# Chapitre V : Classes et Objets

Notions de POO

# Introduction

La Programmation Orientée Objet (POO) est un paradigme de programmation qui permet de structurer le code en organisant les données et les fonctions sous forme **d'objets.**

En Java, ce paradigme est au cœur du langage. Dans ce cours on abordera  les concepts fondamentaux de la POO et on verra vous comment les utiliser efficacement.

# Qu’est-ce qu’un objet ?

Le monde qui nous entoure est constitué d’objets. Ces objets ont deux caractéristiques :

- Un état
- Un comportement.

Exemples :

- Livre
    - état : auteur, titre, ISBN, …
    - comportement : estPrésent, achatExemplaire, …

- Voiture :

  - état : marque, couleur, nbre de places, …
  - Comportement : démarrer, accélérer, ralentir, …

**Objet informatique**

- maintient son état dans des variables (appelées ***champs ou attributs*** )
- implémente son comportement à l'aide de **méthodes**

objet informatique = regroupement logiciel de variables et de méthodes

# Concept de classe

Une classe est un modèle ou un plan qui définit les attributs et les méthodes d'un objet.

La classe peut être vue comme un moule, une fabrique d'objets.

L'état de chaque objet est indépendant des autres.

- Une classe permet d'instancier plusieurs objets
- Chaque objet est instance d'une seule classe

# Définition d'une classe

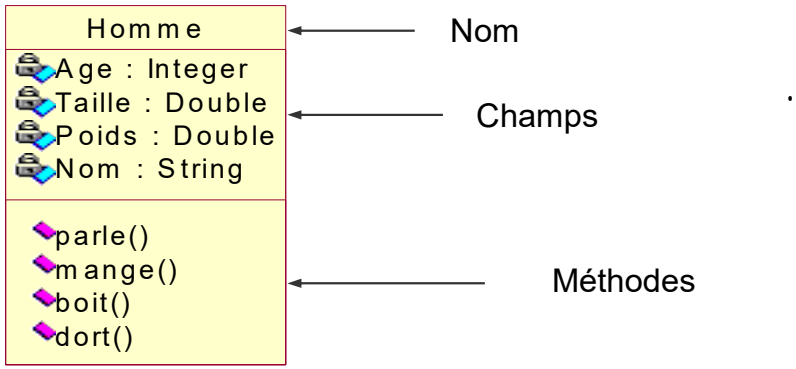
- Une classe est caractérisée par :

  - un nom

  - des champs ou attributs, nommés et ayant une valeur. Ils caractérisent l'état des objets pendant l'exécution du programme.

  - des méthodes représentant le comportement des objets de cette classe. Elles permettent de manipuler les champs des objets et caractérisent les actions pouvant être effectuées par les objets.

# Représentation graphique d'une classe (notation UML – *Unified Modeling Language*)



# Exemple

```
// Définition d'une classe
public class Voiture
{
// Attributs
        String couleur;
        String marque;
        int  annee ;
//Méthodes
        void démarrer() {
                System.out.println("La voiture démarre.");
        }

        void accélérer() {
                System.out.println("La voiture accélère.");
        }
}
```

Nom de la classe

Attributs

Méthodes

# L’instanciation

L’instanciation consiste en la concrétisation d’une classe en un objet « concret ».

- Instance d’une classe
  - représentant physique d'une classe
  - obtenu par moulage du dictionnaire des variables et détenant les valeurs de ces variables.
  - Son comportement est défini par les méthodes de sa classe
  - Par abus de langage « instance » = « objet »

Exemple :

Dans la classe Voiture, votre voiture est une instance particulière de la classe voiture.

Classe = concept, description

Objet = représentant ***concret*** d'une classe

# Les constructeurs

Dans la méthode main,

pour créer un nouvel objet, on utilise l'opérateur new.

L'appel de new pour créer un nouvel objet déclenche, dans l'ordre :

- L'allocation mémoire nécessaire au stockage de ce nouvel objet et l'initialisation par défaut de ses attributs,
- L'initialisation explicite des attributs, s'il y a lieu,
- L'exécution d'un constructeur.

- Un constructeur est une méthode d'initialisation

```
Public class Application
{
        Public static void main(String [] args)
        {
                Voiture titine=new Voiture();
                titine.couleur= "rouge";
        }
}
```

Le constructeur est ici celui défini par défaut

Lorsque l'initialisation explicite n'est pas possible, il est possible de réaliser l'initialisation par l'intermédiaire d'un constructeur.

- Le constructeur est une méthode :
  - de même nom que la classe,
  - sans type de retour.
- Toute classe possède au moins un constructeur. Si le programmeur ne l'écrit pas, il en existe un par défaut, sans paramètres, de code vide !

# Surcharge de constructeur

Pour une même classe, il peut y avoir plusieurs constructeurs, de signatures différentes (surcharge).

- L'appel de ces constructeurs est réalisé avec le new auquel on fait passer les paramètres.

```
Voiture titine = new Voiture("rouge", "toyota", 2023);
```

En fonction des paramètres passés lors de l'appel ( nombre et types), le bon constructeur est automatiquement déclenché. Ce mécanisme est appelé « lookup »

Remarque : si le programmeur crée un constructeur (même si c'est un constructeur avec paramètres), le constructeur par défaut n'est plus disponible. Attention aux erreurs de compilation !

```
Public Class Etudiant
{
        String nom;
        String prenom ;
        int age;
        public Etudiant()
        {
                this.nom=null;
                this.prenom=null;
                this.age=0;
        }
        public Etudiant(String nomEtud, String prenomEtud, int ageEtud)
        {
                this.nom=nomEtud;
                this.prenom=prenomEtud;
                this.age=ageEtud;
        }
}
```

Redéfinition d'un constructeur sans paramètres

On peut définir plusieurs constructeurs qui se différencient par leur signature

# Exemple

```
public class Main {
      public static void main(String[] args) {
      // Création d'un objet " maVoiture " de type Voiture
            Voiture maVoiture = new Voiture();
            maVoiture.couleur = "Noire" ;
            maVoiture.marque = "Renault" ;
            maVoiture.annee = 2020;

      // Utilisation des méthodes
      maVoiture.démarrer();
      maVoiture.accélérer();
   }
}
```