Tester la classe Point

Objectifs :

— Savoir utiliser l'option `-classpath`
— Savoir utiliser la variable d'environement `CLASSPATH`
— Comprendre comment utiliser JUnit pour écrire un programme de test.

La classe PointTest (listing 1) est un programme de test de la classe Point. Il s'appuie sur JUnit (`http://junit.org`). JUnit est un ensemble de classes (un framework) qui facilite l'écriture de programmes de test.

Listing 1 – La classe PointTest

```
1  import org.junit.*;
2  import static org.junit.Assert.*;
3
4  /** Programme de test de la classe Point.
5    * @author Xavier Crégut
6    * @version    1.11
7    */
8  public class PointTest {
9
10     public static final double EPSILON = 1e-6;
11         // précision pour la comparaison entre réels.
12
13     private Point p1;
14     private Point p2;
15
16     @Before
17     public void setUp() {
18         p1 = new Point(1, 2);
19         p2 = new Point(4, -2);
20     }
21
22     @Test
23     public void testInitialisation() {
24         assertTrue(p1 != null);
25         assertTrue(p2 != null);
26         assertTrue(p1.getX() == 1);
27         assertTrue(p1.getY() == 2);
28         assertTrue(p2.getX() == 4);
29         assertTrue(p2.getY() == -2);
30     }
31
32     @Test
33     public void testInitialisationMieux() {
34         // Remarque : faire un test d'égalité sur des réels est à éviter
35         // à cause des erreurs d'arrondi.  En conséquence, il faut
```

```
36          // vérifier que les deux nombres sont égaux à EPSILON près.
37          // C'est ce que fait assertEquals(attendu, réel, précision)
38          assertNotNull(p1);
39          assertNotNull(p2);
40          assertEquals(1.0, p1.getX(), EPSILON);
41          assertEquals(2.0, p1.getY(), EPSILON);
42          assertEquals(4.0, p2.getX(), EPSILON);
43          assertEquals(-2.0, p2.getY(), EPSILON);
44      }
45
46      @Test
47      public void testSetX() {
48          p1.setX(10);
49          assertEquals(10.0, p1.getX(), EPSILON);
50          p1.setX(-5);
51          assertEquals(-5.0, p1.getX(), EPSILON);
52      }
53
54      @Test
55      public void testSetY() {
56          p1.setY(10);
57          assertEquals(10.0, p1.getY(), EPSILON);
58          p1.setY(-5);
59          assertEquals(-5.0, p1.getY(), EPSILON);
60      }
61
62      @Test
63      public void testDistance() {
64          assertEquals(0.0, p1.distance(p1), EPSILON);
65          assertEquals(0.0, p2.distance(p2), EPSILON);
66          assertEquals(5.0, p1.distance(p2), EPSILON);
67          assertEquals(5.0, p2.distance(p1), EPSILON);
68      }
69
70      @Test
71      public void testTranslater1() {
72          p1.translater(2, 4);
73          assertEquals(3.0, p1.getX(), EPSILON);
74          assertEquals(6.0, p1.getY(), EPSILON);
75      }
76
77      @Test
78      public void testTranslater2() {
79          p2.translater(-2, -4);
80          assertEquals(2.0, p2.getX(), EPSILON);
81          assertEquals(-6.0, p2.getY(), EPSILON);
82      }
83
84  }
```

Les annotations [1] sont utilisées pour indiquer le rôle des méthodes de la classe de test.

L'annotation `@Before` indique la méthode, ici `setUp()`, utilisée pour initialiser les attributs

1. Les annotations ont été introduites avec la version 5 de Java. Une annotation permet de marquer un élément du langage afin de lui ajouter une propriété particulière. Les annotations sont exploitées par le compilateur ou lors de

de la classe (les données de test) qui seront utilisés au cours des tests. Cette méthode est appelée avant l'appel de chaque méthode de test. Ici, on déclare deux points p1 et p2 respectivement initialisés avec les coordonnées (1, 2) et (4, -2).

L'annotation `@Test` indique qu'une méthode correspond à un test. C'est une méthode de test.

Dans le code la méthode de test `testInitialisation`, on utilise `assertTrue(Boolean)` qui fait penser au mot-clé `assert` introduit en Java 1.4. Elle vérifie que l'expression passée en paramètre est vraie. Si ce n'est pas le cas, le test sera arrêté et comptabilisé en échec. Cette méthode est en fait une méthode définie par JUnit comme méthode de classe de `org.junit.Assert`.

La méthode `testInitialisationMieux` réalise un test équivalent mais en utilisant d'autres méthodes de vérification disponibles :

- `assertNotNull(poignée)` : vérifie que la poignée est non nulle ;
- `assertNull(poignée)` : vérifie que la poignée est nulle ;
- `assertEquals(attendue, réelle)` : vérifie si l'expression calculée (réelle) correspond bien à l'expression attendue.
- `assertEquals(attendue, réelle, précision)` : Sur les réels (**`float`** ou **`double`**) la méthode précédente n'est pas suffisante car vérifier l'égalité de deux réels n'a pas de sens en général à cause des problèmes d'arrondi. En conséquence cette autre version prend un troisième paramètre qui correspond à la précision de la comparaison ;
- `assertFalse(expression booléenne)` : vérifie si l'expression est fausse ;

Notons que toutes ces méthodes de vérification existent aussi dans une version avec un premier paramètre de type String qui correspond à un message qui est affiché dans le cas où une vérification échoue.

Après avoir exécuté toutes les méthodes de test de la classe de test, un verdict est affiché avec le nombre de tests réussis et le nombre de tests en erreur. L'utilisateur sait aussi pour chaque test s'il a réussi ou non.

Les annotations sont utilisées depuis la version 4 de JUnit. Dans les versions précédentes un résultat similaire était obtenu en imposant des conventions sur le nom des méthodes. La méthode d'initialisation s'appelle `setUp` et le nom d'une méthode de test doit commencer par `test`. Nous avons respecté ces conventions dans cette classe de test.

Exercice 1 : Compilation de la classe PointTest

Dans cet exercice nous allons voir comment compiler la classe `PointTest`.

1.1 Compiler le fichier `PointTest.java` et expliquer les erreurs affichées.

1.2 *Utilisation de l'option -classpath.* En fait, il faut dire au compilateur où se trouve le paquetage `junit.framework`. Il est dans l'archive `/mnt/n7fs/ens/tp_cregut/junit4.jar`. Il faut donc compiler en faisant

```
javac -classpath /mnt/n7fs/ens/tp_cregut/junit4.jar:. PointTest.java
```

Attention à ne pas oublier le répertoire courant dans la liste des endroits (répertoires ou fichiers d'archive) où javac peut aller chercher les classes et fichiers Java.

Compiler le fichier `PointTest.java` en utilisant l'option `-classpath`.

l'exécution du programme. Une annotation est introduite par le caractère « @ » (avant l'élément du langage annoté). Les annotations ne sont pas mises dans des commentaires de documentation.

1.3 *Utilisation de la variable d'environnement `CLASSPATH`.* Comme il peut être fastidieux de toujours utiliser l'option `-classpath`, on peut définir la variable d'environnement. Par exemple, en (ba)sh on peut écrire :

```
export CLASSPATH=/mnt/n7fs/ens/tp_cregut/junit4.jar:.
javac PointTest.java
```

et en (t)csh on peut écrire :

```
setenv CLASSPATH /mnt/n7fs/ens/tp_cregut/junit4.jar:.
javac PointTest.java
```

Compiler le fichier `PointTest.java` en utilisant la variable d'environnement `CLASSPATH`. Dans la suite on considèrera la variable d'environnement `CLASSPATH` définie.

Exercice 2 : Exécution du programme de test
Le programme de test, `PointTest`, n'a pas de méthode principale (`main`), il ne peut donc pas être exécuté directement. Il faut en fait fournir cette classe à la classe `org.junit.runner.JUnitCore` définie par JUnit et qui est chargée de conduire le test. Ainsi, pour lancer les tests de `PointTest`, on fait :

```
java org.junit.runner.JUnitCore PointTest
```

Lancer les tests.

Remarque : Si on veut pouvoir exécuter le programme de test de manière habituelle, avec la machine virtuelle java, il suffit de définir la méthode principale de la classe `PointTest` de la manière suivante.

```
1 public static void main(String[] args) {
2     org.junit.runner.JUnitCore.main(PointTest.class.getName());
3 }
```

Exercice 3 Compléter la classe PointTest en ajoutant deux méthodes de test. L'une qui teste la méthode `getCouleur` sur les deux points p1 et p2 et une deuxième méthode qui teste la méthode `setCouleur`.