**Cognizant Digital Nurture 4.0-Week(2)**

**Name: Gandu Lasya Sri**

**Email: gandulasyasri@gmail.com**

**Superset ID:6428164**

**TDD using JUnit5 and Mockito**

**Exercise 1: Setting Up JUnit**

**Scenario:**

I was asked to set up JUnit in my Java project to begin writing basic unit tests. This was my first time working with JUnit, so I followed simple beginner steps using Maven in Visual Studio Code.

**Step 1: Create a Maven Java Project**

* I opened VS Code
* Installed extensions:
  + Java Extension Pack
  + Test Runner for Java
* Created a Maven project using terminal:

mvn archetype:generate -DgroupId=com.lasya -DartifactId=junitdemo -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

**Step 2: Add JUnit Dependency**

* I opened pom.xml
* Added the following code inside <dependencies>:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

**Step 3: Write a Simple JUnit Test**

* I opened this file:

src/test/java/com/lasya/AppTest.java

* I replaced the code with:

package com.lasya;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class AppTest {

@Test

public void testAddition() {

int a = 5;

int b = 7;

int result = a + b;

assertEquals(12, result); // checks if a + b = 12

}

}

**Step 4: Run the Test**

* I hovered over the method and clicked ▶️ Run Test
* I saw:

testAddition() passed

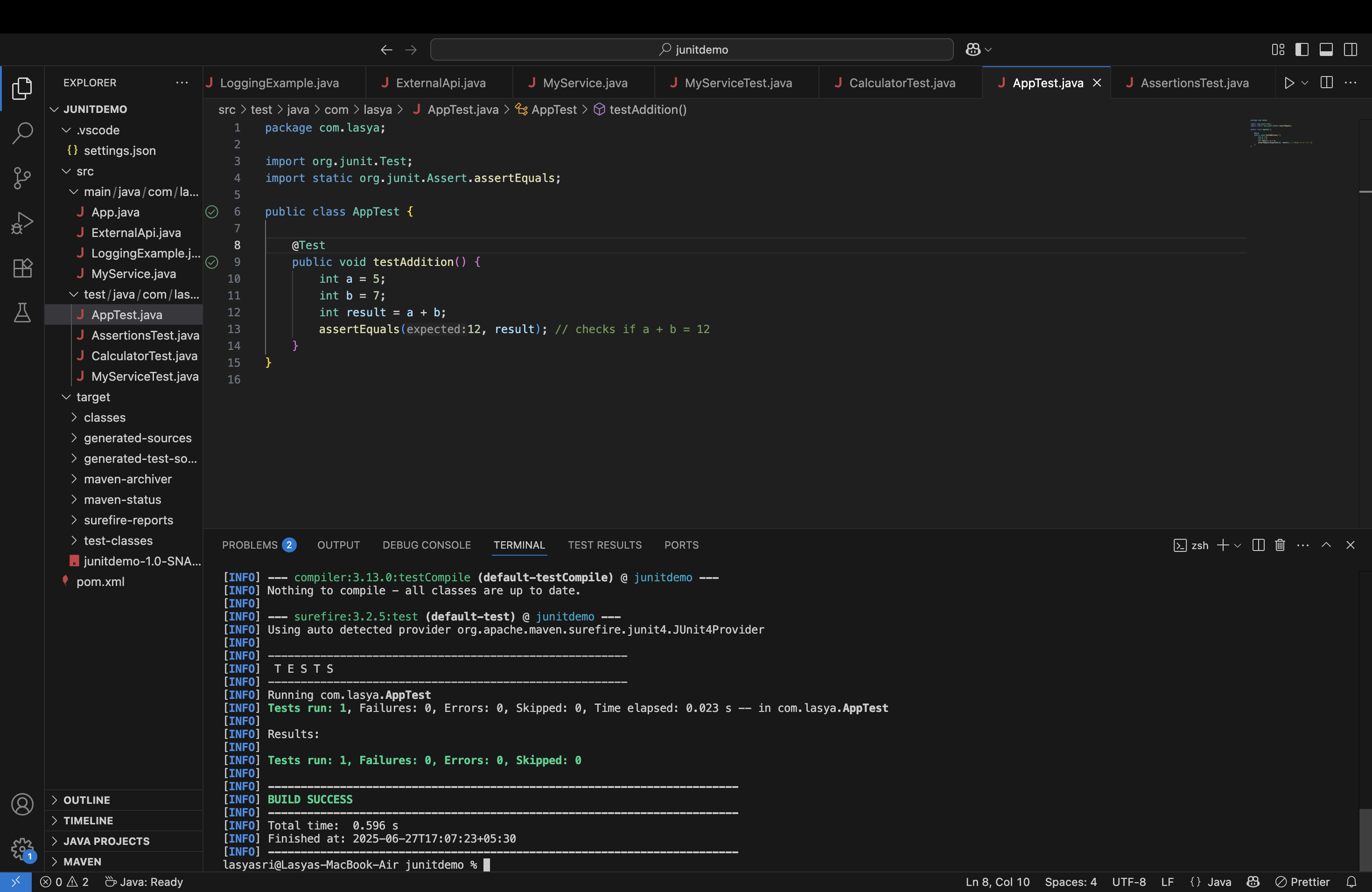
* I also ran this in terminal:

mvn test

And it showed:

Tests run: 1, Failures: 0, BUILD SUCCESS

OUTPUT:



**Exercise 3: Assertions in JUnit**

**Scenario:**

In this exercise, I was asked to write a test method using different types of assertions in JUnit. These assertions are used to check conditions like equality, true/false, and null/not-null. This helped me understand how to validate various types of logic using unit tests.

**1. Setup**

I used the same project structure and pom.xml from the previous exercises, which already had JUnit 4.13.2 included:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

**2. Code: AssertionsTest.java**

I created a file called AssertionsTest.java inside src/test/java/com/lasya/ and wrote this code:

package com.lasya;

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());

}

}

**How I Ran It:**

I used **two ways** to run the test:

**Option 1: Using Run Button in VS Code**

* I hovered over the @Test method and clicked ▶ Run Test
* The test passed and I saw this:

✔ testAssertions() passed

**Option 2: Using Maven in Terminal**

I ran this command in the terminal:

cd ~/junitdemo

mvn test

And the output was:

-------------------------------------------------------

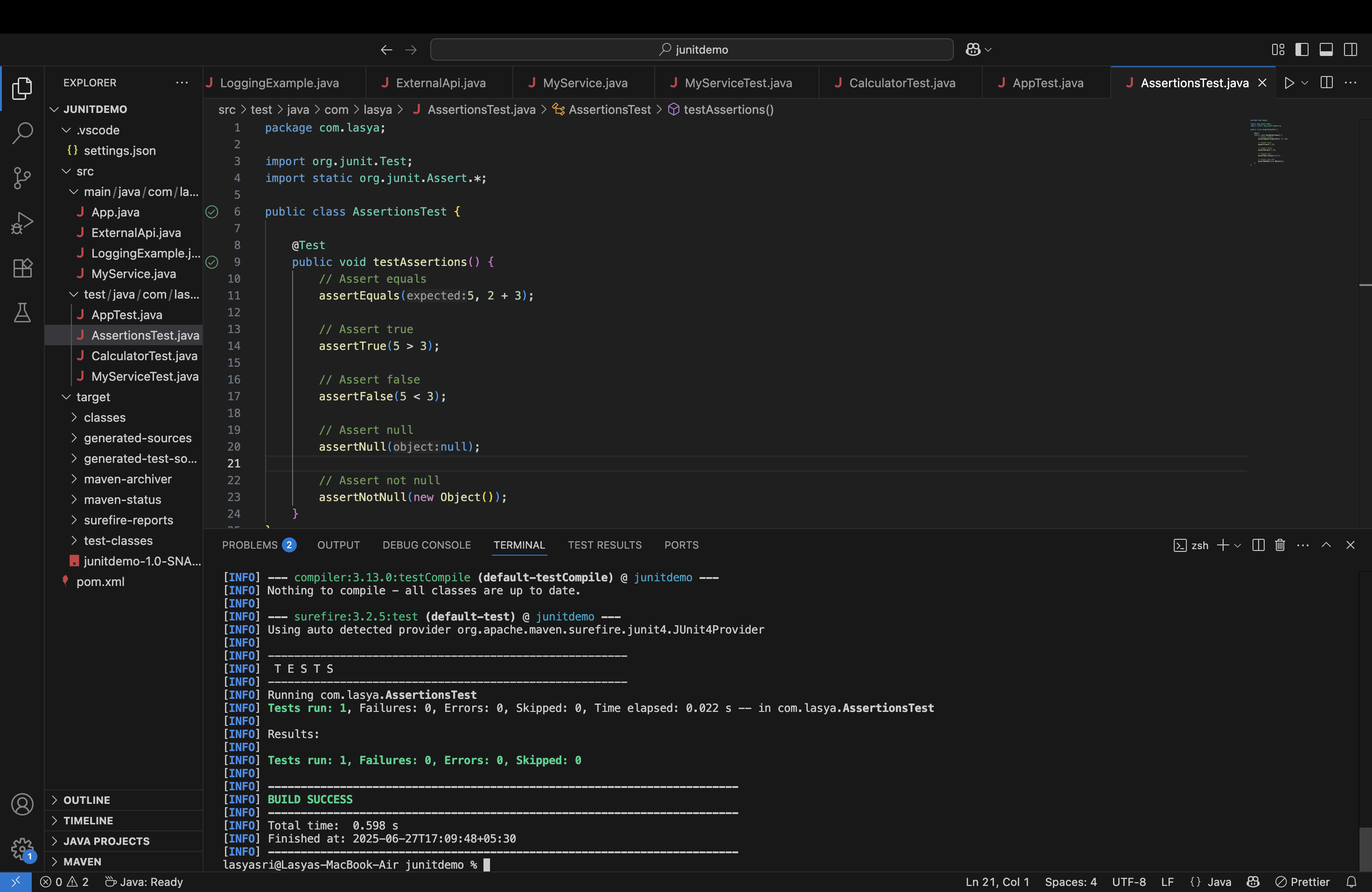
T E S T S

-------------------------------------------------------

Running com.lasya.AssertionsTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] BUILD SUCCESS

OUTPUT:

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

**Scenario:**

In this exercise, I needed to organize my unit tests using the **Arrange-Act-Assert (AAA) pattern** and implement setup and teardown methods using @Before and @After annotations. This helps maintain test structure and isolate each test execution.

**1. Setup**

I used the same Maven project setup from earlier exercises with JUnit 4.13.2 in pom.xml:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

**2. Code: CalculatorTest.java**

I created a new class called CalculatorTest.java in src/test/java/com/lasya/:

package com.lasya;

import org.junit.\*;

import static org.junit.Assert.\*;

public class CalculatorTest {

private int a;

private int b;

@Before

public void setUp() {

// Arrange

a = 10;

b = 5;

System.out.println("Setup complete");

}

@After

public void tearDown() {

System.out.println("Teardown complete");

}

@Test

public void testAddition() {

// Act

int result = a + b;

// Assert

assertEquals(15, result);

}

@Test

public void testSubtraction() {

// Act

int result = a - b;

// Assert

assertEquals(5, result);

}

}

* @Before runs before every test method.
* @After runs after every test method.
* Inside the test methods, I followed the **AAA pattern**:
  + **Arrange**: Setup in @Before
  + **Act**: Perform operation
  + **Assert**: Check result

▶️ **How I Ran It:**

I ran the tests using the Run ▶ button in VS Code and also using Maven:

cd ~/junitdemo

mvn test

And the output was:

-------------------------------------------------------

T E S T S

-------------------------------------------------------

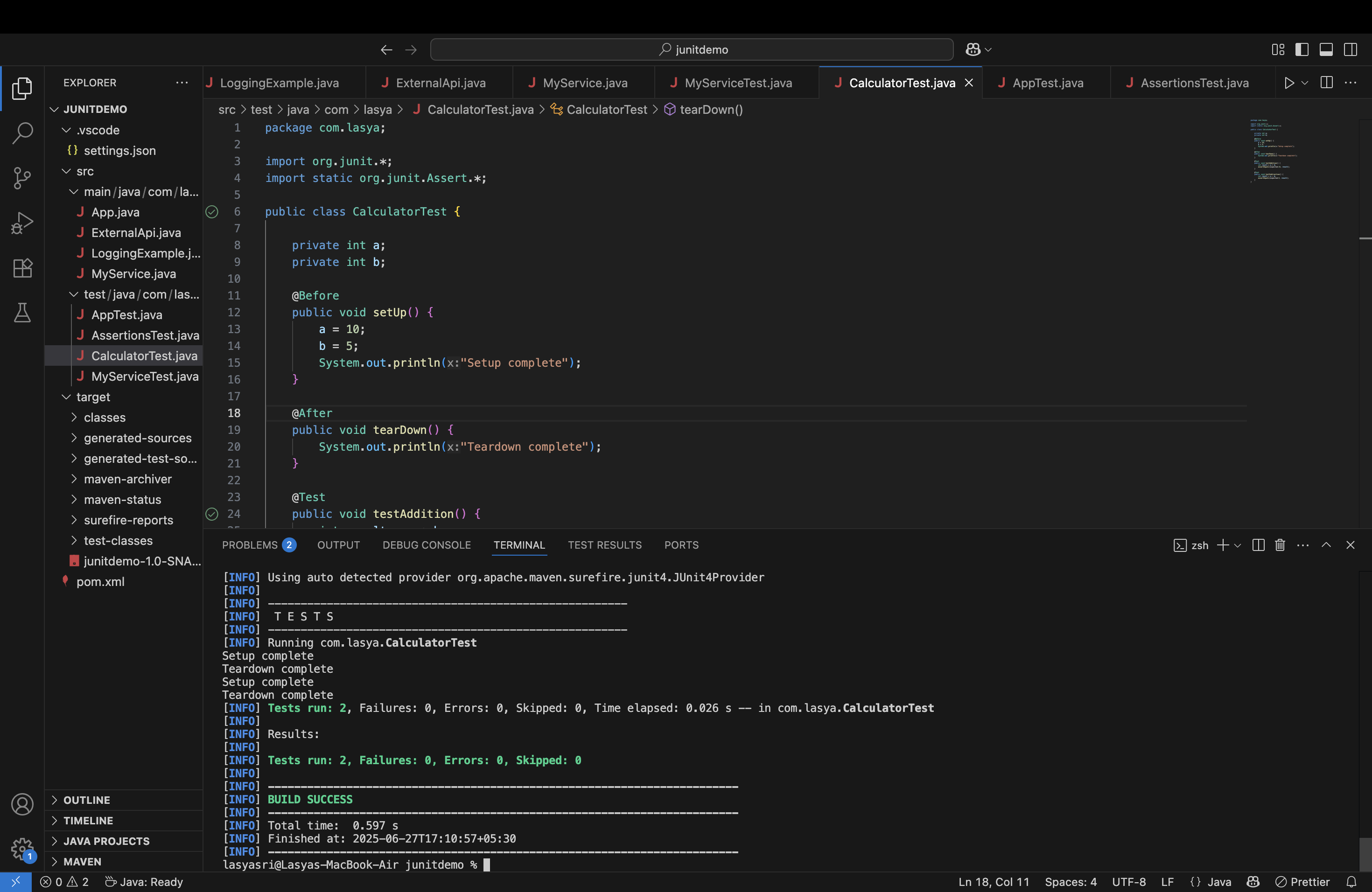
Running com.lasya.CalculatorTest

Setup complete

Teardown complete

Setup complete

Teardown complete

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0

[INFO] BUILD SUCCESS

OUTPUT: