**WEEK-2 JUnit Testing Exercises**

**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

3. Create a new test class in your project.

**package** com.example;

**public** **class** Calculator {

**public** **int** add(**int** a, **int** b) {

**return** a + b;

}

**public** **int** multiply(**int** a, **int** b) {

**return** a \* b;

}

**public** **int** subtract(**int** a, **int** b) {

**return** a - b;

}

}

**3.test class:**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class BasicTest {

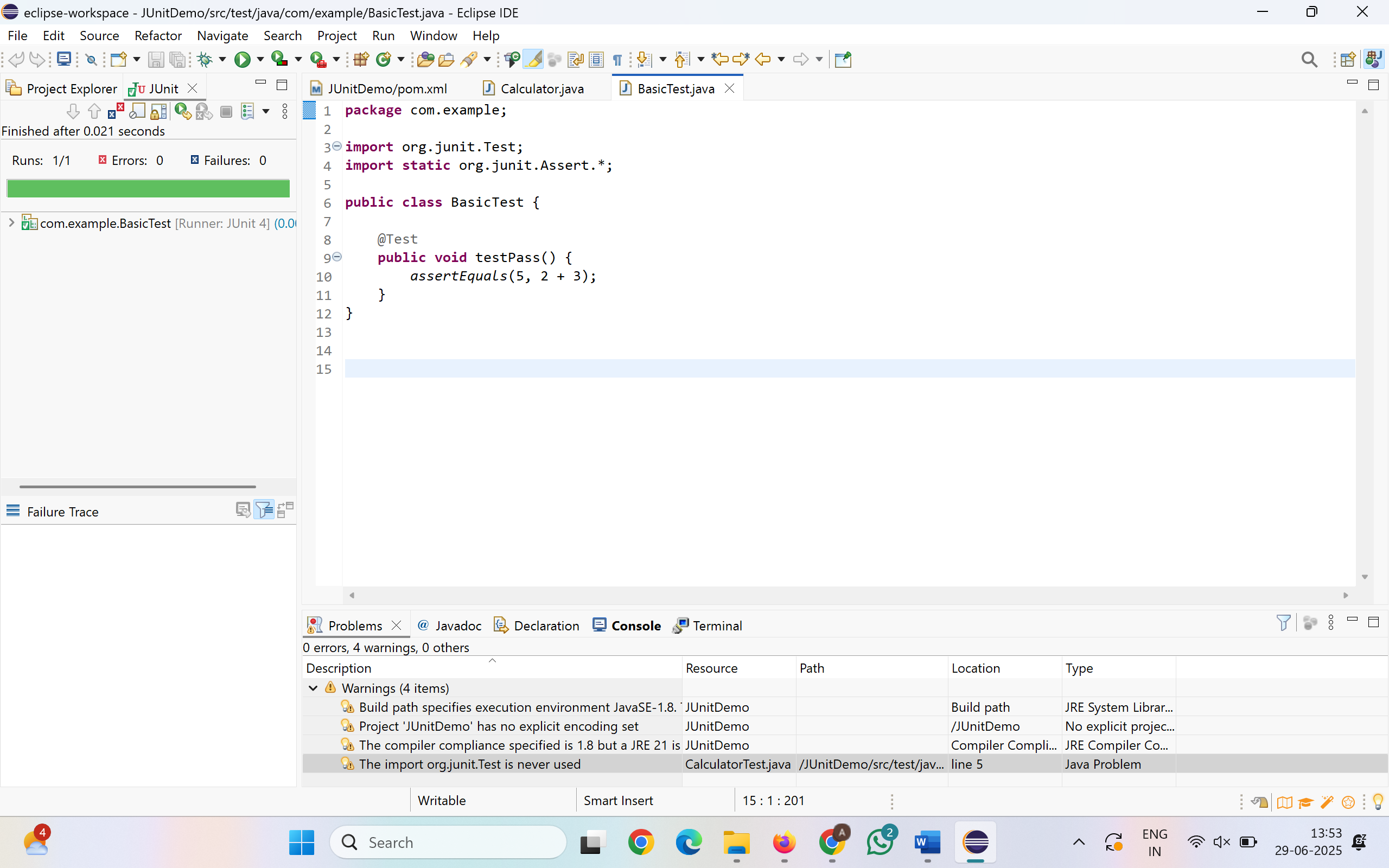
@Test

public void testPass() {

assertEquals(5, 2 + 3);

}

}



**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.

**Calculator.java:**

package com.example;

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.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));

}

@Test

public void testSubtract() {

Calculator calc = new Calculator();

assertEquals(1, calc.subtract(3, 2));

}

@Test

public void testMultiply() {

Calculator calc = new Calculator();

assertEquals(6, calc.multiply(2, 3));

}

@Test

public void testDivide() {

Calculator calc = new Calculator();

assertEquals(2, calc.divide(6, 3));

}

@Test(expected = IllegalArgumentException.class)

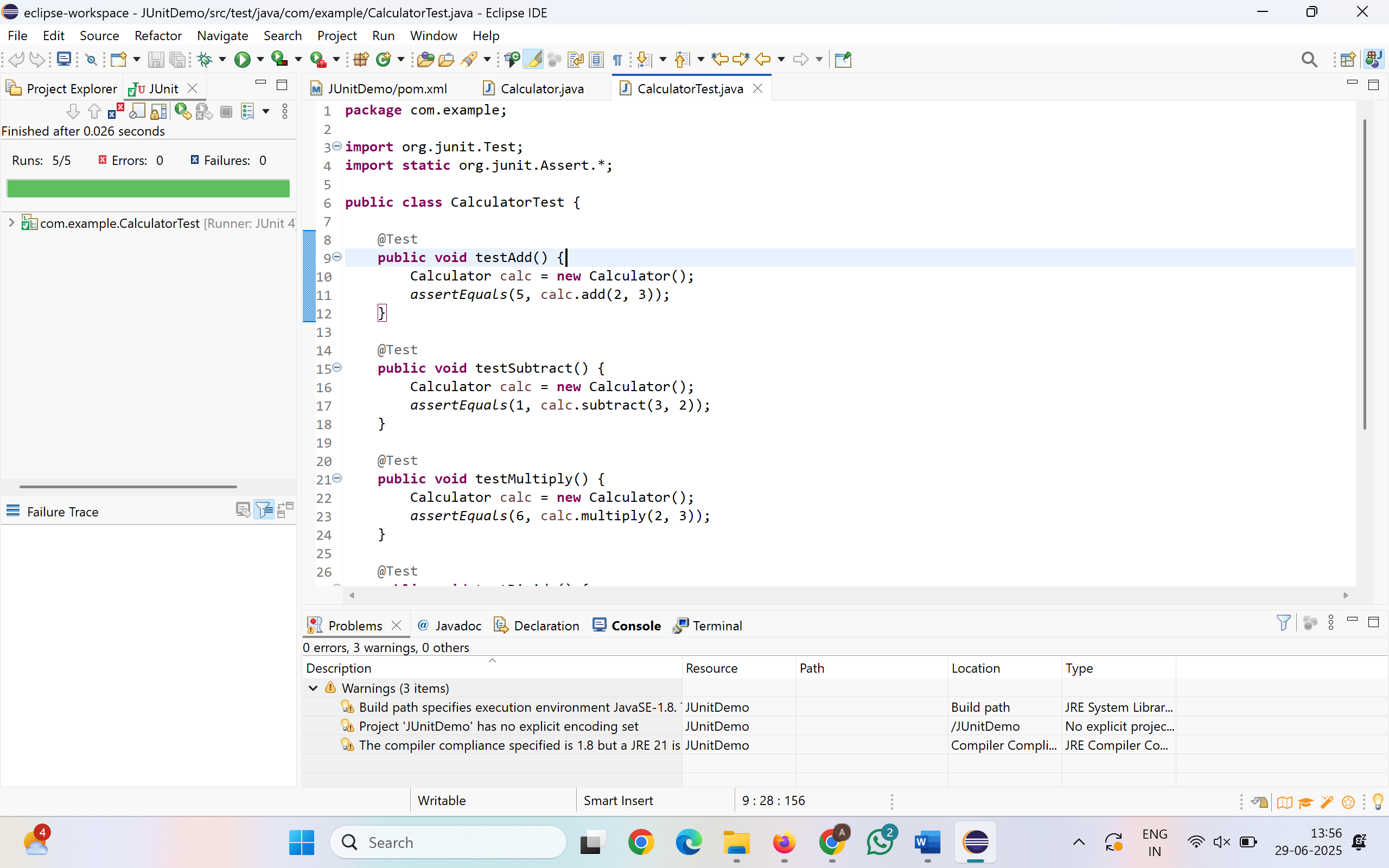
public void testDivideByZero() {

Calculator calc = new Calculator();

calc.divide(10, 0); // Should throw exception

}

}



**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

package com.example;

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

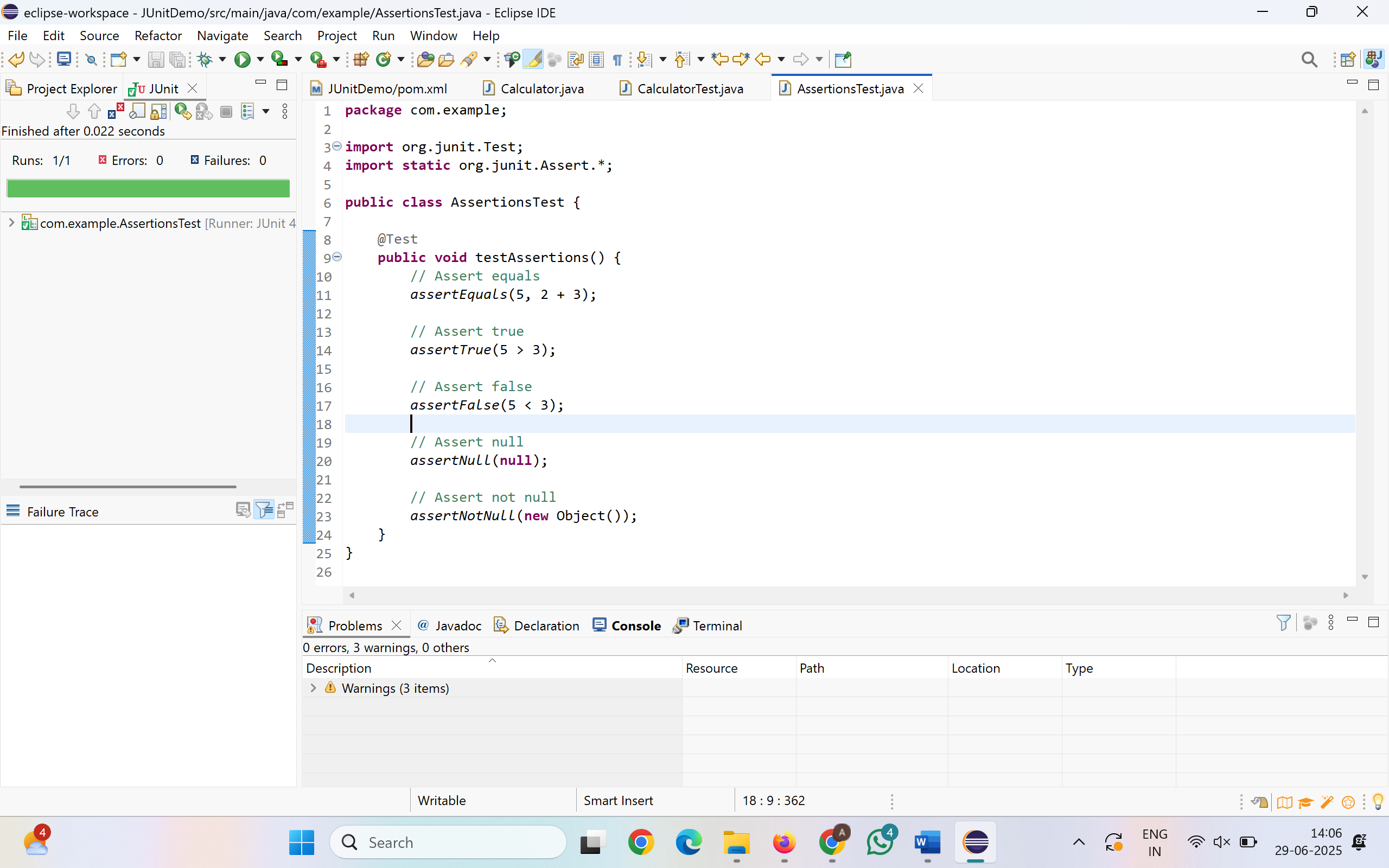
assertNull(null);

// Assert not null

assertNotNull(new Object());

}

}



**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 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("Divide by zero");

return a / b;

}

}

**CalculatorFixtureTest.java:**

package com.example;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorFixtureTest {

private Calculator calc;

@Before

public void setUp() {

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

calc = new Calculator();

}

@After

public void tearDown() {

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

calc = null;

}

@Test

public void testAdd() {

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

*assertEquals*(30, result);

}

@Test

public void testMultiply() {

int result = calc.multiply(5, 4);

*assertEquals*(20, result);

}

@Test

public void testDivide() {

int result = calc.divide(10, 2);

*assertEquals*(5, result);

}

}

