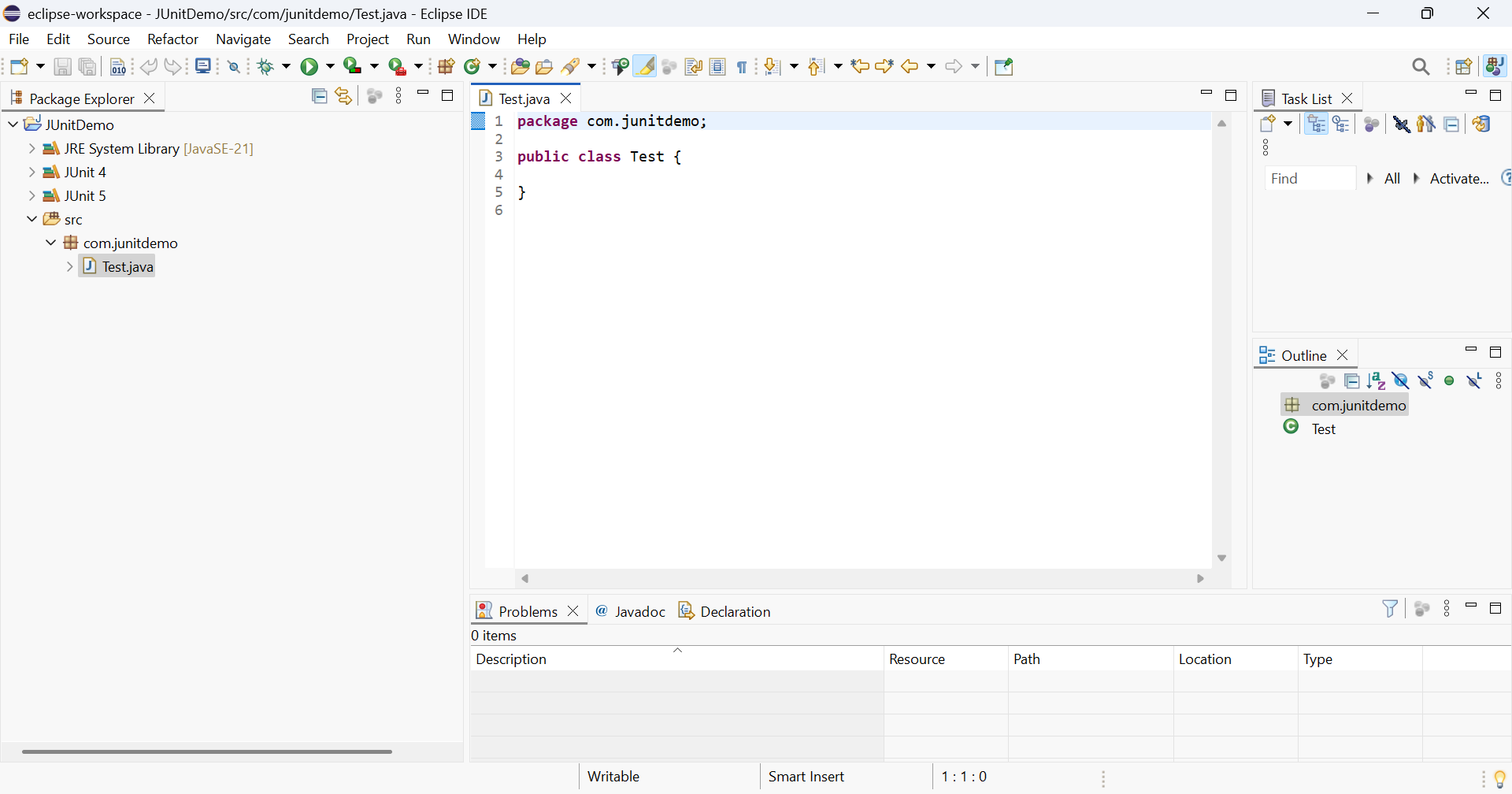
**JUnit\_Basic Testing Exercises:**

**Exercise 1: Setting Up JUnit**

**Scenario:**

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

Solution:



**Exercise 3: Assertions in JUnit**

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

Code:

**package** com.junitdemo;

**import** **static** org.junit.Assert.\*;

**import** org.junit.Test;

**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**);

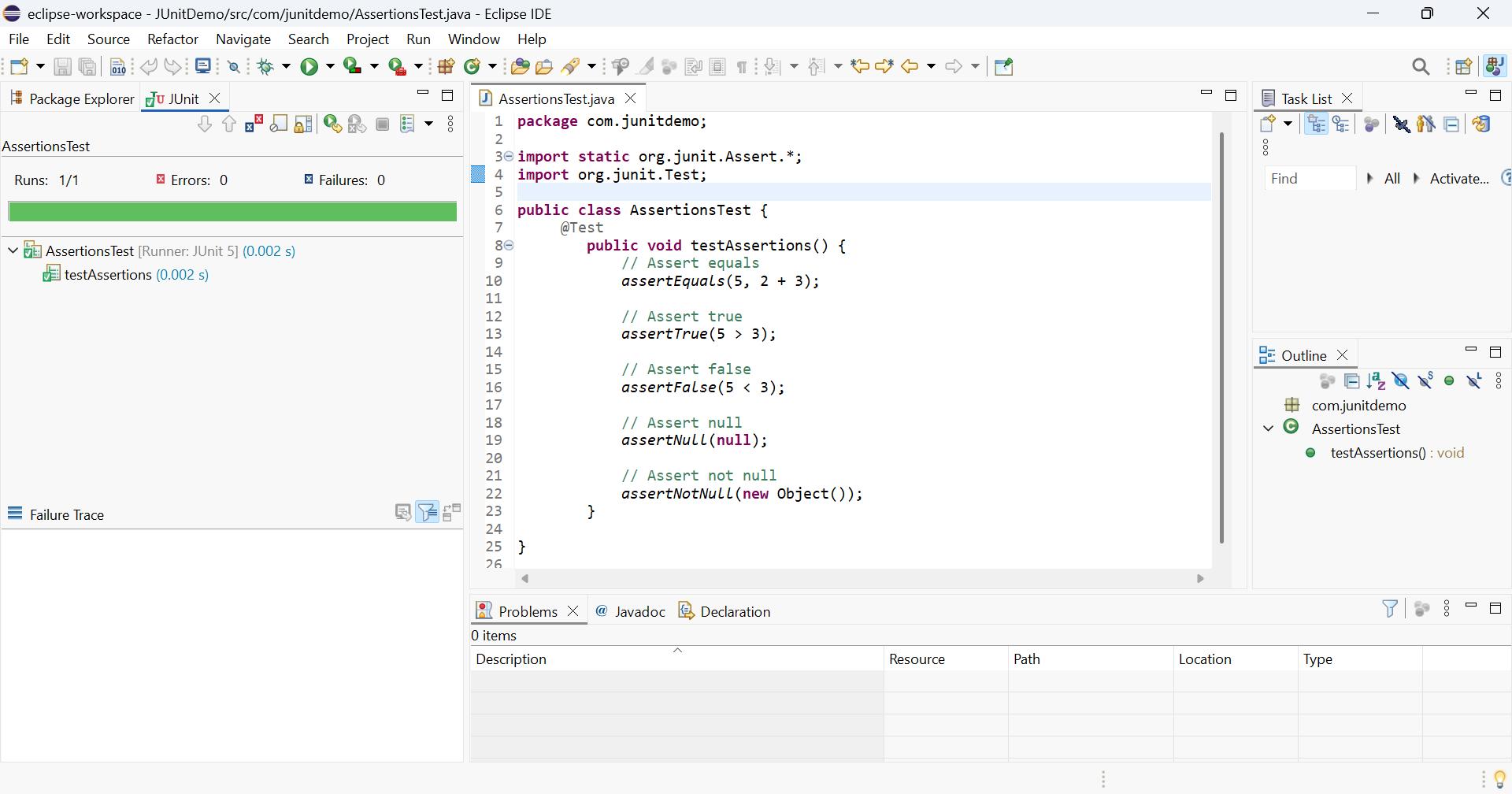
// Assert not null

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

}

}

Output:



**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.**

Code:

Calculator.java

**package** com.junitdemo;

**public** **class** Calculator {

**public** **int** add(**int** a, **int** b)

{

**return** a + b;

}

**public** **int** subtract(**int** a, **int** b)

{

**return** a - b;

}

}

CalculatorTest.java

**package** com.junitdemo;

**import** **static** org.junit.Assert.\*;

**import** org.junit.Before;

**import** org.junit.After;

**import** org.junit.Test;

**public** **class** CalculatorTest {

**private** Calculator calculator;

@Before

**public** **void** setUp() {

calculator = **new** Calculator();

System.***out***.println("Setup completed");

}

@After

**public** **void** tearDown() {

calculator = **null**;

System.***out***.println("Teardown completed");

}

@Test

**public** **void** testAddition() {

**int** result = calculator.add(2, 3); // Act

*assertEquals*(5, result); // Assert

}

@Test

**public** **void** testSubtraction() {

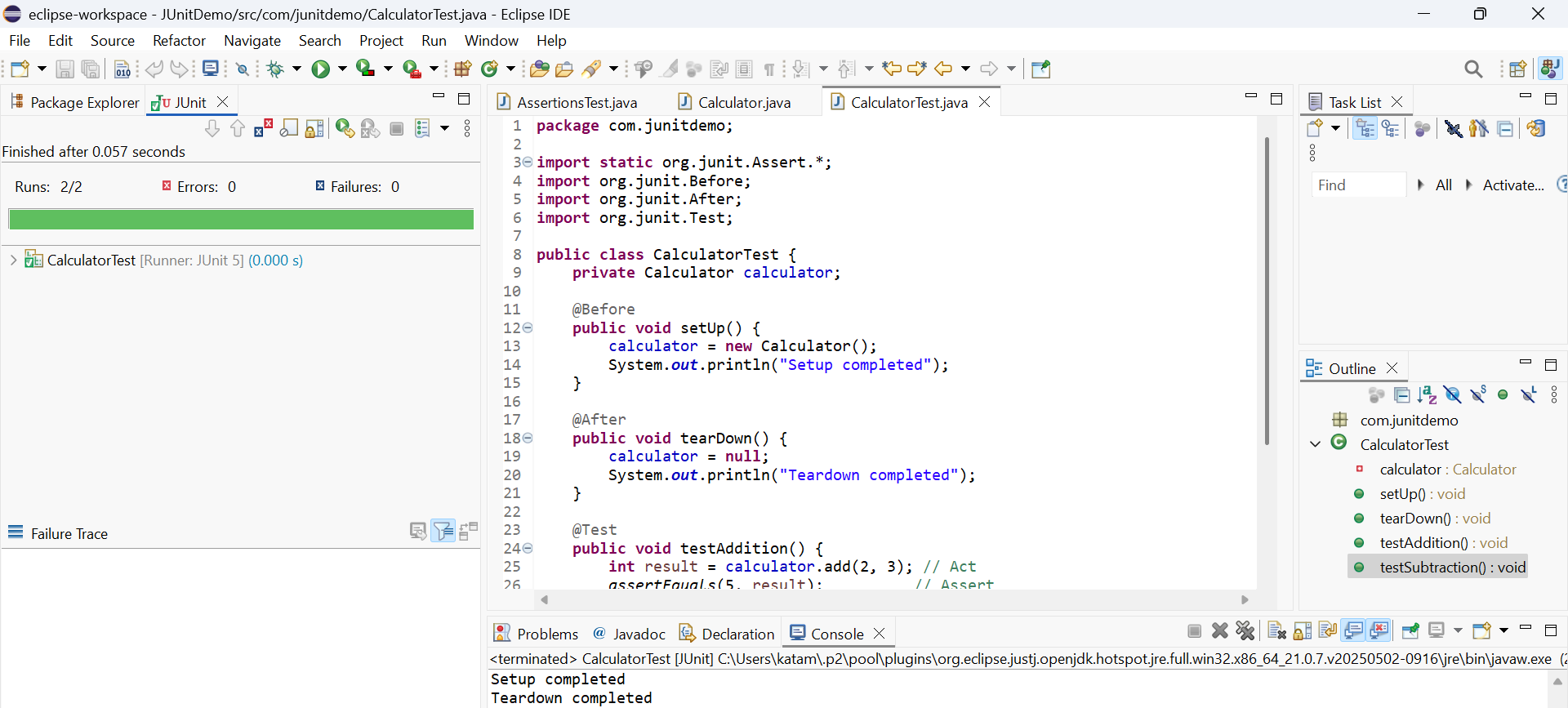
**int** result = calculator.subtract(5, 3); // Act

*assertEquals*(2, result); // Assert

}

}

Output:



**Mockito 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.**

Code:

ExternalApi.java

package com.example.mockitodemo;

public interface ExternalApi {

String getData();

}

MyService.java

**package** com.example.mockitodemo;

**public** **class** MyService {

**private** ExternalApi api;

**public** MyService(ExternalApi api) {

**this**.api = api;

}

**public** String fetchData() {

**return** api.getData();

}

}

MyServiceTest.java

**package** com.example.mockitodemo;

**import** **static** org.mockito.Mockito.\*;

**import** **static** org.junit.jupiter.api.Assertions.\*;

**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);

}

@Test

**public** **void** testVerifyInteraction() {

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

MyService service = **new** MyService(mockApi);

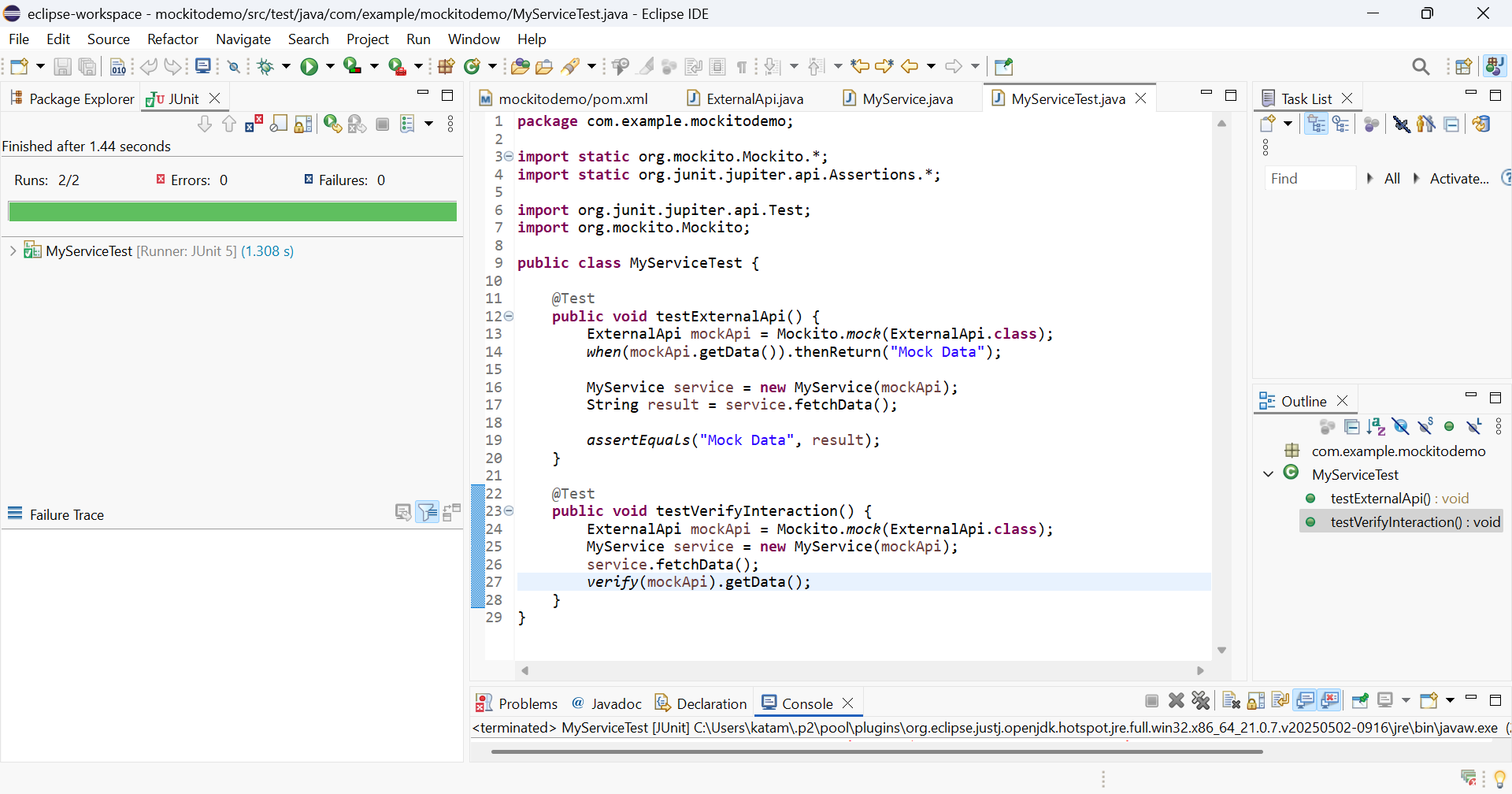
service.fetchData();

*verify*(mockApi).getData();

}

}

Output:

****

**Exercise 2: Verifying Interactions**

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

Code:

ExternalApi.java

package com.example.mockitodemo;

public interface ExternalApi {

String getData();

}

MyService.java

**package** com.example.mockitodemo;

**public** **class** MyService {

**private** ExternalApi api;

**public** MyService(ExternalApi api) {

**this**.api = api;

}

**public** String fetchData() {

**return** api.getData();

}

}

MyServiceTest.java

**package** com.example.mockitodemo;

**import** **static** org.mockito.Mockito.\*;

**import** org.junit.jupiter.api.Test;

**import** org.mockito.Mockito;

**public** **class** MyServiceTest {

@Test

**public** **void** testVerifyInteraction() {

// Step 1: Create a mock

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

// Step 2: Inject into service

MyService service = **new** MyService(mockApi);

// Step 3: Call the method

service.fetchData();

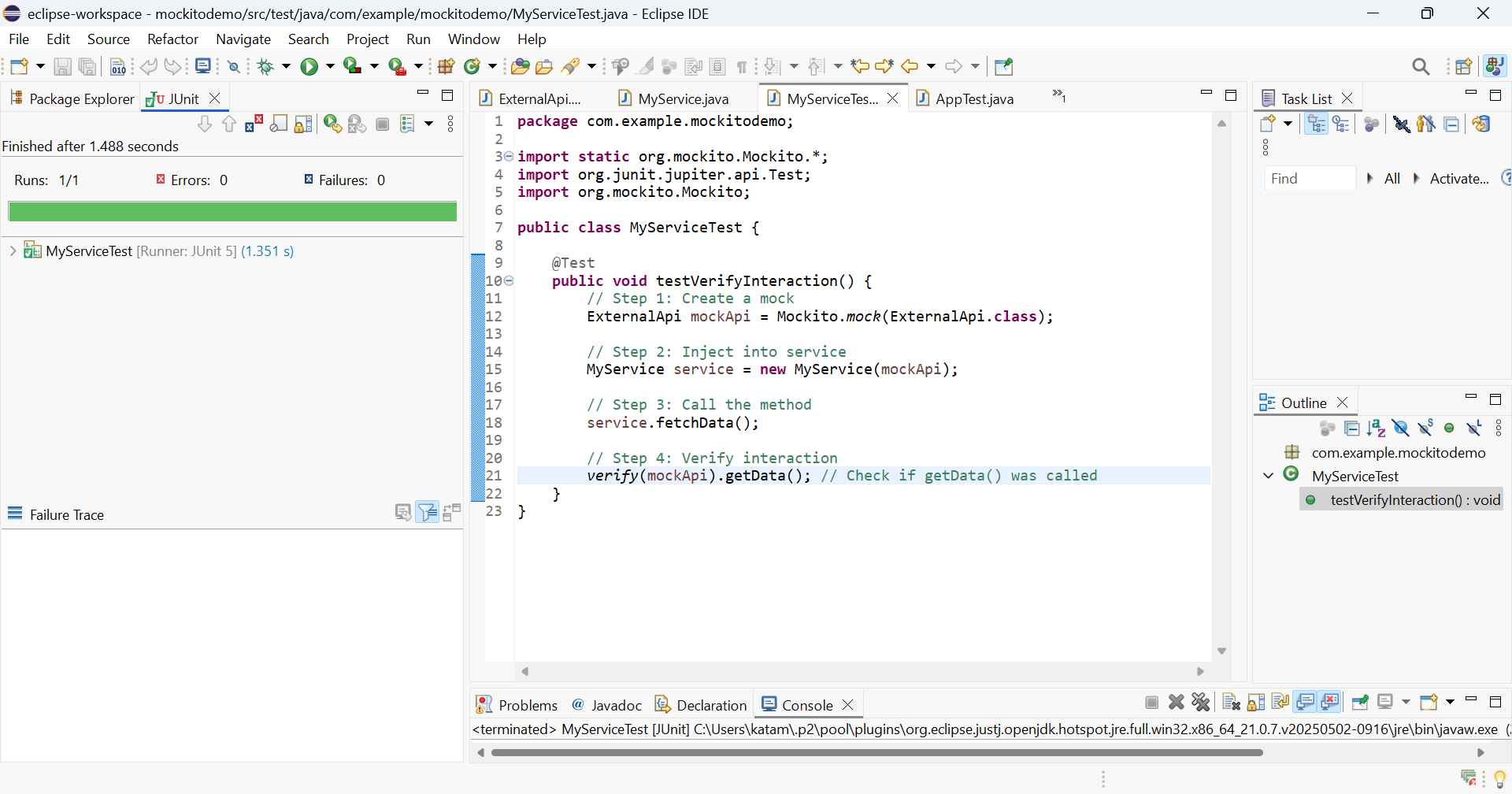
// Step 4: Verify interaction

*verify*(mockApi).getData(); // Check if getData() was called

}

}

Output:

****