Comprendre les annotations

Pour ce TP, en fait pour l'exercice 2, on peut utiliser l'eclipse suivant (un peu vieux) qui contient lombok installé (si vous êtes sur votre machine, vous pouvez installer lombok, sinon faire en ligne de commande):

/mnt/n7fs/ens/tp_cregut/eclipse/eclipse-jee-photon-4.8/eclipse

Exercice 1: Les annotations Deprecated et SuppressWarnings

Les annotations en Java sont précédées par le caractère @. Voyons, à travers plusieurs exemples concrets, comment elles peuvent être utilisées et quand elles sont exploitées.

- **1.** Les annotations Deprecated ¹ et SuppressWarnings ². Les classes de la figure 1 utilisent les annotations Deprecated et SuppressWarnings.
- **1.1.** Après avoir compilé le fichier ExempleDeprecated.java (ou l'avoir ouvert sous Eclipse), expliquer l'intérêt des annotations Deprecated et SuppressWarnings.
- 1.2. Quand sont exploitées ces annotations et par qui?
- **2.** L'annotation Override ³. Considérons les classes de la figure 2 (fichier ExempleOverride.java).
- **2.1.** Est-ce que des erreurs sont signalées lors de la compilation du programme ?
- **2.2.** Que se serait-il passé si l'annotation override n'avait pas été utilisée?
- 2.3. Quand est exploitée cette annotation et par qui?
- 3. JUnit 4. Considérons la classe de test StringBufferTest (listing 3) qui s'appuie sur JUnit 4.
- **3.1.** Expliquer la signification des annotations Test et Before. En particulier, que signifie l'élément expected=IndexOutOfBoundsException.class utilisé en « paramètre » de Test.
- **3.2.** Expliquer comment on fait pour « exécuter » cette classe.
- **3.3.** Quand sont exploitées ces annotations et par qui?

Exercice 2: Lombok

À travers, un exemple concret (et simplifié), nous allons voir quelques précautions à prendre quand on écrit des classes en Java (ou d'autres langages) et comment le projet Lombok ⁴ peut simplifier la vie des programmeurs Java.

L'objectif de la classe Analyseur (listing 4) est d'analyser les données d'un fichier texte dont un exemple est donné au listing 6. Ce fichier contient une donnée par ligne composée de 4 informations : une abscisse, une ordonnée, un numéro d'ordre (ignoré) et une valeur réelle. L'abscisse et l'ordonnée sont des entiers et définissent une position modélisée par la classe Position (listing 5). Les données sont lues par la classe Analyseur et conservées dans un tableau associatif (Map) en sommant les valeurs par position.

- 1. https://docs.oracle.com/javase/8/docs/api/java/lang/Deprecated.html
- 2. https://docs.oracle.com/javase/8/docs/api/java/lang/SuppressWarnings.html
- 3. https://docs.oracle.com/javase/8/docs/api/java/lang/Override.html
- 4. https://projectlombok.org

TP 3 1/8

```
@Deprecated
class AA { }
class BB {
        @Deprecated
        public void m() {
                System.out.println("BB.m();");
}
         }
public class ExempleDeprecated {
        void m1() {
                AA a = new AA();
                BB b = new BB();
                b.m();
        }
        @SuppressWarnings("deprecation")
        void m2() {
                AA a = new AA();
                BB b = new BB();
                b.m():
}
         }
```

FIGURE 1 – Les annotations Deprecated et SuppressWarnings

- 1. Lire le texte des classes fournies et indiquer combien il devrait y avoir d'entrées dans la tableau associatif après le traitement du fichier du listing 6.
- **2.** Exécuter le programme et comparer les résultats obtenus à ceux attendus. Expliquer. Sous Eclipse, le fichier donnees.txt doit être à la racine du projet.
- **3.** Nous n'avons pas obtenu les résultats attendus car deux objets différents seront considérés différents même s'ils ont même abscisse et même ordonnée. Il faut donc redéfinir la méthode equals de Object. De plus, si on redéfinit cette méthode equals, il faut aussi définir la méthode hashCode car deux objets égaux doivent avoir la même valeur de hachage.

On pourrait écrire manuellement ces deux méthodes mais Eclipse propose un assistant ⁵ pour les engendrer. L'utiliser et exécuter à nouveau le programme.

- **4.** Cette classe Position devrait aussi définir les accesseurs et modifieurs sur ses attributs. Là encore, Eclipse peut le faire pour nous. Les engendrer.
- **5.** Décommenter la ligne dans Analyseur qui incrémente la valeur de l'ordonnée de la position. Exécuter le programme. Commenter les résultats.
- **6.** On ne devrait pas modifier un objet utilisé comme clé dans un tableau associatif (lire la documentation de l'interface Map ⁶). Pour le garantir, on décide de rendre notre objet non modifiable

TP 3 2/8

^{5.} Faire clic droit, choisir *Source* puis *generate hasCode()* and equals().

^{6.} https://docs.oracle.com/javase/8/docs/api/java/util/Map.html

```
class A {
        public String unNomTresTresLong(int n) {
                return "A" + n;
}
         }
class B extends A {
        @Override
        public String unNomTresTresLong(int n) {
                return "B" + n;
}
         }
class C extends A {
        @Override
        public String unNonTresTresLong(int n) {
                return "C" + n;
}
         }
class D extends A {
        @Override
        public String unNomTresTresLong(long n) {
                return "D" + n;
}
         }
public class ExempleOverride {
        static void verifier(String attendu, int i, A a) {
                String calcul = a.unNomTresTresLong(i);
                if (! calcul.equals(attendu)) {
                        System.out.println("Erreur: " + calcul + " au lieu de " + attendu);
        }
                 }
        public static void main(String[] args) {
                verifier("A0", 0, new A());
                verifier("B1", 1, new B());
                verifier("C2", 2, new C());
                verifier("D3", 3, new D());
}
         }
```

FIGURE 2 – Intérêt de l'annotation Override

TP 3 3/8

en déclarant les attributs constants. Faire les modifications.

Il faut mettre les attributs en privé et supprimer les modifieurs qui avaient été ajoutés par Eclipse.

7. Utilisons la bibliothèque Lombok ⁷. Il faut commencer par l'installer : aller sur la page du projet, faire download, etc. Malheureusement, nous ne pouvons pas l'installer sur l'Eclipse d'enseignement. Nous allons continuer en ligne de commande...

Revenir à la version initiale de la classe Position (figure 5) et ajouter la décoration @Data ⁸ devant la classe. Si lombok était installée dans Eclipse, on pourrait constater en dépliant la classe Position dans l'explorateur Eclipse que les accesseurs ont été ajoutés ainsi que les méthodes equals et hashCode.

Dans un terminal, on peut compiler la classe Position en faisant :

```
javac -cp lombok.jar:. Position.java
```

Dans un terminal on peut utiliser la commande suivante pour liste les membres de la classe Position :

```
javap Position
```

Ce qui pouvait être engendré automatiquement par Eclipse et par la même polluer notre code est engendré automatiquement grâce à l'annotation @Data lors de la compilation de la classe.

- **8.** Supprimer le constructeur de la classe Position et constater que Lombok l'engendre pour nous. **Remarque :** Pour définir une classe ne contenant que des données immuables, il serait préférable d'utiliser l'annotation @Value qui ajoute automatiquement le modifieur final sur les attributs.
- **9.** Dans la méthode charger de la classe Analyseur, nous pouvons être confrontés aux exceptions FileNotFoundException et IOException. Ici, nous avons décidé de les laisser se propager mais nous ne souhaitions pas avoir à ajouter tous les **throws** correspondant (ces exceptions sont vérifiées par le compilateur). Aussi, nous avons encapsulé l'exception se propageant dans une exception de type RuntimeException, non vérifiée par le compilateur.

Ce motif est courant et Lombok propose une annotation pour le prendre en compte : @SneakyThrows.

De même, au lieu d'utiliser le **try** avec ressource pour garantir que le fichier sera bien fermé, on peut utiliser l'annotation @Cleanup devant la déclaration de la variable in.

Faire ces modifications.

10. Pour finir, vous pouvez consulter la page du projet Lombok 9 et ce billet 10.

Exercice 3: Les annotations en Java

Diagonaliser la documentation fournie par Oracle concernant les annotations : https://docs.oracle.com/javase/tutorial/java/annotations/ et répondre aux questions suivantes :

- 1. Que signifient l'annotation @Target?
- 2. Que signifient l'annotation @Retention?
- 7. https://projectlombok.org
- 8. Ne pas oublier d'ajouter en début de fichier import lombok. Data; pour avoir accès à cette annotation.
- 9. https://projectlombok.org
- $10. \ \mathsf{https://www.toptal.com/java/write-fat-free-java-code-project-lombok}$

TP 3 4/8

- 3. Une annotation peut-elle contenir du code?
- 4. Quel mot-clé permet de définir une annotation?
- 5. Peut-on associer des informations à une annotation?

TP 3 5/8

```
import org.junit.*;
import static org.junit.Assert.*;
public class StringBufferTest {
        private StringBuffer s1;
        @Before public void initialiser() {
                s1 = new StringBuffer("Le texte");
        }
        @Test public void testReverse() {
                s1.reverse();
                assertEquals("etxet eL", s1.toString());
        }
        @Test public void testDelete() {
                s1.delete(2, 7);
                assertEquals("Lee", s1.toString());
        }
        @Test(expected=IndexOutOfBoundsException.class)
        public void erreurSurIndice() {
                assertEquals('e', s1.charAt(-1));
        }
        @Test public void initiale() {
                assertEquals('L', s1.charAt(1));
        }
}
```

FIGURE 3 – Une classe de test JUnit 4

TP 3 6/8

```
import java.io.*;
   import java.util.*;
   public class Analyseur {
            private Map<Position, Double> cumuls;
            public Analyseur() {
                    cumuls = new HashMap<>();
            }
            public void charger() {
11
                    try (BufferedReader in = new BufferedReader(new FileReader("donnees.txt"))) {
12
                             String ligne = null;
13
                             while ((ligne = in.readLine()) != null) {
14
                                     String[] mots = ligne.split("\\s+");
15
                                     assert mots.length == 4;
                                                                      // 4 mots sur chaque ligne
                                     int x = Integer.parseInt(mots[0]);
                                     int y = Integer.parseInt(mots[1]);
18
                                     Position p = new Position(x, y);
19
                                     double valeur = Double.parseDouble(mots[3]);
20
                                     cumuls.put(p, valeur(p) + valeur);
21
                                     // p.setY(p.getY() + 1);
22
                                                                     // p.y += 1;
                             }
23
                    } catch (IOException e) {
24
                             throw new RuntimeException(e);
25
                    }
26
            }
27
28
            public double valeur(Position position) {
29
                    Double valeur = cumuls.get(position);
                    return valeur == null ? 0.0 : valeur;
31
            }
32
33
            public Map<Position, Double> donnees() {
34
                    return Collections.unmodifiableMap(this.cumuls);
            }
36
            public static void main(String[] args) {
38
                    Analyseur a = new Analyseur();
39
                    a.charger();
40
                    System.out.println(a.donnees());
41
                    System.out.println("Nombres de positions : " + a.donnees().size());
42
            }
43
44
   }
```

```
public class Position {
           public int x;
           public int y;
3
           public Position(int x, int y) {
                    this.x = x;
                    this.y = y;
                    // System.out.println("...appel à Position(" + x + ", " + y + ")" + " --> " + this
           }
10
           @Override public String toString() {
11
                    return super.toString() + "(" + x + "," + y + ")";
12
           }
13
14 }
```

FIGURE 5 – La classe Position

```
1 1 1 1 10.5
2 1 1 2 15
3 1 1 3 4.2
4 1 2 1 5
5 2 1 1 18
6 1 2 2 10
7 1 1 4 6.3
8 1 1 5 1.6
9 1 1 6 0.8
10 1 1 7 1.0
11 1 2 3 5
```

FIGURE 6 – Le fichier donnees.txt

TP 3