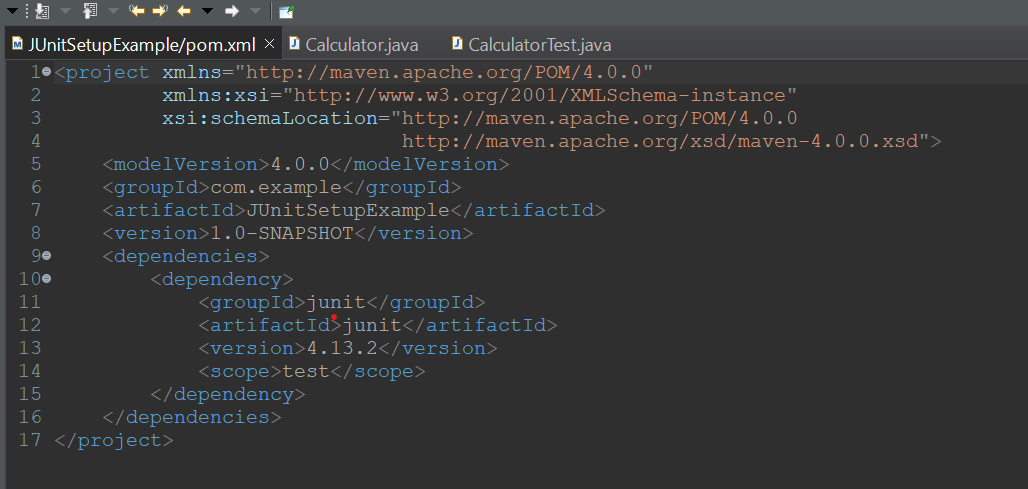
**JUnit Testing Exercises**

**Exercise 1:**

Setting Up JUnit Scenario: You need to set up JUnit in your Java project to start writing unit tests.



**Calculator.java:**

public class Calculator {

// Method to be tested

public int add(int a, int b) {

return a + b;

}

// Optional: main() method for quick manual testing

public static void main(String[] args) {

Calculator calc = new Calculator();

int result = calc.add(10, 20);

System.out.println("Sum = " + result);

}

}

**CalculatorTest.java:**

package com.example.calculator;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

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

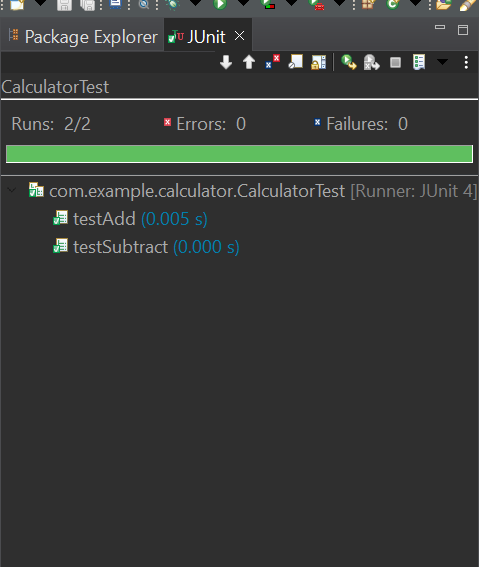
System.out.println("Test output: " + result);

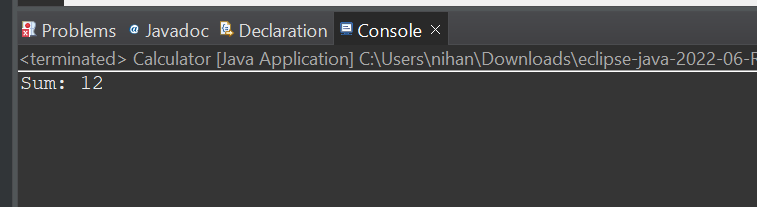
assertEquals(4, result);

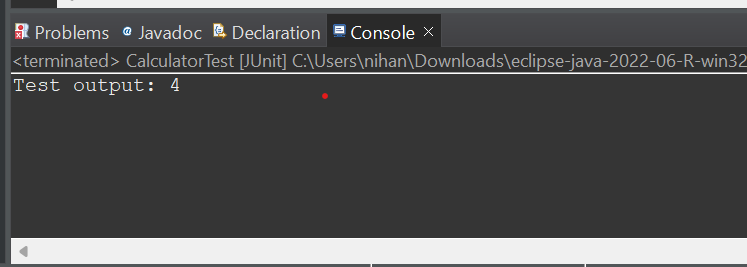
}

}

**Output :**

****

****



**Exercise 3:**

Assertions in JUnit Scenario: You need to use different assertions in JUnit to validate your test results.

**AssertionsTest.java :**

package com.example.calculator;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

System.out.println("Running testAssertions...");

// Assert equals

System.out.println("Asserting 2 + 3 == 5");

assertEquals(5, 2 + 3);

// Assert true

System.out.println("Asserting 5 > 3 is true");

assertTrue(5 > 3);

// Assert false

System.out.println("Asserting 5 < 3 is false");

assertFalse(5 < 3);

// Assert null

Object obj1 = null;

System.out.println("Asserting obj1 is null");

assertNull(obj1);

// Assert not null

Object obj2 = new Object();

System.out.println("Asserting obj2 is not null");

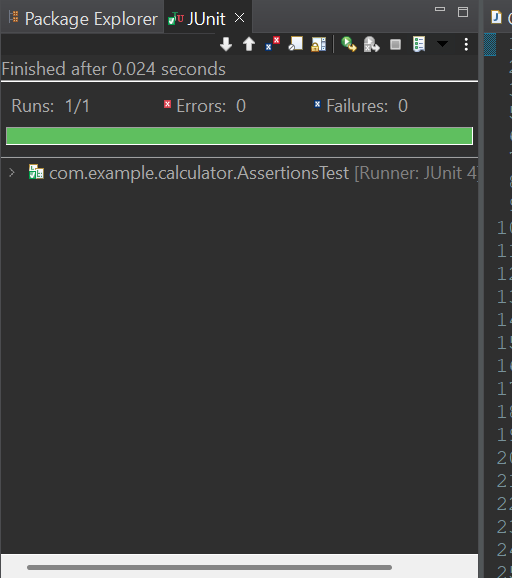
assertNotNull(obj2);

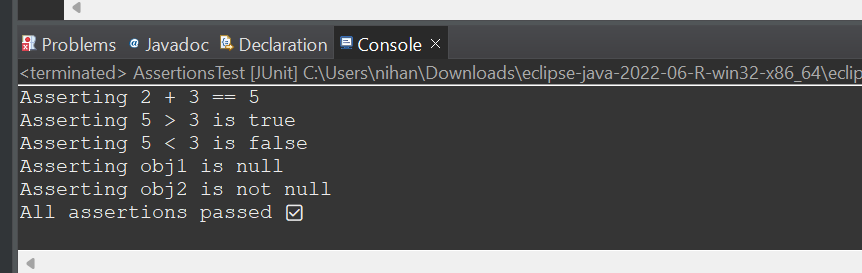
System.out.println("All assertions passed ");

}

}

**Output :**

****

****

**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.

**Calculator.java :**

package com.example.calculator;

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.calculator;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calc;

// Setup method (runs before every test)

@Before

public void setUp() {

System.out.println("Setting up Calculator...");

calc = new Calculator(); // Arrange

}

// Teardown method (runs after every test)

@After

public void tearDown() {

System.out.println("Tearing down Calculator...\n");

calc = null;

}

@Test

public void testAdd() {

// Act

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

// Assert

assertEquals(8, result);

System.out.println("testAdd passed");

}

@Test

public void testSubtract() {

// Act

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

// Assert

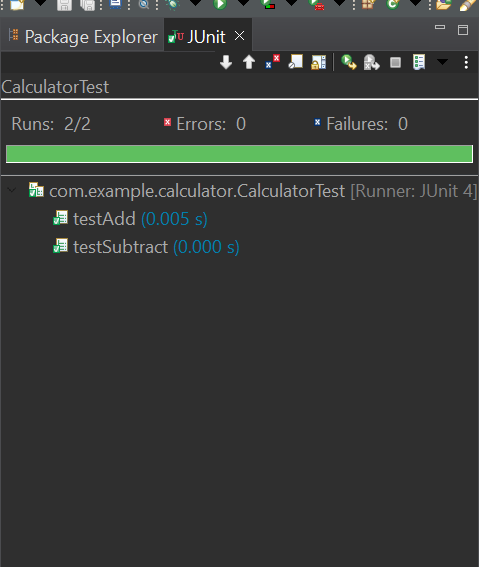
assertEquals(6, result);

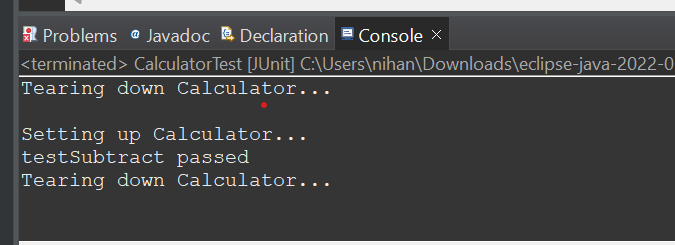
System.out.println("testSubtract passed");

}

}

**Output :**

****

****