**WEEK 2**

**TDD using JUnit5 and Mockito**

**Exercise 1: Setting Up Junit**

**Code:**

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**Calulator.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;

}

public int multiply(int a, int b) {

return a \* b;

}

public int divide(int a, int b) {

if (b == 0) {

throw new IllegalArgumentException("Cannot divide by zero.");

}

return a / b;

}}

**CalulatorTest.java:**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

*@Test*

public void testAdd() {

Calculator calc = new Calculator();

*assertEquals*(9, calc.add(4, 5));

}

*@Test*

public void testSubtract() {

Calculator calc = new Calculator();

*assertEquals*(1, calc.subtract(5, 4));

}

*@Test*

public void testMultiply() {

Calculator calc = new Calculator();

*assertEquals*(20, calc.multiply(4, 5));

}

*@Test*

public void testDivide() {

Calculator calc = new Calculator();

*assertEquals*(2, calc.divide(10, 5));

}

*@Test*(expected = IllegalArgumentException.class)

public void testDivideByZero() {

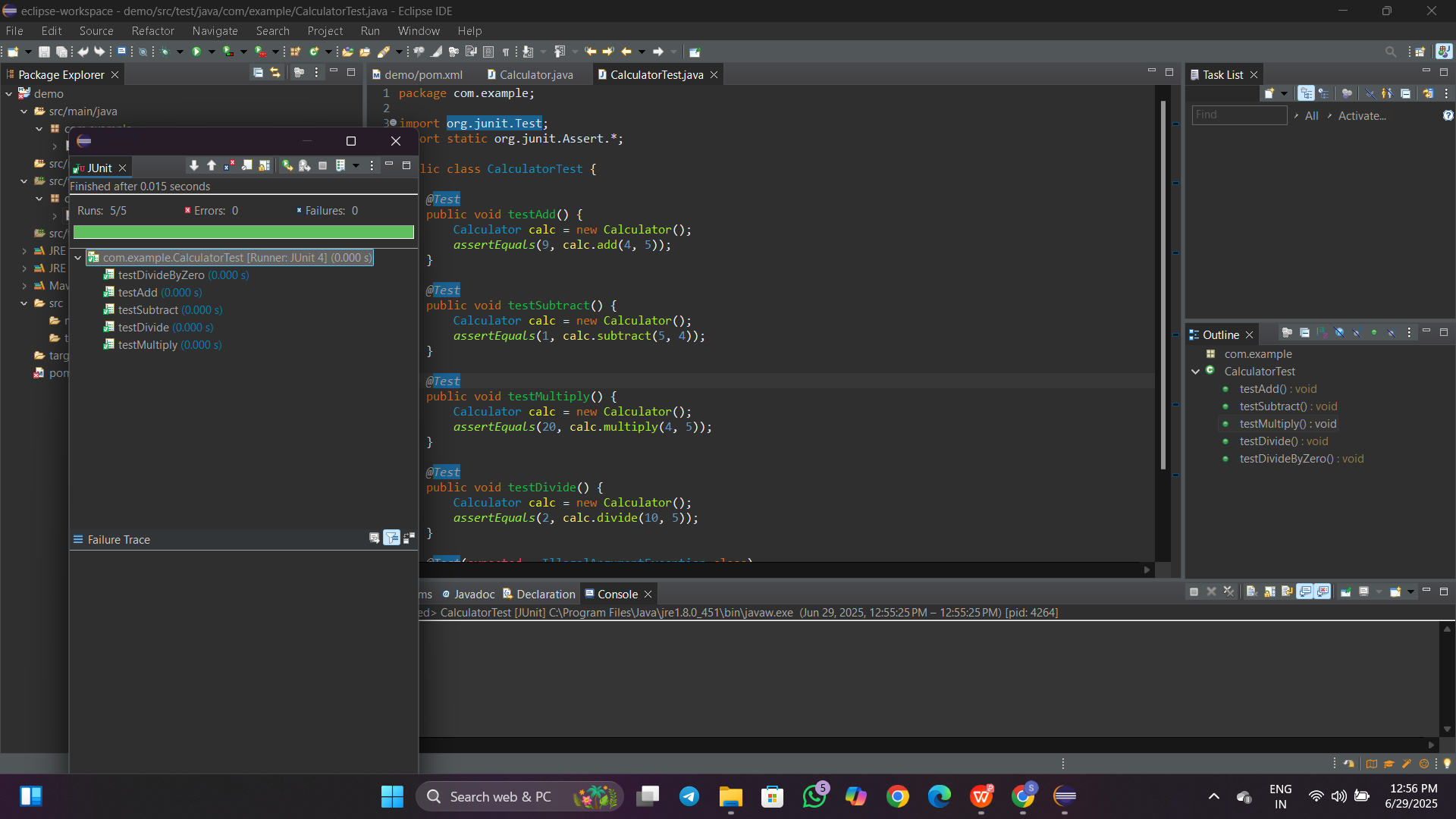
Calculator calc = new Calculator();

calc.divide(10, 0);

}

}

**OUTPUT:**



**Exercise 3: Assertions in Junit**

**Code:**

**AsserttionsTest.java**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

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

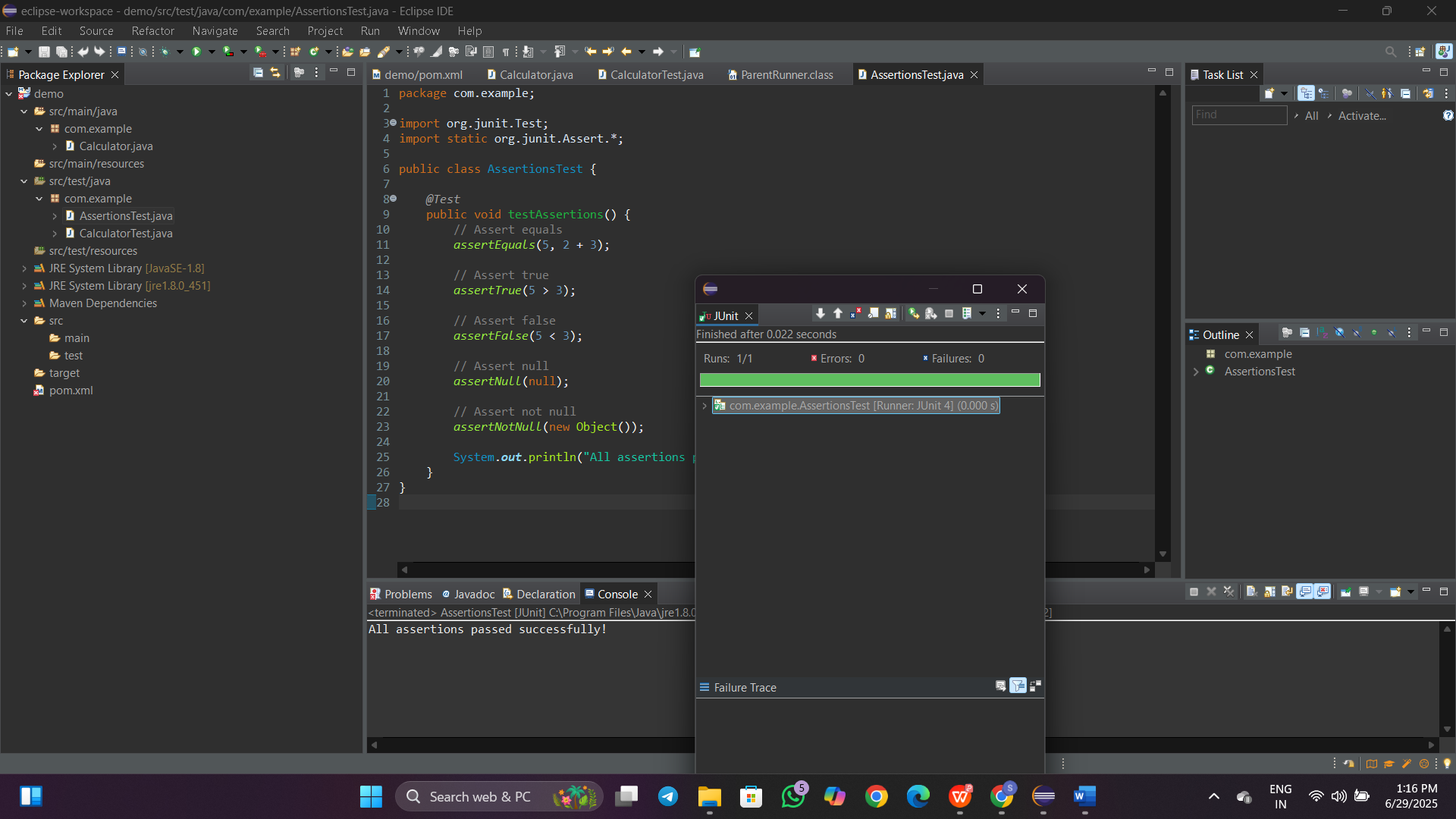
assertNotNull(new Object());

System.*out*.println("All assertions passed successfully!");

}

}

**OUTPUT:**



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

**Teardown Methods in Junit**

**Code:**

**CalculatorLifecycleTest.java**

package com.example;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorLifecycleTest {

private Calculator calc;

@Before

public void setUp() {

System.out.println("🔧 Setup: Creating Calculator object...");

calc = new Calculator(); // Arrange

}

@After

public void tearDown() {

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

calc = null;

}

@Test

public void testAddition() {

// Act

int result = calc.add(10, 5);

// Assert

assertEquals(15, result);

}

@Test

public void testSubtraction() {

// Act

int result = calc.subtract(10, 4);

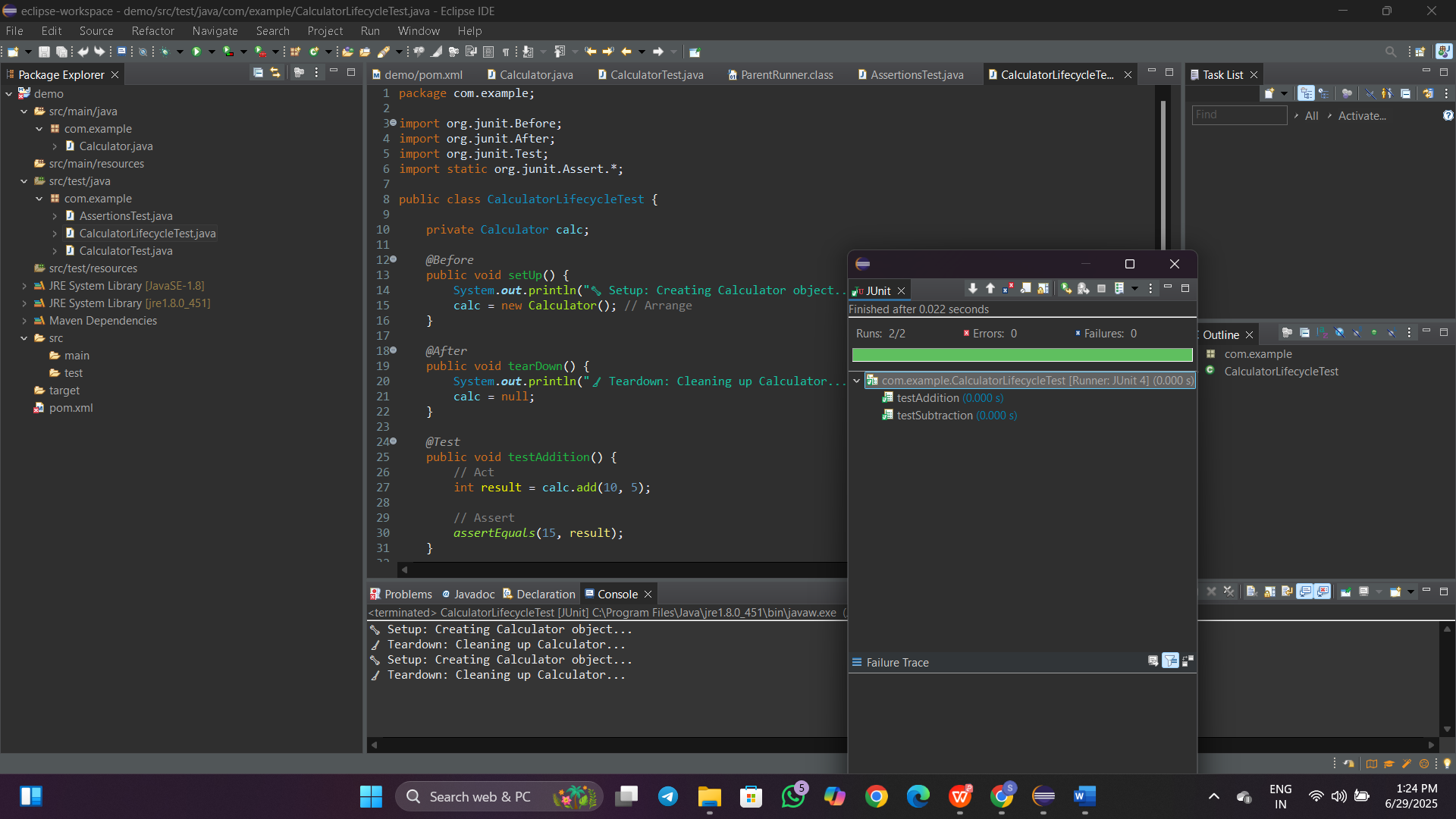
// Assert

assertEquals(6, result);

}

}

**Output:**



**Mockito Hands-On Exercises**

**Exercise 1: Mocking and Stubbing**

**Code:**

**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 org.junit.Test;

import static org.junit.Assert.\*;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testExternalApi() {

ExternalApi mockApi = mock(ExternalApi.class); // Create mock

when(mockApi.getData()).thenReturn("Mock Data"); // Stub return value

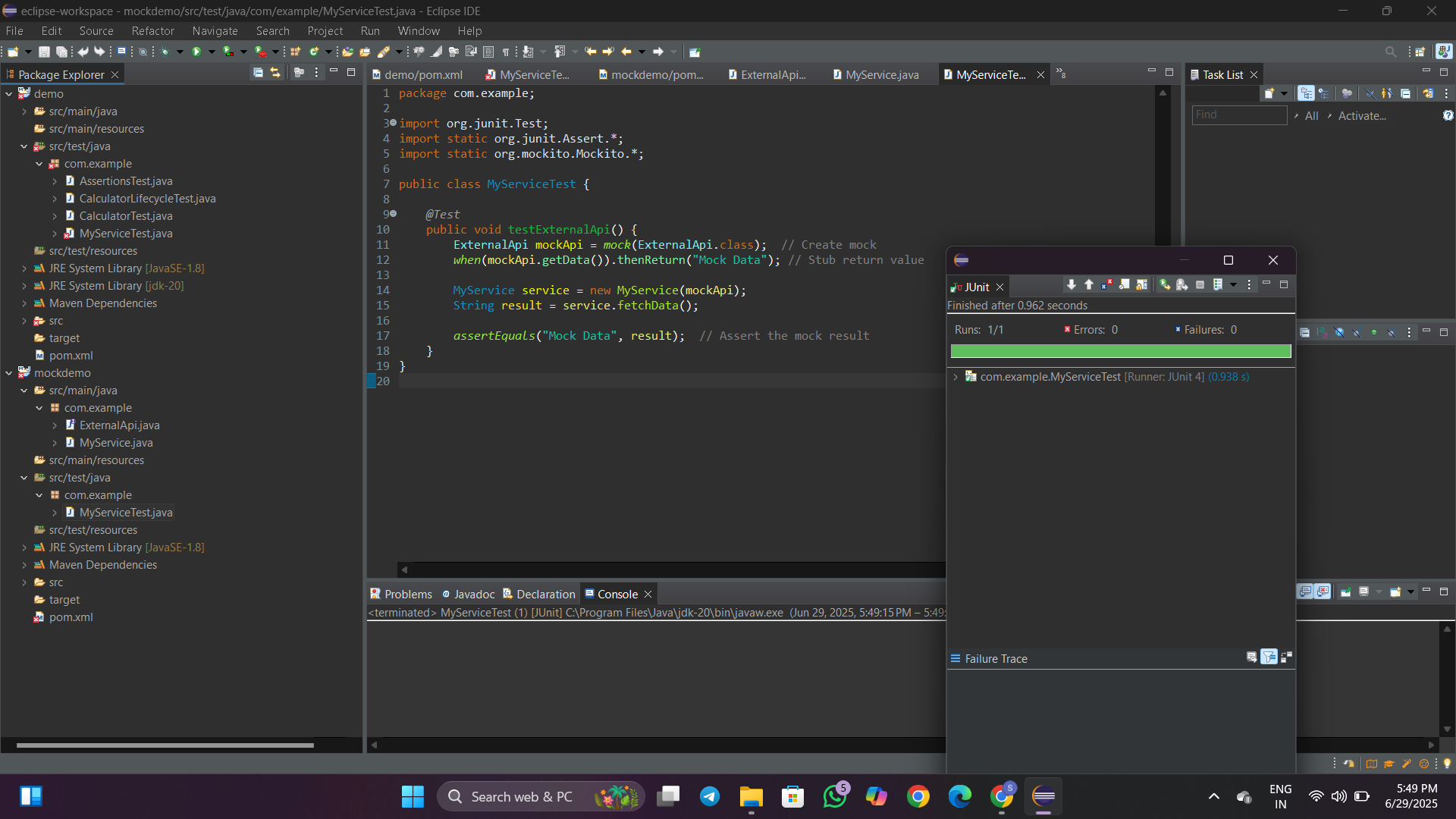
MyService service = new MyService(mockApi);

String result = service.fetchData();

assertEquals("Mock Data", result); // Assert the mock result

}}

**OUTPUT:**



**Exercise 2: Verifying Interactions**

**Code:**

**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 org.junit.Test;

import static org.mockito.Mockito.\*;

import static org.junit.Assert.\*;

public class MyServiceTest {

*@Test*

public void testVerifyInteraction() {

// Step 1: Create mock object

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

// Step 2: Pass mock to service and call method

MyService service = new MyService(mockApi);

service.fetchData(); // triggers mockApi.getData()

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

*verify*(mockApi).getData();

}}

**OUTPUT:** 