TD VIII

Vie de famille: Héritage et Polymorphisme

VIII.1 Héritage et appel des constructeurs

On considère les classes `A`, `B` et `Test` suivantes:

```java
public class A {
    public A() {
        System.out.println("> Constructeur de A()");
        afficher();
    }

    public void afficher() {
        System.out.println("> Affichage de A");
    }
}
```

```java
public class B extends A {
    private int v;

    public B() {
        System.out.println("> Constructeur de B()");
        v = 5;
        afficher();
    }

    public void afficher() {
        System.out.println("> v = " + v);
    }
}
```

```java
public class Test {
    public static void main(String[] args) {
      System.out.println("A objetAA = new A();");
      A objetAA = new A();
      System.out.println("A objetAB = new B();");
      A objetAB = new B();
      System.out.println("B objetBA = new A();");
      B objetBA = new A();
    }
}
```

L'objectif est de deviner, par l'analyse le résultat du code de la méthode `main` .

Question 1.1
Dessinez le diagramme APO de ce qui se passe dans la méthode `main`.

Question 1.2
Que produit la ligne 4 de la classe `Test` (`A objetAA = new A();`) et pourquoi ?

Question 1.3
Que produit la ligne 6 de la classe `Test` (`A objetAB = new B();`) et pourquoi ?

Question 1.4
Que produit la ligne 8 de la classe `Test` (`B objetBA = new A();`) et pourquoi ?

VIII.2 Véhicules en tout genre

Nous allons ici modéliser 3 types de véhicules. Un `Vehicule` a une `vitesse` , un nombre à virgule, toujours positif ou nul, et un nombre de `passagers`, un entier toujours positif. Un `Avion` est un véhicule particulier. Il a une vitesse comprise entre 0 et 1000 km/h. Il a entre 5 et 200 passagers. De plus, il a en plus entre 1 et 8 `moteurs`. Une `Voiture` est un autre véhicule particulier. Il a une vitesse comprise entre 0 et 250 km/h. Il a entre 1 et 8 passager. De plus, il a entre 1 et 5 `portières`. On considère que chacune des classes contient:

- les attributs cités (et uniquement eux),
- un seul constructeur qui prend le nombre de paramètres nécessaires (dans l'ordre donné par l'énoncé),
- des *getters* et des *setters* pour chacun des attributs,
- la redéfinition de la méthode `+ toString(): String` qui renvoie:
  - pour un véhicule standard, une chaîne du type :
    `"Vehicule: [vitesse: 35.5 km/h, passagers: 1352]"`
  - pour un avion, une chaîne du type:
    `"Avion: [vitesse: 800.0 km/h, passagers: 73, moteurs: 6]"`
  - pour une voiture, une chaîne du type:
    `"Voiture: [vitesse: 100.0 km/h, passagers: 2, portieres: 3]"`

Question 2.1
Faire le diagramme UML de classe du problème posé.

Question 2.2
Quelles sont les méthodes qui devront être redéfinies dans les sous-classes? Pourquoi?

Question 2.3
Ecrire les classes `Vehicule` , `Avion` et `Voiture` en prenant bien soin de préserver l'encapsulation et de ne pas réécrire de code.

On consière à présent la méthode `main` suivante:

```
1  public static void main(String[] args) {
2      Vehicule v1 = new Vehicule(35.0, 1352);
3      System.out.println(v1);
4      Vehicule v2 = new Avion(800.0, 73);
5      System.out.println(v2);
6      Vehicule v3 = new Avion(800.0, 73, 6);
7      System.out.println(v3);
8      Avion    v4 = new Avion(-354.3, 42, 3);
9      System.out.println(v4);
10     Avion    v5 = new Vehicule(800.0, 73);
11     System.out.println(v5);
12     Avion    v6 = new Vehicule(800.0, 73, 6);
13     System.out.println(v6);
14 }
```

Question 2.4
Quelles sont les lignes qui vont poser problème à la compilation? Pourquoi?

Question 2.5
Si l'on supprime les lignes qui posent problème, que se passe-t-il à l'exécution?