RAPPORT

1. Encapsulation

Dans la classe client, les informations privées telles que nom, prenom, numerodepermisdeconduire, numerodetelephone, et locationencours ne sont pas accessibles directement en dehors de la classe. Des méthodes getter (ex., getNom(), getPrenom()) permettent un accès contrôlé aux informations nécessaires sans exposer directement les attributs, protégeant ainsi l'intégrité des données.

2. Héritage

L'héritage est appliqué en créant une structure de classes dérivée pour représenter des types de véhicules spécifiques.Dans le code, la classe abstraite Vehicule sert de classe mère, et Voiture et Camion en héritent. La classe Vehicule définit des attributs et des méthodes communs pour tous les véhicules, comme immatriculation, marque, modele, statut, et calculerPrixLocation(). Les classes Voiture et Camion héritent de Vehicule et étendent ses fonctionnalités, telles que les attributs nombredeplace et typedecarburant pour Voiture ou capacitedechargement et nombreessieux pour Camion.

3. Polymorphisme

Le polymorphisme est mis en œuvre à travers l'utilisation de méthodes abstraites et d'interfaces. La méthode abstraite calculerPrixLocation() dans Vehicule est redéfinie dans Voiture et Camion. De plus, l'interface Louable oblige les classes Voiture et Camion à implémenter les méthodes louer() et retourner(). En utilisant le polymorphisme, le programme peut interagir avec des objets de type Louable de manière générique, ce qui rend le code plus flexible.

4. Gestion des Exceptions

Les exceptions personnalisées ClientNonAutoriseException et VehiculeIndisponibleException signalent des erreurs spécifiques de . Par exemple, dans la méthode louer() de Voiture, une VehiculeIndisponibleException est levée si la voiture n'est pas disponible pour la location. De même, dans la méthode louerVehicule(client client, Vehicule vehicule), une VehiculeIndisponibleException est levée si le véhicule n’est pas disponible ou non louable.