**WEEK – 2 : JUNIT\_BASIC TESTING EXERCISES**

**EXERCISE 1: SETTING UP JUNIT**

**CODE IMPLEMENTATION**

Main Class: Calculator.java

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

Test Class: CalculatorTest.java

package com.example;

import static org.junit.Assert.\*;

import org.junit.Test;

public class CalculatorTest {

*@Test*

public void testAdd() {

Calculator calc = new Calculator();

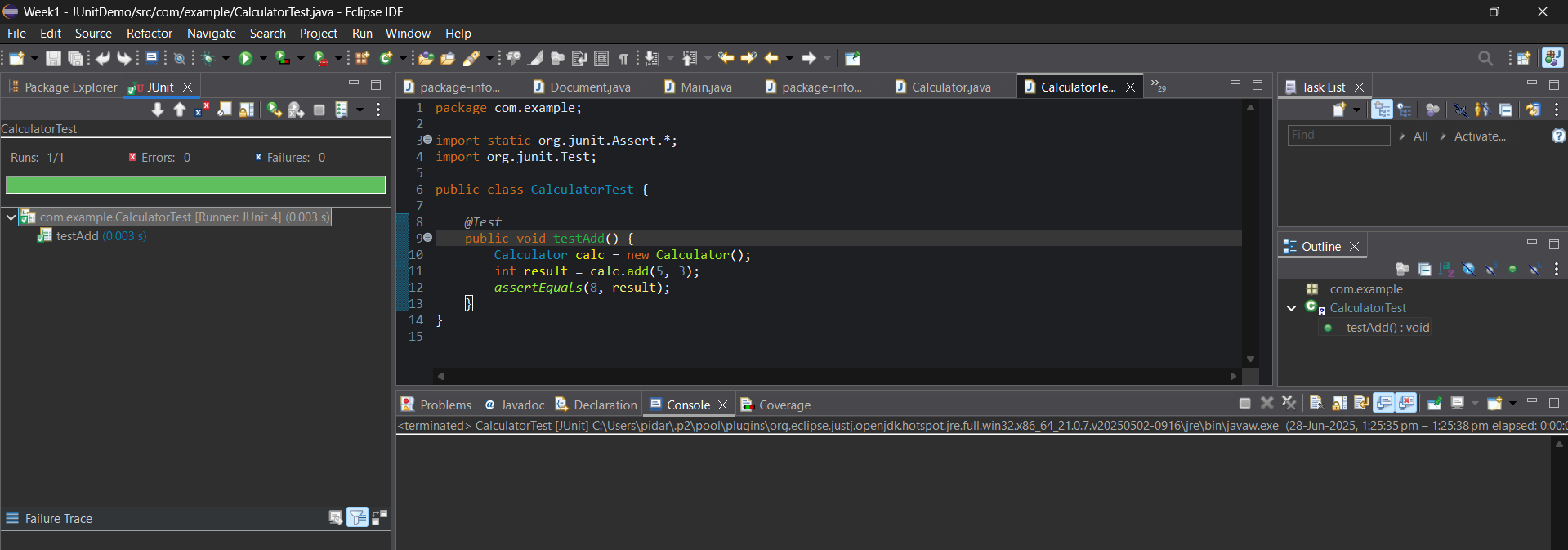
int result = calc.add(5, 3);

*assertEquals*(8, result);

}

}

**OUTPUT**



**EXERCISE 3:** ASSERTIONS IN JUNIT

**CODE IMPLEMENTATION**

AssertionsTest.java

package com.example;

import static org.junit.Assert.\*;

import org.junit.Test;

public class AssertionsTest {

*@Test*

public void testAssertions() {

*assertEquals*(5, 2 + 3);

*assertTrue*(5 > 3);

*assertFalse*(5 < 3);

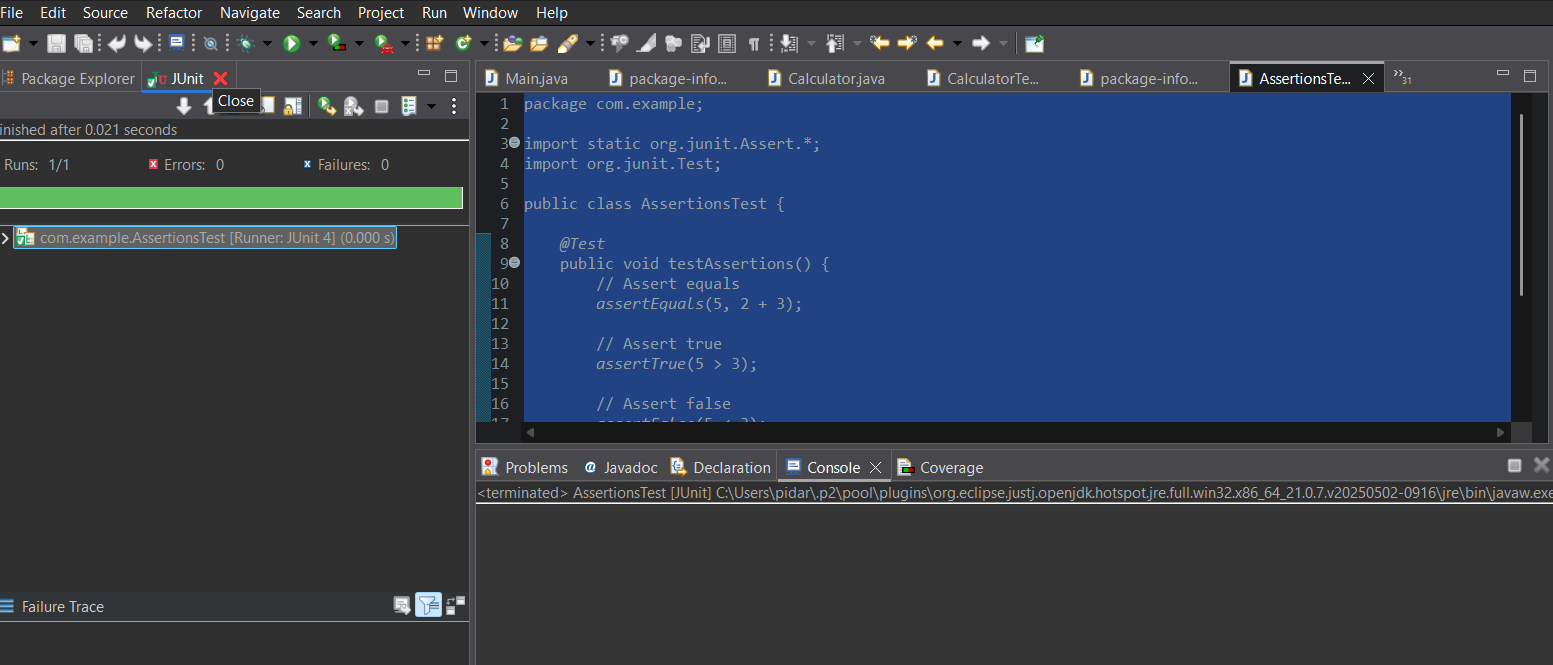
*assertNull*(null);

*assertNotNull*(new Object());

}

}

**OUTPUT**



**EXERCISE 4: ARRANGE-ACT-ASSERT (AAA) PATTERN WITH SETUP AND TEARDOWN** **IN** **JUNIT**

**CODE IMPLEMENTATION**

Calculator.Java

package com.example;

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.example;

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: Calculator initialized");

}

*@After*

public void tearDown() {

calculator = null;

System.***out***.println("🧹 Teardown: Calculator cleaned up");

}

*@Test*

public void testAdd() {

int a = 5, b = 3;

int result = calculator.add(a, b);

*assertEquals*(8, result);

}

public void testSubtract() {

int a = 10, b = 4;

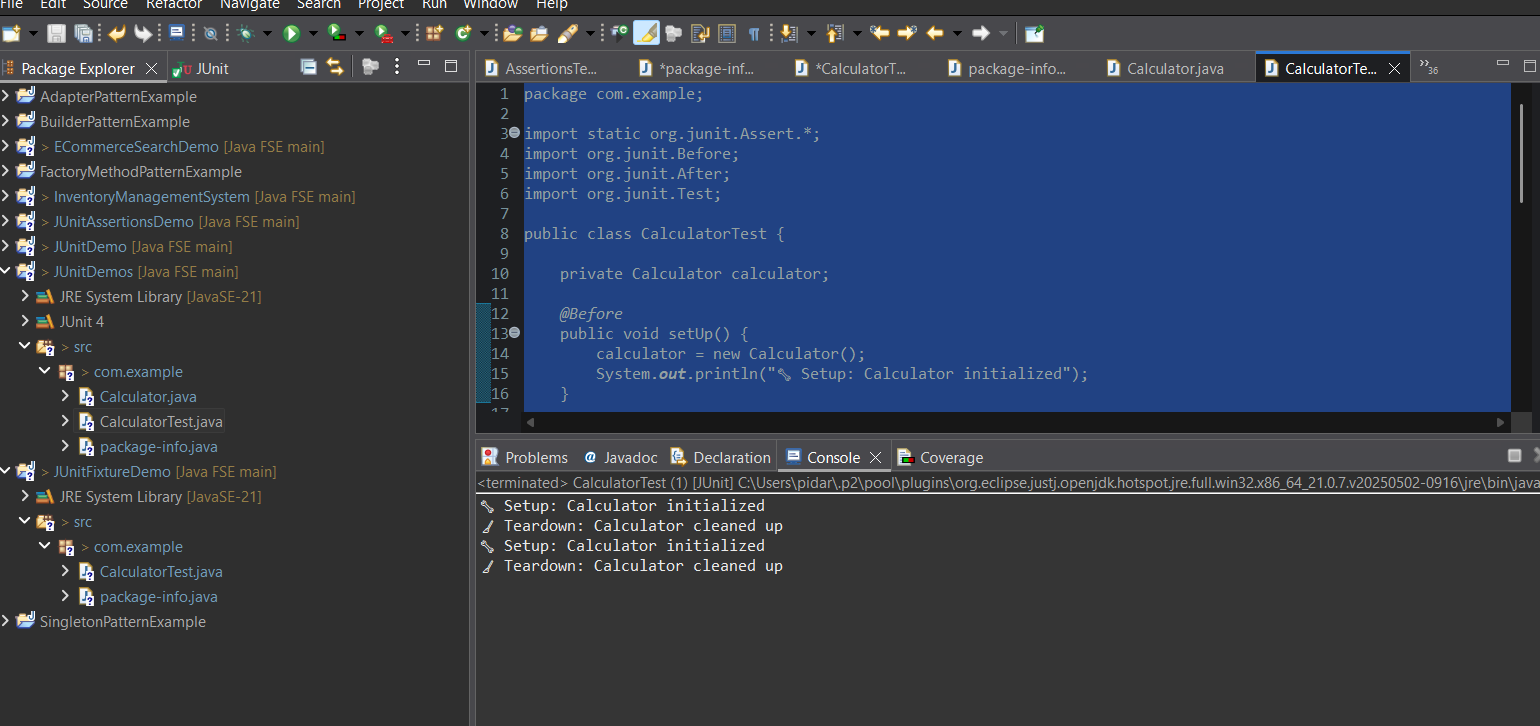
int result = calculator.subtract(a, b);

*assertEquals*(6, result);

}

}

**OUTPUT**



**WEEK 2: MOCKITO EXERCISES**

**EXERCISE 1: MOCKING AND STUBBING**

**Step 1:** Create a java maven project in eclipse.

**Step 2:** Update the pom.xml file and update the Maven project.

**Step 3:** Create a ExternalApi.java in src/main/java

package com.example;

public interface ExternalApi {

String getData();

}

**Step 4:** Create a Service.java file in src/main/java

package com.example;

public class MyService {

private ExternalApi externalApi;

public MyService(ExternalApi externalApi) {

this.externalApi = externalApi;

}

public String fetchData() {

return externalApi.getData();

}

}

**Step 5:** Create a ServiceTest.java file in src/test/java

package com.example;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

import static org.mockito.Mockito.\*;

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

public class MyServiceTest {

*@Test*

public void testExternalApi() {

// Step 1: Create mock

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

// Step 2: Stub method

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

// Step 3: Use mock in service

MyService service = new MyService(mockApi);

String result = service.fetchData();

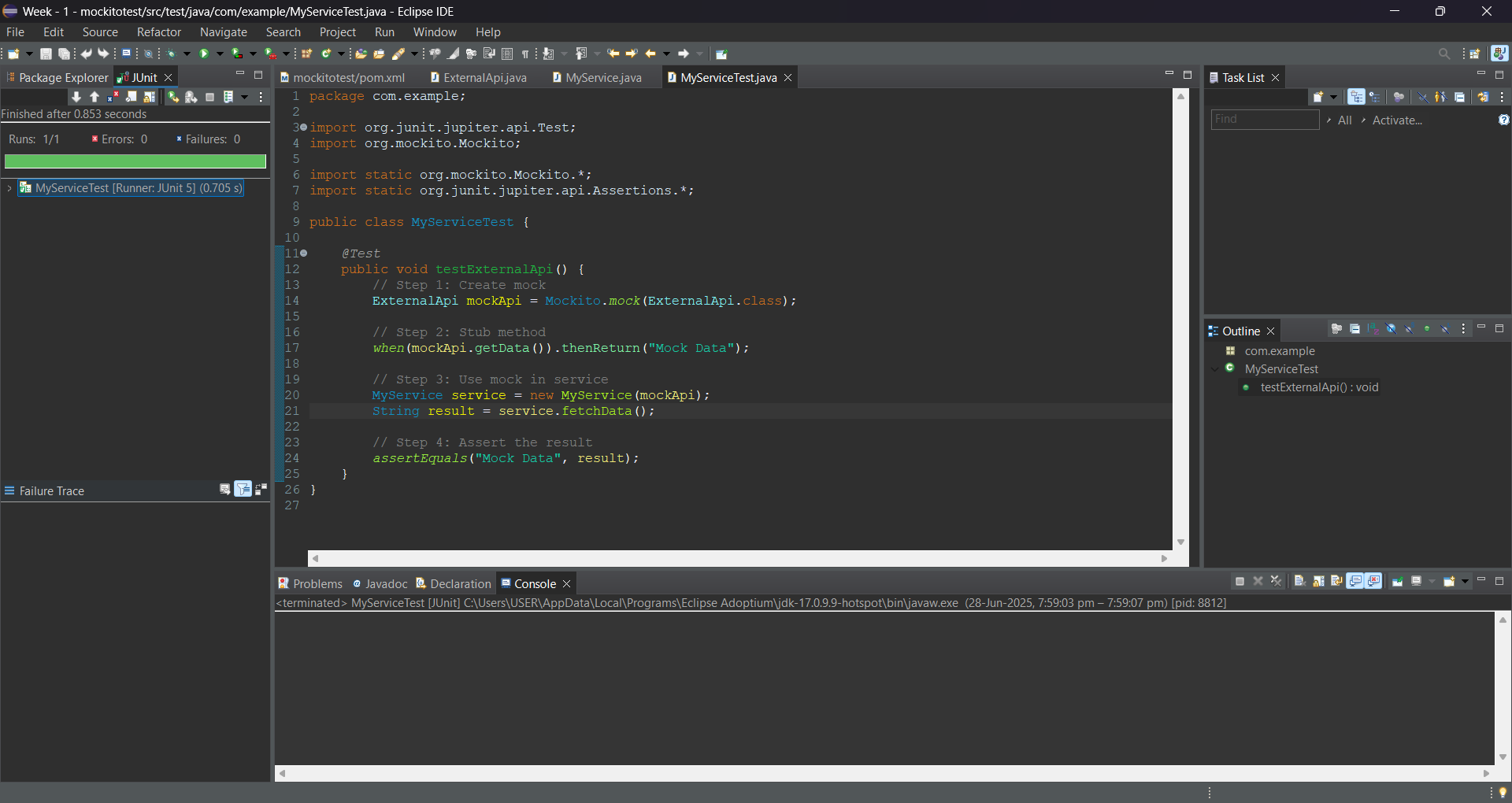
// Step 4: Assert the result

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

}

}

**OUTPUT**



**EXERCISE 2: VERIFYING INTERACTIONS**

**Step 1:** Create a java maven project in eclipse.

**Step 2:** Update the pom.xml file and update the Maven project.

**Step 3:** Create a ExternalApi.java in src/main/java

package com.example;

public interface ExternalApi {

String getData();

}

**Step 4:** Create a Service.java file in src/main/java

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**Step 5:** Create a ServiceTest.java file in src/test/java

package com.example;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

*@Test*

public void testVerifyInteraction() {

// Step 1: Mock the ExternalApi

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

// Step 2: Pass mock to service

MyService service = new MyService(mockApi);

service.fetchData();

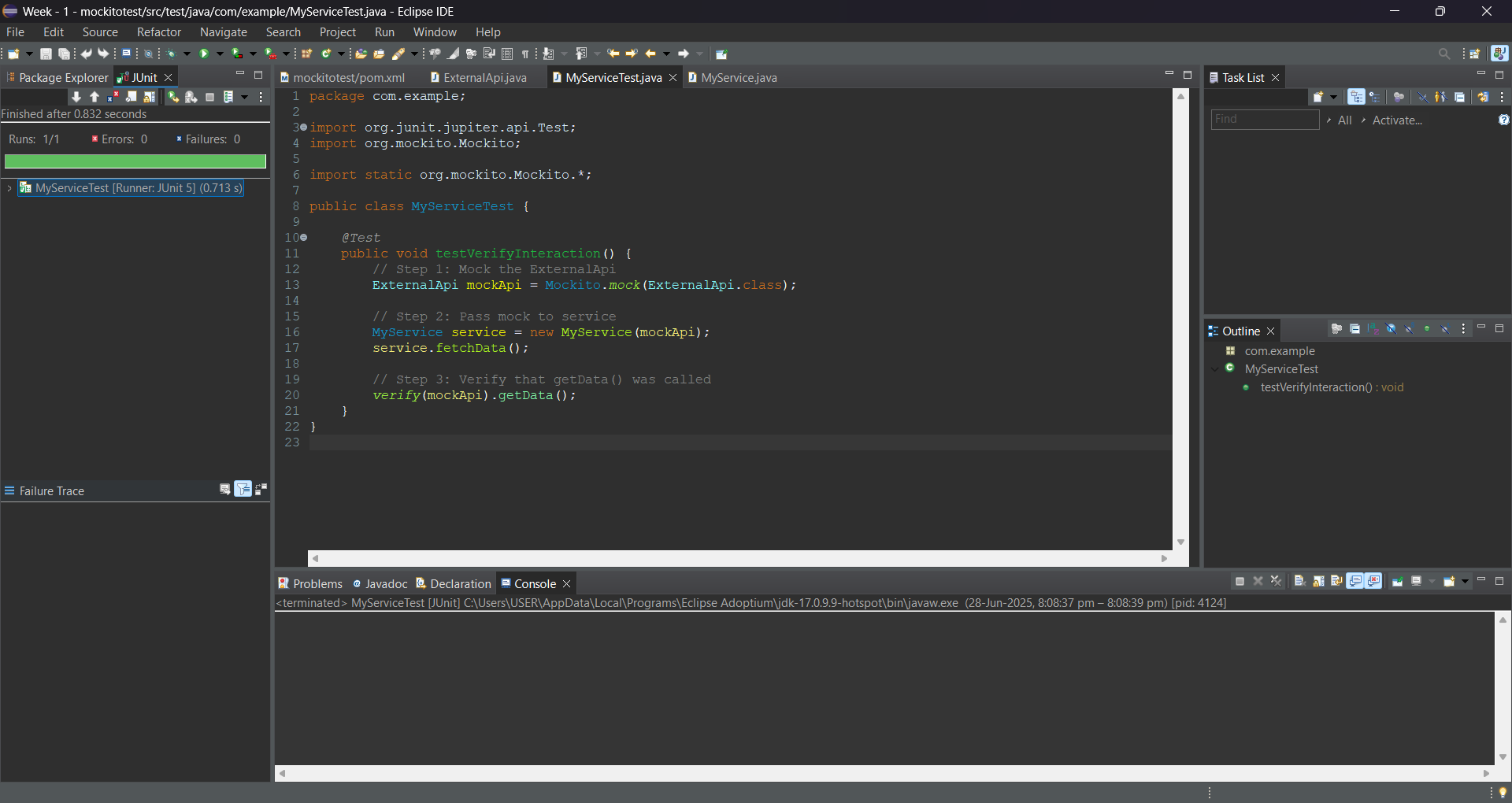
// Step 3: Verify that getData() was called

*verify*(mockApi).getData();

}

}

**OUTPUT**

****