**Module 2: TDD Using Junit5 and Mockito**

**Exercise 1: Setting up Junit**

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

**Exercise 3: Assertions in Junit**

**Code:**

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

// Assert equals

assertEquals(5, 2 + 3);

// Assert true

assertTrue(5 > 3);

// Assert false

assertFalse(5 < 3);

// Assert null

assertNull(null);

// Assert not null

assertNotNull(new Object());

}

}

**OUTPUT:**

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

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

**Code:**

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTestWithFixture {

private Calculator calculator;

@Before

public void setUp() {

// Arrange

calculator = new Calculator();

}

@After

public void tearDown() {

// Teardown

calculator = null;

}

@Test

public void testAdd() {

// Arrange

int a = 2;

int b = 3;

int expected = 5;

// Act

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

// Assert

assertEquals(expected, result);

}

@Test

public void testSubtract() {

// Arrange

int a = 3;

int b = 2;

int expected = 1;

// Act

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

// Assert

assertEquals(expected, result);

}

}

**OUTPUT:**

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0

**Mockito:**

**Exercise 1: Mocking and Stubbing**

**Code:**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testExternalApi() {

// Create a mock object for the external API

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

// Stub the method to return a predefined value

when(mockApi.getData()).thenReturn("Mock Data");

// Create an instance of MyService with the mock API

MyService service = new MyService(mockApi);

// Call the method that uses the external API

String result = service.fetchData();

// Assert that the result is as expected

assertEquals("Mock Data", result);

}

}

**OUTPUT:**

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

**Exercise 2: Verifying Interactions**

**Code:**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

// Create a mock object for the external API

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

// Create an instance of MyService with the mock API

MyService service = new MyService(mockApi);

// Call the method that uses the external API

service.fetchData();

// Verify that the getData method was called

verify(mockApi).getData();

}

}

**OUTPUT:**

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0