Fiche d'exercices : Programmation Orientée Objet

Principes de la POO

- **Encapsulation:** Masquer les détails internes et exposer uniquement les fonctionnalités nécessaires.
- **Abstraction:** Se concentrer sur les caractéristiques essentielles tout en cachant les détails d'implémentation.
- **Héritage:** Permettre à une nouvelle classe d'hériter des propriétés et des méthodes d'une classe existante.
- **Polymorphisme:** Permettre à une méthode d'avoir plusieurs formes différentes.
- **Composition:** Une classe est composée d'une ou plusieurs instances d'autres classes.
- **Association:** Une relation entre deux classes sans qu'aucune des deux ne possède l'autre.

Exercice 1 : Encapsulation

Objectif : Comprendre et utiliser l'encapsulation en programmation orientée objet.

1. **Définir une classe `BankAccount` avec les attributs privés suivants :**

- `_accountNumber` (chaîne de caractères)
- `_balance` (flottant)

2. **Définir des méthodes publiques pour accéder et modifier ces attributs de manière contrôlée :**

- `getAccountNumber()`

- `getBalance()`
- `deposit(amount: float)`
- `withdraw(amount: float)`

3. **Créer une instance de `BankAccount`, effectuer des dépôts et retraits, puis afficher le solde.**

Exercice 2 : Abstraction

Objectif : Utiliser l'abstraction pour simplifier l'implémentation.

1. **Définir une classe abstraite `Shape` avec une méthode abstraite `area()`.**
2. **Définir deux classes dérivées `Circle` et `Rectangle` qui implémentent la méthode `area()`.**
3. **Créer des instances de `Circle` et `Rectangle`, puis afficher leurs aires.**

Exercice 3 : Héritage

Objectif : Comprendre et utiliser l'héritage en programmation orientée objet.

1. **Définir une classe `Person` avec les attributs suivants :**
   - `name` (chaîne de caractères)
   - `firstName` (chaîne de caractères)
   - `age` (entier)
2. **Définir une méthode `introduceSelf` dans la classe `Person` qui affiche les informations de la personne.**
3. **Définir une classe `Student` qui hérite de la classe `Person` et ajouter l'attribut supplémentaire :**
   - `school` (chaîne de caractères)
4. **Définir une méthode `introduceSelf` dans la classe `Student` qui affiche les informations de l'étudiant, y compris l'établissement.**
5. **Créer une instance de `Person` et une instance de `Student`, puis appeler leur méthode `introduceSelf`.**

Exercice 4 : Polymorphisme

Objectif : Comprendre et utiliser le polymorphisme en programmation orientée objet.

1. **Définir une méthode `displayInfo` dans la classe `Person`.**

2. **Définir une méthode `displayInfo` dans la classe `Student` qui remplace celle de `Person`.**

3. **Créer une liste de `Person` contenant des objets `Person` et `Student`.**

4. **Parcourir la liste et appeler la méthode `displayInfo` sur chaque objet.**

Exercice 5 : Composition

Objectif : Utiliser la composition pour créer des relations "partie-tout".

1. **Définir une classe `Address` avec les attributs suivants :**

- `street` (chaîne de caractères)
- `city` (chaîne de caractères)
- `postalCode` (chaîne de caractères)

2. **Définir une classe `Person` (comme précédemment) et ajouter un attribut `address` qui est une instance de la classe `Address`.**

3. **Modifier la méthode `introduceSelf` de la classe `Person` pour inclure les informations de l'adresse.**

4. **Créer une instance de `Address` et une instance de `Person`, puis afficher les informations complètes de la personne.**

Exercice 6 : Association

Objectif : Utiliser l'association pour créer des relations entre objets.

1. **Définir une classe `Company` avec les attributs suivants :**

- `name` (chaîne de caractères)
- `industry` (chaîne de caractères)

2. **Ajouter un attribut `company` dans la classe `Person` pour représenter l'association avec une entreprise.**

3. **Définir une méthode `changeCompany` dans la classe `Person` pour changer l'entreprise associée à une personne.**

4. **Créer une instance de `Person` et une instance de `Company`, puis associer la personne à l'entreprise et afficher les informations.**

Exercice 7 : Agrégation

Objectif : Utiliser l'agrégation pour représenter des relations "tout-partie".

1. **Définir une classe `Course` avec les attributs suivants :**

- `courseName` (chaîne de caractères)

- `credits` (entier)

2. **Ajouter un attribut `courses` (liste de `Course`) dans la classe `Student`.**
3. **Définir des méthodes pour ajouter et supprimer des cours dans la classe `Student`.**
4. **Créer une instance de `Student` et plusieurs instances de `Course`, puis associer les cours à l'étudiant et afficher les informations.**

Exercice 8 : Diagrammes de classes

Objectif : Créer des diagrammes de classes pour représenter les relations entre les classes.

1. **Dessiner un diagramme de classes pour les classes `Person` et `Student` avec la relation d'héritage.**
2. **Dessiner un diagramme de classes pour les classes `Person` et `Address` avec la relation de composition.**
3. **Dessiner un diagramme de classes pour les classes `Student` et `Course` avec la relation d'agrégation.**
4. **Dessiner un diagramme de classes pour les classes `Person` et `Company` avec la relation d'association.**

Projet Final

Objectif : Mettre en pratique tous les concepts de la programmation orientée objet dans un projet complet.

1. **Contexte :** Vous devez créer un système de gestion de bibliothèque.
2. **Classes à définir :**

- `Library` avec les attributs suivants :
  - `name` (chaîne de caractères)
  - `address` (instance de `Address`)
  - `books` (liste de `Book`)
  - `members` (liste de `Member`)
- `Book` avec les attributs suivants :
  - `title` (chaîne de caractères)
  - `author` (chaîne de caractères)
  - `isbn` (chaîne de caractères)
- `Member` avec les attributs suivants :
  - `name` (chaîne de caractères)

- `membershipId` (chaîne de caractères)
- `borrowedBooks` (liste de `Book`)

3. **Relations :**

- `Library` et `Address` (composition)
- `Library` et `Book` (agrégation)
- `Member` et `Book` (association)

4. **Fonctionnalités à implémenter :**

- Ajouter un livre à la bibliothèque.
- Supprimer un livre de la bibliothèque.
- Inscrire un membre à la bibliothèque.
- Désinscrire un membre de la bibliothèque.
- Emprunter un livre.
- Retourner un livre.