**WEEK-02 HANDS ON SOLUTIONS**

**Mockito Hands-On Exercises**

**Exercise 1: Mocking and Stubbing**

Scenario: You need to test a service that depends on an external API.

Use Mockito to mock the external API and stub its methods.

Steps: 1. Create a mock object for the external API.

2. Stub the methods to return predefined values.

3. Write a test case that uses the mock object.

Solution

Code:

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest

{

@Test

public void testExternalApi()

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

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

MyService service = new MyService(mockApi);

String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**EXPLANATION:**

In this exercise, I implemented unit testing using the Mockito library to mock and stub the behavior of an external dependency.

The goal was to test a class named MyService, which relies on another interface ExternalApi to fetch data.

Since ExternalApi is a dependency that could represent a network-based or external service, I used Mockito to simulate its behavior without requiring a real implementation.

To set this up, I first added the necessary dependencies in my Maven pom.xml file.

I included both JUnit 5 (via the junit-jupiter artifact) for unit testing and Mockito Core (version 4.8.1) for mocking.

These dependencies were scoped to test so that they are only used during testing and not included in production builds.

After updating the project, these libraries were automatically downloaded and made available in the test environment.

For the implementation, I created an interface called ExternalApi with a single method getData().

Then, I created a class MyService that takes an ExternalApi object as a constructor parameter and calls its getData() method from within a method named fetchData().

To test this setup, I wrote a JUnit test class MyServiceTest.

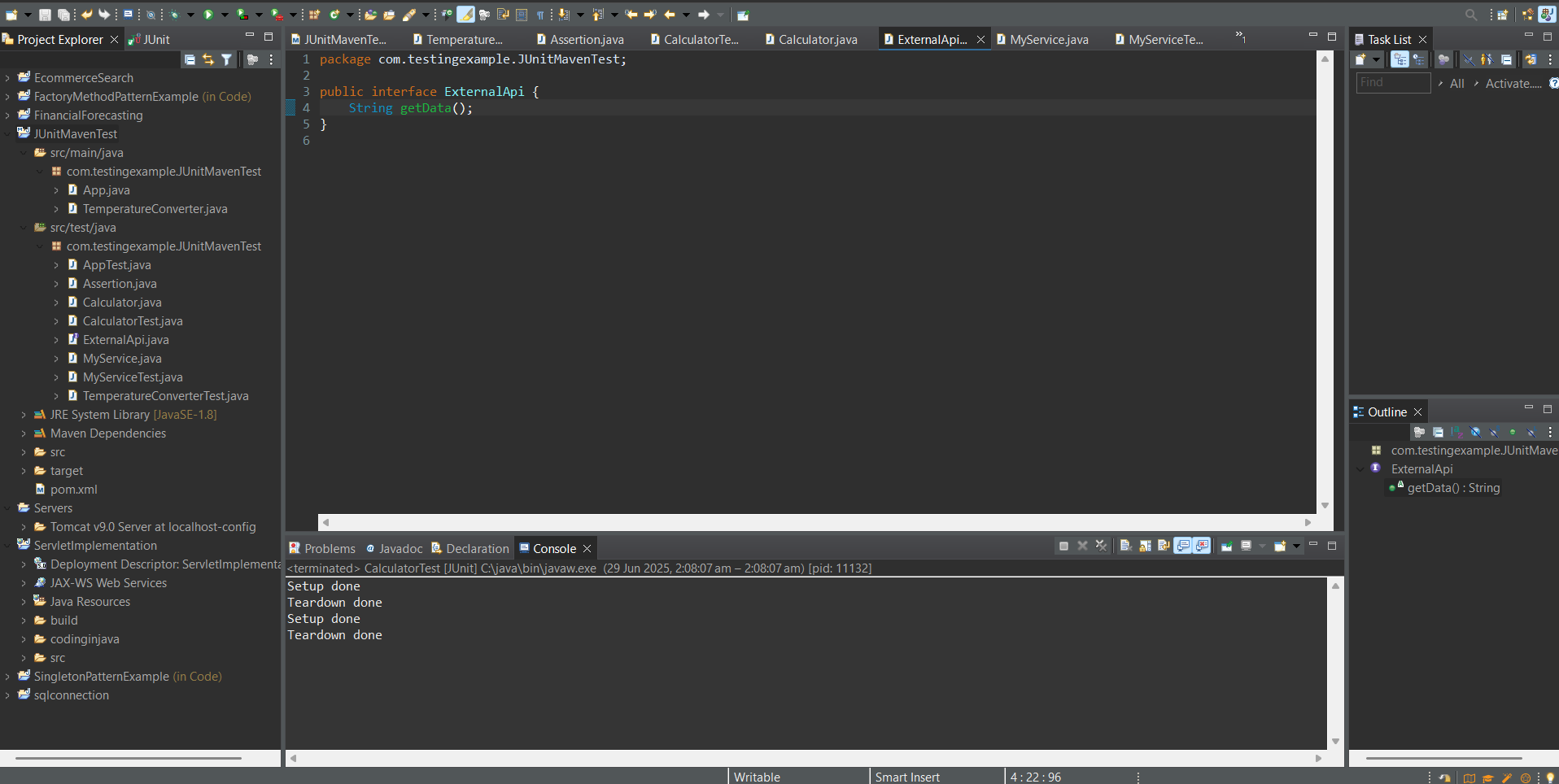
Inside the test method, I used Mockito.mock(ExternalApi.class) to create a mock version of the API.

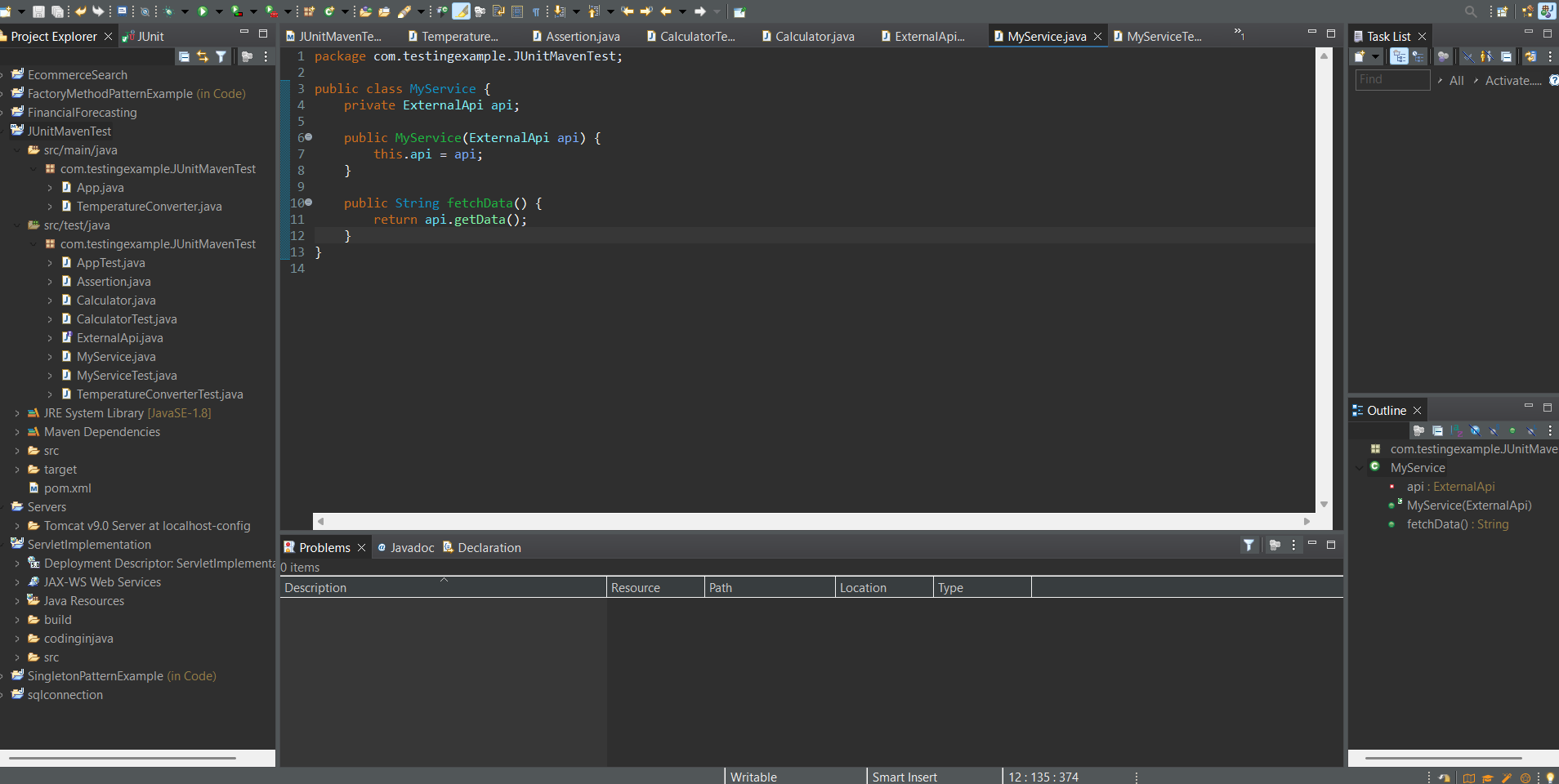
I then stubbed the getData() method using when(mockApi.getData()).thenReturn("Mock Data") so that it would return a fixed value whenever called.

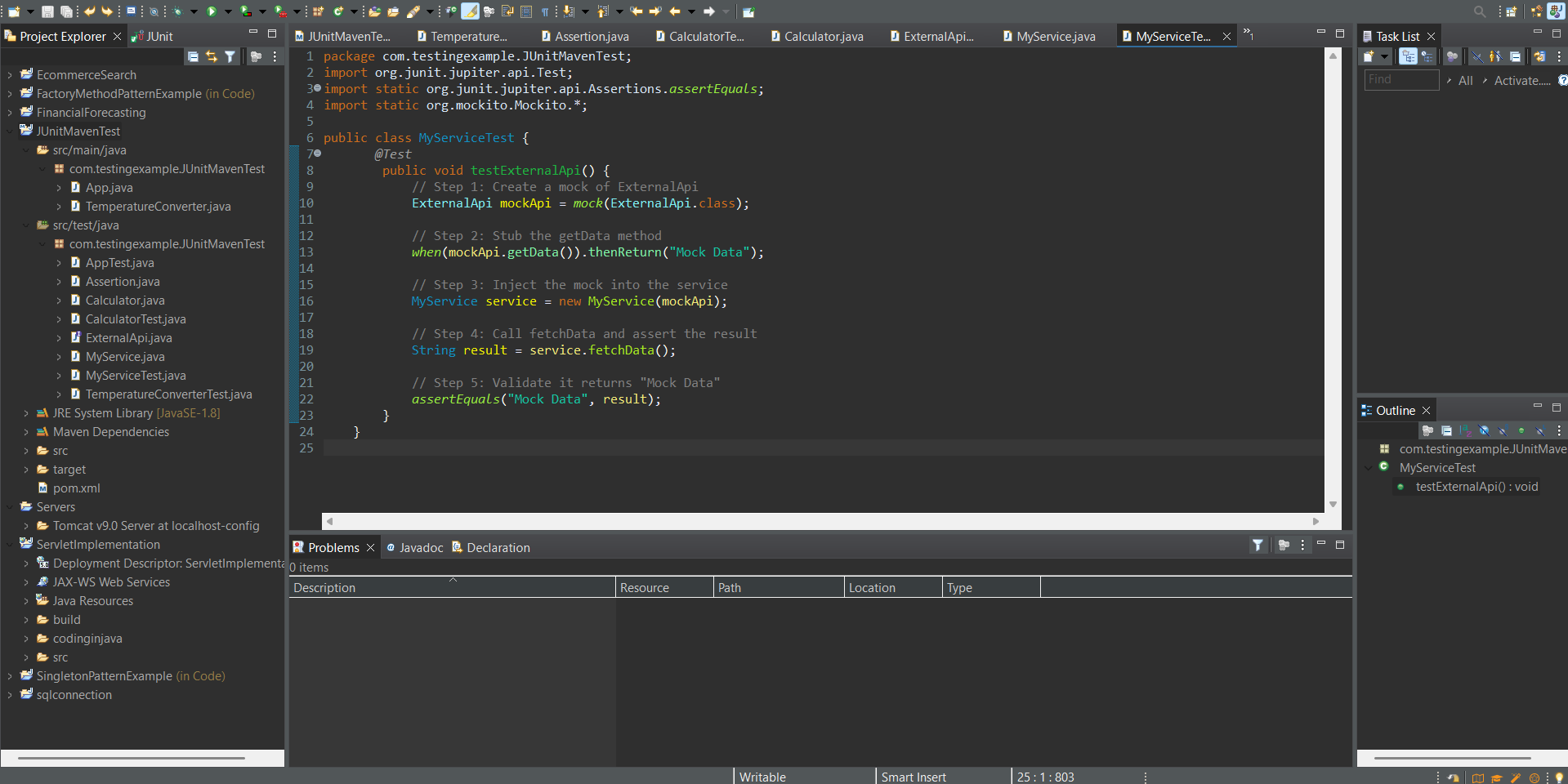
Next, I passed this mock API into the constructor of MyService and invoked the fetchData() method.

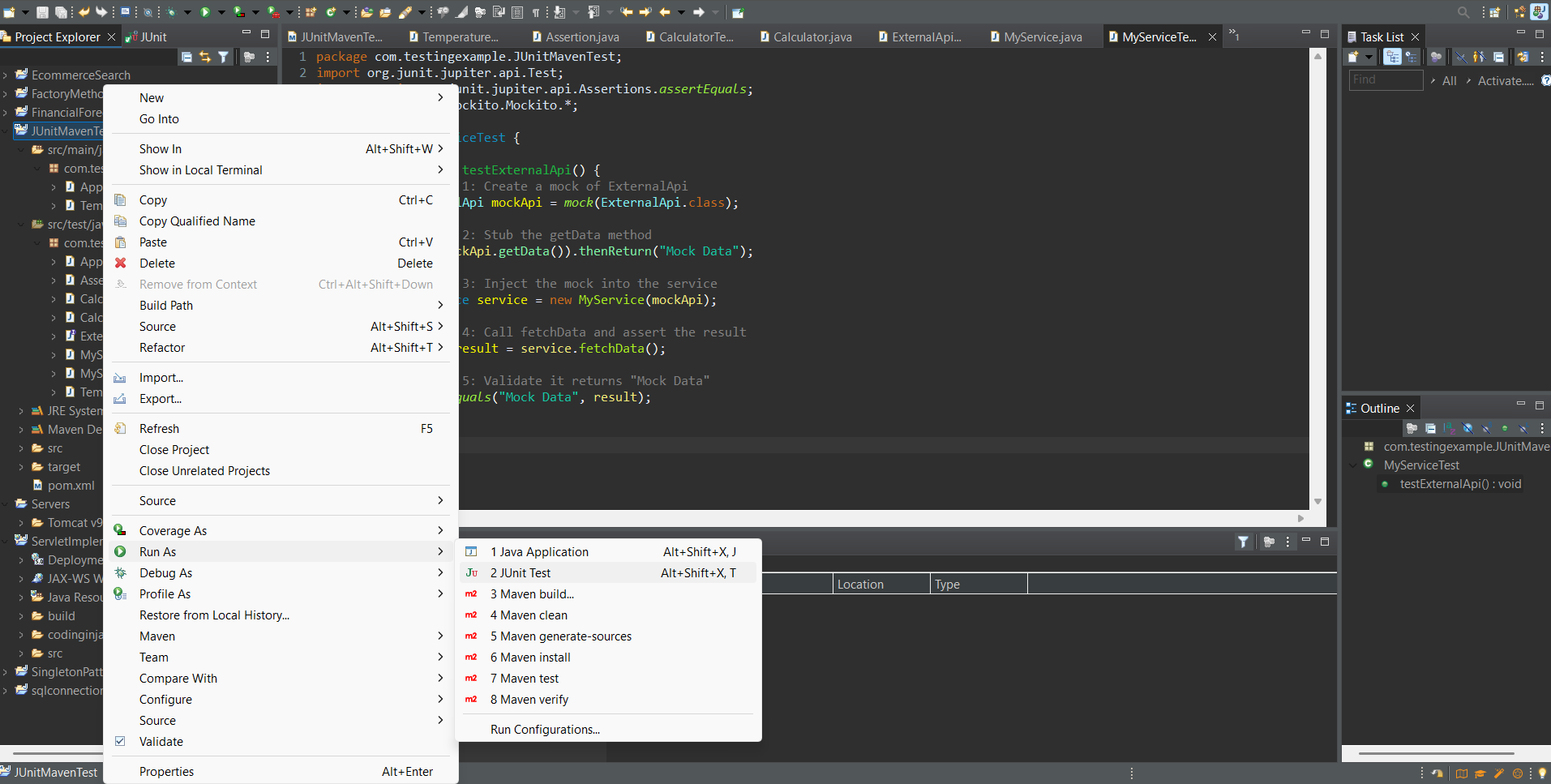
Using assertEquals(), I verified that the method returned the expected stubbed value.

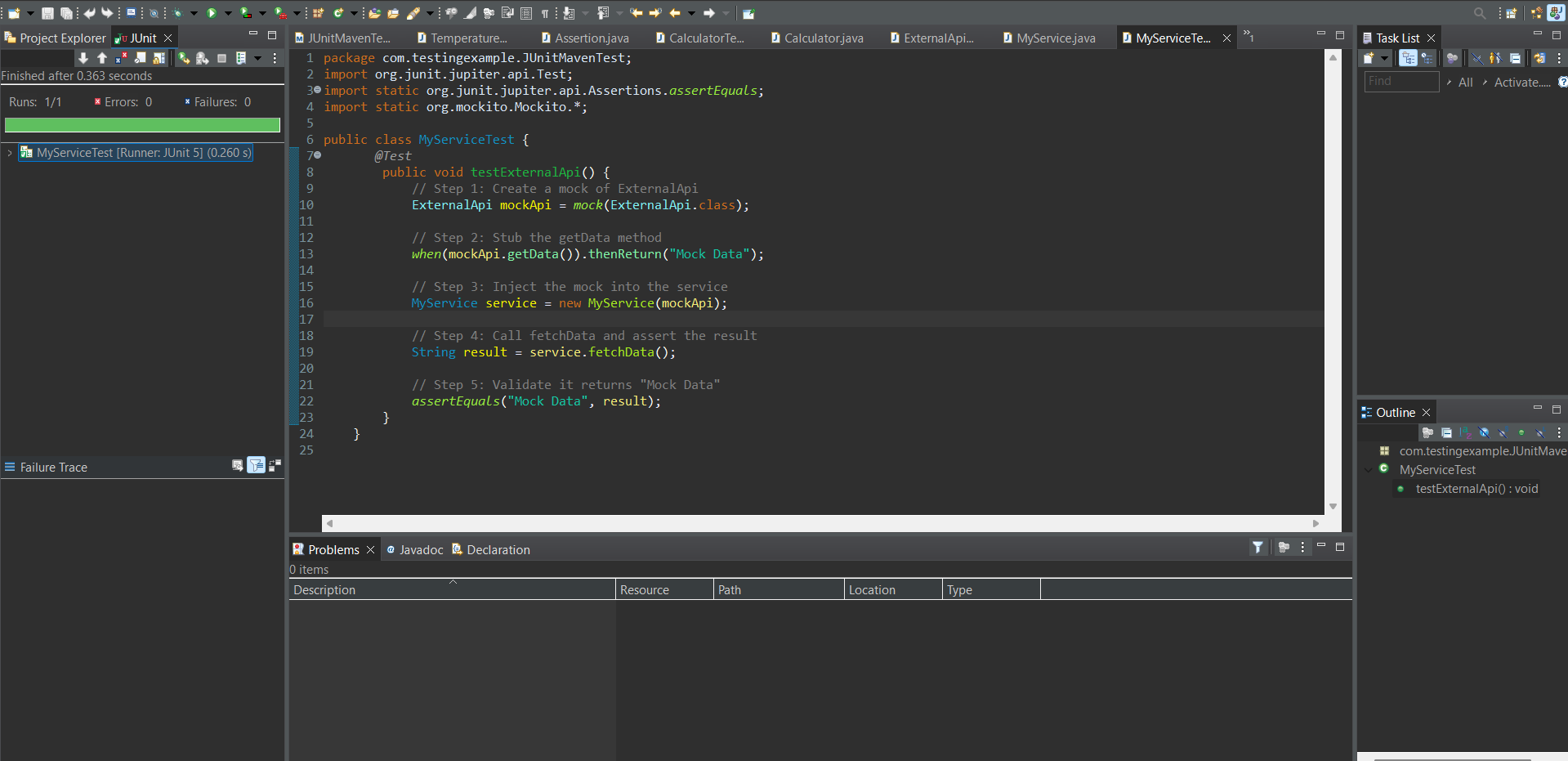
This confirmed that the service behaved correctly when interacting with the mocked dependency. The test successfully passed, and I received a green bar (1/1 tests), indicating that the mocking and stubbing process was correctly implemented using Mockito in conjunction with JUnit 5.

****

****

****

****

****