TD n°7 – Tests unitaires

Premiers tests

Reprendre la classe `MaClasse` vue en cours et sa classe de tests unitaires associée et les développer dans Eclipse.

Vous créerez un nouveau projet TD7 et séparerez le répertoire du code Java (src) de celui des tests unitaires JUnit (tests).

Modifier la valeur attendue du résultat de la fonction calculer afin d'analyser les tests en cas d'échec ou de réussite.

Classe Monnaie

Une monnaie représente une certaine valeur d'argent dans une devise particulière. On définit une classe Money permettant de représenter une certaine valeur d'argent (amount) dans une devise particulière (currency). Il est possible d'ajouter à la valeur d'une monnaie la valeur d'une autre monnaie. Cette opération n'est possible que si les deux monnaies ont même devise.

1. Ecrire la classe Money correspondant à l'énoncé ci-dessus.
2. Ecrire une classe de tests MoneyTest, créant 2 objets Money et testant la méthode ajouter.
3. A partir de cette fonction test :

```
@Test public void testEquals() {
    Money m12CHF= new Money(12, "CHF");
    Money m14CHF= new Money(14, "CHF");

    assertTrue(!m12CHF.equals(null));
    assertEquals(m12CHF, m12CHF);
    assertEquals(m12CHF, new Money(12, "CHF"));
    assertTrue(!m12CHF.equals(m14CHF));
}
```

Redéfinir la méthode `equals` dans la classe Monnaie et vérifier que la méthode passe les tests ci-dessus.

Classe IP

Récupérer la classe IP disponible dans SupportCours. Créer un scenario permettant de tester la méthode `ipValide`.

Les exemples suivants permettent de s'assurer que le code traite bien tous les cas :

- ""
- "127.0.0.1"
- "127.231.1.1"
- "1.2.3.4"
- "12.2.3"
- "12.3.213.123.123"

- ”1231.12.2.3”
- ”.1.2.3”
- ”1.2.3.”
- ”1.2..3”

Utiliser les assertions sur les tableaux `assertArray…` de la classe `org.junit.jupiter.api.Assertions.`

Trouvez l’erreur

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.jupiter.api.Assertions.*;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

public class BinStringTest {
	private BinString binString;

	@BeforeEach
```

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

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

	@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.

Compléments

Transformez les classes main fournies pour les TDs précédents `TestTriplet`, `TestBouquet`, `GestionVehicules` en classes de tests unitaires JUnit.