Projet : Tests unitaires avec JUnit5 et Mockito

Description du Projet

Ce projet vise à démontrer l'utilisation des tests unitaires avec **JUnit5** et **Mockito** pour garantir la fiabilité des composants d'un portefeuille électronique (`Purse`) et d'un code secret (`CodeSecret`).

README-pdf coursJUnit5_Mockito-pdf

Objectifs :

- Implémenter des tests unitaires avec **JUnit5**.
- Utiliser **Mockito** pour créer des mocks afin de simuler des comportements et isoler les tests.
- Valider les fonctionnalités telles que le crédit, le débit et la vérification du code secret.

Prérequis

- **Java 8** ou supérieur
- **Maven** ou **Gradle** (selon votre gestionnaire de dépendances préféré)
- **JUnit5** et **Mockito** configurés dans votre projet

Dépendances Maven :

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>PorteMonnaie</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <dependencies>
        <!-- JUnit 5 -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.9.3</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>5.9.3</version>
            <scope>test</scope>
        </dependency>

        <!-- Mockito -->
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>5.5.0</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.1.2</version>
```

```
            </plugin>
        </plugins>
    </build>
</project>
```

Structure du Projet

```
src/
  main/
      java/
          Purse.java
          CodeSecret.java
  test/
      java/
          PurseUnitTest.java
          CodeSecretUnitTest.java
```

Exécution des Tests

Pour exécuter les tests unitaires, utilisez la commande suivante :

```
mvn test
```

Ou si vous utilisez **Gradle** :

```
gradle test
```

Exemples de Tests

1. Test de la classe `Purse`

Le test suivant vérifie que la méthode `credit()` met correctement à jour le solde du portefeuille :

```java
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Assertions;
import org.mockito.Mockito;

public class PurseUnitTest {

    private String codeJuste = "9876";
    private CodeSecret mockPinCode;

    @BeforeEach
    public void setup() {
        mockPinCode = Mockito.mock(CodeSecret.class);
        Mockito.when(mockPinCode.verifierCode(codeJuste)).thenReturn(true);
    }

    @Test
    public void testCredit() throws MotifBlocageTransaction {
        Purse purse = new Purse(100, 100, mockPinCode);
        double solde = purse.getSolde();
        purse.credit(10);
        Assertions.assertEquals(solde + 10, purse.getSolde());
    }
}
```

2. Test de la classe `CodeSecret`

Ce test vérifie le comportement de la méthode `revelerCode()` lorsque le code est masqué :

```java
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
```

```java
import org.junit.jupiter.api.Assertions;
import org.mockito.Mockito;

public class CodeSecretUnitTest {

    private CodeSecret codeSecret;

    @BeforeEach
    public void setup() {
        MyRandom random = Mockito.mock(MyRandom.class);
        Mockito.when(random.nextInt(10)).thenReturn(5, 4, 3, 2);
        codeSecret = CodeSecret.createCode(random);
    }

    @Test
    public void testRevelerCode() {
        Assertions.assertTrue(isCode(codeSecret.revelerCode()));
        Assertions.assertEquals("xxxx", codeSecret.revelerCode());
    }

    private boolean isCode(String code) {
        if (code.length() != 4) return false;
        try {
            Integer.parseInt(code);
        } catch (NumberFormatException e) {
            return false;
        }
        return true;
    }
}
```

3. Exemple de Mocking avec Mockito

L'exemple suivant montre comment simuler le comportement d'un générateur de nombres aléatoires avec **Mockito** :

```java
MyRandom random = Mockito.mock(MyRandom.class);
Mockito.when(random.nextInt(10)).thenReturn(5, 4, 3, 2);
```

Bonnes Pratiques

Isolation des Tests

- Utilisez les mocks pour remplacer les dépendances complexes et rendre vos tests plus isolés.

Assertions

- `Assertions.assertEquals()` : pour comparer deux valeurs.
- `Assertions.assertThrows()` : pour vérifier qu'une exception est lancée.
- `Assertions.assertTrue()` et `Assertions.assertFalse()` : pour tester des conditions.

Vérifications avec Mockito

Utilisez **Mockito** pour vérifier les appels de méthodes :

```java
Mockito.verify(mockPinCode, Mockito.times(1)).verifierCode("9876");
```

Exécution du Projet en Ligne de Commande

Compilation

```
mvn clean compile
```

Exécution des tests

```
mvn test
```

Génération d'un rapport de couverture de test

Si vous utilisez **Jacoco** pour la couverture de test, utilisez :

```
mvn jacoco:report
```

Le rapport sera généré dans le dossier `target/site/jacoco`.

Conclusion

L'utilisation de **JUnit5** et **Mockito** dans vos tests unitaires vous permet de : - Tester efficacement vos classes en isolant les dépendances. - Utiliser des assertions claires pour vérifier les comportements attendus. - Simuler des scénarios complexes avec des mocks pour tester des composants sans avoir à se soucier des dépendances externes.

En suivant ce cours, vous serez capable de créer des tests unitaires robustes pour vos projets Java, garantissant une meilleure fiabilité et maintenabilité de votre code.

Ressources Additionnelles

- Documentation JUnit5
- Mockito Guide

Ce README vous fournira les informations nécessaires pour comprendre, tester et exécuter le projet en utilisant **JUnit5** et **Mockito**.