RAPPORT

. Concept d’héritage :

Ici le concept d’héritages est utilisé avec la classe « Voiture » et la classe « Camion », qui héritent de la classe « Véhicule ».

Cela permet de ne pas réécrire à répétition les nombreux attributs communs de ces deux classes et d’ajouter des attributs particuliers propre à chacun de ces classes ;

La classe « Véhicule » est déclaré en « abstract » pour permettre à chaque classe fille de personnaliser la méthode « calculPrixLocation() » selon la valeur du tarif, qui est unique à chaque classe.

. Concept d’encapsulation :

Le concept d’encapsulation va permettre de limiter l’accès a certains attributs et méthodes et pour d’autre d’y accéder facilement, selon le besoin.

. Concept de polymorphisme :

Ici le polymorphisme est utilisé lors de la création de l’interface « Louable », qui sera implémenté par deux méthodes abstraite, « louer » et « retourner ».

Ces méthodes seront définies plus tard dans la classe voiture et la classe camion.

Il y a aussi, l’utilisation de la classe abstraite « Vehicule » , en effet les méthodes déclarés comme abstraite ( « calculPrixLocation » et « toString ») dans cette classe seront définies par chaque sous classe à sa manière.

. Gestion d’exception :

Les exceptions personnalisées ont été créées pour gérer des situations spécifiques lors de la location des véhicules.

L’exception « VehiculeIndisponibleException » est utilisée lorsque l’utilisateur tente de louer un véhicule déjà loué.

L’exception « ClientNonAutoriseException » est utilisée pour vérifier si un client a le permis adéquat avant de lui permettre de louer un camion.