27/10/2024

MON RAPPORT DE JAVA

N’GUESSAN KOBENAN EDMOND

**PLAN DE TRAVAIL**

I-LA CLASSE VEHICULE

II-LA CLASSE VOITURE

III-LA CLASSE CAMION

IV-CLASSE CLIENT ET GESTION DES CLIENTS

V-INTERFACE LOUABLE

VI-GESTION DES EXCEPTIONS AVEC VEHICULEINDISPONOBLEAVECEXCEPTION

VII- MENU DE GESTION ET ESPACE UTILISATEUR

VIII-GESTION CENTRALISEE AVEC PARC AUTOMOBILE

I-LA CLASSE VEHICULE

La classe Vehicule utilise l'encapsulation en rendant ses attributs privés (immatriculation, marque, modele, anneeMiseEnService, kilometrage, estLouee). Cela limite l'accès direct à ces informations depuis l'extérieur de la classe, offrant une protection et un contrôle sur les modifications. Pour interagir avec ces données, des méthodes publiques sont mises en place, comme les getters (getImmatriculation, getMarque, etc.) et le setter (setLouee).

* Héritage

Véhicule est une classe abstraite qui sert de modèle générique pour tout type de véhicule. Cela permet de créer des sous-classes spécifiques (comme Voiture, Moto, et Camion), qui héritent des attributs et méthodes de Vehicule. Ainsi, Véhicule représente un concept de base dont les attributs communs sont réutilisables pour toutes les sous-classes, favorisant la modularité et la maintenabilité du code.

* Polymorphisme

Grâce à la méthode abstraite calculerPrixLocation(int duree), chaque sous-classe de Vehicule fournit sa propre implémentation de la méthode, adaptée à ses spécificités. Par exemple, une Voiture et un Camion peuvent avoir des critères différents pour calculer le prix de location. Cela permet de traiter les objets de type Vehicule de manière uniforme tout en appliquant des comportements spécifiques pour chaque type de véhicule.

II-LA CLASSE VOITURE

La classe Voiture illustre l’héritage en réutilisant les attributs et méthodes de la classe Vehicule. Elle introduit les éléments suivants :

* Héritage

En appelant super(...) dans son constructeur, Voiture utilise le constructeur de Vehicule pour initialiser les attributs hérités. Cela réduit le code redondant et garantit que les informations de base sont correctement initialisées.

* Interface Louable

Voiture implémente l'interface Louable, ce qui signifie qu’elle doit définir les méthodes louer() et retourner() pour gérer la location. Cette interface impose une structure standard pour tous les objets pouvant être loués.

* Polymorphisme

Voiture redéfinit la méthode calculerPrixLocation(int duree), où elle utilise un tarif de location spécifique (ex. 50.0 \* duree). Cette personnalisation est un exemple de polymorphisme, où des sous-classes appliquent leur propre logique tout en respectant l'interface commune.

* Encapsulation et Gestion des Exceptions

Dans louer(), la méthode vérifie d’abord si le véhicule est déjà loué via le getter isLouee(). Si c'est le cas, elle lance une exception VehiculeIndisponibleException, assurant que le véhicule ne peut pas être reloué par inadvertance.

III-LA CLASSE CAMION

La classe Camion reprend les concepts POO utilisés dans Voiture et les adapte :

* Gestion de la location et du retour

La méthode louer() vérifie d'abord si le camion est déjà loué, met à jour l’état si disponible, et affiche un message. En cas d’indisponibilité, une exception VehiculeIndisponibleException est lancée.

retourner() permet de mettre fin à la location, confirmant le retour avec un message.

* Encapsulation

Camion utilise des getters (getCapaciteChargement(), getNombreEssieux()) pour accéder aux attributs spécifiques, tels que la capacité de chargement et le nombre d'essieux. Ces méthodes permettent un contrôle sécurisé de ces données.

IV-CLASSE CLIENT ET GESTION DES CLIENTS

La classe Client représente les clients et gère les informations personnelles ainsi que les véhicules loués :

* Encapsulation

Les attributs privés (nom, prenom, numeroPermis, telephone) sont protégés et accessibles via des méthodes publiques (getters). La liste locationsEnCours est manipulée via des méthodes spécifiques (ajouterLocation et retirerLocation) pour contrôler les véhicules actuellement loués par chaque client.

* Gestion de la liste des véhicules loués

Cette classe permet de suivre chaque véhicule loué, fournissant une trace fiable des véhicules en location pour chaque client.

V-INTERFACE LOUABLE

L’interface Louable impose une structure commune pour tous les objets pouvant être loués, en définissant les méthodes suivantes :

louer() : Cette méthode indique que l’objet est loué et lance une exception si ce dernier est déjà loué.

retourner() : Cette méthode enregistre le retour de l’objet.

Grâce à cette interface, tous les objets de type Louable sont gérés de manière uniforme dans le programme.

VI-GESTION DES EXCEPTIONS AVEC VEHICULEINDISPONIBLEEXCEPTION

La classe VehiculeIndisponibleException est une exception personnalisée pour gérer les cas où un véhicule n'est pas disponible pour la location :

* Héritage d’Exception

En héritant de Exception, cette classe crée une exception vérifiée, forçant ainsi le programme à gérer les cas d’indisponibilité, renforçant la fiabilité du code.

* Constructeur

Le constructeur accepte un message, permettant de fournir des informations détaillées sur l’indisponibilité du véhicule, ce qui aide au débogage et à l’interaction utilisateur.

VII- MENU DE GESTION ET ESPACE UTILISATEUR

Le menu gère les opérations principales :

* Ajouter un véhicule ou un client

ajouterVehicule : Ajoute un véhicule selon son type, en validant que seuls des véhicules valides (ex. voiture ou camion) peuvent être saisis.

ajouterClient : Ajoute un client en vérifiant que toutes les informations nécessaires sont complètes.

* Location et retour de véhicule

louerVehicule : Vérifie la disponibilité d'un véhicule et l’état du client. Si le véhicule est disponible, il est loué.

retournerVehicule : Permet à un client de retourner un véhicule loué, en mettant à jour le parc.

* Affichage des véhicules

afficherVehiculesDisponibles et afficherVehiculesLoues affichent respectivement les véhicules disponibles et loués, donnant une vue d’ensemble rapide sur l’état actuel du parc.

VIII-GESTION CENTRALISEE AVEC PARC AUTOMOBILE

La classe ParcAutomobile centralise la gestion des clients et des véhicules :

* Gestion des véhicules

ajouterVehicule ajoute un véhicule à la liste vehicules.

getVehiculesDisponibles et afficherVehiculesDisponibles affichent les véhicules disponibles, en vérifiant l'état de location.

rechercherVehicule permet de retrouver un véhicule par son immatriculation.

* Gestion des clients

ajouterClient et rechercherClient permettent l'ajout et la recherche de clients par numéro de permis.