TDD using JUnit5 and Mockito

**Candidate Name:** Rohit Tudu  
**Superset ID:** 6393839

1. **JUNIT Hands-On Exercises**

**Exercise 1: Setting Up JUnit**

**Scenario:**

You need to set up JUnit in your Java project to start writing unit tests.

**Steps:**

1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).

2. Add JUnit dependency to your project. If you are using Maven, add the following to your

pom.xml:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

1. Create a new test class in your project.

**Solution Checklist**

1. Java project created in IDE (IntelliJ/VSCode)
2. pom.xml includes:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

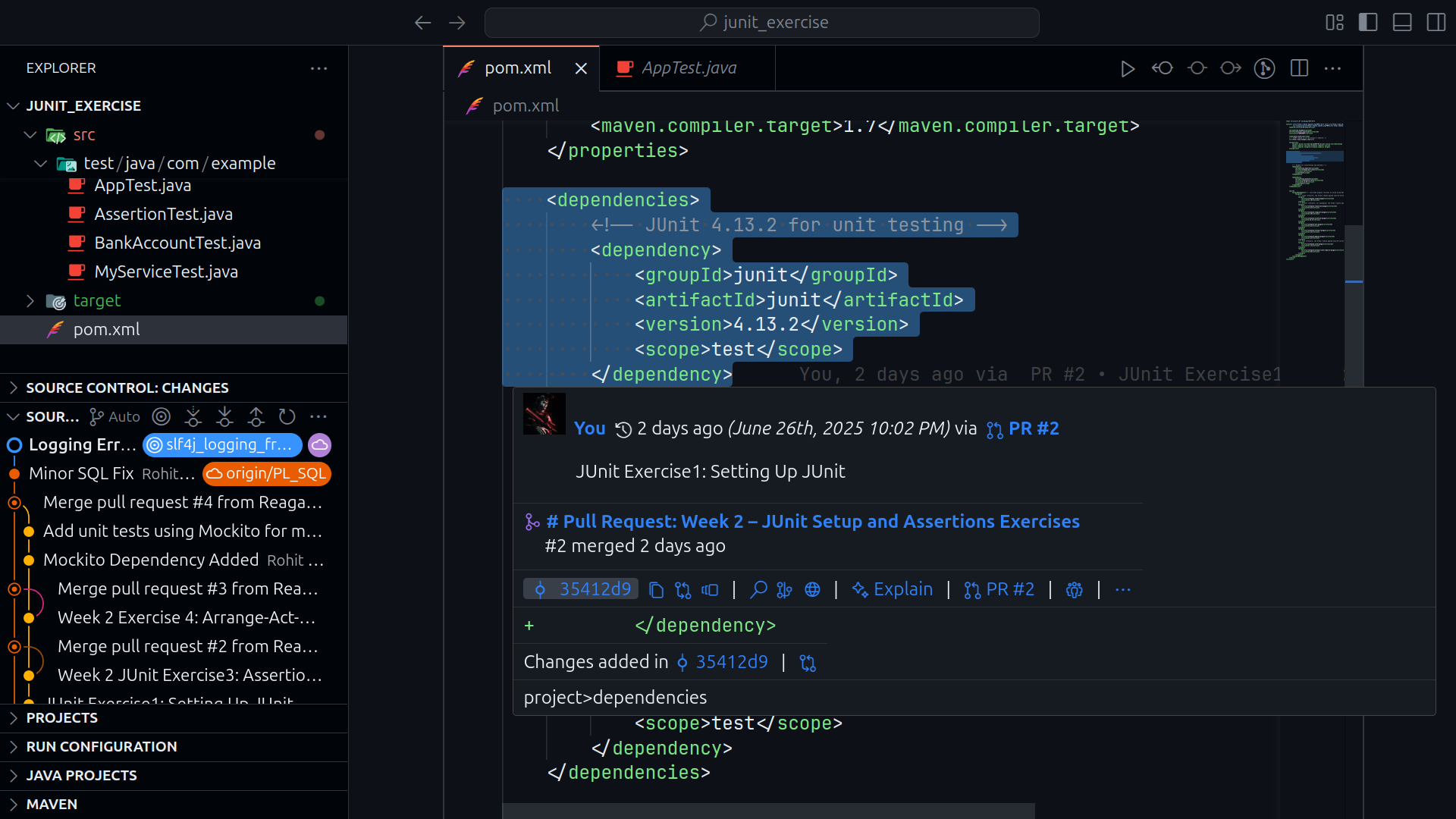
<version>4.13.2</version>

<scope>test</scope>

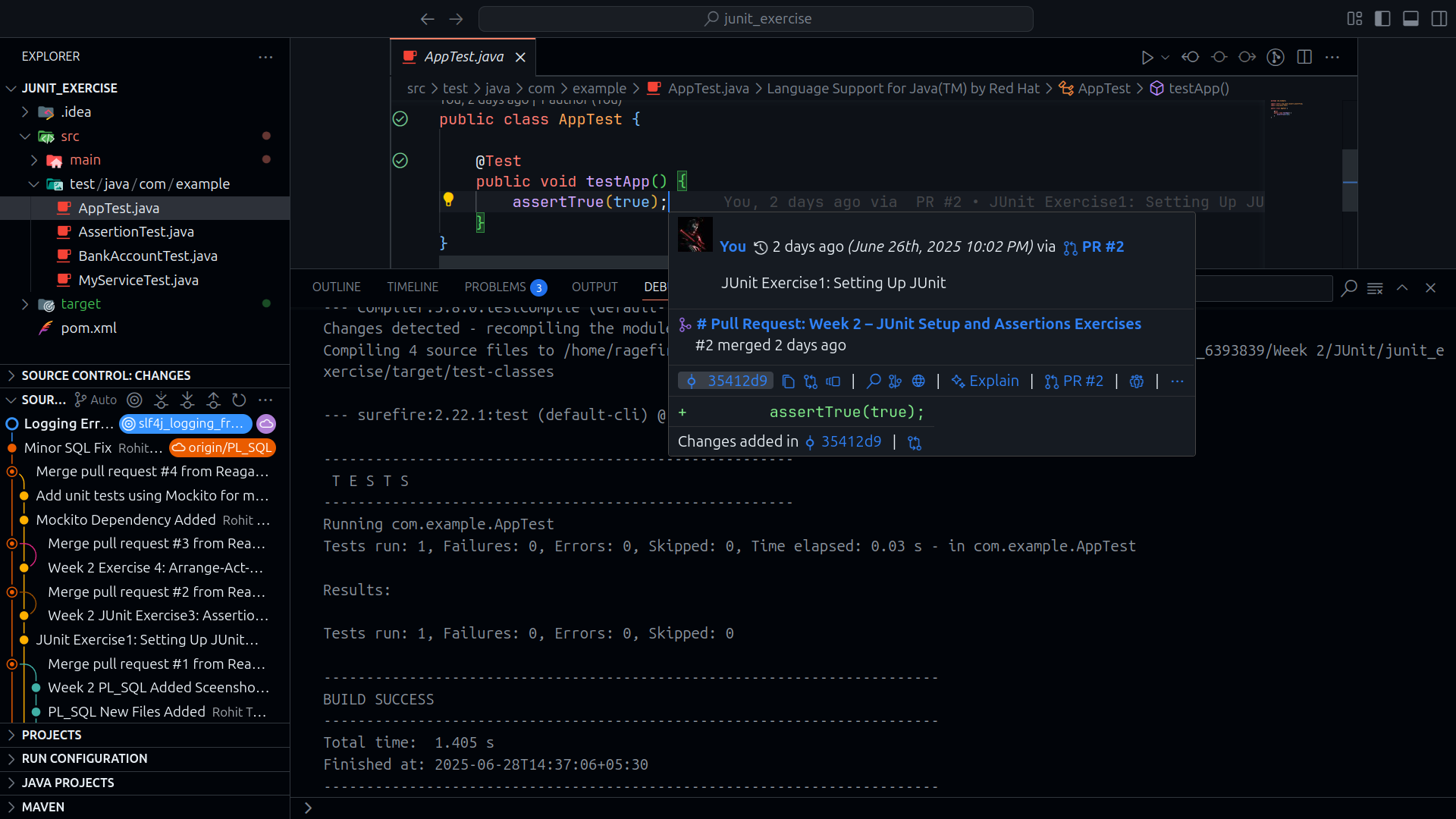
</dependency>

1. Maven project refreshed and JUnit downloaded
2. Test class created in src/test/java
3. Test method added with @Test annotation
4. Test runs successfully (command: mvn test)

**OUTPUT**



**RUNNING TEST**



**Exercise 3: Assertions in JUnit**

**Scenario:**

You need to use different assertions in JUnit to validate your test results.

**Steps:**

1. Write tests using various JUnit assertions.

**Solution Code:**

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());

}

}

**SOLUTION CHECKLIST**

1. Test class **AssertionsTest.java** created
2. Test method **testAssertions**() added with **@Test** annotation
3. Following JUnit assertions implemented inside the test method:

**assertEquals**(5, 2 + 3);

**assertTrue**(5 > 3);

**assertFalse**(5 < 3);

**assertNull**(null);

**assertNotNull**(new **Object**());

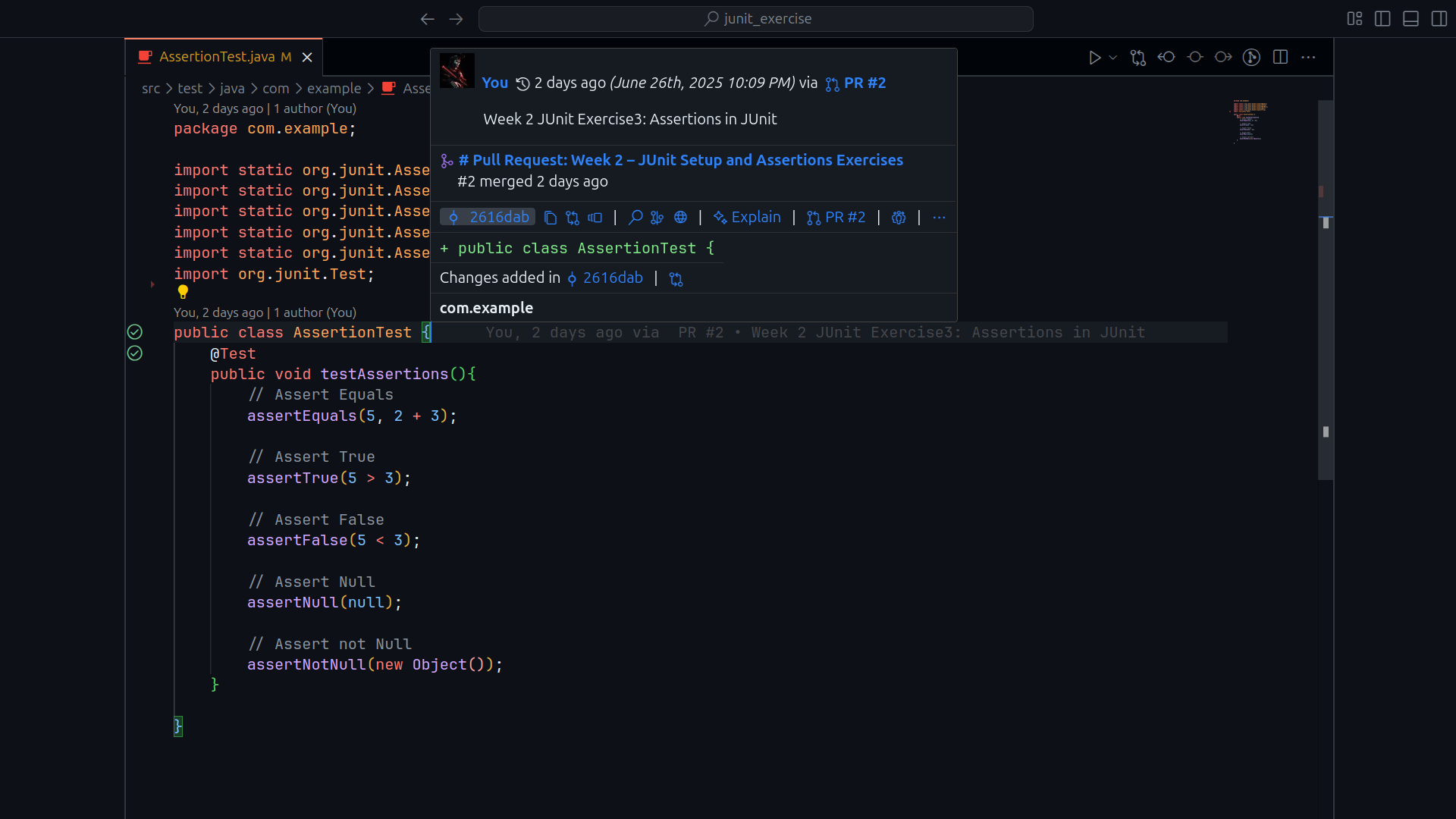
1. Required imports added:

import org.junit.Test;

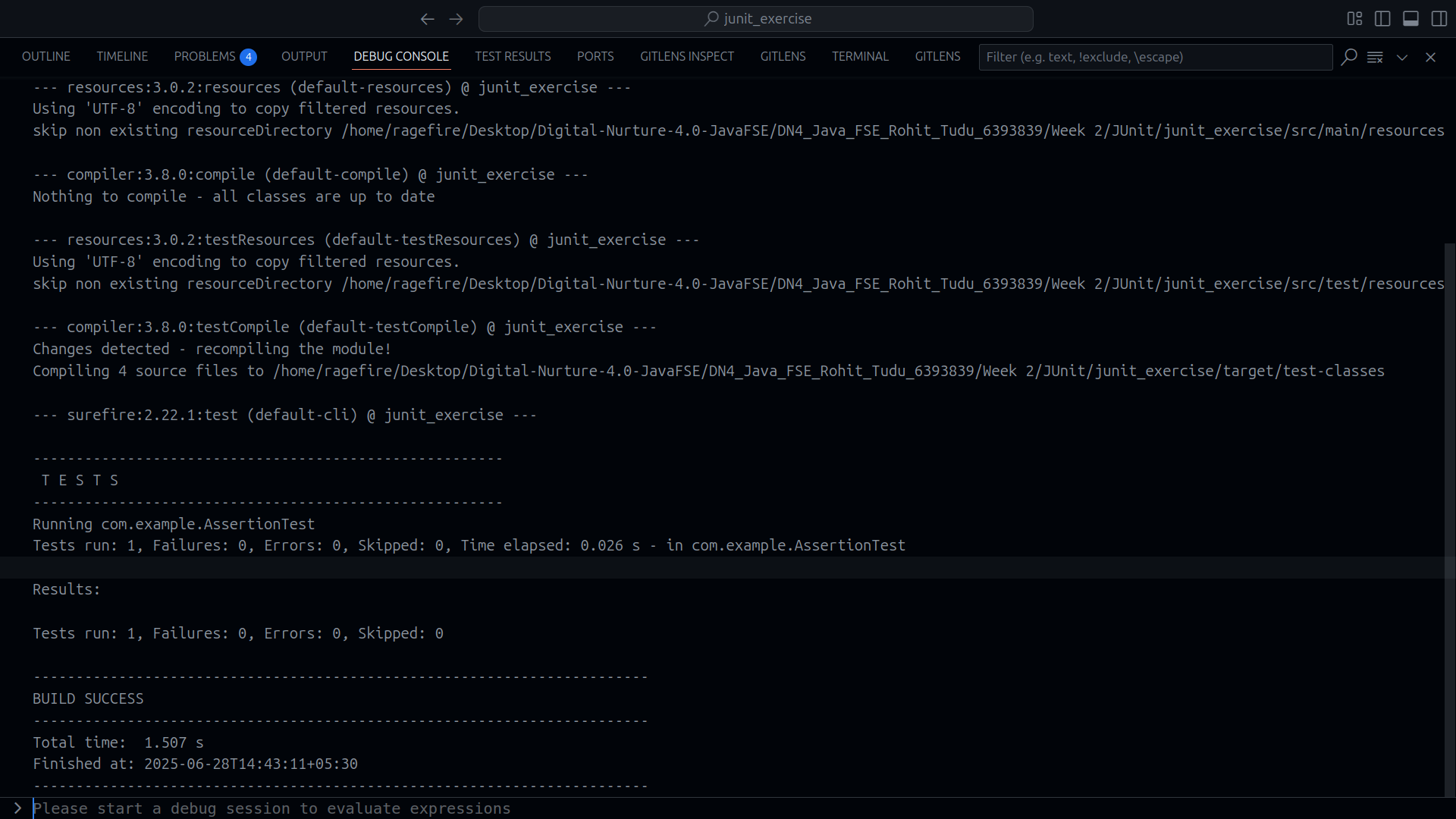
import static org.junit.Assert.\*;

1. Test executed using **mvn test** and ran successfully

**OUTPUT**



**RUNNING TEST**



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

**Scenario:**

You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

**Steps:**

1. Write tests using the AAA pattern.

2. Use **@Before** and **@After** annotations for setup and teardown methods.

**SOLUTION CHECKLIST**

1. Class BankAccount created under **src/main/java/com/example** with the following:

Constructor accepting initial balance

**deposit**(double **amount**) method

**withdraw**(double **amount**) method with check for sufficient funds

**getBalance**() method to return current balance

1. Test class **BankAccountTest** created under **src/test/java/com/example**
2. Used **Arrange-Act-Assert** (AAA) pattern in all test methods:

**testDeposit**() tests deposit functionality

**testWithdraw**() tests withdrawal within available balance

**testWithdrawMoreThanBalance**() tests prevention of overdrawing

1. Setup and Teardown methods implemented:

**@Before setUp**() initializes a fresh BankAccount object before each test

**@After tearDown**() cleans up by setting the object to null

1. Included print statements ("Setup" and "Teardown") to verify execution flow
2. Required imports added:

import org.junit.Before;

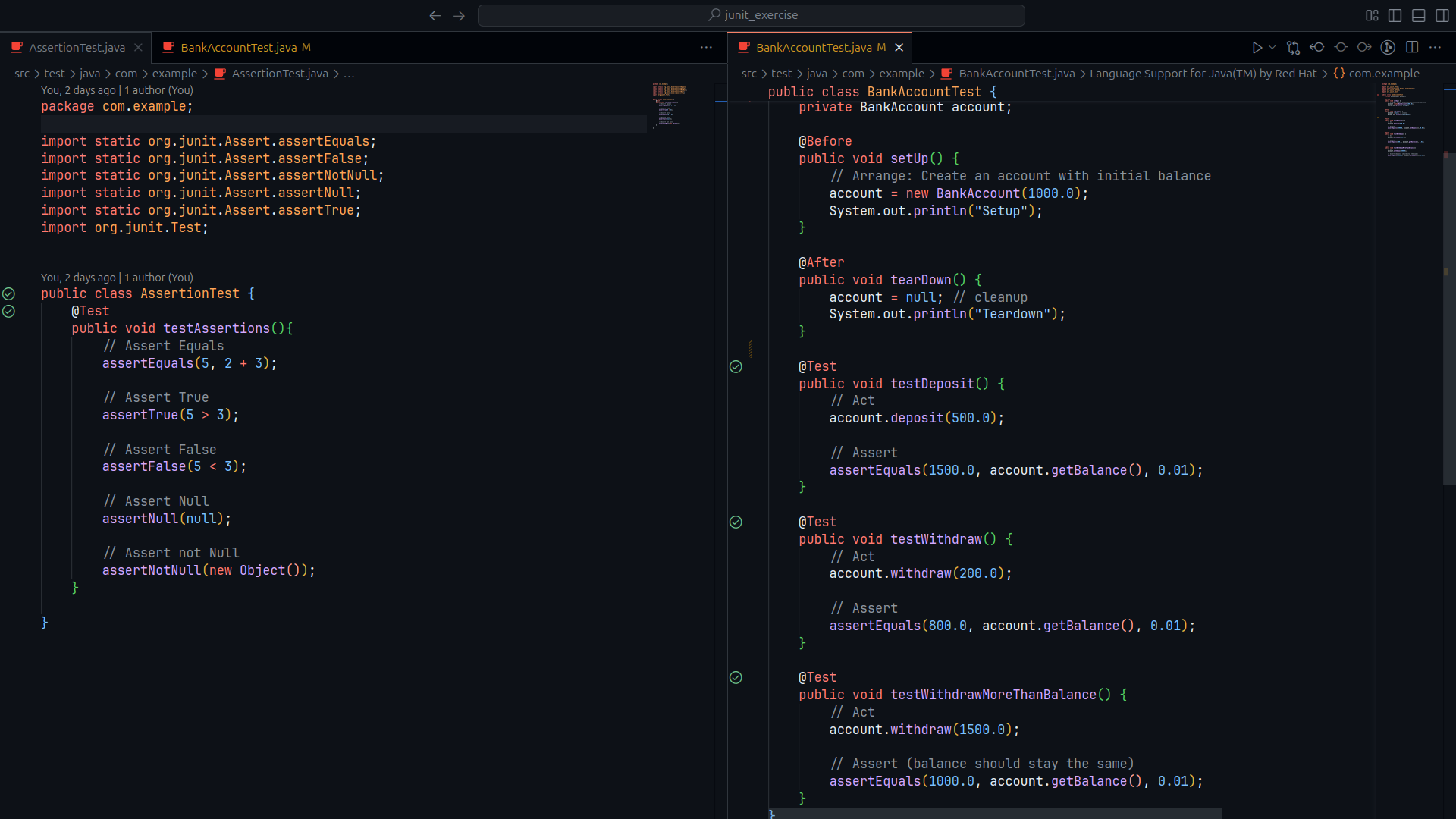
import org.junit.After;

import org.junit.Test;

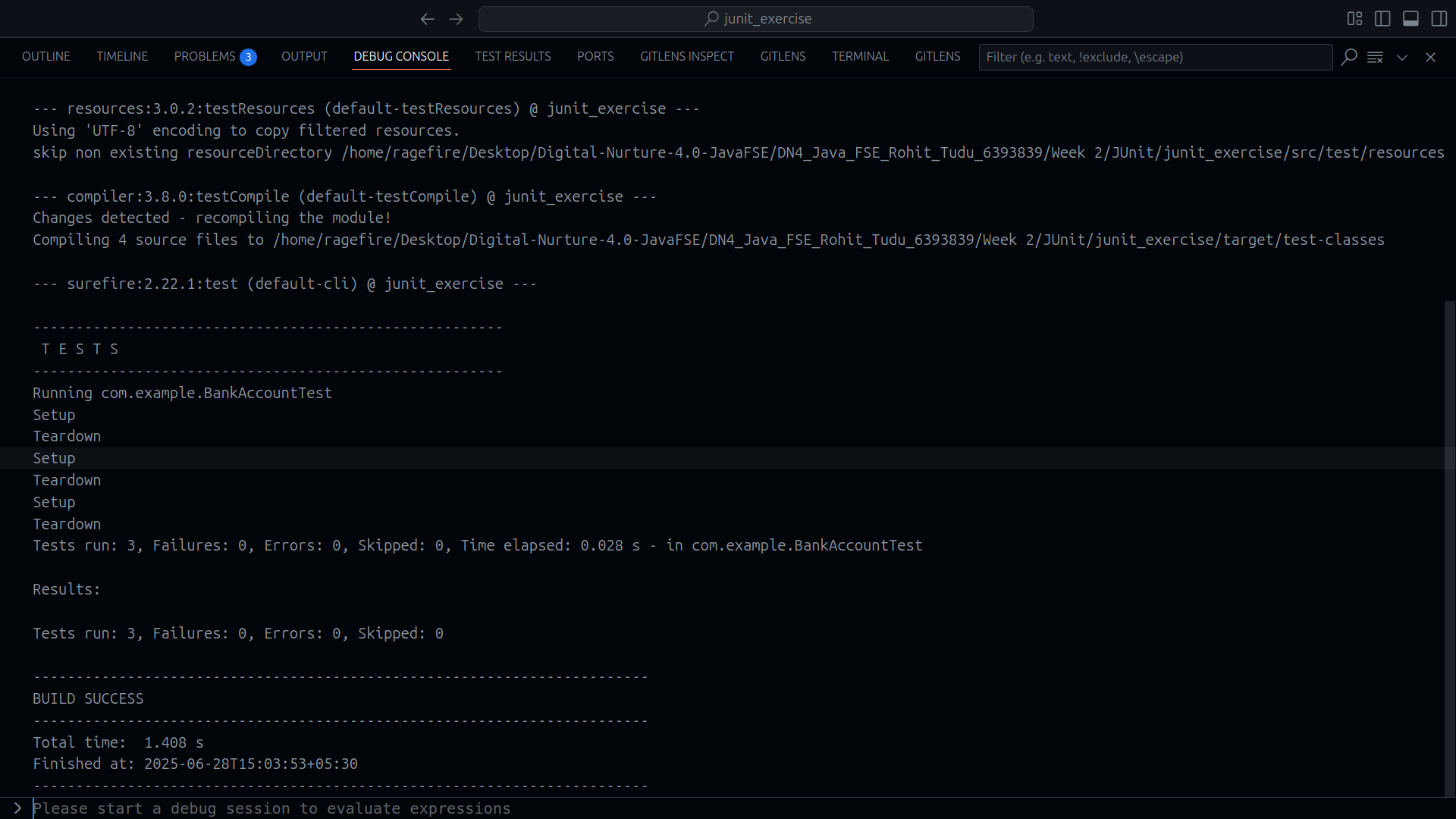
import static org.junit.Assert.assertEquals;

1. Tests executed using **mvn test** and passed successfully

**OUTPUT**



**RUNNING TEST**



1. **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);

}

}

**Exercise 2: Verifying Interactions**

**Scenario:**

You need to ensure that a method is called with specific arguments.

**Steps:**

1. Create a mock object.
2. Call the method with specific arguments.
3. Verify the interaction.

**Solution Code:**

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();

}

}

**SOLUTION CHECKLIST (COMBINED EX\_1 & EX\_2)**

1. Interface/class ExternalApi created with a method getData()
2. Class MyService implemented with:
   1. Constructor that accepts an ExternalApi instance
   2. Method **fetchData**() that calls **externalApi.getData**()
3. Test class **MyServiceTest** created under **src/test/java/com/example**

🧪 Exercise 1: **Mocking and Stubbing**

1. In **testExternalApi**():
   1. Created mock object using **Mockito.mock**(ExternalApi.class)
   2. Stubbed **getData**() to return "Mock Data" using **when**(...).**thenReturn**(...)
   3. Created a **MyService** instance using the mock Called **fetchData**() and asserted result with:

**assertEquals**("Mock Data", result);

🧪 Exercise 2: **Verifying Interactions**

1. In **testVerifyInteraction**():
   1. Created mock object of **ExternalApi**
   2. Passed it to **MyService** and called **fetchData**()
   3. Verified interaction using:
      1. **verify**(mockApi).getData();
2. Required imports added:

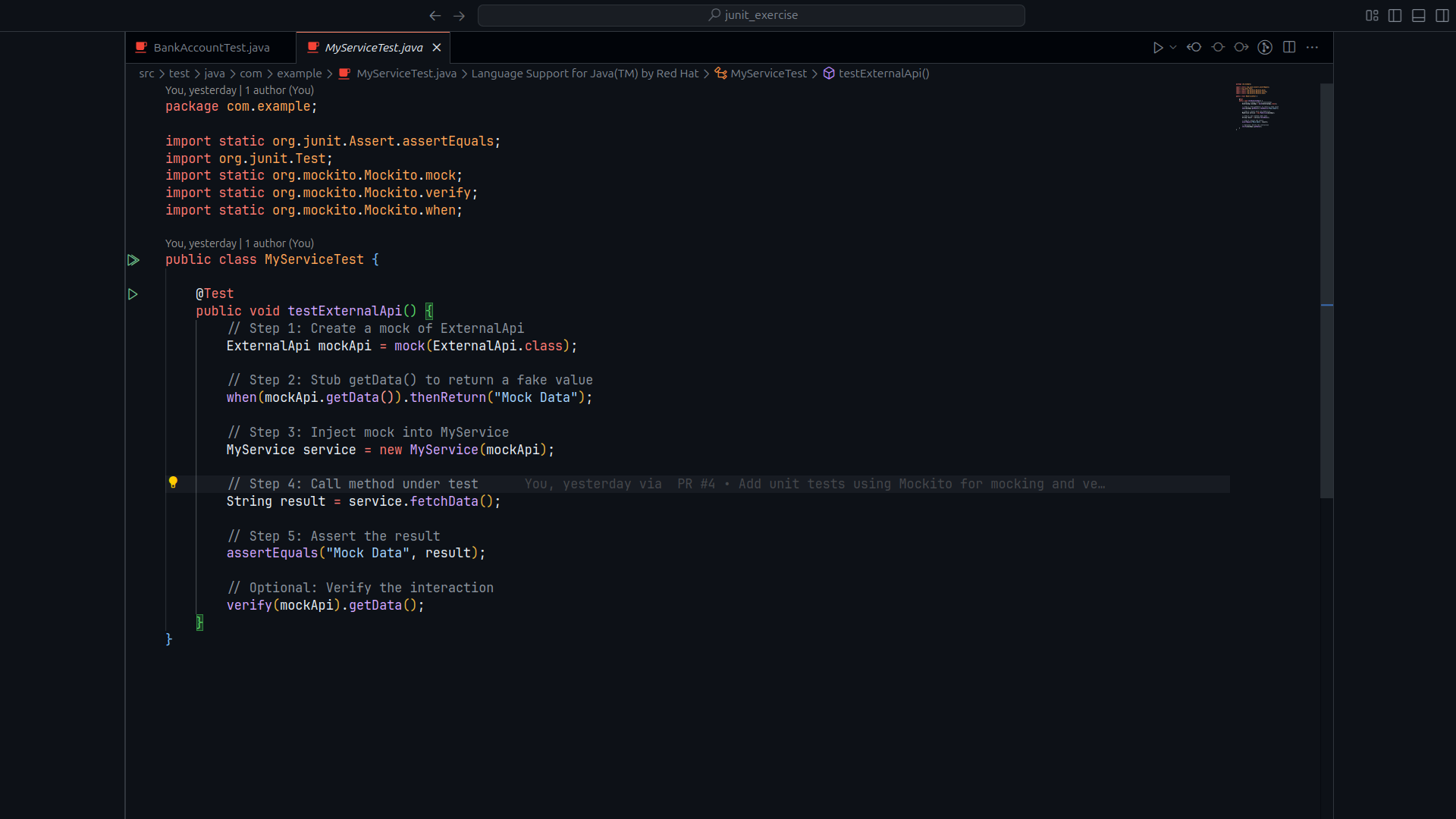
import static org.mockito.Mockito.\*;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

1. Tests executed using mvn test and passed successfully

**OUTPUT**



**RUNNING TEST**

