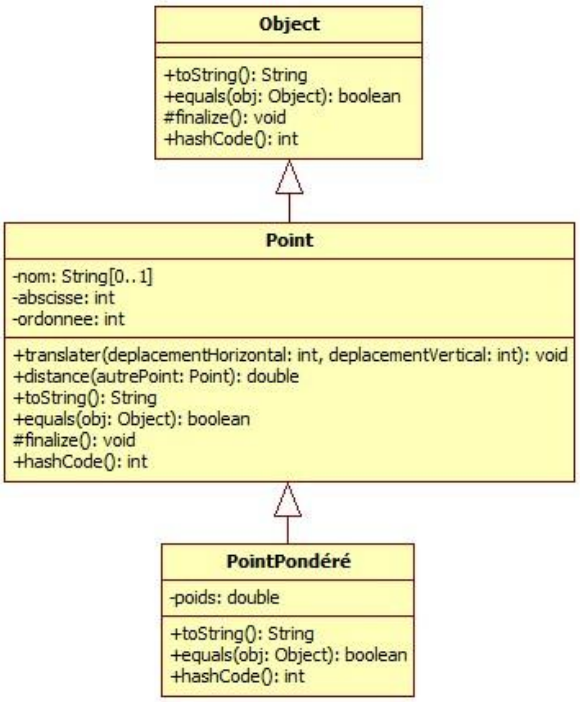
TD4 – Héritage : spécialisation



Introduction.

Le but de ce TD est de traduire en Java la notion d'héritage en s'intéressant à la spécialisation de classes existantes et au mécanisme de liaison tardive.

Nous ajoutons aux classes des séances précédentes la classe PointPondéré qui spécialise la classe Point en ajoutant un poids à un point. Nous nous intéresserons également à l'héritage par défaut de la classe Object pour toutes classes écrites en Java.

Exercice 1. Spécialisation de la classe Object

Toute classe Java hérite implicitement de la classe Object et peut donc utiliser ou redéfinir ses méthodes. Dans les séances précédentes, vous avez déjà redéfini la méthode de conversion en chaîne de caractère toString dans vos premières classes : Point, Cercle et Triangle. Nous allons nous intéresser aux autres méthodes de la classe Object dans le cadre de la classe Point.

1.1 – Méthode **finalize()**

a. Chercher dans la documentation de l'API Java ce que fait cette méthode.
b. Quel est le droit d'accès de cette méthode ? A votre avis pourquoi ?
c. Redéfinir cette méthode dans la classe Point afin de tracer son appel dans la console. Le message affichera le point (soi-même) et l'évènement (cf question a)).
d. Ecrire un programme TesGarbageCollector qui dans une boucle affecte une référence de type Point avec un nouveau Point et l'affiche. Que remarquez-vous ?

1.2 – Méthode **equals()**

a. Redéfinir cette méthode pour la classe Point. En Java 7, utiliser la classe Objects pour sa réalisation.
b. Réaliser le test suivant. Le résultat est-il correct ?

```
Point pointA, pointB;
pointA = new Point("A", 0, 0);
pointB = pointA;
System.out.println("pointA -> " + pointA +    "  / pointB -> " +
        pointB);
System.out.println("pointA == pointB -> " + (pointA == pointB));
System.out.println("pointA.equals(pointB) -> "
         + (pointA.equals(pointB)));

pointB = new Point("A", 0, 0);
// même test
```

1.3 – Méthode **hashCode()**

a. Lire la documentation de l'API sur la cohérence entre cette méthode et la méthode equals.
b. Proposer une fonction de calcul pour le hashcode d'un point. Implémenter cette fonction. En Java 7, utiliser la classe Objects pour sa réalisation.

Exercice 2. La Classe PointPondéré

2.1 - Ecrire la classe **PointPondéré** avec :

a. son **constructeur** prenant en paramètres : coordonnées, nom et poids

b. des **accesseurs** en lecture et écriture pour son poids.

c. la redéfinition de **toString**, **equals** et **hashCode**. Format pour l'affichage : `{Point ; Poids}`.

2.2 – Références sur Point et PointPondéré

a. Prévoir sans l'exécuter le déroulement du test suivant :

```
1     Point pointA, pointB;
2     PointPondere pointPondereA, pointPondereB;
3     pointA = new Point("A", 0, 0);
4     pointPondereA = pointA;
5     System.out.println(pointA);
6     System.out.println((PointPondere) pointA);
7     System.out.println(pointPondereA);
8     System.out.println((Point) pointPondereA);
9     System.out.println("Poids de A : " + pointA.getPoids());
10    System.out.println("Poids de A : " + pointPondereA.getPoids());
11    pointPondereB = new PointPondere("B", 1, 2, 1.5);
12    pointB = pointPondereB;
13    System.out.println(pointB);
14    System.out.println((PointPondere) pointB);
15    System.out.println(pointPondereB);
16    System.out.println((Point) pointPondereB);
17    System.out.println("Poids de B : " + pointB.getPoids());
18    System.out.println("Poids de B : " + pointPondereB.getPoids());
```

b. Compiler les tests puis corriger ou commenter les éventuels problèmes. Exécuter-les et comparer avec vos prévisions. Noter bien les problèmes détectés à la compilation et ceux à l'exécution.

2.3 – Redéfinition de l'égalité

a. Faire de même avec le test suivant :

```
1     Point pointB;
2     PointPondere pointPondereA, pointPondereB;
3     pointPondereB = new PointPondere("B", 1, 2, 1.5);
4     pointPondereA = new PointPondere("B", 1, 2, 4.5);
5     System.out.println("pointPondereA -> " + pointPondereA +
6                 "  / pointPondereB -> " + pointPondereB);
7     System.out.println("pointPondereA.equals(pointPondereB) -> "
8                 + (pointPondereA.equals(pointPondereB)));
9     pointB = pointPondereB;
10    System.out.println("pointPondereA -> " + pointPondereA +
11                "  / pointB -> " + pointPondereB);
12    System.out.println("pointPondereA.equals(pointB) -> "
13                + (pointPondereA.equals(pointB)));
14    System.out.println("pointB.equals(pointPondereA) -> "
15                + (pointB.equals(pointPondereA)));
16    pointB = new Point("B", 1, 2);
17    System.out.println("pointPondereA -> " + pointPondereA +
18                "  / pointB -> " + pointB);
19    System.out.println("pointPondereA.equals(pointB) -> "
20                + (pointPondereA.equals(pointB)));
21    System.out.println("pointB.equals(pointPondereA) -> "
22                + (pointB.equals(pointPondereA)));
```

Exercice 3. Fin du Triangle et du Cercle

Finir les classes Triangle et Cercle puis leur ajouter les méthodes **equals** et **hashCode**.