
TP Polymorphisme

Exercice 1

Dans cet exercice, respectez au maximum les principes suivants :

1. Dès qu'un objet est créé, il doit être correctement initialisé : toutes ses propriétés (attributs) doivent avoir des valeurs correctes.
2. Une hiérarchie d'héritage doit être extensible : idéalement, on devrait pouvoir ajouter une classe (par exemple ici un nouveau type de véhicule) sans toucher aux classes existantes. Notamment, une classe n'est pas censée connaître ses sous-classes.
3. L'héritage permet de factoriser le code : essayez ici de le faire le plus possible (même si ça peut vous sembler lourd sur un si petit exemple).
4. Lorsqu'une valeur est commune à tous les objets d'une classe (par exemple ici la valeur de "poids à vide" ou de "charge maximale"), elle n'a pas vocation à être stockée dans chaque objet. Elle peut être stockée dans un attribut statique et retournée par une méthode (statique ou non selon l'utilisation que l'on veut en faire).

Une entreprise de Patagonie développe un moteur à huile de rutabaga, et dote plusieurs véhicules de ce moteur. Avant d'effectuer des tests grandeur nature, l'entreprise conçoit un programme de simulation du comportement des véhicules. Ce programme est écrit en Java. On ne s'intéresse ici qu'à une toute petite partie de ce programme. Le problème que l'on veut résoudre dans un premier temps est celui de la représentation des véhicules et du calcul de la vitesse maximum pouvant être atteinte par un véhicule.

Tout véhicule possède une immatriculation (propre à chaque véhicule) et un poids à vide (propre à chaque type de véhicule). Certains véhicules peuvent transporter un chargement : on appelle alors charge le poids de ce chargement. La charge d'un véhicule ne doit pas dépasser un certain poids, que l'on appellera charge maximale, dépendant du type de véhicule.



Les différents types de véhicules dotés du fameux moteur sont les suivants : les petits bus, les camions citernes, et les camions bâchés.

- Un petit bus a un poids à vide de 4 tonnes, et peut atteindre une vitesse maximale de 150 km/h. Il ne possède pas de chargement (le poids des passagers est considéré comme négligeable par rapport au poids à vide).
- Un camion-citerne a un poids à vide de 3 tonnes et une charge maximale de 10 tonnes. Sa vitesse maximale dépend de sa charge :
 - 130 km/h si la charge est nulle
 - 110 km/h si la charge est inférieure ou égale à 1 tonne
 - 90 km/h si la charge est supérieure à 1 tonne et inférieure ou égale à 4 tonnes
 - 80 m/h pour une charge supérieure à 4 tonnes
- Un camion bâché a un poids à vide de 4 tonnes et une charge maximum de 20 tonnes. Sa vitesse maximale dépend également de sa charge (mais à charge égale, un camion-citerne a une vitesse maximale plus faible, car le liquide qu'il transporte est plus instable qu'un chargement solide) :
 - 130 km/h si la charge est nulle
 - 110 km/h si la charge est inférieure ou égale à 3 tonnes
 - 90 km/h si la charge est supérieure à 3 tonnes et inférieure ou égale à 7 tonnes
 - 80 km/h au-delà.

Question 1.

Dessiner une hiérarchie de classes, dont la classe racine s'appelle Véhicule, permettant de :

- décrire un véhicule par une chaîne de caractères : cette chaîne comportera l'immatriculation, le poids à vide, et, le cas échéant, la charge maximale et la charge du véhicule.
- calculer la vitesse maximale d'un véhicule.

Vous indiquerez le nom des attributs et des méthodes de chaque classe, et mentionnerez, s'il y a lieu, quelles classes et méthodes sont abstraites. Il n'est pas nécessaire de s'occuper des constructeurs à ce stade de la conception.

Question 2.

1. Écrire les classes participant à la définition d'un camion bâché (c'est-à-dire la classe Véhicule, et la classe permettant de représenter un camion bâché, ainsi que les



éventuelles classes intermédiaires si ces deux classes ne sont pas en relation d'héritage direct).

2. Programmer toutes les classes participant à la définition de tous les types de véhicules. Ajouter une méthode `main()` créant deux camions bâchés, et affichant leur description ainsi que leur vitesse maximale

Question 3.

On s'intéresse maintenant à la définition d'une classe `Convoi`, permettant de gérer un convoi de véhicules. Cette classe doit savoir calculer la vitesse maximale d'un convoi, sachant que cette vitesse correspond à la plus petite des vitesses maximales des véhicules du convoi.

1. Définir un ou des attributs de la classe `Convoi` permettant de représenter les véhicules d'un convoi. On supposera qu'au cours de la vie d'un objet instance de la classe `Convoi`, on ne modifie pas la composition du convoi de véhicules (en particulier, le nombre de véhicules est stable). (donc utiliser un tableau)
2. Écrire les méthodes suivantes :
 - un constructeur qui prend en paramètre un tableau de véhicules
 - une méthode `toString` qui affiche une chaîne de caractères décrivant le convoi, c'est-à-dire chacun des véhicules composant le convoi.
 - une méthode calculant la vitesse maximale d'un convoi de véhicules.
3. Dans la méthode `main` créer deux convois et affichant les caractéristiques de chacun de ces convois, ainsi que leur vitesse maximale. Vérifiez que vos calculs sont corrects !

Question 4.

Les ingénieurs de l'entreprise fournissent une formule permettant de calculer la consommation de carburant d'un véhicule en fonction de sa vitesse et de son poids total (poids à vide + charge éventuelle). Cette formule est la même pour tous les types de véhicules.

On supposera donnée, dans une classe annexe nommée `Utilitaires`, la méthode d'entête suivante : `public static float consommation (int v, int p)`, qui retourne la consommation (en litres pour 100 km) d'un véhicule de poids `p` allant à une vitesse `v`.

1. Intégrer à la hiérarchie de classes une ou des méthodes permettant de calculer la consommation d'un véhicule allant à une certaine vitesse (qui n'est pas forcément sa vitesse maximale). Et ajouter à la classe `Convoi` une méthode permettant le calcul de la consommation totale d'un convoi allant à sa vitesse maximale (c'est-à-dire, on le rappelle, à la plus petite des vitesses maximales des véhicules du convoi).



2. Ajouter la classe Utilitaires avec sa méthode consommation. En ce qui concerne la formule de calcul, peu importe si elle n'est pas vraisemblable (par exemple, $\text{consommation} = \text{vitesse-en-km/h} / 10 + \text{poids-en-tonnes}$) ; il est plus important que vous puissiez vérifier que le calcul de la consommation d'un convoi est correct.
3. Modifier la hiérarchie de classes pour intégrer le calcul de la consommation des véhicules et des convois. Dans la méthode main() afficher la consommation de chacun des deux convois à leur vitesse maximale.