Introduction à la programmation orientée objet (POO) en Java

1. Qu'est-ce que la Programmation Orientée Objet (POO) ?

La Programmation Orientée Objet (**POO** - *Object-Oriented Programming*) est un paradigme de programmation incontournable qui repose sur le concept d'"objets". Contrairement à la programmation procédurale classique (comme le langage C) qui s'articule uniquement autour de fonctions et d'une exécution séquentielle, la POO structure le logiciel en modélisant des entités interactives. Chaque objet regroupe son propre état (ses données ou caractéristiques) et son comportement (ses fonctions ou méthodes). **Java** est l'un des langages les plus populaires et puissants reposant intégralement sur ce modèle : en Java, tout ou presque est un objet.

> **Remarque importante :** L'objectif principal de la POO est de simuler des entités du monde réel dans le code (ex: un Client, un Compte Bancaire, un Produit). Cela rend les programmes plus organisés, hautement réutilisables, et grandement plus faciles à faire évoluer et à maintenir, particulièrement pour les projets d'entreprise complexes.

2. Les Concepts Fondamentaux de la POO

Pour maîtriser Java, il est impératif de comprendre les piliers sur lesquels repose la POO :

- **La Classe (Class) :** C'est le modèle, le plan de construction ou le moule. Elle définit les caractéristiques (attributs) et les comportements (méthodes) que posséderont les objets créés à partir d'elle. Une classe est un concept abstrait.
- **L'Objet (Object ou Instance) :** C'est la matérialisation concrète d'une classe en mémoire. Par exemple, si la classe est "Voiture", un objet sera la "Voiture rouge de marque X appartenant à Y".
- **L'Encapsulation (Encapsulation) :** C'est le principe de protection des données. Il consiste à masquer les détails internes de fonctionnement de l'objet pour le protéger des modifications extérieures incohérentes ou indésirables. En Java, on utilise les modificateurs d'accès (`private`, `protected`) et on expose des méthodes sécurisées appelées **Getters et Setters**.
- **L'Héritage (Inheritance) :** Permet à une nouvelle classe (classe fille) d'acquérir les attributs et méthodes d'une classe existante (classe mère) grâce au mot-clé `extends`. Cela évite la duplication de code et crée une hiérarchie logique (ex: Un "Chat" hérite de la classe "Animal").

apcpedagogie.com

- **Le Polymorphisme (Polymorphism) :** Du grec "plusieurs formes", c'est la capacité pour une méthode d'avoir des comportements différents selon l'objet exact qui l'invoque. Il se manifeste par la surcharge (Overloading) et la redéfinition (Overriding) de méthodes.
- **L'Abstraction (Abstraction) :** Consiste à simplifier la réalité en ne conservant que les fonctionnalités essentielles d'une entité, en masquant la complexité de son implémentation. En Java, cela se réalise via les `Interfaces` ou les classes abstraites (`abstract`).

3. Les Avantages de la POO en Java

- **Modularité :** Le code est divisé en petits blocs (objets) indépendants, facilitant grandement le travail en équipe.
- **Maintenance simplifiée :** Un bug dans un objet est plus facile à isoler et à corriger sans casser le reste du système.
- **Réutilisabilité :** Grâce à l'héritage et à la création de classes génériques, le code écrit une fois peut être réutilisé dans de multiples projets.

4. Exemple Pratique et Détaillé en Java

Voici un exemple illustrant la définition d'une classe, l'encapsulation (avec `private`) et l'instanciation d'un objet en Java :

```
// 1. Définition de la classe (Le modèle)
public class Voiture {

    // 2. Attributs encapsulés (privés) pour protéger les données
    private String couleur;
    private int vitesse;

    // 3. Constructeur : méthode spéciale appelée lors de la création de l'objet
    public Voiture(String couleur) {
        this.couleur = couleur;
        this.vitesse = 0; // La vitesse initiale est de 0
    }

    // 4. Méthodes d'accès (Getters / Setters)
    public String getCouleur() {
        return this.couleur;
    }

    public int getVitesse() {
        return this.vitesse;
    }
```

apcpedagogie.com

```
    // 5. Comportement (Méthode)
    public void accelerer(int ajoutVitesse) {
        this.vitesse += ajoutVitesse;
        System.out.println("La voiture " + this.couleur + " accélère. Vitesse actuelle : " +
this.vitesse + " km/h.");
    }

    // --- POINT D'ENTRÉE DU PROGRAMME ---
    public static void main(String[] args) {

        // Instanciation : Création d'un nouvel objet "Voiture" concrète
        Voiture maVoiture = new Voiture("Rouge");
        Voiture voitureAmi = new Voiture("Bleue");

        // Utilisation des objets : chaque objet possède son propre état
        maVoiture.accelerer(50);
        maVoiture.accelerer(30);

        voitureAmi.accelerer(60);
    }
}
```

Analyse du code :

Dans cet exemple, `Voiture` est la **Classe** qui agit comme un plan de construction global. `maVoiture` et `voitureAmi` sont deux **Objets** (ou instances) distincts créés en mémoire. Les attributs sont protégés (`private`) selon le principe d'**Encapsulation**, ce qui signifie que l'on interagit avec la voiture uniquement *via* ses méthodes publiques sécurisées comme `accelerer()`.

Rédigé et structuré par : **Med Khalil Kribi**

Source : https://apcpedagogie.com/introduction-a-la-programmation-orientee-objet-poo-en-java · téléchargé le 03/11/2026 · © APC Pédagogie · apcpedagogie.com