**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:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

3. Create a new test class in your project.

**Maven Project:**

**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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.poorani.testing</groupId>

<artifactId>calculator-junit-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<name>JUnit Calculator Demo</name>

</project>

**Calculator.java**

package com.poorani.testing;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorTest.java**

package com.poorani.testing;

import static org.junit.Assert.\*;

import org.junit.Test;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

int result1 = calc.add(5, 10);

int result2 = calc.add(10, 10);

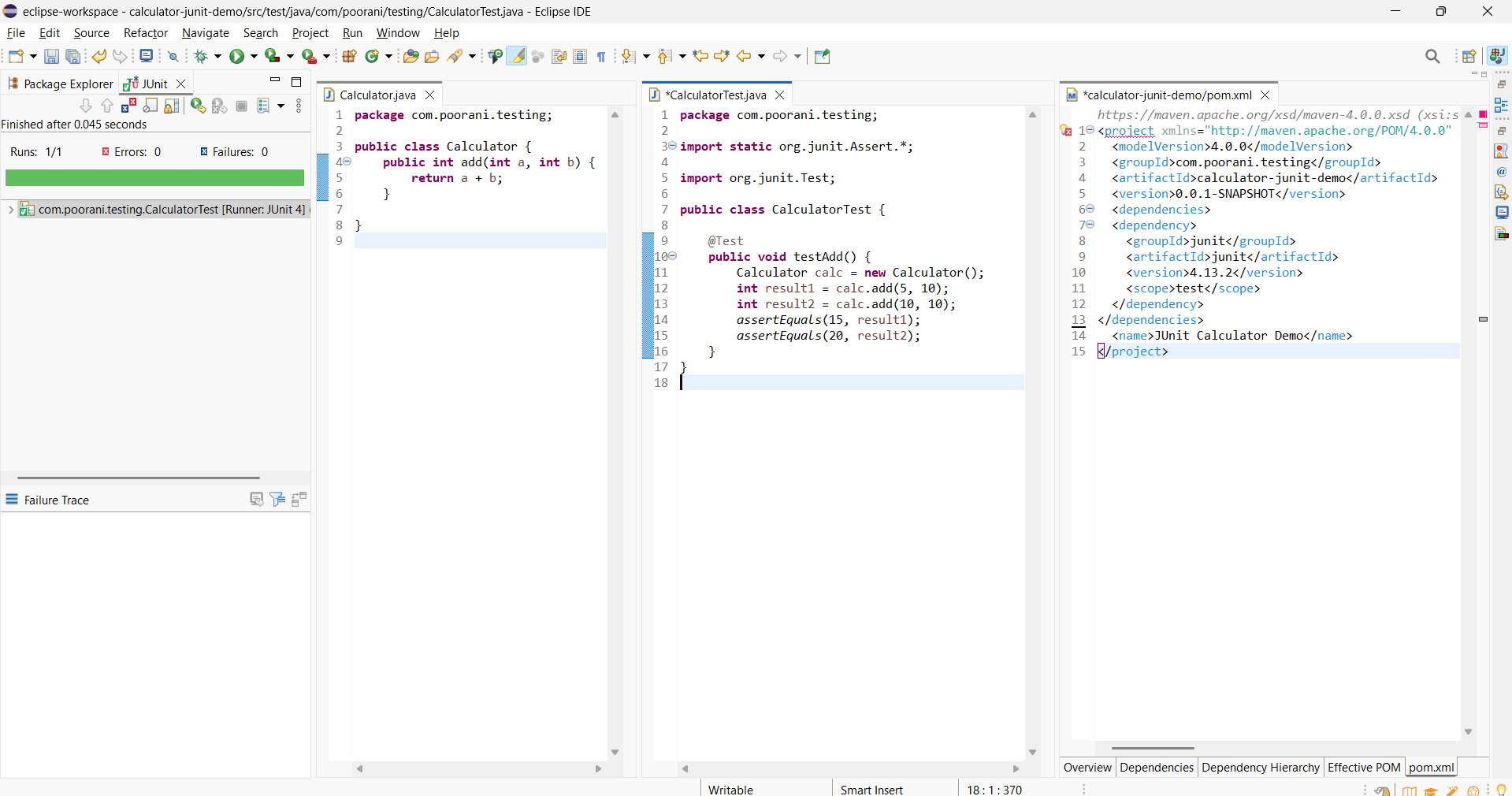
*assertEquals*(15, result1);

*assertEquals*(20, result2);

}

}

**Output:**



**Exercise 2: Writing Basic JUnit Tests**

**Scenario:**

You need to write basic JUnit tests for a simple Java class.

**Steps:**

1. Create a new Java class with some methods to test.

2. Write JUnit tests for these methods.

**Maven Project:**

**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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>JUnitDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**Calculator.java**

package com.poorani.testing;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

public int multiply(int a, int b) {

return a \* b;

}

public int divide(int a, int b) {

if (b == 0) throw new IllegalArgumentException("Division by zero not allowed");

return a / b;

}

}

**CalculatorTest.java**

package com.poorani.testing;

import static org.junit.Assert.*assertEquals*;

import org.junit.Test;

public class CalculatorTest {

Calculator calc = new Calculator();

@Test

public void testAdd() {

*assertEquals*(15, calc.add(10, 5));

}

@Test

public void testSubtract() {

*assertEquals*(5, calc.subtract(10, 5));

}

@Test

public void testMultiply() {

*assertEquals*(50, calc.multiply(10, 5));

}

@Test

public void testDivide() {

*assertEquals*(2, calc.divide(10, 5));

}

@Test(expected = IllegalArgumentException.class)

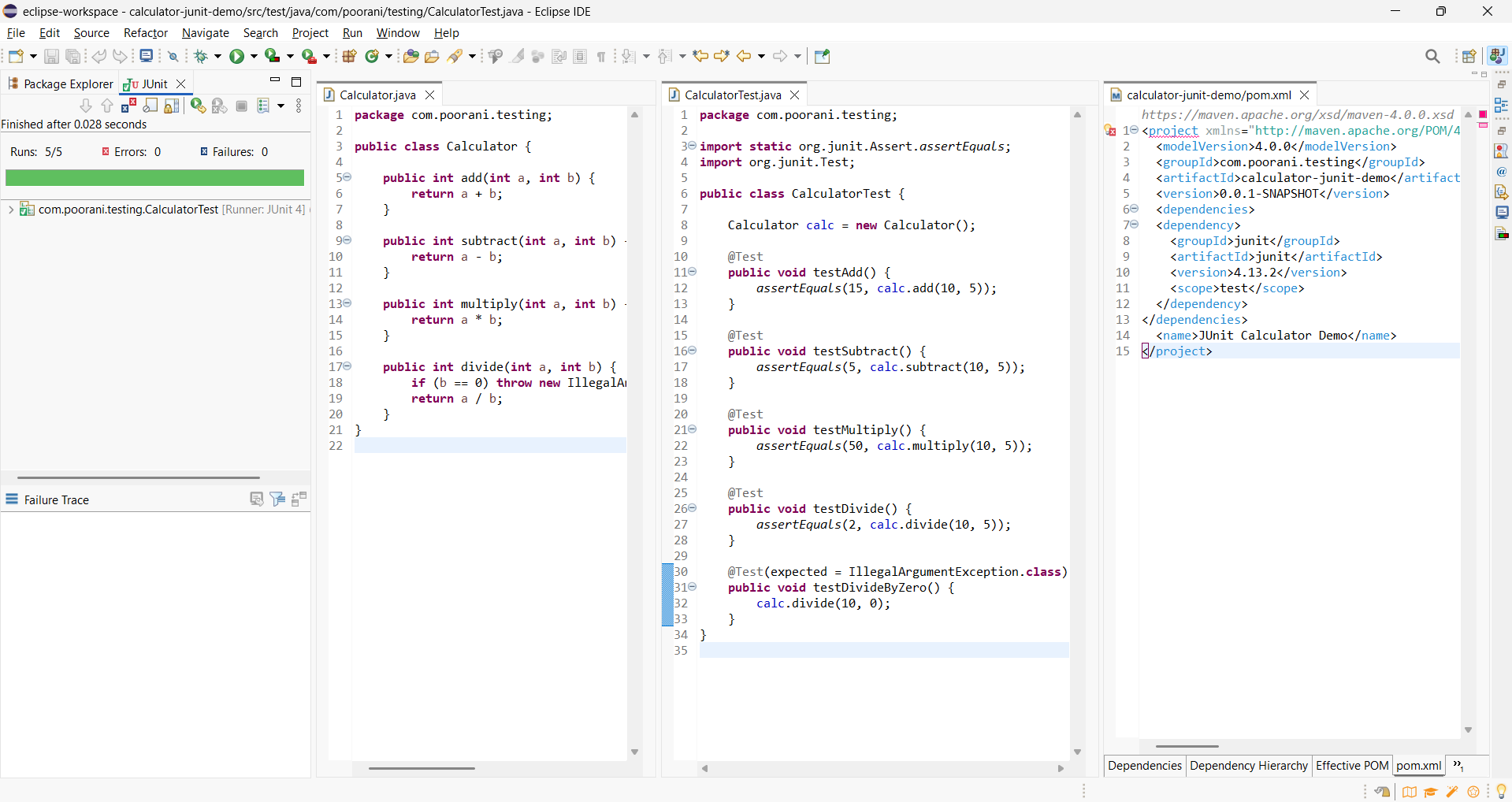
public void testDivideByZero() {

calc.divide(10, 0);

}

}

**Output:**



**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());

}

}

**Maven Project:**

**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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>AssertionsPractice</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**AssertionsTest.java**

package com.poorani.testing;

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());

// Assert same (two references point to the same object)

String str = "JUnit";

*assertSame*(str, str);

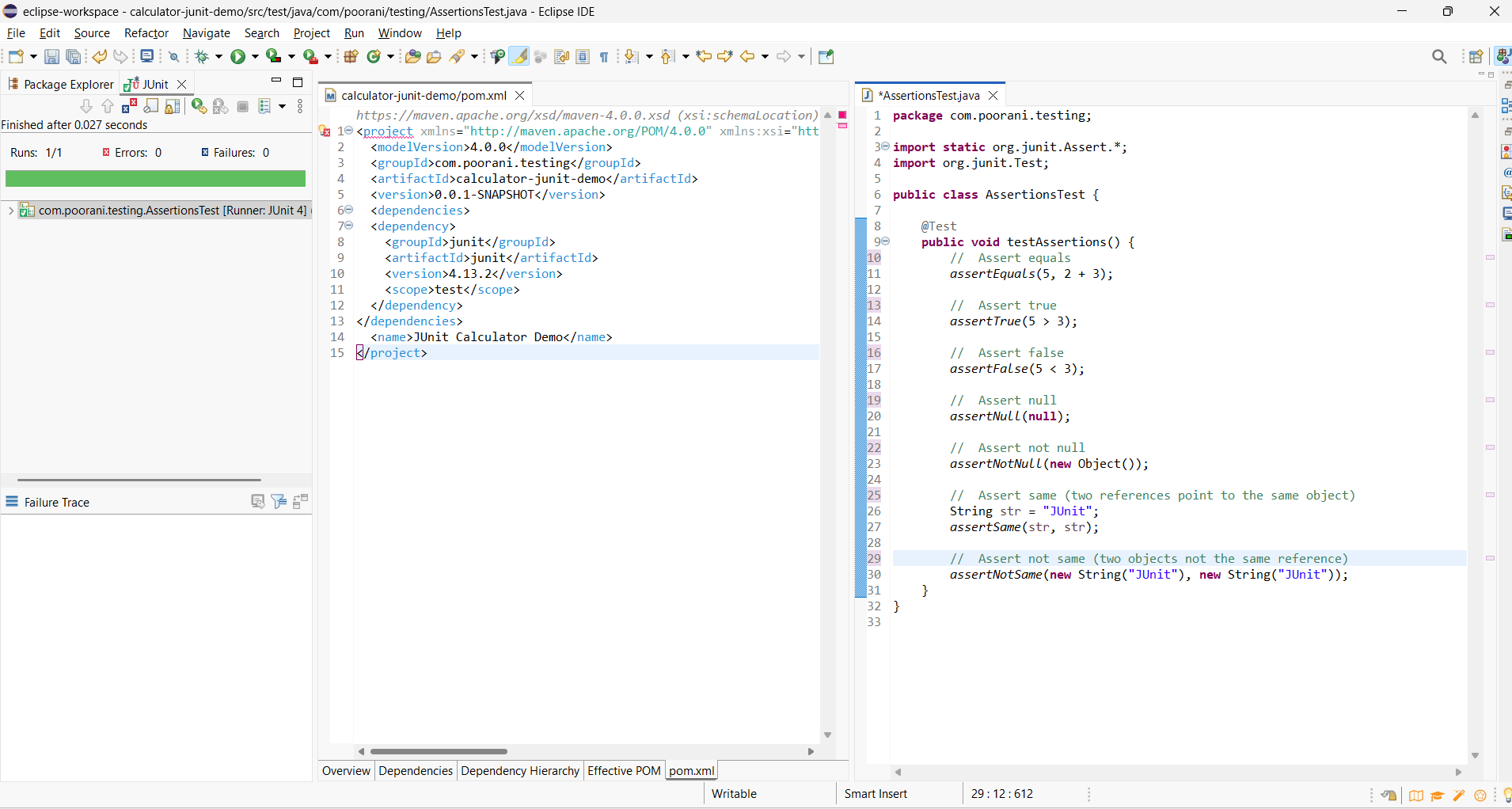
// Assert not same (two objects not the same reference)

*assertNotSame*(new String("JUnit"), new String("JUnit"));

}

}

**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.

**Steps:**

1. Write tests using the AAA pattern.

2. Use @Before and @After annotations for setup and teardown methods.

**Program:**

**Calculator.java**

package com.poorani.testing;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

public int multiply(int a, int b) {

return a \* b;

}

public int divide(int a, int b) {

if (b == 0) throw new IllegalArgumentException("Cannot divide by zero");

return a / b;

}

}

**CalculatorTest.java**

package com.poorani.testing;

import static org.junit.Assert.\*;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

public class CalculatorTest {

private Calculator calc;

// Setup method (runs before each @Test)

@Before

public void setUp() {

calc = new Calculator(); // Arrange

System.*out*.println("Setup: New Calculator instance created.");

}

// Teardown method (runs after each @Test)

@After

public void tearDown() {

System.*out*.println("Teardown: Test completed.\n");

}

@Test

public void testAddition() {

// Act

int result = calc.add(10, 5);

// Assert

*assertEquals*(15, result);

}

@Test

public void testSubtraction() {

int result = calc.subtract(10, 5);

*assertEquals*(5, result);

}

@Test

public void testMultiplication() {

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

*assertEquals*(12, result);

}

@Test

public void testDivision() {

int result = calc.divide(20, 5);

*assertEquals*(4, result);

}

@Test(expected = IllegalArgumentException.class)

public void testDivisionByZero() {

calc.divide(10, 0);

}

}

**Output:**

