Digital Nurture 4.0 – Deep Skilling Program

Java Full Stack Engineer (FSE)

Week 2 – Mandatory Exercises Submission

Name: Khushi

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

Email:khushisnagraj12@gmail.com

Skills Covered:

TDD using Junit5 and Mockito

JUnit Testing Exercises

**Exercise 1:Setting up Junit**

**Program:**

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

</dependencies>

package com.example;

public class Hello {

public String sayHello() {

return "Hello, JUnit 5!";

} }

package com.example;

import org.junit.jupiter.api.Test;

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

class HelloTest {

@Test

void testSayHello() {

Hello hello = new Hello();

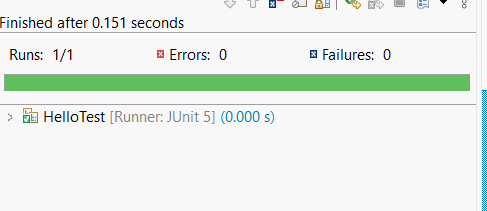
String result = hello.sayHello();

assertEquals("Hello, JUnit 5!", result);

    }

}

**OUTPUT:**



**Exercise 3: Assertions in JUnit**

**Program:**

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

</dependencies>

package com.sahithi.junitdemo;

import org.junit.jupiter.api.Test;

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

class AssertionsTest {

@Test

void testAssertions() {

// assertEquals: checks if two values are equal

assertEquals(5, 2 + 3);

// assertTrue: checks if condition is true

assertTrue(5 > 3);

// assertFalse: checks if condition is false

assertFalse(5 < 3);

// assertNull: checks if value is null

Object obj = null;

assertNull(obj);

// assertNotNull: checks if value is NOT null

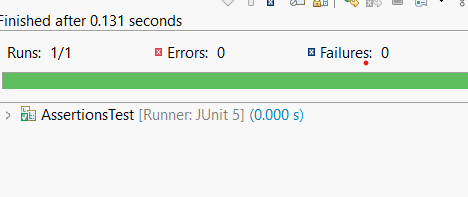
Object notNullObj = new Object();

assertNotNull(notNullObj);

}

}

**OUTPUT:**

****

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

**Teardown Methods in JUnit**

**Program:**

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

</dependencies>

package com.sahithi.junitdemo;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

public int multiply(int a, int b) {

return a \* b;

}

public int divide(int a, int b) {

if (b == 0) throw new IllegalArgumentException("Cannot divide by zero");

return a / b;

}

}

package com.sahithi.junitdemo;

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

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

class CalculatorTest {

private Calculator calculator;

@BeforeEach

void setUp() {

// Arrange: Initialize the object before each test

calculator = new Calculator();

System.out.println("Setup complete.");

}

@AfterEach

void tearDown() {

// Cleanup after each test

calculator = null;

System.out.println("Teardown complete.");

}

@Test

void testAdd() {

// Act

int result = calculator.add(10, 5);

// Assert

assertEquals(15, result);

}

@Test

void testSubtract() {

int result = calculator.subtract(10, 5);

assertEquals(5, result);

}

@Test

void testMultiply() {

int result = calculator.multiply(4, 3);

assertEquals(12, result);

}

@Test

void testDivide() {

int result = calculator.divide(10, 2);

assertEquals(5, result);

}

@Test

void testDivideByZeroThrowsException() {

Exception exception = assertThrows(IllegalArgumentException.class, () -> {

calculator.divide(10, 0);

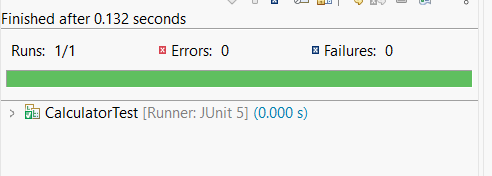
});

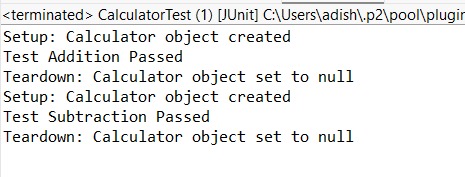
assertEquals("Cannot divide by zero", exception.getMessage());

}

}

**OUTPUT:**

****



Mockito Hands-On Exercises

**Exercise 1: Mocking and Stubbing**

**Program:**

<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>mockito-exercise1</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0</version>

</plugin>

</plugins>

</build>

</project>

package com.example.api;

public interface ExternalApi {

String getData();

}

package com.example.service;

import com.example.api.ExternalApi;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

package com.example.test;

import com.example.api.ExternalApi;

import com.example.service.MyService;

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() {

// Step 1: Create mock

ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Stub method

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

// Step 3: Use mock in service

MyService service = new MyService(mockApi);

// Step 4: Call and assert

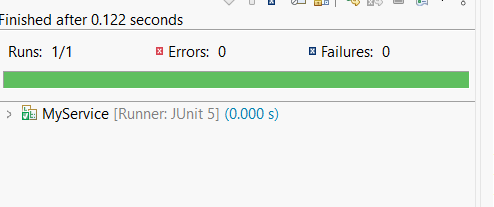
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**OUTPUT:**

****

**Exercise 2: Verifying Interactions**

**Program:**

<dependencies>

<!-- JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0</version>

</plugin>

</plugins>

</build>

package com.example.api;

public interface ExternalApi {

String getData();

}

package com.example.service;

import com.example.api.ExternalApi;

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

package com.example.test;

import com.example.api.ExternalApi;

import com.example.service.MyService;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

// Step 1: Create a mock

ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Pass mock to service

MyService service = new MyService(mockApi);

// Step 3: Call method

service.fetchData();

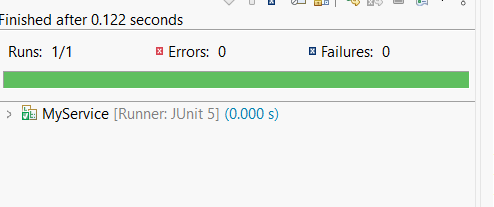
// Step 4: Verify interaction

verify(mockApi).getData(); // This ensures getData() was called

}

}

**OUTPUT:**

****