**Rapport sur la Programmation Orientée Objet (POO)**

La Programmation Orientée Objet (POO) est une méthode de programmation qui organise le code en "objets", instances de classes. Elle améliore la modularité, la maintenabilité et encourage la réutilisation du code.

**Les Concepts de Base**

**Encapsulation** :Dans mon code, l'encapsulation est illustrée par des attributs privés dans les classes, accessibles via des méthodes publiques. Par exemple, estDisponible est un attribut privé, accessible via la méthode isDisponible. Cela protège les données de modifications non contrôlées.

**Héritage**: Le concept d'héritage est démontré par les classes Voiture et Camion qui héritent de la classe abstraite Vehicule. Elles réutilisent et étendent les fonctionnalités de la classe parent, ce qui permet d'avoir une base commune de fonctionnalités tout en ajoutant des spécificités propres à chaque type de véhicule.

**Polymorphisme :** Le polymorphisme permet à des objets de différentes classes d'être traités de manière uniforme via une interface commune. Ici, Louable est une interface que Voiture et Camion implémentent, ce qui leur impose de définir les méthodes louer et retourner. Ainsi, les objets Voiture et Camion peuvent être manipulés de manière interchangeable via l'interface Louable.

Abstraction La classe Vehicule est une classe abstraite contenant une méthode abstraite calculerPrixLocation. Cette méthode doit être implémentée par toutes les classes qui héritent de Vehicule, comme Voiture et Camion. L'abstraction permet de définir un cadre général sans implémentation détaillée, laissant les sous-classes fournir leur propre implémentation.

**Implémentation du Code**

**Interface Louable** : L'interface Louable définit les méthodes louer et retourner, qui doivent être implémentées par les classes qui la réalisent. Cela garantit que toutes les classes louables auront ces méthodes, permettant une manipulation polymorphique.

**Classe Abstraite Vehicule :** La classe Vehicule représente un véhicule avec des attributs communs tels que l'immatriculation, la marque, le modèle, l'année de mise en service et le kilométrage. Elle implémente l'interface Louable et définit des comportements de base tout en laissant certaines méthodes comme calculerPrixLocation abstraites pour être implémentées par les sous-classes.

**Classes Voiture et Camion** : Les classes Voiture et Camion héritent de Vehicule et redéfinissent la méthode calculerPrixLocation selon leurs propres critères. Elles implémentent aussi les méthodes louer et retourner définies dans l'interface Louable, gérant ainsi la disponibilité des véhicules.

**Classe Client** : La classe Client gère les informations relatives aux clients (nom, prénom, numéro de permis, numéro de téléphone) et maintient une liste des véhicules loués par chaque client. Elle permet d'ajouter et de supprimer des locations.

**Classe ParcAutomobile** : Cette classe gère l'ensemble des véhicules disponibles pour la location. Elle permet d'ajouter des véhicules, de lister les véhicules disponibles et de trouver un véhicule spécifique par son immatriculation.

**Classe Main** : La classe principale contient un menu interactif permettant aux utilisateurs d'ajouter des véhicules, des clients, de louer et de retourner des véhicules, et d'afficher les véhicules disponibles. Elle sert de point d'entrée au programme, orchestrant les interactions entre les différentes classes.

Exemple de Cas Pratique

Voici un exemple d'utilisation du code dans un scénario concret :

Ajout d'un véhicule

Ajout d'un client

Location et retour d'un véhicule