

## Égalité entre points et point nommés

### Exercice 1 : Comprendre la surcharge et la redéfinition

L'objectif de cet exercice est de comprendre les notions de surcharge et de redéfinition en s'appuyant sur les classes `Point` (listing 1) et `PointNomme` (listing 2).

On s'intéresse à l'égalité logique de deux points. Contrairement à l'égalité physique qui est réalisée par la comparaison des poignées (il y a égalité physique de deux poignées si elles référencent le même objet), il s'agit de vérifier si les deux points ont les mêmes valeurs d'attributs <sup>1</sup>.

**1.1.** Définir une méthode `isEqual` <sup>2</sup> dans la classe `Point`.

**1.2.** On considère les déclarations suivantes :

```
1 PointNomme pn1 = new PointNomme("A", 1, 2);
2 PointNomme pn2 = new PointNomme("A", 1, 2);
3 PointNomme pn3 = new PointNomme("B", 1, 2);
4 PointNomme pn4 = new PointNomme("A", 1, 1);
5 PointNomme pn5 = new PointNomme("B", 1, 1);
6 Point p1 = new Point(1, 1);
7 Point p2 = new Point(1, 1);
8 Point p3 = p2;
9 Point q1 = pn4;
10 Point q2 = pn5;
11 Point q3 = new PointNomme("A", 1, 1);
```

**1.2.1.** Indiquer, pour les expressions suivantes, les méthodes exécutées et les résultats obtenus.

```
1 p1 == p2
2 p3 == p2
3 p1.isEqual(p2)
4 p1.isEqual(pn4)
5 pn1.isEqual(pn3)
```

**1.2.2.** Dans le dernier cas, indiquer comment faire pour que le résultat de l'expression soit « faux » et modifier en conséquence les classes `Point` et `PointNomme`.

**1.2.3.** Indiquer, pour les expressions suivantes, les méthodes exécutées et les résultats obtenus.

```
1 q1.isEqual(p1)
2 q3.isEqual(pn4)
3 pn4.isEqual(p1)
```

**1.2.4.** Indiquer comment faire pour que le résultat de la dernière expression soit « faux » et modifier en conséquence les classes `Point` et `PointNomme`.

**1.2.5.** Indiquer alors, pour l'expression suivante, les méthodes exécutées et les résultats obtenus.

```
1 pn4.isEqual(q3)
```

---

1. L'égalité logique est en fait plus difficile à définir et dépend de la classe considérée.  
2. La méthode qui correspond à l'égalité logique en Java est la méthode `equals` de la classe `Object`. Tout ce qui sera dit ici sur `isEqual` s'appliquera à `equals`.

**1.2.6.** On souhaite que l'expression précédente s'évalue à « vrai ». Indiquer les éventuelles modifications à apporter.

**1.2.7.** Indiquer alors, pour les expressions suivantes, les méthodes exécutées et les résultats obtenus.

```
1 pn4.isEqual(pn5)
2 pn4.isEqual(q2)
3 q1.isEqual(pn5)
4 q1.isEqual(q2)
```

**1.2.8.** Indiquer et commenter le résultat des deux expressions suivantes :

```
1 p1.isEqual(pn4)
2 pn4.isEqual(p1)
```

## Listing 1 – La classe Point

```
1  /** Définition d'un point avec ses coordonnées cartésiennes. */
2  public class Point {
3      private double x; // abscisse
4      private double y; // ordonnée
5
6      /** Construire un point à partir de son abscisse et de son ordonnée.
7          * @param x abscisse
8          * @param y ordonnée */
9      public Point(double x, double y) {
10         this.x = x;
11         this.y = y;
12     }
13
14     /** Abscisse du point */
15     public double getX() {
16         return x;
17     }
18
19     /** Ordonnée du point */
20     public double getY() {
21         return y;
22     }
23
24     /** Changer l'abscisse du point
25         * @param x la nouvelle abscisse */
26     public void setX(double x) {
27         this.x = x;
28     }
29
30     /** Changer l'ordonnée du point
31         * @param y la nouvelle ordonnée */
32     public void setY(double y) {
33         this.y = y;
34     }
35
36     @Override public String toString() {
37         return "(" + x + "," + y + ")";
38     }
39
40     /** Distance par rapport à un autre point */
41     public double distance(Point autre) {
42         double dx2 = Math.pow(autre.x - x, 2);
43         double dy2 = Math.pow(autre.y - y, 2);
44         return Math.sqrt(dx2 + dy2);
45     }
46
47     /** Translater le point.
48         * @param dx déplacement suivant l'axe des X
49         * @param dy déplacement suivant l'axe des Y */
50     public void translater(double dx, double dy) {
51         x += dx;
52         y += dy;
53     } }
```

Listing 2 – La classe PointNommé

```
1  /** Un point nommé est un point avec un nom. */
2  public class PointNomme extends Point {
3      private String nom;
4
5      /** Construire un point nommé. */
6      public PointNomme(String nom, double x, double y) {
7          super(x, y);
8          this.nom = nom;
9      }
10
11     /** Nom du point nommé */
12     public String getNom() {
13         return nom;
14     }
15
16     /** Changer le nom du point nommé
17      * @param nom le nouveau nom */
18     public void setNom(String nom) {
19         this.nom = nom;
20     }
21
22     @Override public String toString() {
23         return nom + ":" + super.toString();
24     } }
```