Digital Nurture 4.0 – Deep Skilling Program

Java Full Stack Engineer (FSE)

Week 2 – Mandatory Exercises Submission

Name: Adishree Devoor

GitHub Profile: https://github.com/Adishree16

Email: [adishreedevoor18@gmail.com](mailto:adishreedevoor18@gmail.com)

Skills Covered:

* TDD using Junit5 and Mockito

**JUnit Testing Exercises**

**Exercise 1: Setting Up Junit**

**Program:**

Calculator.java

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

CalculatorTest

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

int result = calc.add(3, 4);

assertEquals(7, result);

System.out.println("Test passed: 3 + 4 = " + result);

}

}

pom.xml Dependency (Snippet Only)

<dependency>

<groupId>junit</groupId>

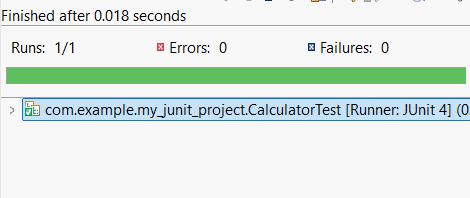
<artifactId>junit</artifactId>

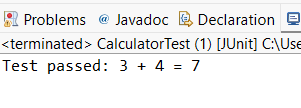
<version>4.13.2</version>

<scope>test</scope>

</dependency>

**Output:**





**Exercise 3: Assertions in Junit**

**Program:**

AssertionsTest.java

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);

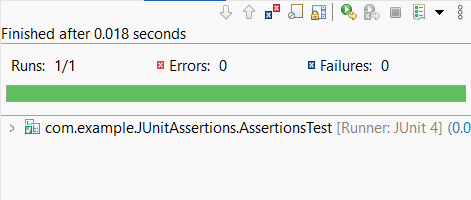
// Assert not null

assertNotNull(new Object());

}

}

**Output:**



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

**Program:**

Calculator.java

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

CalculatorTest.java

package com.example;

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

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

class CalculatorTest {

private Calculator calculator;

@BeforeEach

void setUp() {

calculator = new Calculator();

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

}

@AfterEach

void tearDown() {

calculator = null;

System.out.println("Teardown: Calculator object set to null");

}

@Test

void testAddition() {

int result = calculator.add(5, 3);

assertEquals(8, result);

System.out.println("Test Addition Passed");

}

@Test

void testSubtraction() {

int result = calculator.subtract(10, 4);

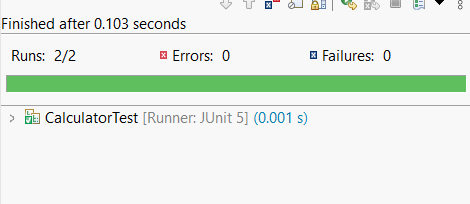
assertEquals(6, result);

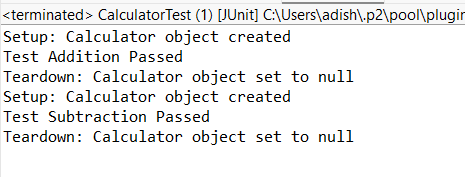
System.out.println("Test Subtraction Passed");

}

}

**Output:**



****

**Mockito Hands-On Exercises**

**Exercise 1: Mocking and Stubbing**

**Program:**

ExternalApi

package com.example;

public interface ExternalApi {

String getData();

}

MyService

package com.example;

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

MyServiceTest

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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

public class MyServiceTest {

@Test

public void testExternalApi() {

// 1. Create mock

ExternalApi mockApi = mock(ExternalApi.class);

// 2. Stub method

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

// 3. Use in service

MyService service = new MyService(mockApi);

// 4. Test

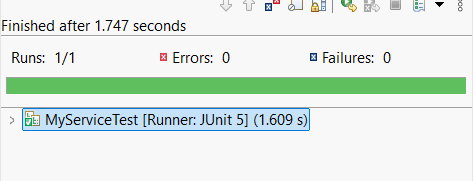
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**Output:**



**Exercise 2: Verifying Interactions**

**Program:**

ExternalApi.java

package com.example;

public interface ExternalApi {

String getData();

}

MyService.java

package com.example;

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

MyServiceTest.java

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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

public class MyServiceTest {

@Test

public void testExternalApi() {

// 1. Create mock

ExternalApi mockApi = mock(ExternalApi.class);

// 2. Stub method

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

// 3. Use in service

MyService service = new MyService(mockApi);

// 4. Test

String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**Output:**

