Introduction à la Programmation Orientée Objet

Rezak AZIZ

CNAM

Objectifs

- Introduire l'approche orientée objet.
- Comprendre les principes de la POO
  - Abstraction
  - Encapsulation
  - Héritage
  - Polymorphisme
- Comprendre la relation entre **classe** et **objet**.

Résolution d'un problème informatique

Pour résoudre un problème, il faut :

- Comprendre et analyser le problème
- Concevoir une solution
- Implémenter la solution
- Tester la solution

Ceci doit être fait en considérant :

- La maintenance
- L'évolution
- La réutilisation

du logiciel.

Paradigmes de programmation

- Selon Larousse, Un paradigme est un "Modèle théorique de pensée qui oriente la recherche et la réflexion scientifiques."
- Un **paradigme** = modèle de pensée pour analyser, concevoir et coder un programme.
- Quatre principaux paradigmes :
  1. **Procédural:** Fortran, C, Pascal, ...
  2. Fonctionnel : LISP,...
  3. Logique : Prolog, ...
  4. **Orienté Objet**: C++, Java, ...

Définition

La POO organise les programmes en collections d'objets coopératifs :

- Chaque objet est une instance d'une classe.
- Les classes forment une hiérarchie liée par héritage.

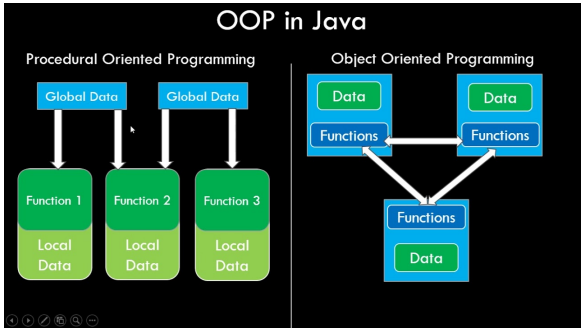
Procédural vs Orienté Objet

Procédural

- Programme = suite d'instructions séquentielles.
- Découpage en fonctions.
- Manipuler des Variables globales et locales.

Orienté Objet

- Programme = collection d'objets en interaction.
- Centré sur les données.
- Coopération via messages.



Exemple Orienté Objet

- Objet `Conducteur` demande à l'objet `Voiture` de démarrer.
- Interaction = **message** envoyé d'un objet à un autre.

Les objets communiquent entre eux via des messages (appel de méthodes).

```
class Voiture {
   int vitesse;
   void demarrer() {
      this.vitesse = 40;
   }
}
class Conducteur {
   String nom = "John";
   void conduire(Voiture v) {
      v.demarrer();
   }
}
```

Qu'est-ce qu'un objet ?

Définition

Un objet regroupe des données (**état**) et des traitements (**comportement**).

Exemples :

- Voiture : marque, couleur (état) ; démarrer, avancer (comportement).
- Étudiant : nom, matricule (état) ; étudier, passer examen (comportement).

Objet = Identité + État + Comportement

- **Identité** : chaque objet est unique.
- **État** : valeurs de ses attributs.
- **Comportement** : méthodes qu'il peut exécuter.

Deux objets peuvent avoir le même état mais rester distincts.

Qu'est-ce qu'une classe ?

Définition

Une **classe** est un modèle (ou moule) décrivant les caractéristiques communes d'objets.

Exemple :

- **Attributs** : marque, couleur, année
- **Méthodes** : démarrer(), avancer(), freiner()

Objets concrets :

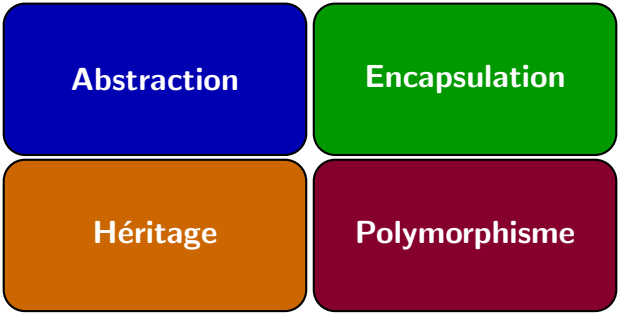
- `maClio` : (Renault, rouge, 2018)
- `taClio` : (Renault, rouge, 2018)
- `tesla` : (Tesla, blanche, 2023)

Exemple

Proposer une modélisation objet (attributs et méthodes) pour chacun des objets suivants :

- Un Point
- Une Personne
- Un compte Bancaire
- Une bibliothèque

Les 4 piliers de la Programmation Orientée Objet



Ces quatre principes structurent la pensée orientée objet et définissent la manière de modéliser le monde réel en logiciel.

Abstraction

- Identifier et regrouper, selon le point de vue de l'observateur, les caractéristiques et comportements communs et essentiels à des entités afin d'en faciliter la manipulation.
- L'abstraction permet de se concentrer sur l'essentiel et d'ignorer les détails d'implémentation.

Étudiant (Scolarité)
Attributs
- nom, prénom
- filière, niveau
- moyenne
Méthodes
+ inscrire()
+ calculer_moyenne()

Étudiant (Médecin)
Attributs
- age, poids, taille
- allergies, stress
Méthodes
+ effectuer_bilan()
+ prescrire_traitement()

Encapsulation

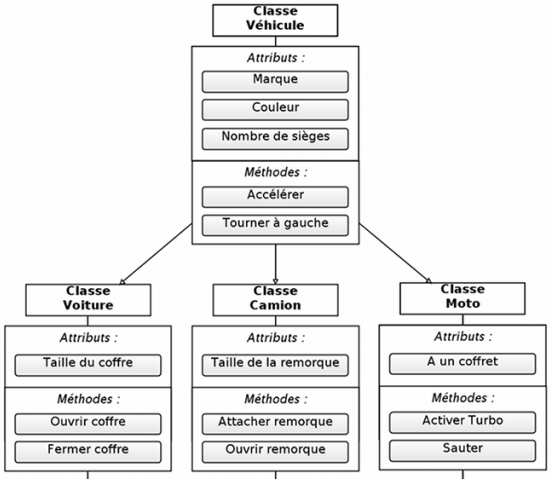
- L'encapsulation consiste à regrouper dans une même classe :
  - les données (attributs)
  - et les méthodes qui les manipulent.
- Elle permet de protéger l'état interne de l'objet :
  - les données ne sont pas accessibles directement depuis l'extérieur,
  - seules les méthodes autorisées peuvent les modifier.

Principe Clé

- **Vu de l'extérieur, un objet se résume à ses méthodes visibles (interface).**
- **Les données internes sont cachées**, ce qui garantit :
  - la sécurité (pas de manipulation directe),
  - la lisibilité (on sait comment interagir avec l'objet),
  - la maintenance (les changements internes n'affectent pas le reste du code).

Héritage

- Permet de créer de nouvelles classes à partir de classes existantes.
- Les nouvelles classes s'appelle des classes dérivées, des classes filles ou des sous classes.
- La classe original s'appelle classe de base, classe mère ou super classe.
- Facilite la réutilisation du code.



Polymorphisme

Definition

Le terme polymorphisme décrit la caractéristique d'un élément qui peut se présenter sous différentes formes.

- **Polymorphisme d'objet**
- **Polymorphisme de méthodes**

Polymorphisme d'objet

Definition

Il permet une certaine flexibilité dans la manipulation des objets en exploitant la relation d'héritage entre les classes. Pour cela, il applique la règle suivante : Si la classe A est dérivée de la classe B alors un objet de la classe A peut aussi être considéré comme étant un objet de la classe B.

Polymorphisme d'objet

Definition

- Le polymorphisme d'objet permet de traiter de manière uniforme des objets issus de classes différentes mais partageant une même hiérarchie.
- Si une classe `A` hérite d'une classe `B`, alors un objet de `A` peut être manipulé comme un objet de `B`.
- Ce mécanisme permet d'écrire du code plus générique et réutilisable.

```
class Animal {
    void manger() {
    System.out.println("Je mange");
    }
}
class Chien extends Animal {
    void aboyer() {
    System.out.println("Le chien aboie");
    }
}
public class TestPolymorphisme {
    public static void main(String[] args) {
        Animal a1 = new Chien();
        a1.manger(); // Je mange
        a1.parler(); // Le chien aboie
    }
}
```

Polymorphisme de méthodes

Principe

- Même nom de méthode, comportements différents.
- Deux formes :
  - **Surcharge** : même méthode avec différents paramètres.
  - **Redéfinition** : une sous-classe adapte une méthode héritée.

```
class Animal {
  void parler() {
    System.out.println("Un animal fait un son.");
  }

  //surcharge
  void parler(String lieu) {
    System.out.println("Un animal fait un son à "\
    + lieu + ".");
    }
}

class Chien extends Animal {
    // redéfinition
    void parler() {
    System.out.println("Le chien aboie.");
    }
}
```