Université Abou Bekr Belkaid

Master 2 Génie Logiciel (2020/2021)

TP N^0 2

—

Validation et vérification.
Tests des exceptions

—

Le but de ce TP, est d'apprendre à écrire des codes de test assez concis et clairs, qui garantissent la qualité du code de production et qui permettent d'avoir des feed back rapides. La situation sur laquelle on va travailler, c'est qu'on a à tester des exceptions.

0.1 Introduction

Junit fournit différentes méthodes pour tester la lever des exceptions. Nous allons détaillé ici les méthodes proposer. Prenons en compte la classe Calculatrice qui permet de calculer la racine carrée d'un entier ainsi que la division de deux entiers.

```
public class Calculatrice {

  public double racineCarree(int x)throws IllegalArgumentException{
    if (x <= 0){
      throw new IllegalArgumentException("Impossible de calculer la
          racine carree d'un nombre negative");
    } else {
      return sqrt(x);
    }
  }

  public double diviser(int x, int y) throws IllegalArgumentException{
    if (y==0){
      throw new IllegalArgumentException("impossible de diviser par 0);
    } else {
      return x/y;
    }
```

```
  }
}
```

Nous souhaitons écrire un test qui détermine avec précision laquelle des exceptions est levée.

1 fail() avec try catch

```
@Test
public void racineCarreTest(){
  try {
    calculatrice.racineCarree(-10);
    fail("Une exception doit etre lever si on calcule la racine carre d'un
        nombre negatif");
  }catch(IllegalArgumentException aExp){
    assert(aExp.getMessage().contains("nombre negative"));
  }}
```

Le test passe si l'exception spécifiée dans le try-catch est levée, sinon il échoue à l'exécution de la méthode fail() avec le message suivant "Une exception doit être lever si on calcule la racine carré d'un nombre négatif".

2 Expected annotaion

```
@Test(expected=IllegalArgumentException.class)
public void calculatriceTest() {
  calculatrice.diviser(100,0);
  calculatrice.racineCarre(-10);
}
```

Le test passe quand l'exception spécifiée dans la propriété expected de l'annotation @Test est levée.

3 JUnit @Rule et ExpectedException

```
      @Rule
public ExpectedException thrown = ExpectedException.none();

@Test
public void racineCarreTest(){
  thrown.expect(IllegalArgumentException.class);
  thrown.expectMessage(JUnitMatchers.containsString("nombre
      negative"));
 calculatrice.racineCarre(-10);
}
```

On commence par déclarer thrown qui peut être utilisé dans toutes les méthodes de la classe de tests. Contrairement à Expected on peut faire une assertion sur le contenu du message, avec des matchers de Junit si on veut avoir plus de précisions. Le message suivant s'affichera si aucune exception n'est levée "Expected test to throw (exception with message a string containing "nombre négative" and an instance of java.lang.IllegalArgumentException".

Exerice 1 :

- Écrire un programme permettant de calculer toutes les racines carrées des nombres compris entre A et B, A et B étant deux nombres entiers tels que A ¡ B.
- Écrire un programme permettant d'afficher une matrice de taille MxN remplie par des nombres aléatoires compris entre A et B. Les valeurs M, N, A et B doivent être passées en paramètres.

Écrivez l'intégralité de la classe test. Cette classe doit comprendre :

- Des assertions.
- Formatez correctement les sorties de vos tests en utilisant les annotations @Before et @After.
- Des tests vérifiant que les exceptions sont bien levées quand elles doivent l'être.

Exerice 2 :

Exploitez JUnit et les tests d'exceptions vues précédemment pour tester le programme ShoppingCart. Spécifications :

- Une fois créé, le panier contient 0 articles.
- Lorsqu'il est vide, le panier contient 0 articles.
- Lorsqu'un nouveau produit est ajouté, le nombre d'articles doit être incrémenté.
- Lorsqu'un nouveau produit est ajouté, le nouveau solde doit être la somme des soldes précédent plus le coût du nouveau produit.
- Lorsqu'un élément est supprimé, le nombre d'éléments doit être réduit.
- Lorsqu'un produit qui n'est pas dans le panier est retiré, un ProductNotFoundException doit être levé.