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 PointNommé (listing 2).

On souhaite pouvoir tester l'égalité logique de deux points. Contrairement à l'égalité physique qui est réalisée par la comparaison des poignées, il s'agit de vérifier si les deux points ont les mêmes valeurs d'attributs [1].

1.1 Définir une méthode isEqual dans la classe Point.

1.2 On considère les déclarations suivantes :

```
1  PointNommé pn1 = new PointNommé(1, 2, "A");
2  PointNommé pn2 = new PointNommé(1, 2, "A");
3  PointNommé pn3 = new PointNommé(1, 2, "B");
4  PointNommé pn4 = new PointNommé(1, 1, "A");
5  PointNommé pn5 = new PointNommé(1, 1, "B");
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 PointNommé(1, 1, "A");
```

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

```
1  p1 == p2
2  p2 == p3
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 PointNommé.

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

```
1  q1.isEqual(p1)
2  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 PointNommé.

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.

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 q1.isEqual(pn5)
3 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   *  @author     Xavier Crégut (cregut@enseeiht.fr)
3   *  @version    1.2
4   */
5  public class Point
6  {
7      private double x;           // abscisse
8      private double y;           // ordonnée
9
10     /** Construire un point à partir de son abscisse et de son ordonnée.
11       * @param  x       abscisse
12       * @param  y       ordonnée */
13     public Point(double x_, double y_) {
14         x = x_;
15         y = y_;
16     }
17
18     /** Abscisse du point */
19     public double getX()                { return x; }
20
21     /** Ordonnée du point */
22     public double getY()                { return y; }
23
24     /** Changer l'abscisse du point
25       * @param x_       la nouvelle abscisse
26       */
27     public void setX(double x_)         { x = x_; }
28
29     /** Changer l'ordonnée du point
30       * @param y_       la nouvelle ordonnée
31       */
32     public void setY(double y_)         { y = y_; }
33
34     /** Afficher le point */
35     public void afficher() {
36         System.out.print("(" + getX() + "," + getY() + ")");
37     }
38
39     /** Distance par rapport à un autre point */
40     public double distance(Point autre) {
41         return Math.sqrt(Math.pow(autre.getX() - getX(), 2)
42                          + Math.pow(autre.getY() - getY(), 2));
43     }
44
45    /** Translater le point.
46      * @param dx_ déplacement suivant l'axe des X
47      * @param dy_ déplacement suivant l'axe des Y
48      */
49     public void translater(double dx_, double dy_) {
50         x += dx_;
51         y += dy_;
52     }
53 }
```

Listing 2 – La classe PointNommé

```
1  /**     Un point nommé est un point avec un nom.
2   *      @author         Xavier Crégut (cregut@enseeiht.fr)
3   *      @version        1.1
4   */
5  public class PointNommé
6      extends Point
7  {
8
9      /** Nom du point nommé */
10     private String nom;
11
12     /**  Construire un point à partir de son abscisse, de son ordonnée et de son
13       *  nom.
14       *  @param x_      abscisse
15       *  @param y_      ordonnée
16       *  @param nom_    nom
17       */
18     public PointNommé(double x_, double y_, String nom_) {
19         super(x_, y_);  // toujours en première ligne !
20         setNom(nom_);
21     }
22
23     /** Nom du point nommé */
24     public String getNom() {
25         return nom;
26     }
27
28     /** Changer le nom du point nommé
29       *  @param nom_    le nouveau nom
30      */
31     public void setNom(String nom_) {
32         nom = nom_;
33     }
34
35     /** Afficher le point nommé */
36     public void afficher()
37     {
38         System.out.print(getNom() + ":");
39         super.afficher();               // utilisation du afficher de Point
40     }
41
42 }
```