**TDD using Junit5 and Mockito**

**Exercise 1: Setting Up Junit**

package javasample;

import org.junit.Test;

import static org.junit.Assert.\*;

public class Excerise1{

@Test

public void testAddition() {

int result = 2 + 3;

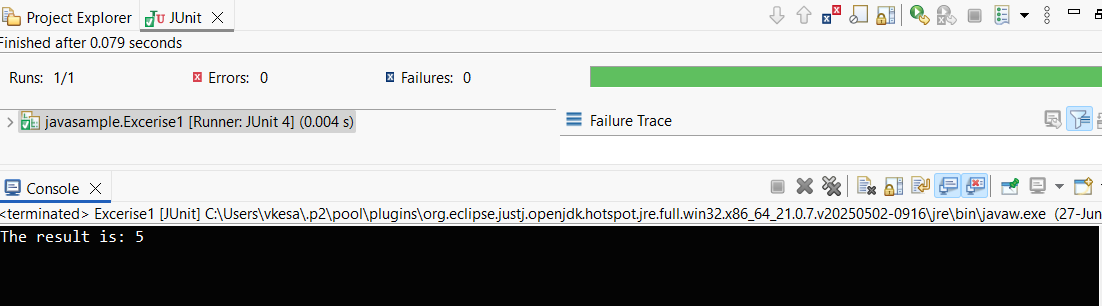
System.*out*.println("The result is: " + result); // This will show in the console

*assertEquals*(5, result);

}

}

**OUTPUT**

****

**Exercise 3: Assertions in Junit**

package javasample;

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());

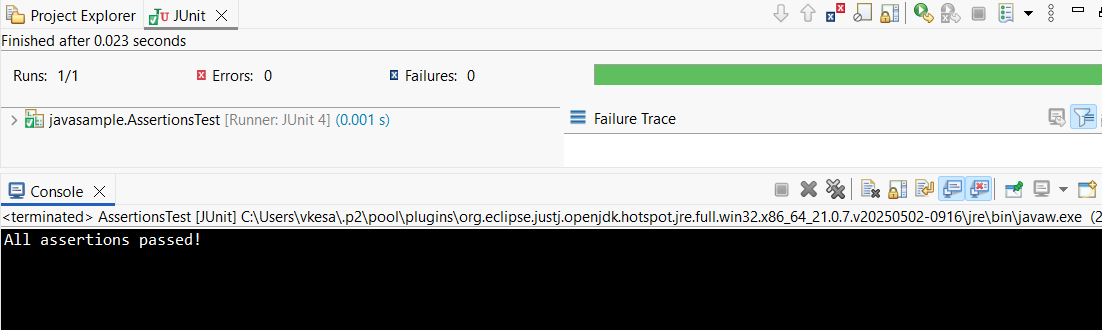
// Console output (optional)

System.*out*.println("All assertions passed!");

}

}

**OUTPUT**

****

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

**Calculator.java**

package ArrangeActAssert;

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 ArrangeActAssert;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calculator;

// Setup method: runs before each test

@Before

public void setUp() {

System.*out*.println("Setting up test...");

calculator = new Calculator();

}

// Teardown method: runs after each test

@After

public void tearDown() {

System.*out*.println("Cleaning up after test...");

calculator = null;

}

@Test

public void testAddition() {

// Arrange

int a = 5;

int b = 3;

// Act

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

// Assert

*assertEquals*(8, result);

}

@Test

public void testSubtraction() {

// Arrange

int a = 10;

int b = 4;

// Act

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

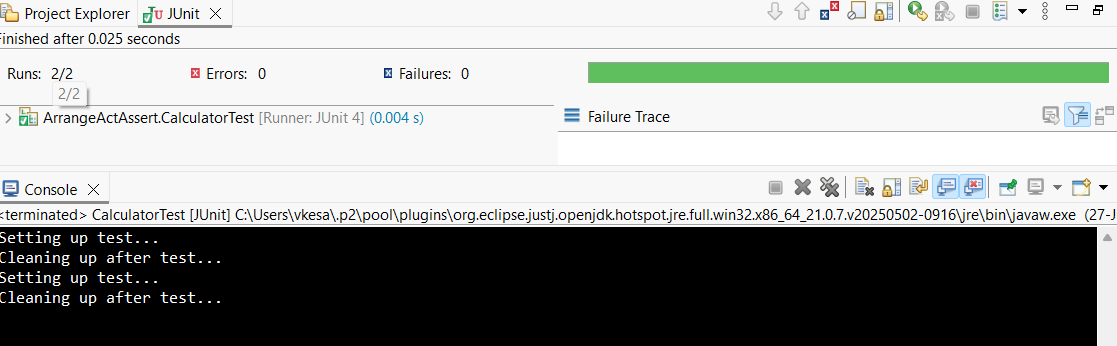
// Assert

*assertEquals*(6, result);

}

}

**OUTPUT**

****

**Exercise 1: Mocking and Stubbing**

**ExternalApi.java**

package com.example.demo;

public interface ExternalApi {

String getData();

}

**MyService.java**

package com.example.demo

public class MyService {

private ExternalApi externalApi;

public MyService(ExternalApi externalApi) {

this.externalApi = externalApi;

}

public String fetchData() {

return externalApi.getData();

}

}

**MyServiceTest.java**

package com.example.demo;

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

import static org.mockito.Mockito.\*

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();

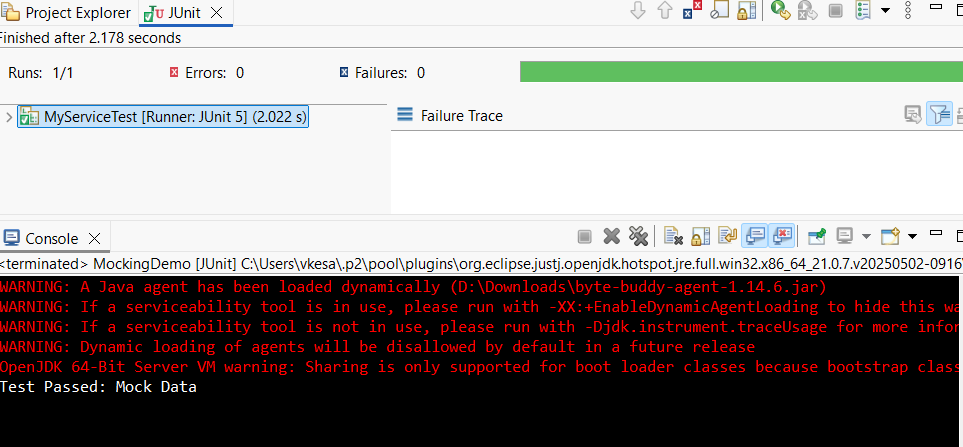
*assertEquals*("Mock Data", result);

System.*out*.println("Test Passed: " + result);

}

}

**OUTPUT**



**Exercise 2: Verifying Interactions**

**ExternalApi.java**

package VERIFYINTERACTION;

public interface ExternalApi {

String getData();

}

**MyService.java**

package VERIFYINTERACTION;

public class MyService {

private ExternalApi externalApi;

public MyService(ExternalApi externalApi) {

this.externalApi = externalApi;

}

public String fetchData() {

return externalApi.getData();

}

}

**MyServiceTest.java**

package VERIFYINTERACTION;

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 a mock object

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

// Step 2: Call the method using MyService

MyService service = new MyService(mockApi);

service.fetchData();

// Step 3: Verify that getData() was called

*verify*(mockApi).getData();

System.*out*.println("Verified: getData() was called on mockApi");

}}

**OUTPUT**

