Université de Rouen	Année 2012-2013
Faculté des sciences	Tests
Master 1 GIL	Fiche de TP n° 4 (2 séances)

1 Couverture de codes

Utilisez le plugin EclEmma pour évaluer la couverture de code du test suivant :

Source de src/CoverageTest.java :

```
public class CoverageTest {
    public static void methodIf(boolean a,
                                 boolean b) {
        if (a || b) {
            b = true;
        }
    }

    private static void lanceException() {
        throw new IllegalArgumentException();
    }

    public static void methodException()
    throws IllegalArgumentException {
        lanceException();
    }
}
```

Source de tests/testCoverageTest.java :

```
public class testCoverageTest {
    @Test
    public void testMethodIf() {
        CoverageTest.methodIf(true, true);
    }

    @Test(expected=IllegalArgumentException.class)
    public void testMethodException() {
        CoverageTest.methodException();
    }
}
```

1. Que signifie les couleurs vert/jaune/rouge dans la coloration du source par EclEmma ?
2. Quelles informations sont fournies par l'onglet « Coverage » lorsqu'on regarde les propriétés d'un fichier source ?
3. Modifiez le test précédent pour obtenir un maximum de lignes vertes.
4. Pourquoi reste-t-il des lignes rouges ? Est-ce vraiment un problème ?
5. EclEmma permet-il de vérifier qu'un test satisafit TER1 ? Et TER2 ?

2 TDD

Dans la suite du TP, on se propose de suivre la méthodologie de développement dirigé par les tests (TDD). Ce TP est la copie plus ou moins fidèle d'un TP proposé en SVL à l'université de Lille 1 (rendons à César ce qui est à César).

2.1 Robots

On se propose de programmer une classe Robot rudimentaire, fournissant :

```
public Robot(int height, int width);
public void avance() throws CannotGoThereException;
public void tourne();
```

ainsi que les accesseurs que vous considérerez comme utiles, de sorte que :

- le robot possède une position dans la salle (deux coordonnées entières) ainsi qu'une orientation (un type énuméré Nord, Sud, etc.) ;
- les entiers fournis au constructeur indiquent la taille d'une salle dans laquelle le robot construit se retrouve plongé ;
- avance() le fait avancer d'une case ou protester s'il ne peut avancer en restant dans la salle ;

– tourne() le fait tourner d'un quart de tour dans le sens des aiguilles d'une montre.

2.2 Télécommande à robots

On veut maintenant pouvoir télécommander ledit robot en se donnant les fonctionnalités suivantes :

```
public class Telecommande {
  public Telecommande(Robot r);

  public void orienteToiAuNord();
  public void orienteToiAuSud();
  public void orienteToiALEst();
  public void orienteToiALOuest();

  public void avanceDe(int cases) throws CannotGoThereException;
  public void avanceEtRebonditDe(int cases);
}
```

de sorte que :

– les méthodes orienteToi... fassent ce qu'on imagine ;
– avanceDe fait avancer le robot du nombre de cases indiqué en paramètre ou protester s'il ne peut avancer en restant dans la salle ;
– le comportement de avanceEtRebonditDe est d'avancer dans la direction actuellement et de poursuivre dans la direction opposée si on atteint le mur.

2.3 Travail demandé

Spécifier ce que la description sommaire du robot et de la télécommande laisse sous-spécifié (au moins la position et orientation initiales du robot dans la salle et ce qui se passe quand les dimensions de la salle ne sont pas raisonnables).

Suivant l'approche TDD et en suivant ladite spécification, implémenter les tests unitaires JUnit et le code applicatif nécessaires pour passer ces tests. On cherchera à réaliser des tests satisfaisants au moins TER2.