TD n°12 – Tests unitaires

Exercice 1

Cet exercice se réalise en groupe. Arrangez-vous pour constituer 6 groupes équilibrés dans chaque TD et choisir équitablement les sujets de départ (chaque groupe aura les 2 sujets à coder).

1. Choix du sujet

Chaque groupe d'étudiants choisit un sujet dans la liste ci-dessous :

- Écrire un programme permettant de calculer dans un tableau toutes les racines carrées des nombres compris entre A et B, A et B étant deux nombres entiers tels que A < B.
- Ecrire un programme permettant de référencer tous les diviseurs d'un nombre entier N plus grand que 1 dans un tableau de booléens (le contenu du tableau sera à vrai si l'indice est un diviseur de N).

2. Pour le sujet choisi…

Ecrire le squelette de la classe principale, et commentez-le.

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

- Des assertions sur les tableaux.
- Des tests vérifiant que les exceptions sont bien levées quand elles doivent l'être.
- Des tests vérifiant que les boucles s'effectuent dans des temps raisonnables, en particulier pour des grandes valeurs de paramètres.

3. Echange des sujets

- Echanger vos classes avec un binôme ayant traité un sujet différent du vôtre.
- Implémenter la classe principale.
- Appliquer les tests de l'autre groupe.
- Itérer jusqu'à obtention d'un programme fonctionnel et satisfaisant l'ensemble des tests.

Exercice 2

Cet exercice a pour but d'apprendre à définir une classe de Test JUnit regroupant plusieurs tests, et corriger le code en fonction du résultat des tests.

Voici le code de la classe à tester.

```
// A class that adds up a string based on the ASCII value of its
// characters and then returns the binary representation of the sum.

public class BinString {
	public BinString(){}

	public String convert(String s){
		return binarise(sum(s));
	}

	public int sum(String s){
		if (s == "") return 0;
		if (s.length() == 1)
			return ((int)s.charAt(0));
		return ((int) s.charAt(0))+sum(s.substring(1));
	}
```

```
	public String binarise(int x){
		if (x == 0) return "";
		if (x%2 == 1) return "1"+binarise(x/2);
		return "0"+binarise(x/2);
	}
}
```

- A l'aide de la documentation, expliquer ce que fait la classe BinString.
- Comprendre comment fonctionnent les méthodes de la classe en exécutant les traces suivantes :
  - sum("") puis sum("de")
  - binarise(0) puis binarise(7)

Voici maintenant le code de la classe test.

```
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class BinStringTest {
	private BinString binString;

	@Before
	protected void setUp() {
		binString = new BinString();
	}

	@Test
	public void binString(){
		assertNotNull("L'instance n'est pas créée",binString);
	}

	@Test
	public void testSumFunction() {
		int expected = 0;
		assertEquals(expected, binString.sum(""));
		expected = 100;
		assertEquals(expected, binString.sum("d"));
		expected = 265;
		assertEquals(expected, binString.sum("Add"));
	}

	@Test
	public void testBinariseFunction() {
		String expected = "101";
		assertEquals(expected, binString.binarise(5));
		expected = "11111100";
		assertEquals(expected, binString.binarise(252));
	}

	@Test
	public void testTotalConversion() {
		String expected = "1000001";
		assertEquals(expected, binString.convert("A"));
	}

}
```

- A l'aide de la documentation de Junit, expliquer ce que fait la classe BinStringTest.
- Lancer les tests et commenter le résultat. Corriger la classe de tests si besoin.
- En vous aidant des résultats des tests, corriger les erreurs dans la classe principale.
- Relancer les tests jusqu'à ce que le programme fonctionne.