

## Abstraction, interfaces et tableaux

### Exercice 1. Véhicules

**Pour cet exercice, ni version, ni test unitaire ! On développe les classes et on les modifie au fur à et à mesure des questions.**

On considère une classe de véhicules. Chacun a un numéro d'identification (attribué) automatiquement et une distance parcourue (initialisée à 0), et parmi eux, on distingue les véhicules à moteur qui ont une capacité de réservoir et un niveau d'essence (initialisé à 0) et les véhicules sans moteur qui n'ont pas de caractéristique supplémentaire. Parmi les véhicules à moteur, les voitures ont un nombre de places, les camions, un volume transporté. Parmi les véhicules sans moteur, les vélos ont un nombre de vitesse.

Question 1. Construire en UML le graphe hiérarchique des classes décrites ci-dessus.

Question 2. Ecrire le code java des classes **Vehicule**, **AMoteur**, **SansMoteur** avec tous les constructeurs nécessaires et la méthode **toString()**, qui retourne une chaîne de caractères décrivant l'état de l'objet.

Question 3. Ecrire une méthode **void rouler (double distance)** qui fait avancer un véhicule. A quel niveau de la hiérarchie faut-il l'écrire ?

Question 4. Ecrire les méthodes **void approvisionner (double nbLitres)** et **boolean enPanne()**. A quel niveau de la hiérarchie faut-il les écrire ?

Question 5. Ecrire la classe **Velo** avec constructeur et méthode **toString()** et une méthode **void transporter (String depart, String arrivee)** qui affiche par exemple :

Le vélo n°2 a roulé de Paris à Nice.

Question 6. Ecrire la classe **Voiture** avec constructeur, **toString()** et une méthode : **void transporter (int n, int km)** qui affiche par exemple :

La voiture n°3 a transporté 5 personnes sur 250 km ou bien  
Plus d'essence si elle est à sec.

Question 7. Ecrire la classe **Camion** avec constructeur, **toString()** et une méthode : **void transporter (String materiau, int km)** qui affiche par exemple :

Le camion n°4 a transporté des tuiles sur 400 km ou bien  
Plus d'essence s'il est à sec.

Question 8. Peut-on factoriser la déclaration de la méthode **transporter** et si oui, à quel niveau ?

Question 9. On considère le **main** suivant (à placer dans un fichier **T\_Vehicule.java**) :

```

public static void main (String [] args)
{
    Vehicule v1 = new Velo (17);           // nb vitesses
    Vehicule v2 = new Voiture (40.5, 5);    // capacité réservoir, nb Places
    Vehicule v3 = new Camion (100.0, 100.0); // capacité réservoir, volume
    System.out.println ("Vehicules : "+"\\n" +v1 +"\\n" +v2 +"\\n" +v3 +"\\n");
    v2.approvisionner (35.0) ;              // litres d'essence
    v3.approvisionner (70.0);
    v1.transporter ("Dijon ", "Valence ") ;
    v2.transporter (5, 300) ;
    v3.transporter (" des tuiles", 1000) ;
}

```

Ce programme est-il correct ? Le corriger si nécessaire. Qu'affiche-t-il ?

Question 10. Définir des classes *interface* pour l'approvisionnement et le transport. Modifier les classes en conséquence.