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 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 [1].

1.1. Définir une méthode `isEqual` [2] 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 PointNommé.

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 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.
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

```java
/** Définition d'un point avec ses coordonnées cartésiennes. */
public class Point {
   private double x;  // abscisse
   private double y;  // ordonnée

   /** Construire un point à partir de son abscisse et de son ordonnée.
     * @param x abscisse
     * @param y ordonnée */
   public Point(double x, double y) {
      this.x = x;
      this.y = y;
   }

   /** Abscisse du point */
   public double getX() {
      return x;
   }

   /** Ordonnée du point */
   public double getY() {
      return y;
   }

   /** Changer l'abscisse du point
     * @param x la nouvelle abscisse */
   public void setX(double x) {
      this.x = x;
   }

   /** Changer l'ordonnée du point
     * @param y la nouvelle ordonnée */
   public void setY(double y) {
      this.y = y;
   }

   @Override public String toString() {
      return "(" + x + "," + y + ")";
   }

   /** Distance par rapport à un autre point */
   public double distance(Point autre) {
      double dx2 = Math.pow(autre.x - x, 2);
      double dy2 = Math.pow(autre.y - y, 2);
      return Math.sqrt(dx2 + dy2);
   }

  /** Translater le point.
    * @param dx déplacement suivant l'axe des X
    * @param dy déplacement suivant l'axe des Y */
   public void translater(double dx, double dy) {
      x += dx;
      y += dy;
}  }
```

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 }  }
```