**Exercise 1: Setting Up JUnit Scenario: You need to set up JUnit in your Java project to start writing unit tests. Steps: 1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse). 2. Add JUnit dependency to your project. If you are using Maven, add the following to your pom.xml: junit junit 4.13.2 test 3. Create a new test class in your project.**

**Calculator class**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**calculatorTest class**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

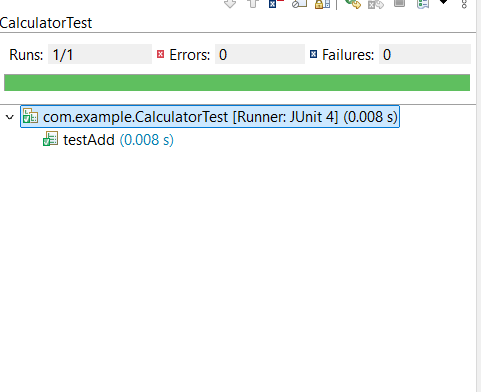
public void testAdd() {

Calculator calc = new Calculator();

*assertEquals*(5, calc.add(2, 3));

}

}



**Exercise 3: Assertions in JUnit Scenario: You need to use different assertions in JUnit to validate your test results. Steps: 1. Write tests using various JUnit assertions. Solution Code: 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()); } }**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionTest {

@Test

public void testAssertions() {

// Assert equals

*assertEquals*("Sum should be 5", 5, 2 + 3);

// Assert true

*assertTrue*("5 should be greater than 3", 5 > 3);

// Assert false

*assertFalse*("5 should not be less than 3", 5 < 3);

// Assert null

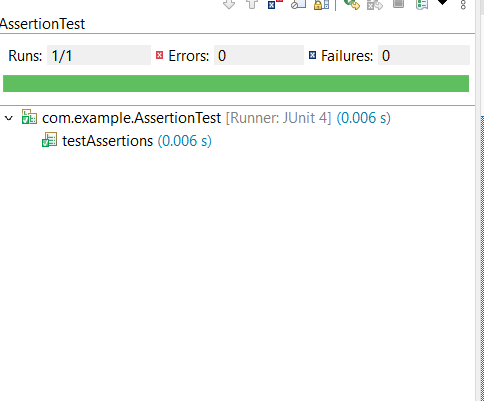
*assertNull*("Value should be null", null);

// Assert not null

*assertNotNull*("Object should not be null", new Object());

}

}



**Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit Scenario: You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods. Steps: 1. Write tests using the AAA pattern. 2. Use @Before and @After annotations for setup and teardown methods.**

**Calculator class**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

**calculatorTest class**

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() {

calculator = new Calculator();

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

}

// Teardown method (runs after each test)

@After

public void tearDown() {

calculator = null;

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

}

@Test

public void testAddition() {

// Arrange

int a = 10;

int b = 20;

// Act

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

// Assert

assertEquals("Addition result should be 30", 30, result);

}

@Test

public void testSubtraction() {

// Arrange

int a = 15;

int b = 5;

// Act

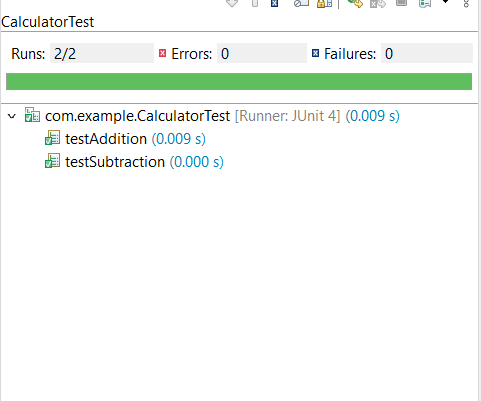
int result = calculator.subtract(a, b);

// Assert

assertEquals("Subtraction result should be 10", 10, result);

}

}



**Exercise 1: Mocking and Stubbing Scenario**

You need to test a service that depends on an external API. Use Mockito to mock the external API and stub its methods. Steps: 1. Create a mock object for the external API. 2. Stub the methods to return predefined values. 3. Write a test case that uses the mock object. Solution Code: 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); } }

**Program**

**File 1: ExternalApi.java**

package com.example;

public interface ExternalApi {

String getData();

}

**File 2: MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**File 3: MyServiceTest.java**

package com.example;

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);

}

}

**Required Dependencies in pom.xml**

<dependencies>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

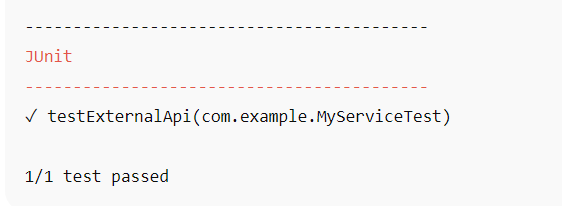
<artifactId>mockito-core</artifactId>

<version>5.2.0</version>

<scope>test</scope>

</dependency>

</dependencies>

****