**WEEK -2**

**TDD using JUnit5 and Mockito**

**1) Setting Up Junit**

**\* pom.xml :**

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

**\*App.java:**

package com.example;

public class App {

    public int add(int a, int b){

        return a + b;

    }

}

**\*AppTest.java:**package com.example;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class AppTest {

    @Test

    public void testAdd() {

        App app = new App();

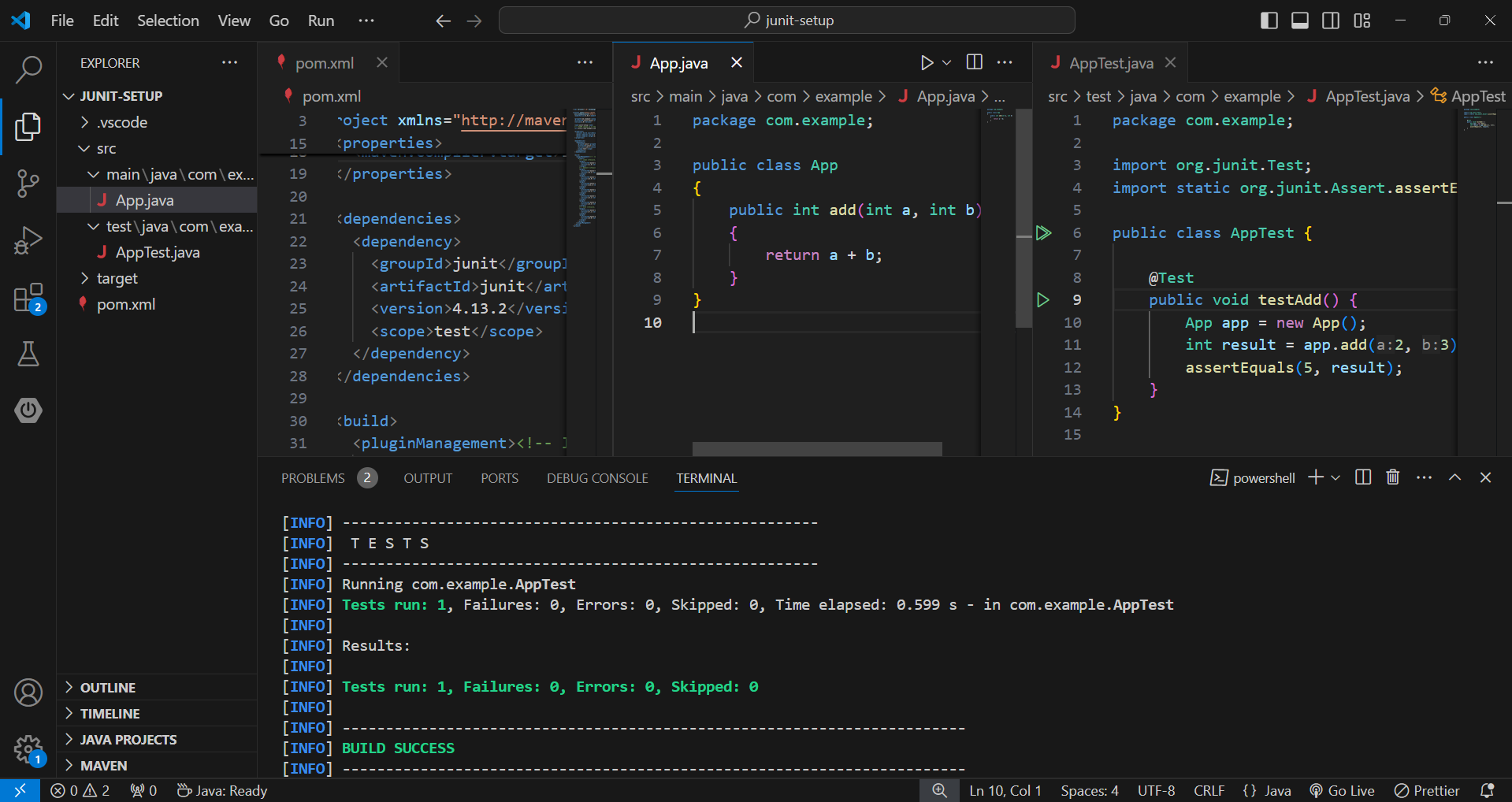
        int result = app.add(2, 3);

        assertEquals(5, result);

    }

}

**Output:**



**2)** **Assertions in Junit**

**\* AssertionsTest.java :**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

    @Test

    public void testAssertions() {

        assertEquals(5, 2 + 3);

        assertTrue(5 > 3);

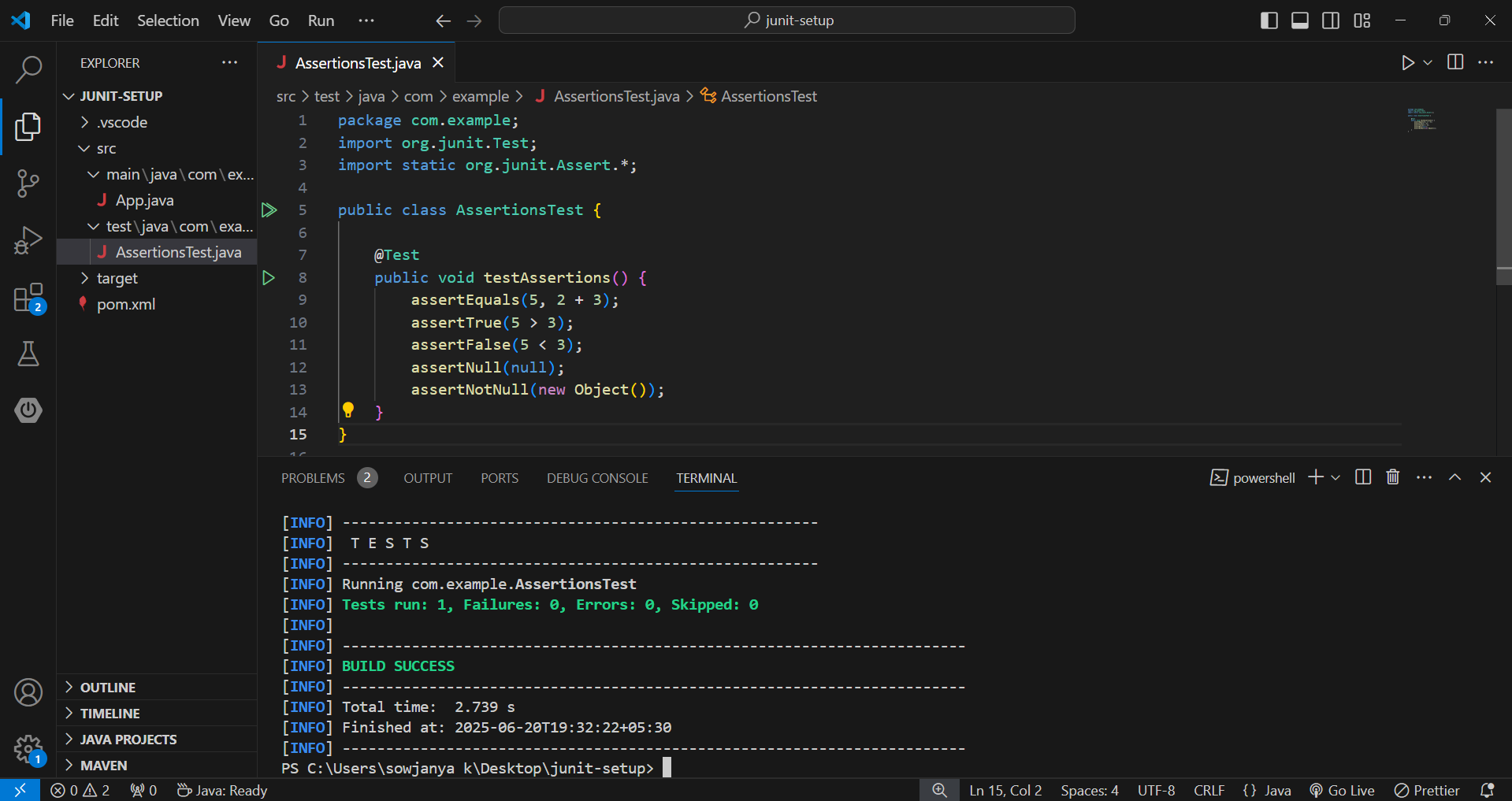
        assertFalse(5 < 3);

        assertNull(null);

        assertNotNull(new Object());

    }

}

**Output:****3) Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit**

**\* Calculator.java :**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**\* CalculatorTest.java :**

package com.example;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calculator;

@Before

public void setUp() {

System.out.println("Setup: Creating Calculator instance");

calculator = new Calculator();

}

@After

public void tearDown() {

System.out.println("Teardown: Cleaning up");

calculator = null;

}

@Test

public void testAddition() {

int result = calculator.add(4, 5);

assertEquals(9, result);

}

@Test

public void testAdditionNegative() {

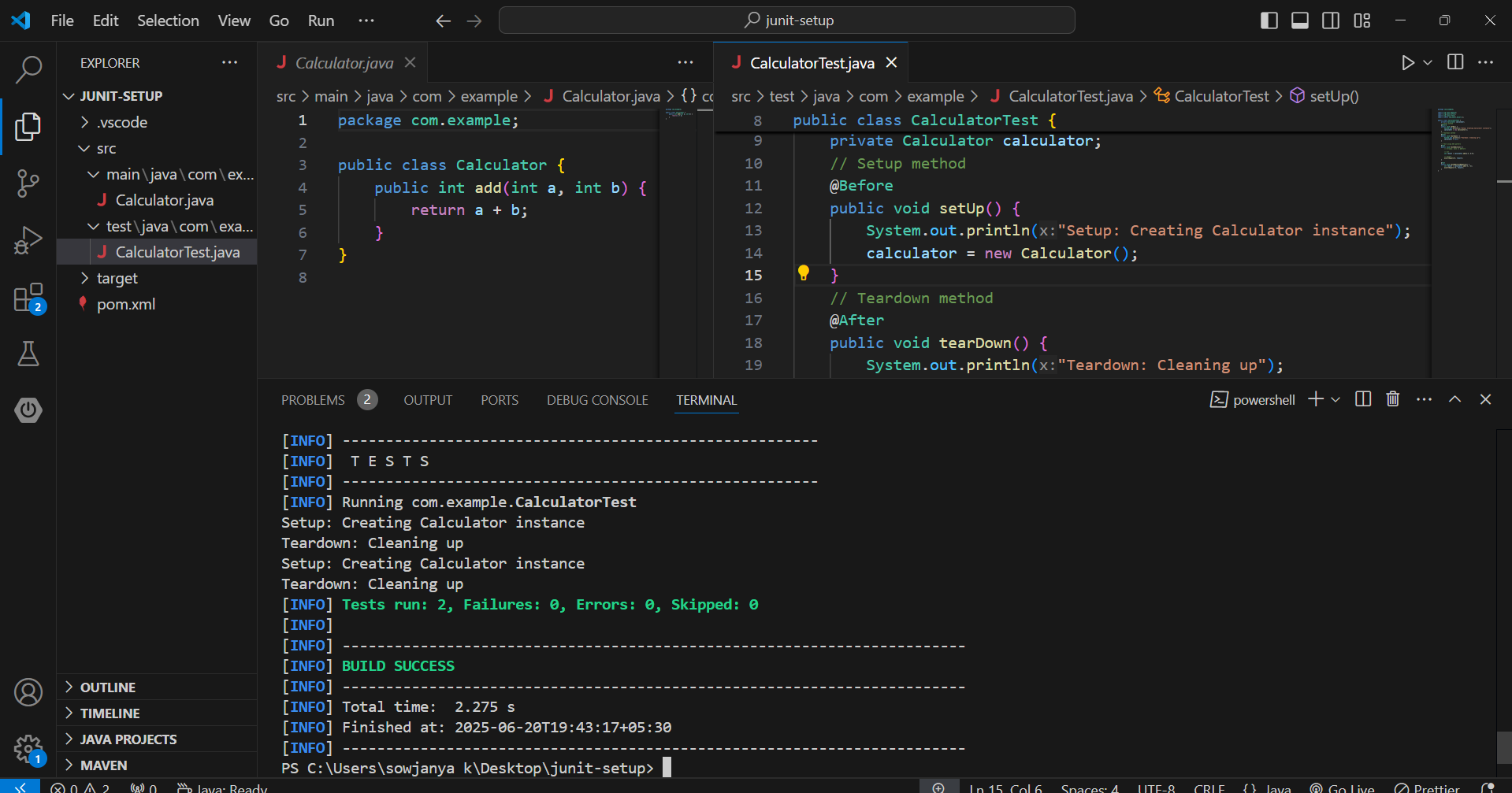
int result = calculator.add(-3, -2);

assertEquals(-5, result);

}

}

**Output :**



**4) Mocking and Stubbing**

**\* pom.xml :**

<dependencies>

<!-- JUnit 5 -->

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

</dependencies>

**\* ExternalApi.java :**

package com.example;

public interface ExternalApi {

String getData();

}

**\* MyService.java :**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**\* MyServiceTest.java :**

package com.example;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito. \*;

public class MyServiceTest {

@Test

public void testExternalApi() {

// Step 1: Mock the external API

ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Stub its method

when(mockApi.getData()).thenReturn("Mock Data");

// Step 3: Inject into service and test

MyService service = new MyService(mockApi);

String result = service.fetchData();

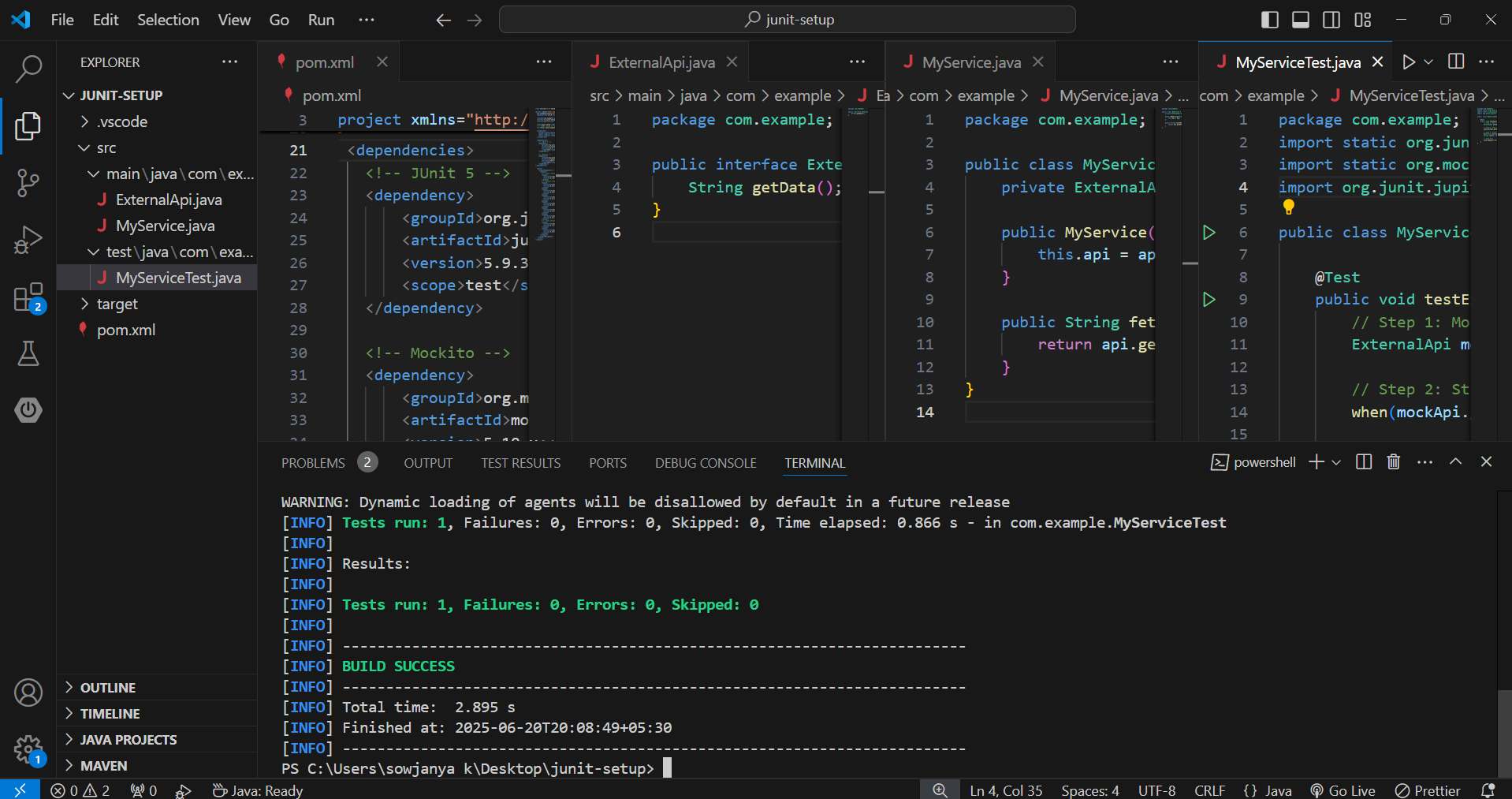
// Assertion

assertEquals("Mock Data", result);

}

}

**Output:**

****

**5) Verifying Interactions**

**\* ExternalApi.java :**

package com.example;

public interface ExternalApi {

String getData();

}

**\* MyService.java :**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**\* MyServiceTest.java :**

package com.example;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

MyService service = new MyService(mockApi);

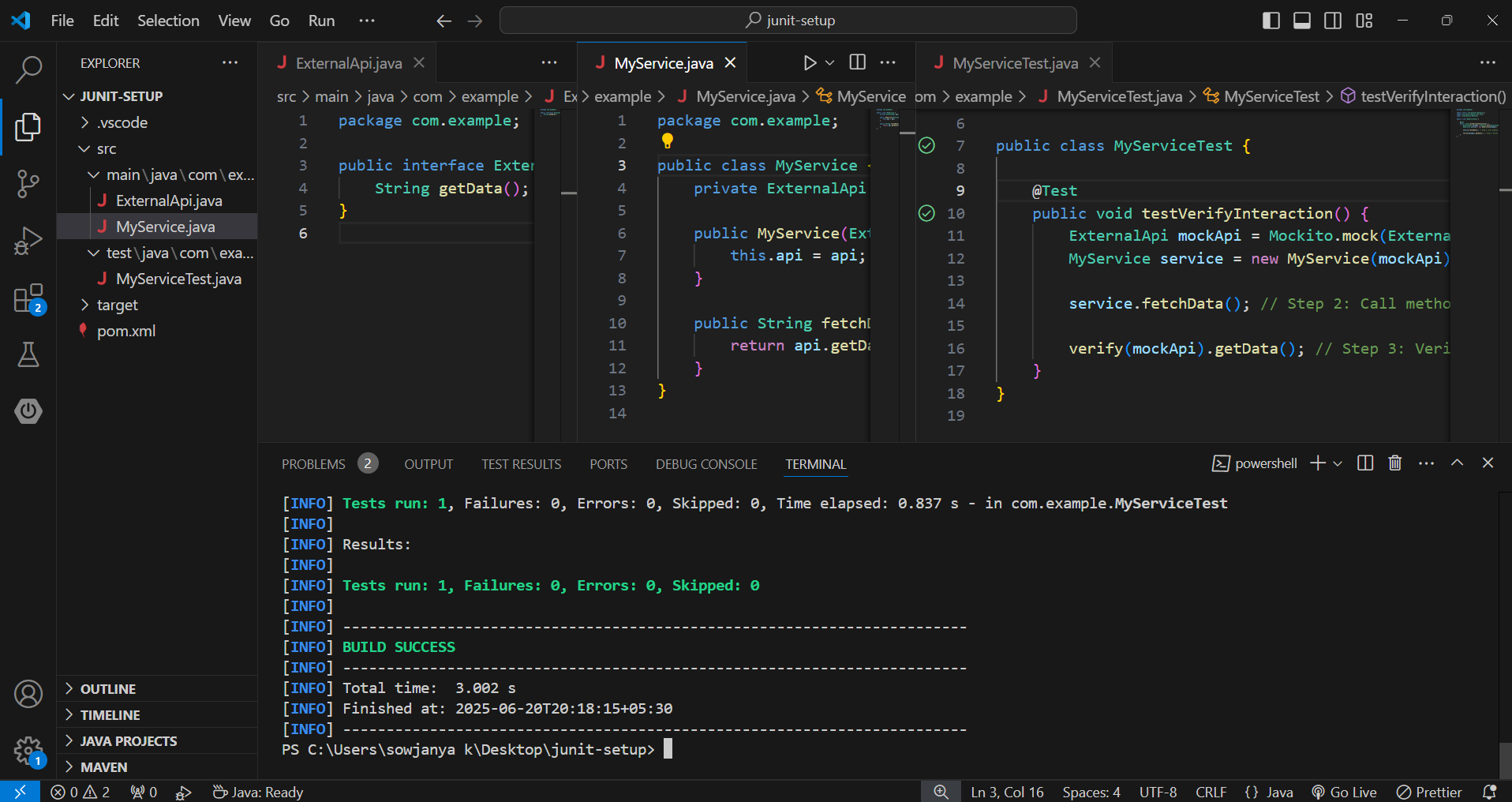
service.fetchData();

verify(mockApi).getData();

}

}

**Output :**

****