# **Exercise 2: Verifying Interactions**

## **Scenario**

You need to ensure that a method on a mock object is called with the expected arguments. This verifies that the service interacts correctly with its dependency.

### **Create a Mock Object**

I used Mockito to create a mock of the ExternalApi interface.

### **Call the Method**

I called the service method which uses the mock object internally.

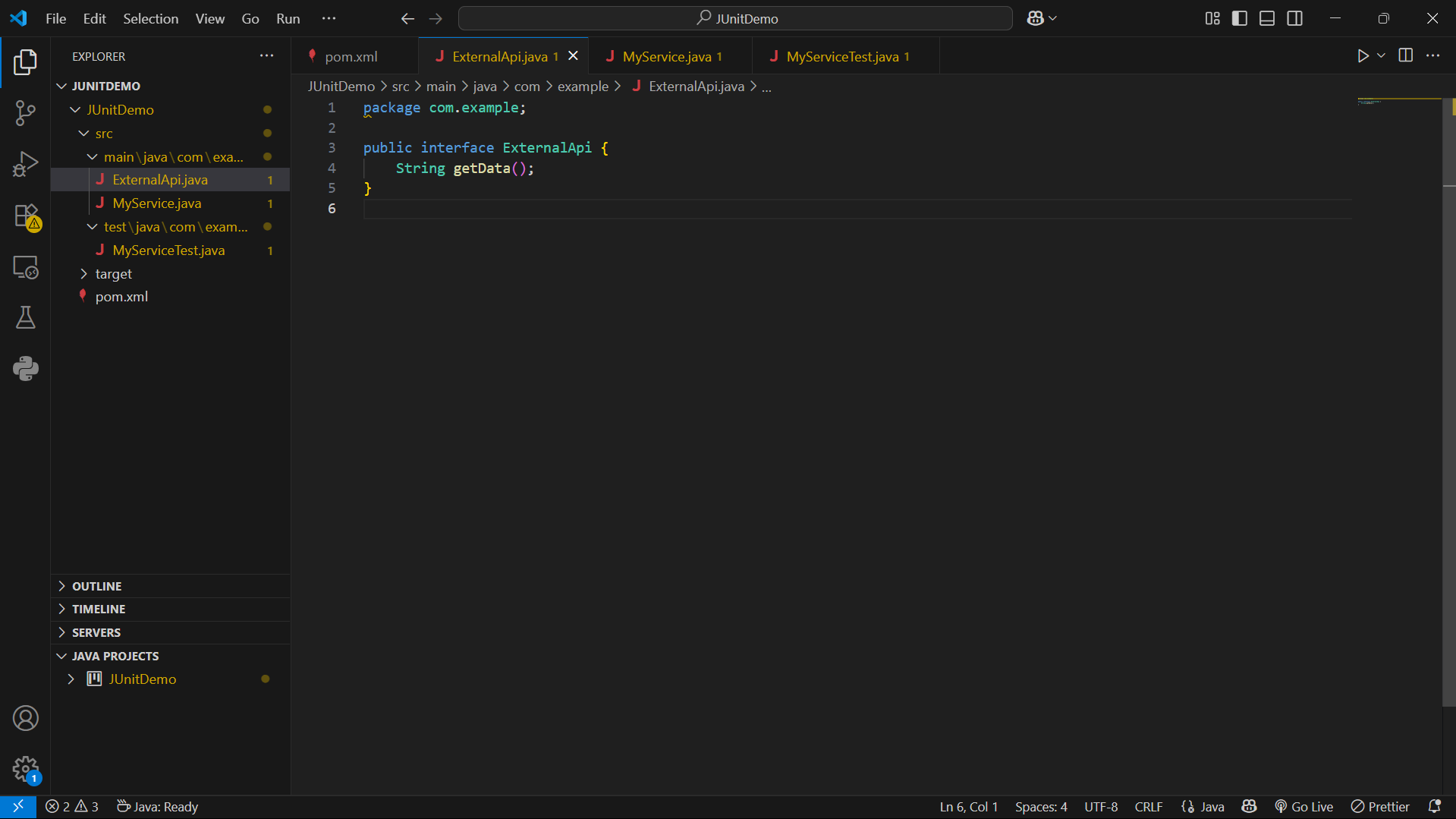
### **Verify the Interaction**

I used verify() to check that the mocked method was called.

## 

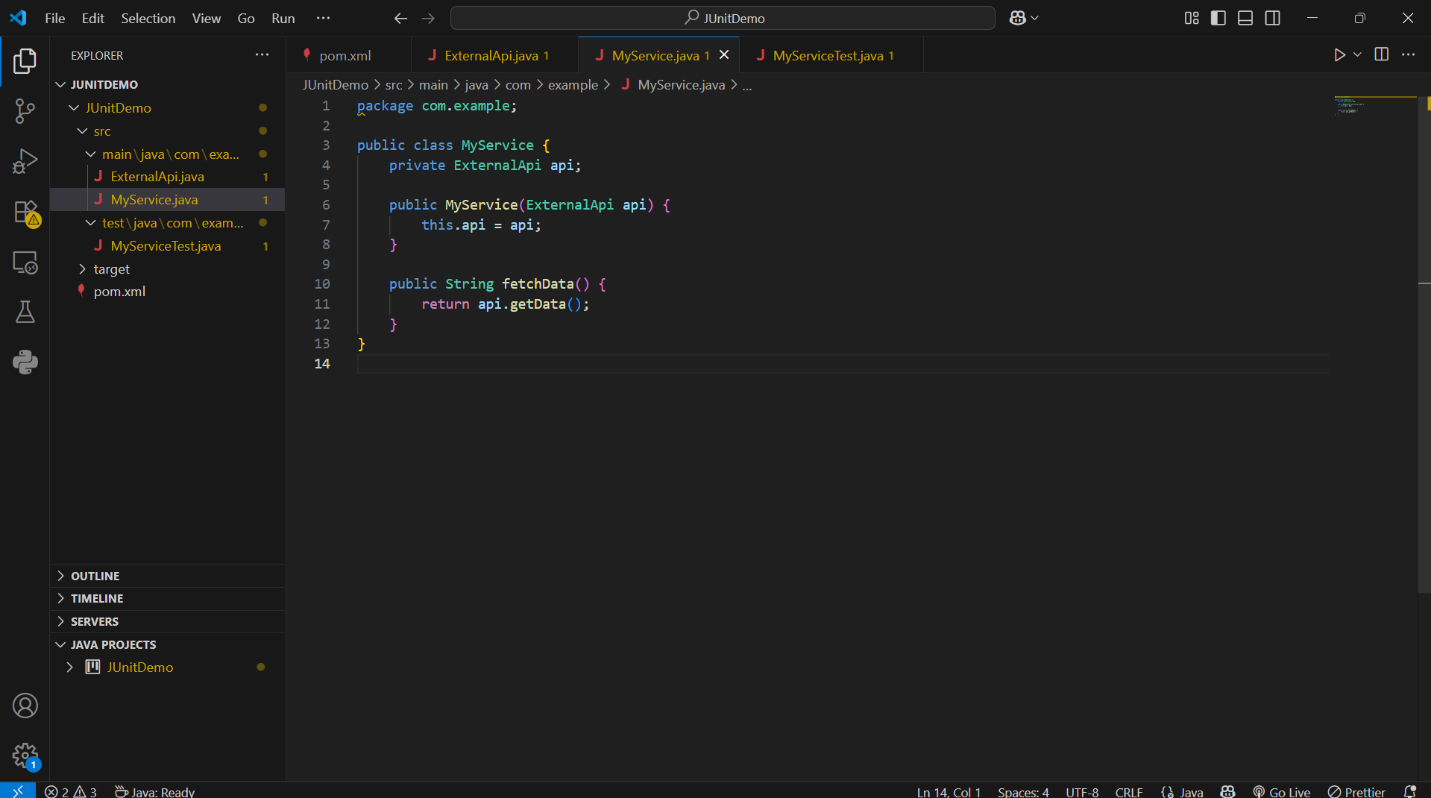
## **ExternalApi.java**

package com.example;  
  
public interface ExternalApi {  
 String getData();  
}



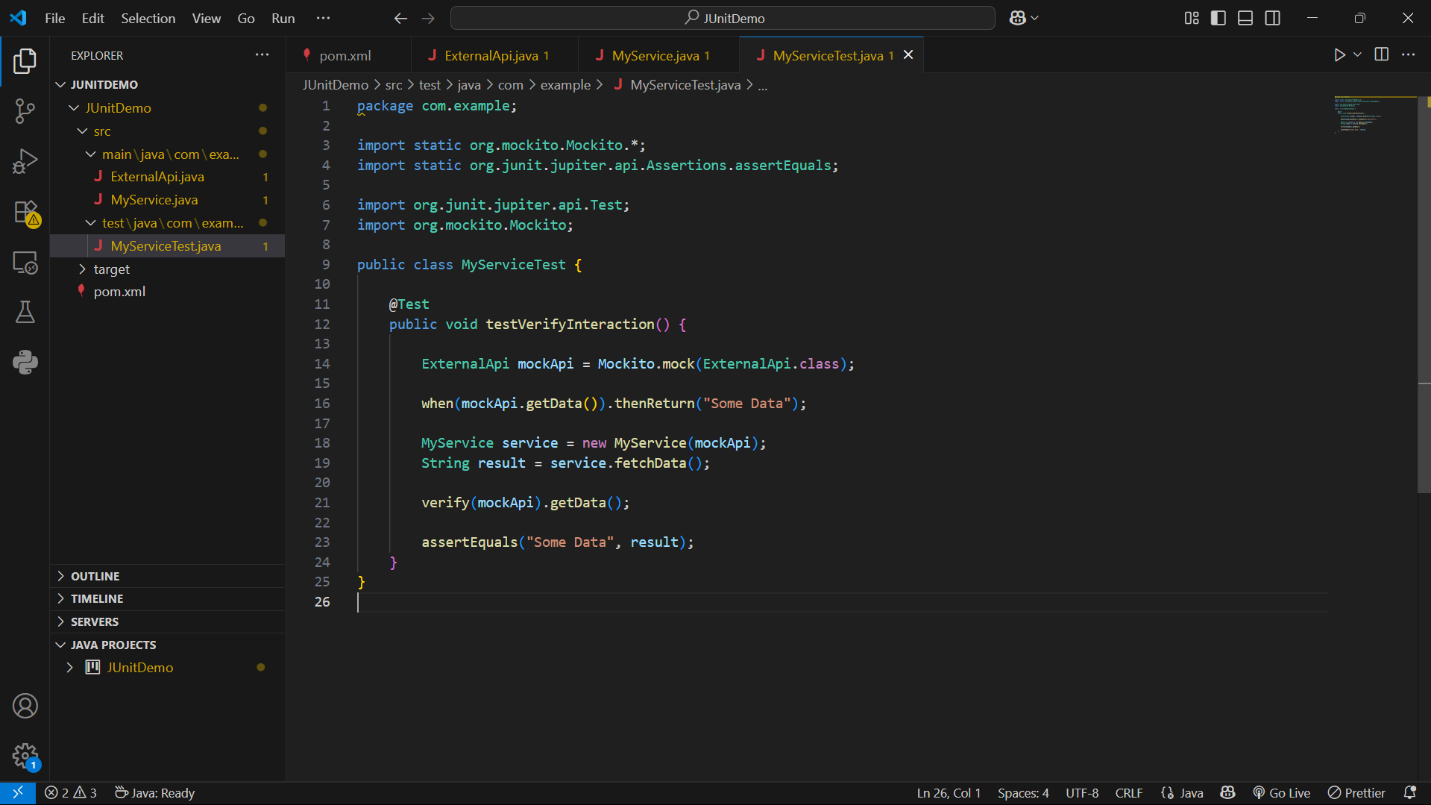
## **MyService.java**

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 static org.junit.jupiter.api.Assertions.assertEquals;  
  
import org.junit.jupiter.api.Test;  
import org.mockito.Mockito;  
  
public class MyServiceTest {  
  
 @Test  
 public void testVerifyInteraction() {  
   
 ExternalApi mockApi = Mockito.mock(ExternalApi.class);  
  
   
 when(mockApi.getData()).thenReturn("Some Data");  
  
   
 MyService service = new MyService(mockApi);  
 String result = service.fetchData();  
  
   
 verify(mockApi).getData();  
  
   
 assertEquals("Some Data", result);  
 }  
}



## **How to Run**

In the VS Code terminal, run:

mvn test

## 

## **Sample Output**

-------------------------------------------------------  
 T E S T S  
-------------------------------------------------------  
Running com.example.MyServiceTest  
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  
  
Results :  
  
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  
  
[INFO] BUILD SUCCESS

