**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

3. Create a new test class in your project.

**Solution:**

Main class:

**package** com.example;

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

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

**return** a + b;

}

}

Test class:

**package** com.example;

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

**import** org.junit.Test;

**public** **class** CalculatorTest {

@Test

**public** **void** testAdd() {

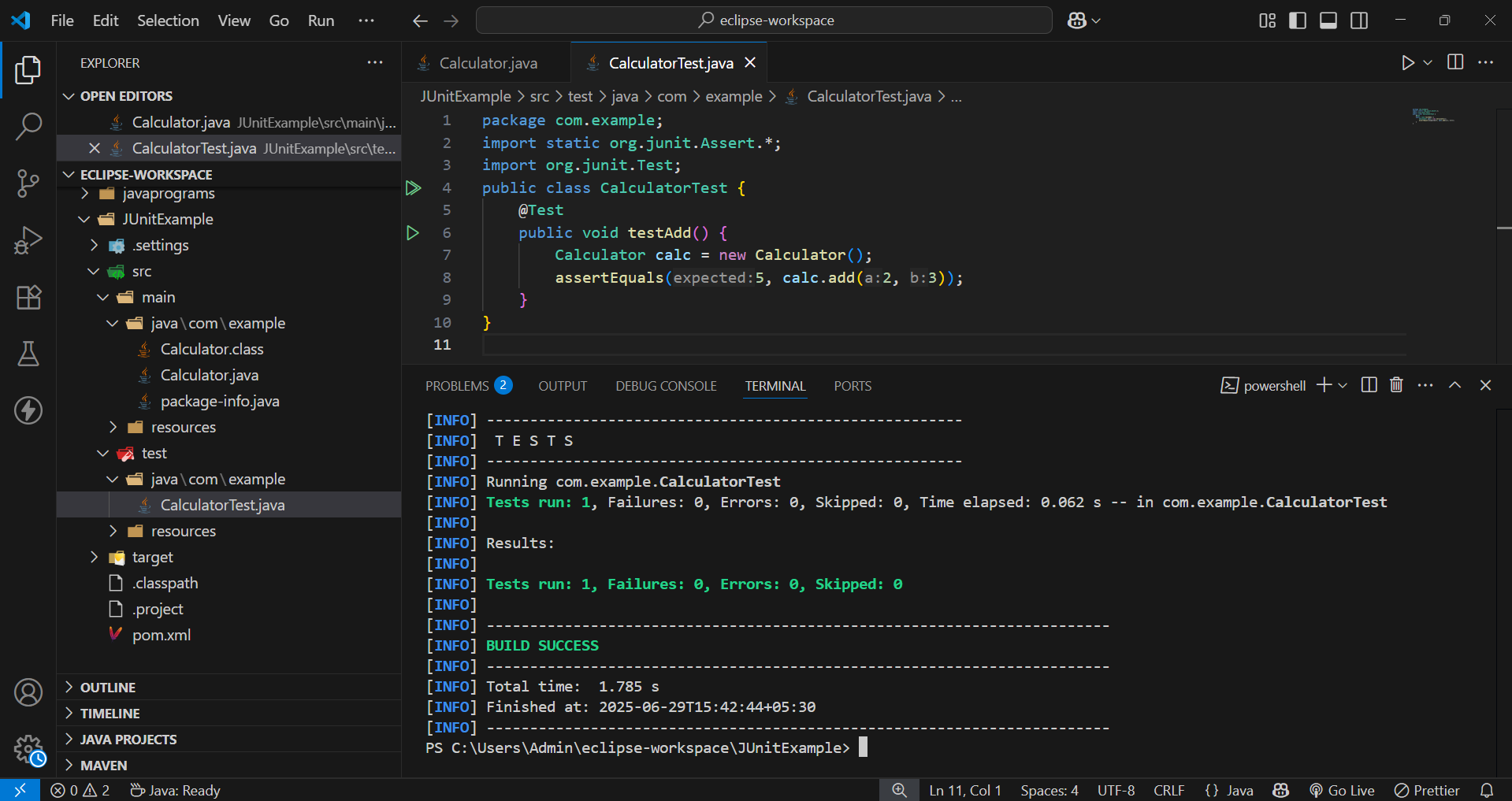
Calculator calc = **new** Calculator();

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

}

}

**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.

**Solution:**

Main class:

**package** com.example;

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

// Method to add two integers

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

**return** a + b;

}

// Method to subtract two integers

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

**return** a - b;

}

// Method to multiply two integers

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

**return** a \* b;

}

// Method to divide two integers

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

**if** (b == 0) {

**throw** **new** IllegalArgumentException("Cannot divide by zero");

}

**return** a / b;

}

}

Test class:

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

Calculator calc = new Calculator();

@Test

public void testAdd() {

int result = calc.add(2, 3);

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

assertEquals(5, result);

}

@Test

public void testSubtract() {

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

System.out.println("Subtraction Result (5 - 4): " + result);

assertEquals(1, result);

}

@Test

public void testMultiply() {

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

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

assertEquals(12, result);

}

@Test

public void testDivide() {

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

System.out.println("Division Result (10 / 5): " + result);

assertEquals(2, result);

}

@Test(expected = IllegalArgumentException.class)

public void testDivideByZero() {

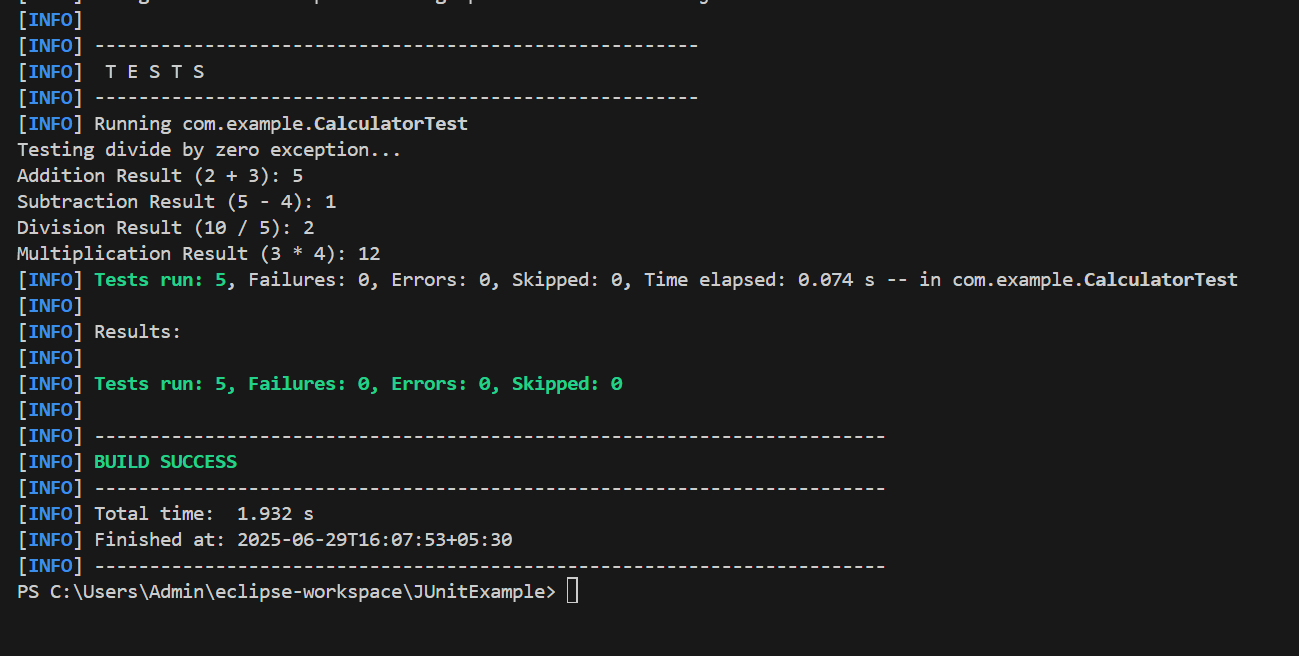
System.out.println("Testing divide by zero exception...");

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.
2. 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());

}

}

**SOLUTION :**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

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

assertEquals(5, 2 + 3);

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

assertTrue(5 > 3);

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

assertFalse(5 < 3);

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

assertNull(null);

System.out.println("value is null");

assertNotNull(new Object());

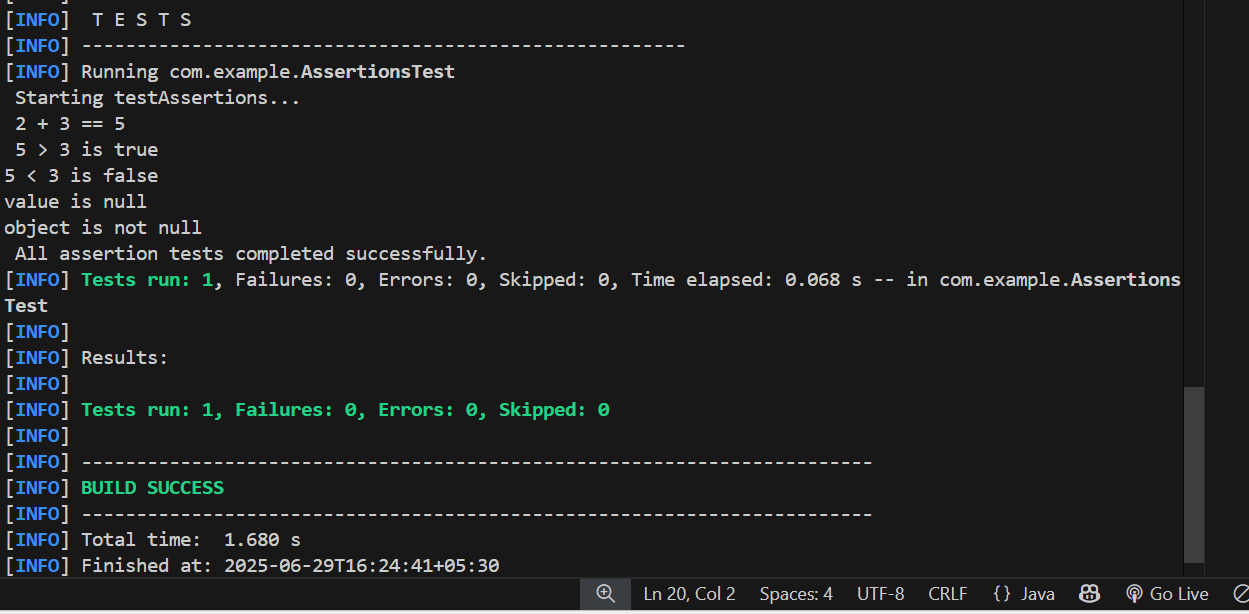
System.out.println("object is not null");

System.out.println(" All assertion tests completed successfully.");

}

}

**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

**Solution:**

Main class:

**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;

}

}

Test class:

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 calc;

    // Setup method: runs before each test

    @Before

    public void setUp() {

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

        calc = new Calculator();

    }

    // Teardown method: runs after each test

    @After

    public void tearDown() {

        calc = null;

        System.out.println("Cleaning up test...\n");

    }

    @Test

    public void testAdd() {

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

        // Act

        int result = calc.add(5, 7);

        // Assert

        assertEquals(12, result);

        System.out.println(" Asserted that 5 + 7 = 12");

    }

    @Test

    public void testSubtract() {

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

        // Act

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

        // Assert

        assertEquals(6,result);

        System.out.println(" Asserted that 10 - 4 = 6");

    }

}

**OUTPUT :**

