TDD using JUnit5 and Mockito

1. JUnit\_Basic Testing Exercises

Exercise 1: Setting Up Junit

**Scenario:** You need to set up JUnit in your Java project to start writing unit tests.

**Answer :**

<!-- pom.xml -->

<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>junit-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- JUnit 4 dependency -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

package com.example;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calculator = new Calculator();

int sum = calculator.add(2, 3);

assertEquals(5, sum);

}

}

Exercise 3: Assertions in Junit

**Scenario:** You need to use different assertions in JUnit to validate your test results.

**Answer :**

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.Test;

public class AssertionsTest {

@Test

public void testAssertions() {

// Assert equals: Check if two values are equal

assertEquals(5, 2 + 3, "2 + 3 should be 5");

// Assert true: The condition should be true

assertTrue(5 > 3, "5 is greater than 3");

// Assert false: The condition should be false

assertFalse(5 < 3, "5 is not less than 3");

//Assert null: The object should be null

Object obj1 = null;

assertNull(obj1, "Object should be null");

//Assert not null: The object should not be null

Object obj2 = new Object();

assertNotNull(obj2, "Object should not be null");

    }

}

Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit

**Scenario:** You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

**Answer :**

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calculator;

@Before // runs before each test

public void setUp() {

calculator = new Calculator();

System.out.println("Setup: Calculator created");

}

@After // runs after each test

public void tearDown() {

calculator = null;

System.out.println("Teardown: Calculator cleared");

}

@Test

public void testAddition() {

// Arrange

int a = 10;

int b = 5;

// Act

int result = calculator.add(a, b);

// Assert

assertEquals(15, result);

}

@Test

public void testSubtraction() {

// Arrange

int a = 10;

int b = 5;

// Act

int result = calculator.subtract(a, b);

// Assert

assertEquals(5, result);

    }

}