TP6

Exercice – points 1D, 2D et 3D

Nous avons vu dans un TP précédent qu'un point dans le plan pouvait être représenté par une abscisse et une ordonnée. Mais nous pouvons aussi considérer qu'un tel point n'est rien d'autre qu'un point en une dimension (représenté uniquement par une abscisse) qui possède une seconde dimension, son ordonnée. De même, un point en 3D peut être vu comme un point en 2D possédant une troisième dimension, sa côte. Ainsi, un point 3D est un point 2D, qui est lui-même un point 1D.

1. Créez une classe `Point1D` et écrivez les méthodes qui définissent la forme canonique de la classe (*i.e.*, constructeurs, accesseurs, `toString`, `equals`).
2. Créez une classe `Point2D` qui étend la classe `Point1D` et écrivez les méthodes qui définissent la forme canonique de la classe (attention de ne pas redéfinir inutilement ce dont `Point2D` dispose déjà et évitez les duplications de code).
3. Pourquoi `p1.equals(p2)` ne donne pas le même résultat que `p2.equals(p1)`, où `p1` est de type `Point1D` et p2 de type `Point2D` ?
4. Créez une classe `Point3D` qui étend la classe `Point2D` et écrivez les méthodes qui définissent la forme canonique de la classe (attention de ne pas redéfinir inutilement ce dont `Point3D` dispose déjà et évitez les duplications de code).
5. Écrivez une méthode de classe `afficherEgaux` qui prend en paramètre un tableau de points (des `Point1D` mais aussi des `Point2D` et des `Point3D`) et un point `p` de type `Point1D`, et qui affiche tous les points du tableau qui sont égaux à `p`.
6. Écrivez un programme de test.
7. Réflexion :
   (a) Pourquoi est-il possible d'écrire `Point1D p = new Point2D(1, 2);` ? Est-il possible d'écrire l'instruction précédente suivie de `System.out.println(p.getY());` ?
   (b) Quels inconvénients y aurait-il eu à définir les classes `Point1D`, `Point2D` et `Point3D` sans héritage ?
   (c) Sans héritage, comment serait le code des méthodes `equals`, sachant qu'un `Point1D` doit toujours pouvoir être comparé à un `Point2D` ou un `Point3D` ?
   (d) Sans héritage entre `Point1D`, `Point2D` et `Point3D`, comment créer un tableau contenant à la fois des instances de `Point1D`, `Point2D` et `Point3D` ?

Exercice – la Poste

Nous souhaitons écrire une application permettant d'afficher les courriers présents dans une boîte aux lettres, de déterminer le nombre de courriers invalides, et de calculer le coût total d'affranchissement des courriers. Une boîte aux lettres peut contenir un nombre limité de courriers. Les courriers peuvent être des lettres, des colis ou encore des magazines. Chaque courrier possède une adresse de destination, un poids (en grammes) et un mode d'expédition (*i.e.*, normal ou express). Le mode d'expédition par défaut d'un courrier est normal. Les lettres ont également

un format (A3, A4, A5, *etc.*), et les colis un volume (en litres). Les règles utilisées pour affranchir le courrier sont les suivantes :

- En mode d'expédition normal,
  - Le montant nécessaire pour affranchir une lettre dépend de son format et de son poids : tarif de base + poids (en kilos), où le tarif de base pour une lettre A4 est de 2.50 euros et de 3.50 euros pour les autres formats
  - Le montant nécessaire pour affranchir un colis dépend de son poids et de son volume : 0.25 * volume (en litres) + poids (en kilos)
  - Le montant nécessaire pour affranchir un magazine dépend de son poids : 5 * poids (en kilos)
- En mode d'expédition express, les montants précédents sont doublés, quel que soit le type de courrier
- Un courrier n'est pas valide si l'adresse de destination est vide
- Un colis n'est pas valide si son adresse de destination est vide ou s'il dépasse un volume de 50 litres
- Seuls les courriers valides sont affranchis

L'affichage des courriers dans la console Java doit correspondre à :

```
Lettre
Destination : Bordeaux
Poids : 100.0 grammes
Express : oui
Prix : 7.2 euros
Format : A3

Magazine
Destination : Cergy
Poids : 1500.0 grammes
Express : non
Prix : 7.5 euros

Colis [invalide]
Destination : Londres
Poids : 3000.0 grammes
Express : oui
Prix : 0.0 euros
Volume : 70.0 litres
```

1. Proposez un diagramme de classes qui modélise le problème.
2. Écrivez en Java une implémentation des classes du diagramme, en évitant les duplications de code.
3. Écrivez un programme de test qui :
   (a) Crée une boîte aux lettres pouvant contenir 50 courriers,
   (b) Crée des lettres, des colis et des magazines, à la fois valides et invalides,
   (c) Ajoute ces courriers à la boîte aux lettres,
   (d) Affiche les courriers de la boîte aux lettres,
   (e) Affiche le coût total d'affranchissement des courriers de la boîte aux lettres,
   (f) Affiche le nombre de courriers invalides dans la boîte aux lettres.