**JUnit Testing**

**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.java :**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int sub(int a, int b) {

return a - b;

}

public int mul(int a, int b) {

return a \* b;

}

public int div(int a, int b) {

if (b == 0) throw new ArithmeticException("Division by zero");

return a / b;

}

}

**CalculatorTest.java :**

package com.example;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calculator;

*@Before*

public void setUp() {

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

calculator = new Calculator(); // Arrange

}

*@After*

public void tearDown() {

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

calculator = null;

}

*@Test*

public void testAddition() {

int result = calculator.add(2, 3); // Act

System.***out***.println("Addition Result: 2 + 3 = " + result);

*assertEquals*(5, result); // Assert

}

*@Test*

public void testSubtraction() {

int result = calculator.sub(10, 4);

System.***out***.println("Subtraction Result: 10 - 4 = " + result);

*assertEquals*(6, result);

}

*@Test*

public void testMultiplication() {

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

System.***out***.println("Multiplication Result: 3 \* 4 = " + result);

*assertEquals*(12, result);

}

*@Test*

public void testDivision() {

int result = calculator.div(20, 4);

System.***out***.println("Division Result: 20 / 4 = " + result);

*assertEquals*(5, result);

}

*@Test*(expected = ArithmeticException.class)

public void testDivisionByZero() {

System.***out***.println("Division by zero test (should throw exception):");

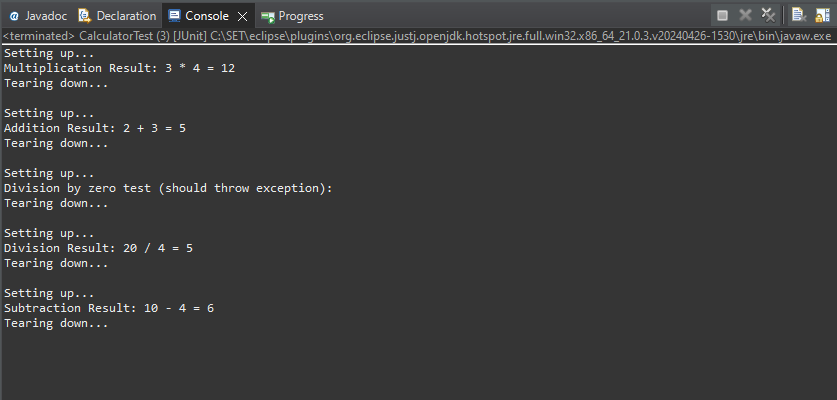
calculator.div(10, 0);

}

}

s

**Output :**

****