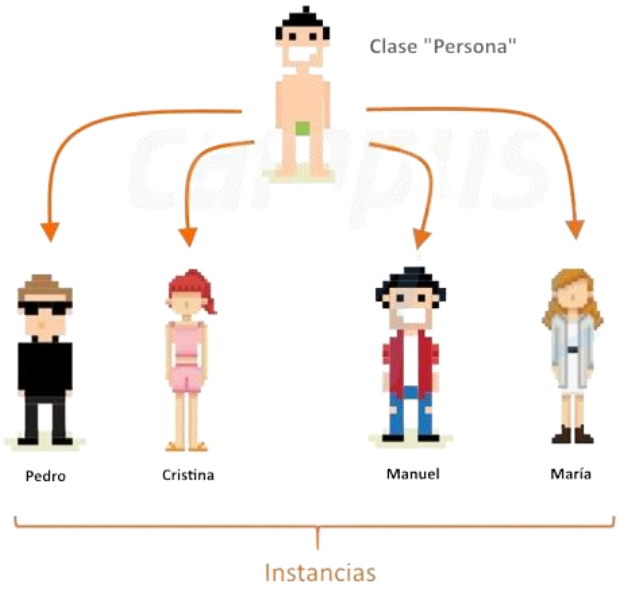
Una classe per si sola no serveix de res, perquè no és més que un concepte, sense entitat real. Per a poder utilitzar una classe en un programa el que cal fer és **instanciar-la**, es a dir, crear un **Objecte**.

Instanciar una classe consisteix a crear un nou objecte concret d'aquesta.

OBJECTE = Instància de classe

Per tant, un objecte és ja una **entitat concreta** que es crea a partir de la plantilla que és la classe. **Aquest nou objecte té ja "existència" real, ja que ocupa memòria i es pot utilitzar en el programa.**

Així un objecte pot ser una persona que es diu **Ada Byron** , de **46** anys, i que en el nostre programa podria **parlar**, **caminar** o **menjar**, que són els comportaments que estan definits en la classe.



La Classe defineix de manera genèrica com són les persones, i els Objectes són les persones concretes.

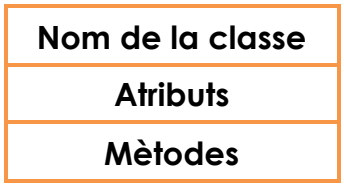
D'aquesta manera, si hem de manejar persones podem anar creant-les a mesura que les necessitem, i actuar sobre elles individualment.

Cada persona té les seues pròpies dades **(atributs)** i les seues pròpies accions **(mètodes).**

En definitiva, una classe és com un motlle, i a partir d'ella es poden crear OBJECTES, és a dir, abans de poder utilitzar un objecte, s'ha de definir la classe a la que pertany.

- La definició d'una classe inclou:
  - **Atributs**. Les variables membre de la classe. Poden ser: ***public*** (accessibles des d'una altra classe), ***private*** (només accessibles per codi de la seua pròpia classe) o ***protected*** (accessibles per les subclasses).
  - **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***.
  - Cada Objecte que creem (**instanciem**) d'una classe tindrà estos atributs i mètodes.

- En notació UML l'estructura d'una classe és la següent:



- La **sintaxi general** d'una classe en JAVA és:

```
[modificador d'accés] class nomDeClasse {

        [modificador d'accés] [static] tipus atribut1;
        [modificador d'accés] [static] tipus atribut2;

                //…més atributs...

        [modificador d'accés] [static] tipus mètode1 (llistaDeParametres) {
                //...codi del mètode...
        }

        [modificador d'accés] [static] tipus mètode2 (llistaDeParametres) {
                //...codi del mètode…
        }

        //… més mètodes...
}
```

- On ***[modificador d'accés]*** pot ser: **public, private** o **protected** , este camp és opcional, si es deixa en blanc equival a **public**. Estos modificadors es poden posar tant als **atributs** com als **mètodes**.

- [**static**]-> La paraula opcional **static** serveix per a fer que el mètode o l'atribut afectat es puga utilitzar de manera genèrica Els atributs i mètodes així definits es diuen **atributs de classe i mètodes de classe** respectivament.

- Els **atributs** i **mètodes** d'una classe es diuen **membres d'una classe**.

- Els **atributs** (variables) d'una classe es diuen **variables d'instància** perquè cada instància de la classe (és a dir, cada objecte de la classe), **conté les seus pròpies variable atribut**. Per tant, les dades de cada objecte són individuals i independent dels altres objectes.

- Es diu **mètode** a una funció d'una classe. La seua sintaxi és igual que les funcions del tema anterior:
  [modificador d'accés] **tipus_de_retorn** nomMètode (llistaDeParametres) {
  //...codi del mètode…
  }

Cal tindre en compte que la classe **Persona** ens servirà per a crear tants **objectes Persona** com necessitem, cadascun amb el seu **nom, cognoms** i **edat,** és a dir, amb els seus atributs.

Els **mètodes** ens permetran manipular les dades de cada objecte.

També és important entendre que **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 també un arxiu ***MainPersona.java*** (que només tindrà la funció principal a la qual estem acostumats).

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

Des de la funció **main**() de ***Programa.java*** podrem crear objectes de tipus **Persona** a més de qualsevol altre codi que necessitem.

Classe Persona (exemple).

Persona
Atributs: nom cognoms edat altura pes gènere
Mètodes: public void parlar() public void caminar() public void correr() public void menjar() public void dormir() public void Llistar() public boolean esMajorEdat()

```
package exemplePersona;

public class Persona {

  // Atributs o característiques
     dels objectes de la classe

      String nom;
      String cognom;
      int edat;
      double altura;
      double pes;
      char genere;

  // mètodes o funcions (accions)

  /* en este exemple, les accions dels objectes persona o MÈTODES, només
     mostren frases per pantalla */

  public void menjar() {
      System.out.println("Soc " + nom + " i estic menjant");
  }

  public void caminar() {
      System.out.println("Soc " + nom + " i estic caminant");
  }

  public void correr() {
    System.out.println("Sóc " + nom + " i estic corrent");
  }

  public void dormir() {
    System.out.println ("Soc " + nom + " i estic dormint");
  }

  public void llistar() {
    System.out.println("Em dic" + nom + "i tinc"+ edat + "anys"+ " i la meua
                       mida és" + altura +" cm");
  }

   public boolean esMajorEdadt() {
            if (edat >= 18) return true;
            return false;
  }

} // Fi classe Persona
```

3.- Instanciació d'un objecte (Operador NEW)

Al crear una classe, el que estem fent es **definir un tipus de dades** que pot utilitzar-se per a instanciar (crear) objectes d'eixa classe. Per a instanciar un objecte cal seguir el mateixos passos que al declarar qualsevol tipus de dades:

1.- Declarem una variable del tipus de la classe.

2.- Creem (construïm) l'objecte assignant el **CONSTRUCTOR** de la classe a la variable.

Per a crear l'objecte utilitzem l'operador new.

Exemple: (de la classe Persona anterior)

```
// Declarem la variable del tipus de la classe Persona
        Persona manolo;
// Creem un objecte Persona i l'assignem a la variable Jaume amb l'operador new
        manolo = new Persona();
```

Evidentment, també es pot fer en una única línia de codi:

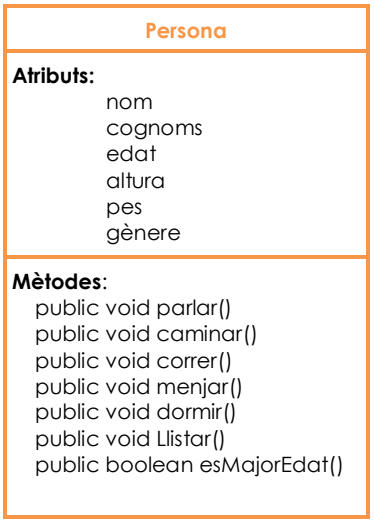
```
        Persona manolo = new Persona();
```

L'operador **new**, s'utilitza per a crear objectes, assigna dinàmicament (és a dir, durant temps d'execució) memòria a un objecte i **retorna una referència**.
Aquesta referència s'emmagatzema en una variable.

Una volta creat l'objecte, podem accedir als seus atributs i mètodes de la següent manera:

```
manolo.nom = "Manolo";
manolo.cognoms = "García";
manolo.edat = 45;
manolo.altura = 1.71;
manolo.pes = 70.5;
manolo.genere='M';

manolo.caminar();
manolo.llistar();
```



Exemple Classe Persona i Programa Principal

```
package exemplePersona;

public class Persona {

// Atributs o característiques dels objetes de la classe

        String nom;
        String cognoms;
        int edat;
        double altura;
        double pes;
        char genere;

// métodes o funcions (accions)

/*
* en aquest exemple, les accions dels objectes
* persona o MÈTODES, només mostren per pantalla
*/

  public void menjar() {

    System.out.println("Soc " + nom + " i estic
                        menjant");
  }

  public void caminar() {

    System.out.println("Soc " + nom + " i estic
                caminant");
  }

  public void correr() {

    System.out.println("Soc " + nom + "  i estic
                corrent");
  }

  public void dormir() {

     System.out.println("Soc " + nom + " i estic
                dormint");
  }

  public void llistar() {
     System.out.println("El menu nom és " + nom
     + " " + cognoms + " i tinc " + edat + " anys"
     + " i la meua mida és " + altura + " cm "
     + " pese " + pes + " i gènere " + genere);
  }

  public boolean esMajorEdadt() {
          if (edat >= 18)
                  return true;
          return false;
  }
}
```

```
package exemplePersona;

public class MainPersona {

   public static void main(String[] args) {

// Creem l'objecte Pepa, que serà una instància de la
classe PERSONA.

        Persona pepa = new Persona();

// Assignem valors als atributs de l'objecte (definits en la
classe PERSONA)

        pepa.altura = 170;
        pepa.nom = "Josefa Mª";
        pepa.cognoms = " A.V.";
        pepa.edat = 49;
        pepa.genere = 'F';

// Utilitzem mètodes de l'objecte pepa

        pepa.dormir();
        pepa.llistar();

// Constructor d'Objecte

        Persona p1 = new Persona();

// Atributs de l'Objecte
        p1.altura = 180;
        p1.edat = 20;
        p1.nom = "Maria Isabel";

// Mètodes de l'objecte
        p1.correr();
        p1.dormir();
        p1.llistar();

        boolean major = p1.esMajorEdadt();

        System.out.println("El valor 'esMajorEdat'
        de " + p1.nom + " és: " + major);

// Construïm objecte i utilitzem els seus atributs i //
mètodes.

        Persona dolo = new Persona();
        dolo.nom = "Dolores";
        dolo.llistar();

  }

} // Fí de MainPersona
```

```
Soc Josefa Mª i estic dormint
El meu nom és Josefa Mª  A.V. i tinc 49 anys i la meua mida és 170.0 cm  pese 0.0 i gènere F

Soc Maria Isabel i estic corrent
Soc Maria Isabel i estic dormint
El meu nom és Maria Isabel null i tinc 20 anys i la meua mida és 180.0 cm  pese 0.0 i gènere
El valor 'esMajorEdat' de Maria Isabel és: true

El meu nom és Dolores null i tinc 0 anys i la meua mida és 0.0 cm  pese 0.0 i gènere
```

4.- Assignació de variables de referència a Objectes

Les variables de referència a objectes permeten **accedir a l'objecte** (són una referència a l'objecte, no l'objecte).

```
Persona p1 = new Persona(); // p1 fa referència a un objecte creat amb new
```

Per això, actuen de manera diferent al que caldria esperar quan té lloc una assignació.

- Per exemple, si realitzem la següent assignació en variables de tipus primitius:

```
int var1, var2;

var1 = 3;
var2 = var1;
var2 = 7;

System.out.println("El valor de var1 és: " + var1 + " i el de var2 és: " + var2);
```

Ens donaria com a resultat:

```
El valor de var1 és: 3 i el de var2 és: 7
```

Amb objectes no funciona igual, en el següent fragment de codi (de la classe Persona anterior)

```
Persona dona = new Persona();
Persona home = new Persona();

dona.nom = "Antonia";
dona.genere = 'F';

home = dona;

dona.llistar();
home.llistar();
```

Ens donaria com a resultat:

```
El meu nom és Antonia null i tinc 0 anys i la meua mida és 0.0 cm  pese 0.0 i gènere F
El meu nom és Antonia null i tinc 0 anys i la meua mida és 0.0 cm  pese 0.0 i gènere F
```

Si ara modifiquem els atributs de l'objecte home:

```
home.nom = "Pepe";
home.genere = 'M';

dona.llistar();
home.llistar();
```

```
El meu nom és Pepe null i tinc 0 anys i la meua mida és 0.0 cm  pese 0.0 i gènere M
El meu nom és Pepe null i tinc 0 anys i la meua mida és 0.0 cm  pese 0.0 i gènere M
```

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