Université Mohammed V - Agdal S. ZITI
Faculté des Sciences – Rabat 2012/2013
Département d'Informatique

Travaux dirigés 6
Programmation orientée objet (Java)
SMA5

Exercice: Gestion d'un Car-ferry

On cherche au cours de cet exercice à modéliser l'embarquement de véhicules dans un Car-ferry, le paquetage **Ferry** contient :

1. Une classe **Personne** abstraite contenant des informations sur les noms, les prénoms, la date de naissance et les adresses et une méthode abstraite ***voyager***
2. Une classe **Voyageur** qui hérite de la classe personne et contenant en plus un attribut désignant la date de voyage.
3. Une Classe **Conducteur** héritant de la classe **Voyageur** et contenant en plus un attribut désignant le numéro de permis et une méthode ***ObtenirPermis*** en fonction de l'âge des personnes (18 ans pour passer le permis) elle permet de lever une exception en cas d'un âge non réglementaire
4. Une classe Véhicule caractérisé par son modèle, son immatriculation, son kilométrage, son poids à vide, sa longueur (en mètres), et sa réserve de carburant (en nombre de litres).
   4.1. Un véhicule peut être conduit par une personne adulte. Ajouter à la classe **Vehicule** les instructions nécessaires pour qu'un véhicule puisse éventuellement désigner son conducteur qui sera un objet de la classe conducteur. Le même conducteur peut-il être le conducteur de plusieurs véhicules ?
   4.2. Ecrire un ***constructeur*** permettant d'instancier des objets de la classe **Vehicule** dont on précisera simplement le modèle, l'immatriculation, la longueur et le poids. Les autres propriétés seront initialisées à 0. Expliquez comment vous traitez le fait, qu'à la construction, un véhicule n'a pas encore de conducteur ?
   4.3. Ecrire une méthode ***ChangeConducteur*** pour changer le conducteur d'un véhicule.
   4.4. Réécrire la méthode ***toString*** indiquant les caractéristiques du véhicule et de son éventuel conducteur.
   4.5. Ecrire la méthode ***GetPoids*** retournant le poids du véhicule, c'est-à-dire la somme du poids à vide, du poids du carburant (on fera la supposition qu'un litre de carburant pèse un kg) et du poids de l'éventuel conducteur.

4.6. Ecrire la méthode ***allerAlaPompe(float quantite)*** permettant d'ajouter du carburant au véhicule et retournant la nouvelle quantité de carburant.

4.7. Ecrire la méthode ***rouler (float distance, float taux_consommation)*** permettant au véhicule d'avancer et retournant le nouveau kilométrage du véhicule. Une exception sera levée si le véhicule n'a pas de conducteur ou si la quantité de carburant disponible n'est pas suffisante pour cette opération

5. Une classe **Camion** : Un camion est un véhicule pouvant transporter une cargaison de colis.

5.1. Ecrire une classe **Camion** héritant de la classe **Vehicule** permettant de mémoriser la cargaison sous la forme d'une collection d'objets de la classe **Colis** donnée en annexe. Choisir un type de collections approprié, en justifiant votre réponse.

5.2. Récrire le constructeur de cette classe afin d'initialiser la cargaison avec une collection vide.

5.3. Ajouter la méthode ***int ajouter (Colis c)*** qui retourne le nombre de colis dans la cargaison après l'ajout

5.4. Ajouter la méthode ***boolean retirer(Colis c)*** qui retourne vrai si il a été trouvé et retiré de la cargaison

5.5. Réecrire la méthode ***PoidCamion()*** permettant de connaître le poids du camion : on calculera la somme des poids du camion à vide, du carburant, de l'éventuel conducteur et de la cargaison.

5.6. Réécrire la méthode ***toString()*** donnant les caractéristiques du véhicule et de sa cargaison.

6. Une classe **Voiture** : une voiture est un véhicule pouvant transporter un nombre limité des passagers.

6.1. Ecrire une classe **Voiture** héritant de la classe **Vehicule** permettant définir le nombre maximum de passagers et de mémoriser la liste de ses passagers sous la forme d'une collection d'objets de la classe **Voyageur**.

6.2. Choisir un type de collections approprié pour mémoriser les passagers, en justifiant votre réponse.

6.3. Récrire le **constructeur** de cette classe pour définir le nombre maximum de passagers et initialiser la collection des passagers.

6.4. Ajouter la méthode ***int monter (voyageur p)*** pour faire monter ou descendre un passager de la voiture (Lever une exception si on essaye de faire monter un passager

alors qu'il n'y a plus de place dans la voiture) et retourne le nombre de personnes dans la voiture après l'ajout

6.5. Ajouter la méthode ***boolean descendre(Voyageur p)*** pour faire descendre un passager de la voiture et retourne vrai si la personne a été trouvée et retirée des passagers de la voiture

6.6. Réecrire la méthode ***PoidVoiture()*** permettant de donner le poids de la voiture, qui calculera la somme des poids de la voiture à vide, du carburant, des passagers et de l'éventuel conducteur.

6.7. Réécrire la méthode ***toString***() donnant les caractéristiques d'une voiture et de ses passagers.

7. Une Classe **CarFerry** : un CarFerry est un véhicule pouvant transporter des voitures ou des camions. La cale du bateau a une longueur et une charge limitées, qui sont des propriétés caractéristiques du **CarFerry**. Les véhicules embarquent dans la cale en se répartissant à droite ou à gauche du bateau, en 2 piles de véhicules (premier entré, dernier sorti)

7.1. Faire un schéma de la structure mémoire d'un objet de la classe CarFerry, qui embarquerait plusieurs voitures et plusieurs camions.

7.2. Ecrire une classe **CarFerry** héritant de la classe **Vehicule**, permettant de modéliser un tel bateau.

7.3. Réécrire le ***constructeur*** définissant les limites de longueur et de charge de la cale, et initialisant les piles de véhicules.

7.4. Récrire la méthode ***PoidsCarFerry()*** permettant de calculer le poids total du CarFerry.

7.5. Ecrire la méthode ***Embarquer (Vehicule v)*** permettant d'embarquer un véhicule, en " l'empilant " à droite ou à gauche de la cale, de manière à équilibrer le poids du bateau. Veiller à ne pas dépasser la longueur et la charge limitées du CarFerry.

7.6. Ecrire la méthode ***Debarquer (Vehicule v)*** permettant de débarquer un véhicule, en " dépilant " soit à droite, soit à gauche de la cale, de manière à équilibrer le poids du bateau.

7.7. Ecrire une méthode ***PropVoiture()*** calculant la proportion de voitures par rapport au nombre de véhicules transportés.

7.8. Ecrire une méthode ***Chercher(voyageur p)*** permettant de savoir si une personne (conducteur d'une voiture ou d'un camion ou passager d'une voiture) se trouve sur le CarFerry.