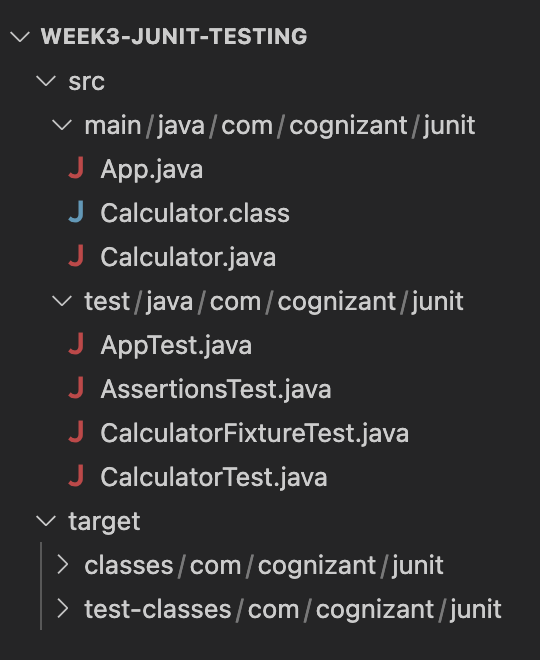
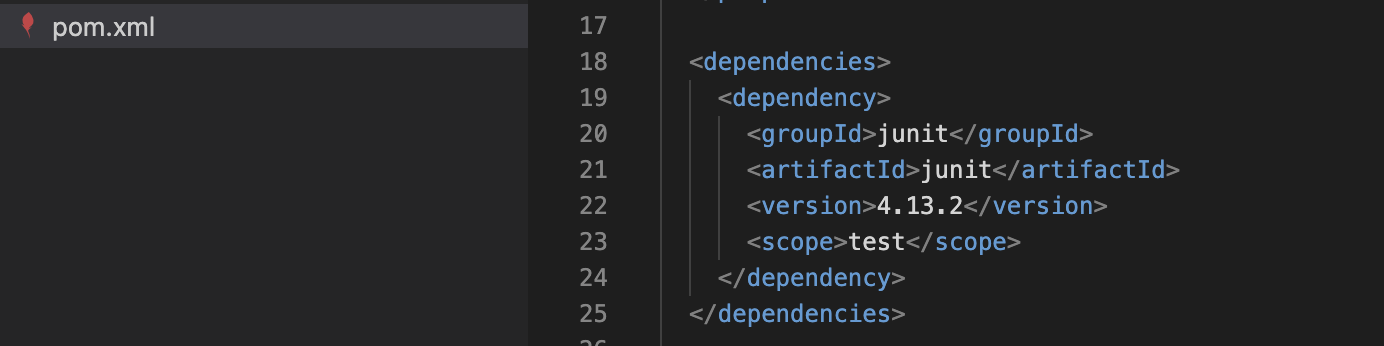
**Exercise 1: Setting Up JUnit (MANDATORY)**

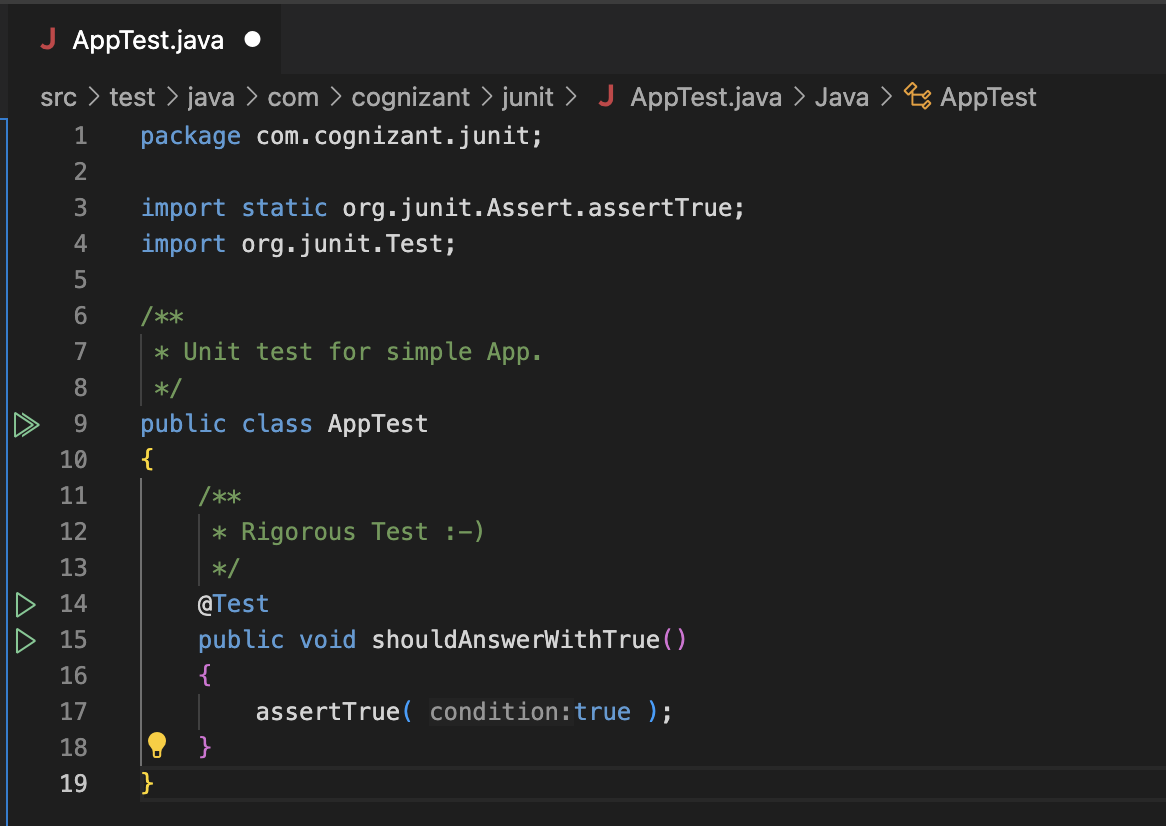
1. **Project Creation:** Created a project



1. **Adding JUnit Dependencies:** pom.xml file with added dependencies

****

1. **Create a new test class in your project:** Adding AppTest.java file



**Exercise 2: Writing Basic JUnit Tests**

Created a Java class **Calculator.java,** in which I implemented four basic arithmetic methods: **add**, **subtract**, **multiply**, and **divide**.

Then created a test class **CalculatorTest.java** inside **src/test/java/com/cognizant/junit/** to write unit tests for each method using JUnit.

**Calculator.java Code:**

package com.cognizant.junit;

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 ArithmeticException("Cannot divide by zero");

}

return a / b;

}

}

**CalculatorTest.java Code:**

package com.cognizant.junit;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class CalculatorTest {

Calculator calc = new Calculator();

@Test

public void testAdd() {

assertEquals(10, calc.add(6, 4));

}

@Test

public void testSubtract() {

assertEquals(2, calc.subtract(6, 4));

}

@Test

public void testMultiply() {

assertEquals(24, calc.multiply(6, 4));

}

@Test

public void testDivide() {

assertEquals(2, calc.divide(8, 4));

}

@Test(expected = ArithmeticException.class)

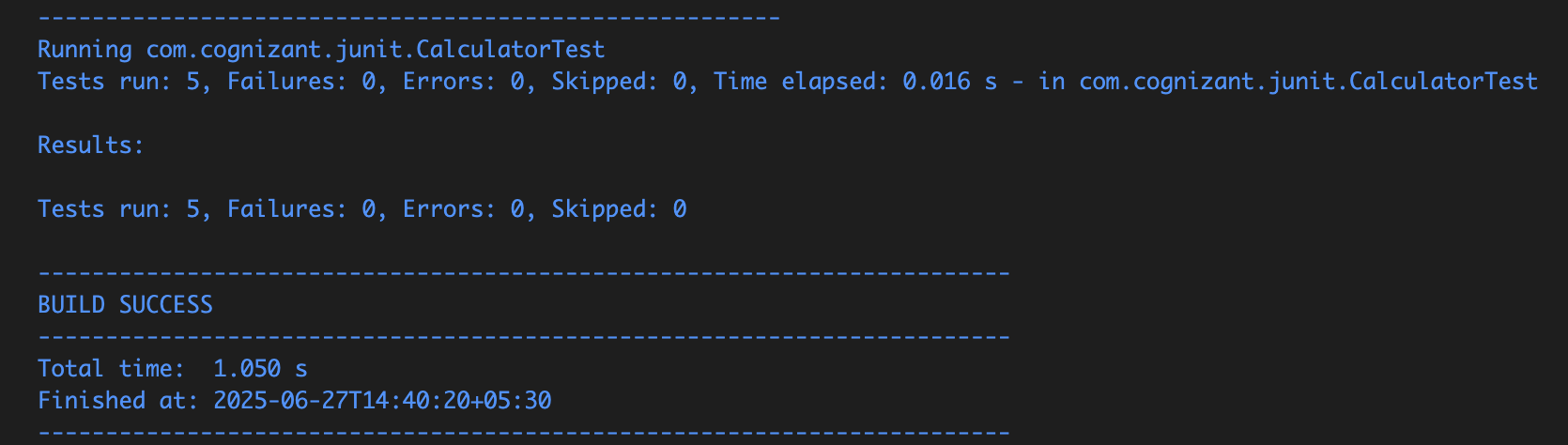
public void testDivideByZero() {

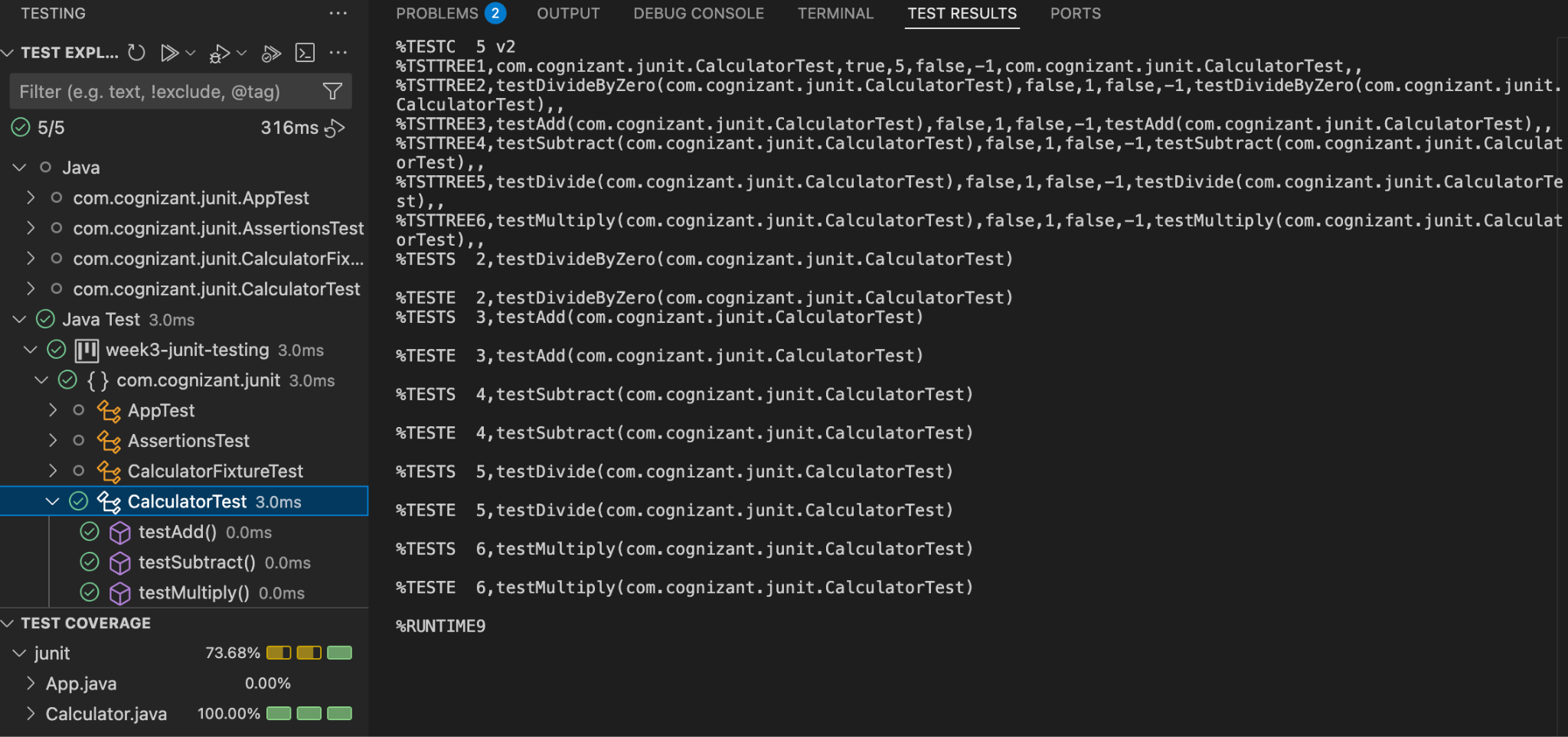
calc.divide(5, 0);

}

}

**Output:** Executing all the methods





**Exercise 3: Assertions in JUnit (MANDATORY)**

**AssertionsTest.java Code:** Write tests using various JUnit assertions.

package com.cognizant.junit;

import static org.junit.Assert.assertEquals;

import static org.junit.Assert.assertFalse;

import static org.junit.Assert.assertNotNull;

import static org.junit.Assert.assertNull;

import static org.junit.Assert.assertTrue;

import org.junit.Test;

public class AssertionsTest {

@Test

public void testAssertions() {

assertEquals("Sum should be 5", 5, 2 + 3);

assertTrue("5 should be greater than 3", 5 > 3);

assertFalse("3 should not be greater than 5", 3 > 5);

String nullStr = null;

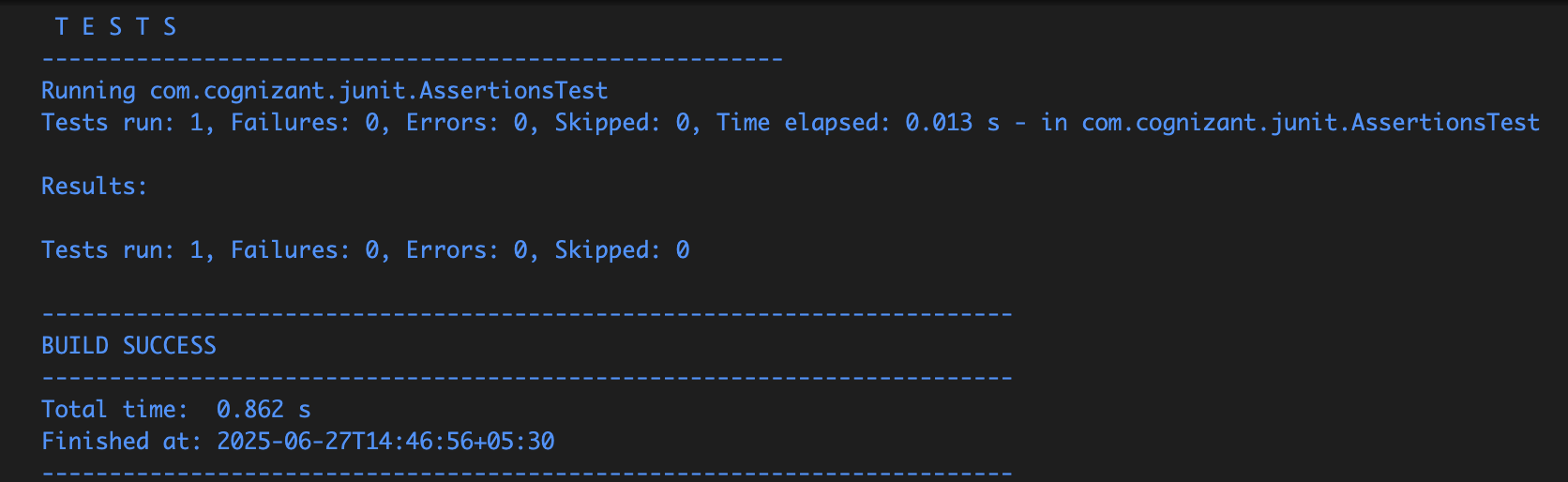
assertNull("Should be null", nullStr);

String notNullStr = "JUnit";

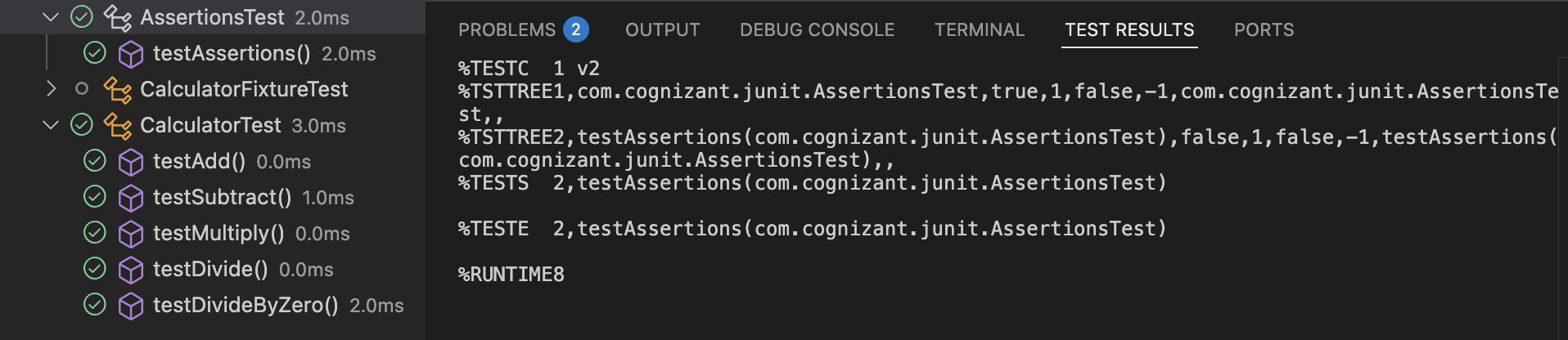
assertNotNull("Should not be null", notNullStr);

}

}

**Output:   
**

**Running all assertions:**

****

**Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and**

**Teardown Methods in JUnit (MANDATORY)**

Created a test class **CalculatorFixtureTest.java** to demonstrate the **Arrange-Act-Assert (AAA)** pattern and the use of **@Before** and **@After** methods in JUnit.

package com.cognizant.junit;

import org.junit.After;

import static org.junit.Assert.assertEquals;

import org.junit.Before;

import org.junit.Test;

public class CalculatorFixtureTest {

private Calculator calculator;

@Before

public void setUp() {

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

calculator = new Calculator();

}

@After

public void tearDown() {

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

calculator = null;

}

@Test

public void testAdditionAAA() {

int a = 10, b = 5;

int result = calculator.add(a, b);

assertEquals("Addition should be 15", 15, result);

}

@Test

public void testMultiplicationAAA() {

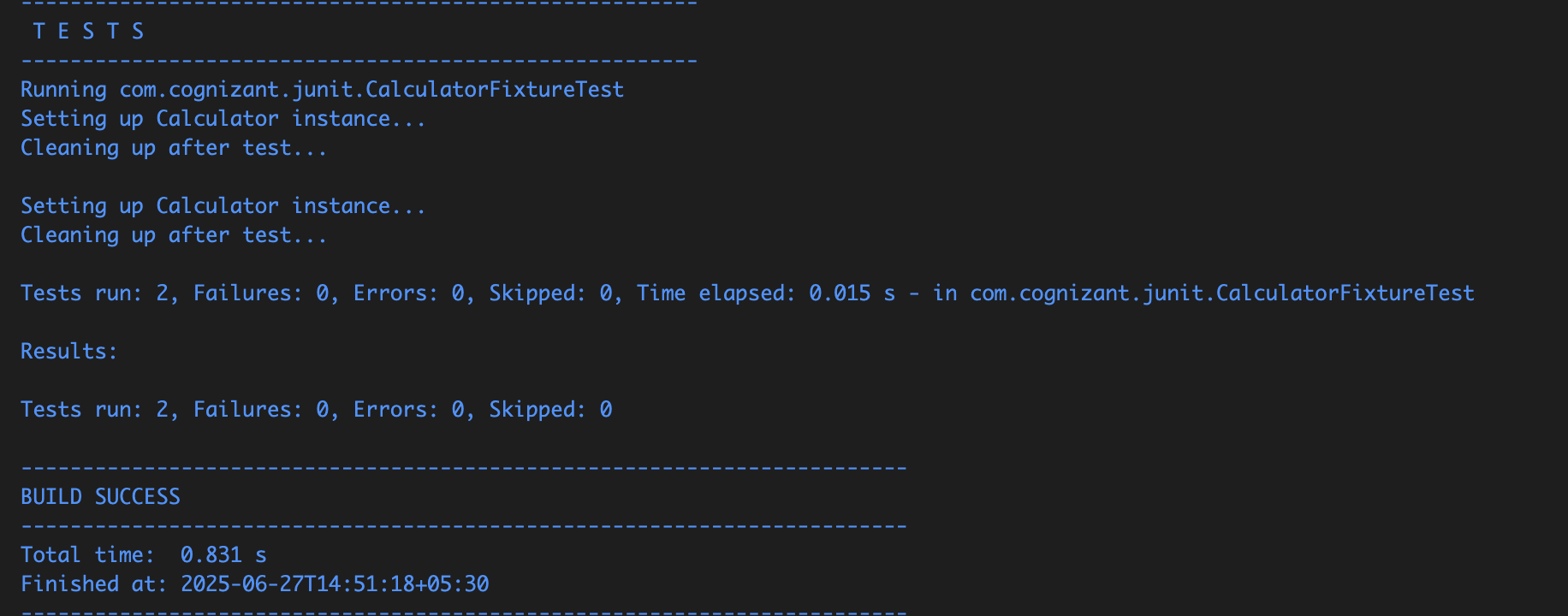
int a = 3, b = 4;

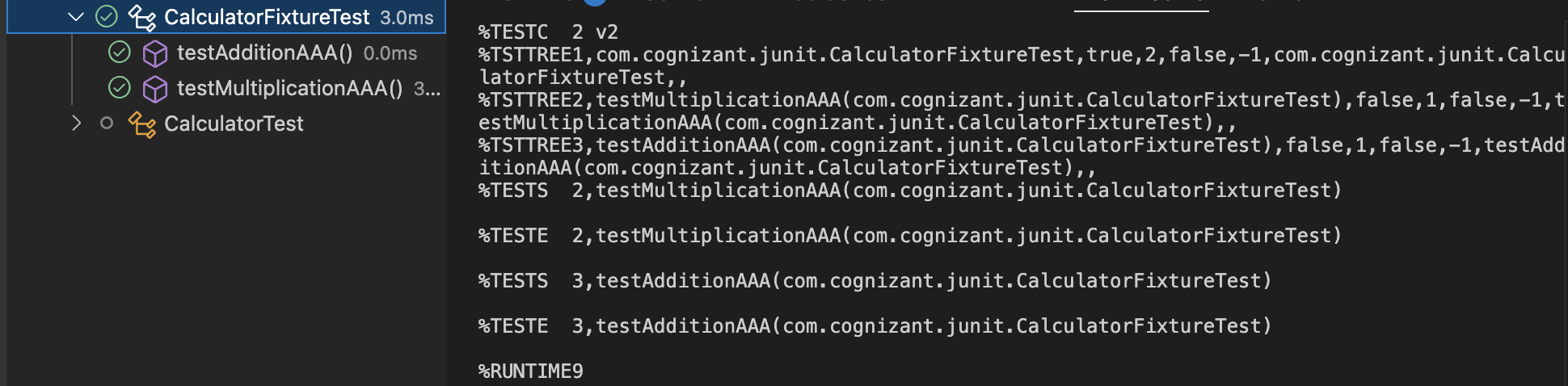
int result = calculator.multiply(a, b);

assertEquals("Multiplication should be 12", 12, result);

}

}

**Output:   
**

****