Exercise 1: Setting Up Junit

Pom.xml--<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

CalculatorTest.java--import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAddition() {

int result = 2 + 3;

assertEquals(5, result);

}

}

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--------------------------------------------------------

T E S T S

Running AssertionsTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: X sec

Results :

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

AAA CODE FOR CALCULATOR CLASS--import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calculator;

@Before

public void setUp() {

// Setup: runs before each test

calculator = new Calculator();

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

}

@After

public void tearDown() {

// Teardown: runs after each test

System.out.println("Teardown: Test finished.\n");

}

@Test

public void testAddition() {

// Arrange

int a = 2;

int b = 3;

// Act

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

// Assert

assertEquals(5, result);

}

@Test

public void testSubtraction() {

// Arrange

int a = 5;

int b = 3;

// Act

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

// Assert

assertEquals(2, result);

}

}

OUTPUT-Setup: Calculator instance created.

Teardown: Test finished.

Setup: Calculator instance created.

Teardown: Test finished.

-------------------------------------------------------

T E S T S

-------------------------------------------------------

Running CalculatorTest

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: X sec

Results :

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

3.MOCKITO EXERCISES

Exercise 1: Mocking and Stubbing

CODE-import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testExternalApi() {

// 1️⃣ Create mock

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

// 2️⃣ Stub method

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

// 3️⃣ Use mock in service

MyService service = new MyService(mockApi);

String result = service.fetchData();

// 4️⃣ Assert

assertEquals("Mock Data", result);

}

}

ASSUMPTIONS-// Example ExternalApi interface

public interface ExternalApi {

String getData();

}

// Example service

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**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() {**

**// 1️⃣ Create mock**

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

**// 2️⃣ Use service**

**MyService service = new MyService(mockApi);**

**service.fetchData();**

**// 3️⃣ Verify interaction**

**verify(mockApi).getData();**

**}**

**}**

**OUTPUT--------------------------------------------------------**

**T E S T S**

**-------------------------------------------------------**

**Running MyServiceTest**

**Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: X sec**

**Results :**

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