**Week-2**

**Skill : TDD using JUnit5 and Mockito**

**Hands-on -1 : Exercise 1: Setting Up JUnit**

//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>pgm2</groupId>

<artifactId>pgm2</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>pgm2</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

//Calculator.java

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

//calculatorTest.java

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

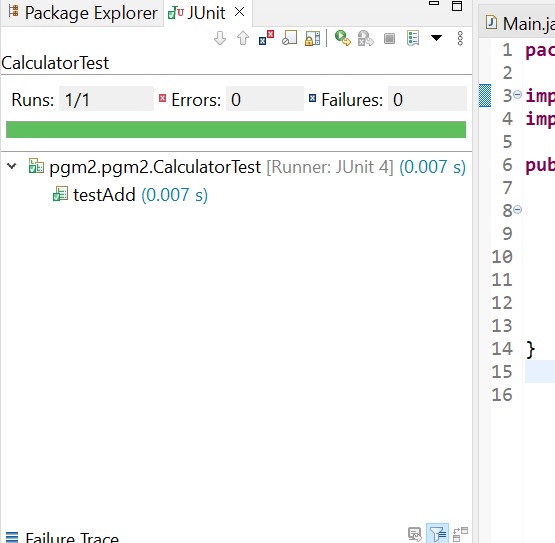
int result = calc.add(5, 3);

assertEquals(8, result);

}

}

Output:-



Hands-on 2 :- Exercise 3: Assertions in JUnit

//AssertionsTest

import static org.junit.Assert.\*;

import org.junit.Test;

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);

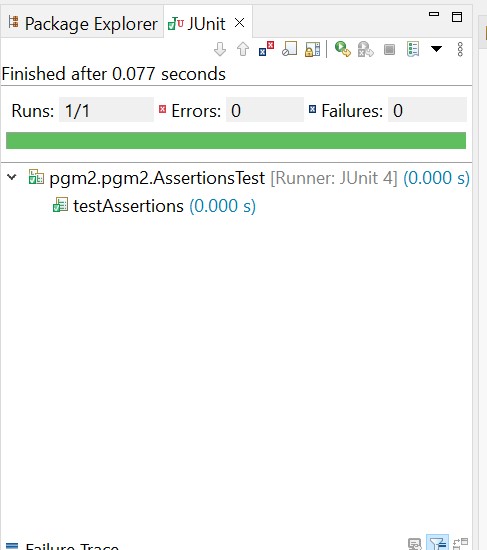
// Assert not null

assertNotNull(new Object());

}

}

Output:-



**Hands-on 3:- Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit**

//Calculator.java

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int multiply(int a, int b) {

return a \* b;

}

}

//CalculatorTest.java

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calc; // shared across tests

@Before

public void setUp() {

// 🔹 Arrange: Create Calculator before each test

calc = new Calculator();

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

}

@After

public void tearDown() {

// 🔹 Teardown: Nullify or release resources

calc = null;

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

}

@Test

public void testAddition() {

// 🔹 Act

int result = calc.add(2, 3);

// 🔹 Assert

assertEquals(5, result);

}

@Test

public void testMultiplication() {

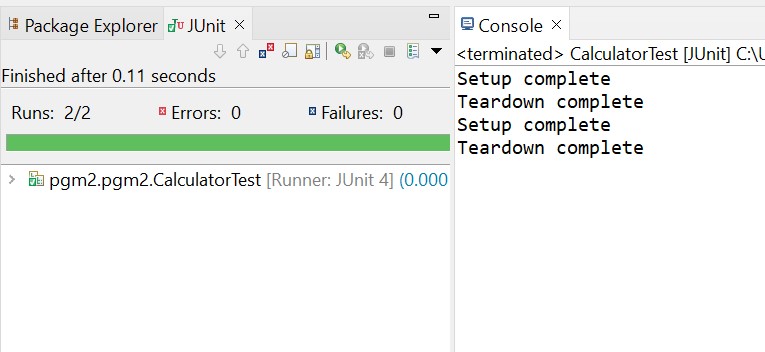
int result = calc.multiply(4, 3);

assertEquals(12, result);

}

}

Output:-



**Hands-on 4 :- Exercise 1: Mocking and Stubbing**

//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>pgm2</groupId>

<artifactId>pgm2</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>pgm2</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<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>

</project>

//ExternalApi.java

public interface ExternalApi {

String getData();

}

//MyService.java

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

//MyServiceTest.java

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() {

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

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

MyService service = new MyService(mockApi);

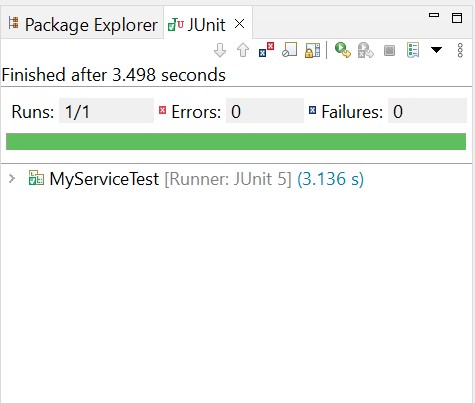
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

Output:-



**Hands-on 5:- Exercise 2: Verifying Interactions**

//MyServiceTest.java

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

// Step 1: Create mock object

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

// Step 2: Inject mock into service and call method

MyService service = new MyService(mockApi);

service.fetchData();

// Step 3: Verify interaction — was getData() called?

verify(mockApi).getData();

}

}

Output:-

