**JUnit Testing Exercises**

**Exercise 1: Setting Up JUnit**

**Calculator.java:**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**calculatorTest.java:**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class calculatorTest {

@Test

public void testAdd() {

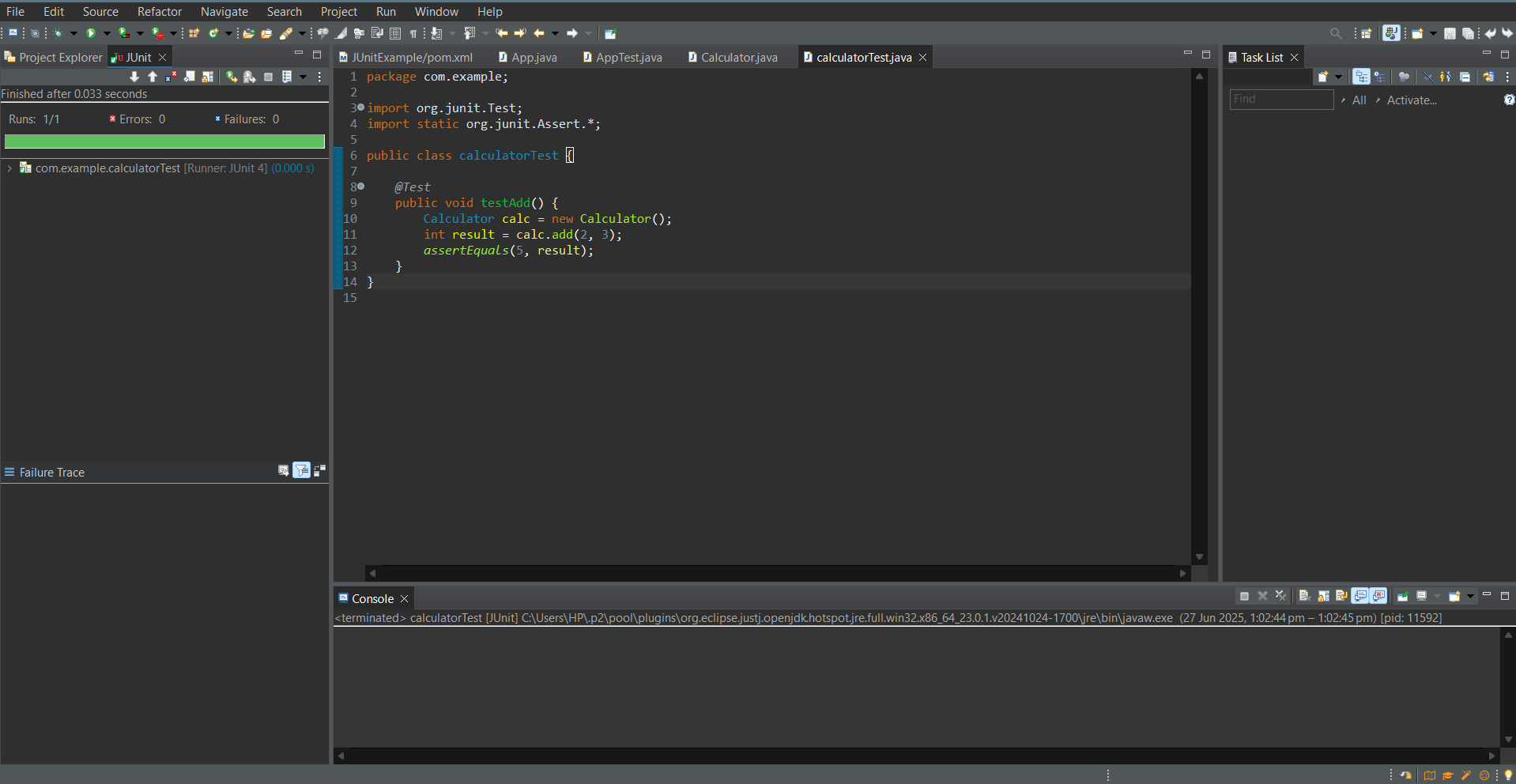
Calculator calc = new Calculator();

int result = calc.add(2, 3);

assertEquals(5, result);

}

}



**Exercise 3: Assertions in JUnit**

**AssertionsTest.java:**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

assertEquals(5, 2 + 3);

assertTrue(5 > 3);

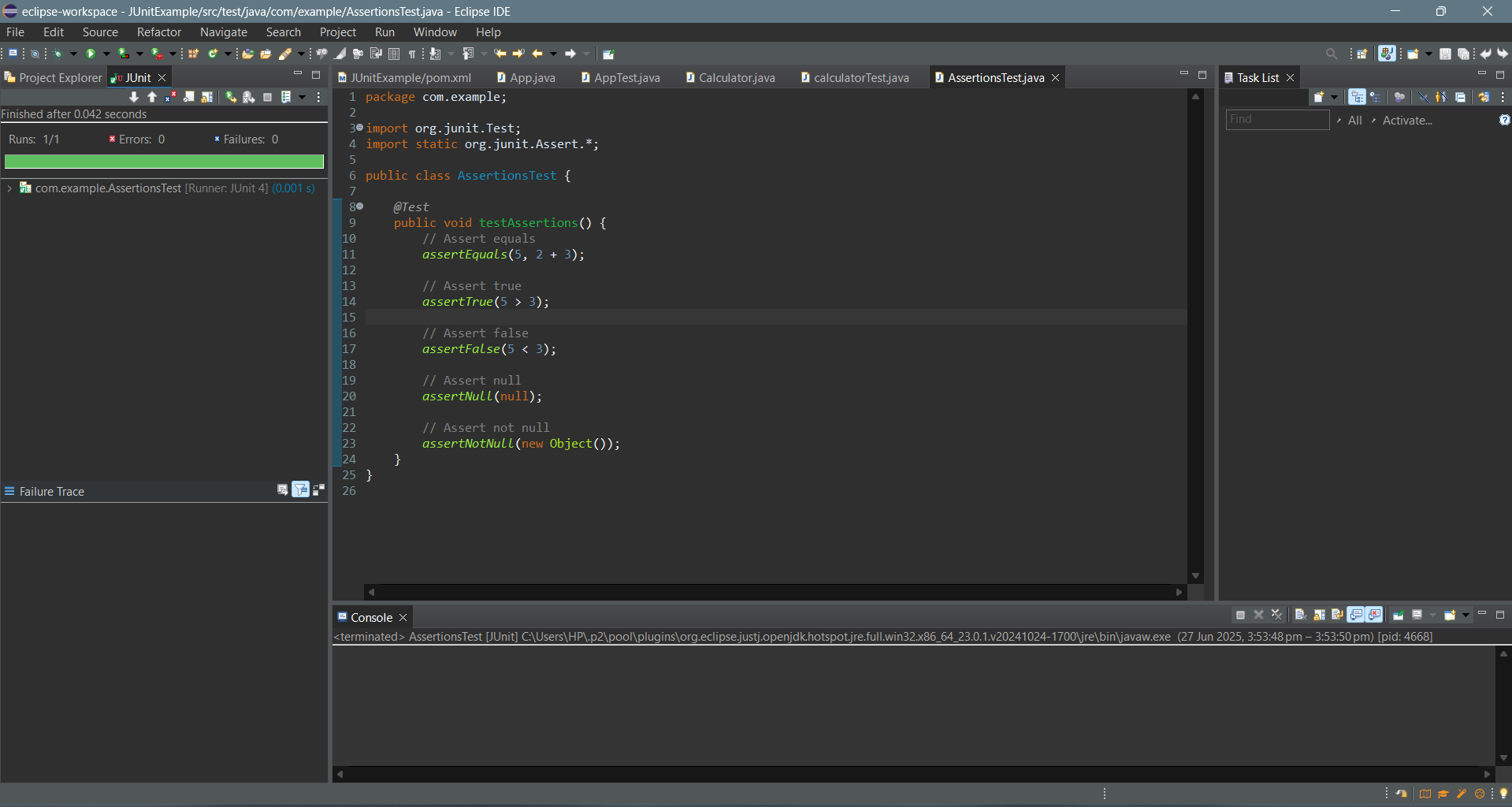
assertFalse(5 < 3);

assertNull(null);

assertNotNull(new Object());

}

}



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

**Calculator.java:**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int multiply(int a, int b) {

return a \* b;

}

}

**CalculatorTest.java:**

package com.example;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calc;

@Before

public void setUp() {

// Arrange: initialize Calculator before each test

System.out.println("Setting up...");

calc = new Calculator();

}

@After

public void tearDown() {

System.out.println("Tearing down...");

calc = null;

}

@Test

public void testAdd() {

// Act

int result = calc.add(2, 3);

assertEquals(5, result);

}

@Test

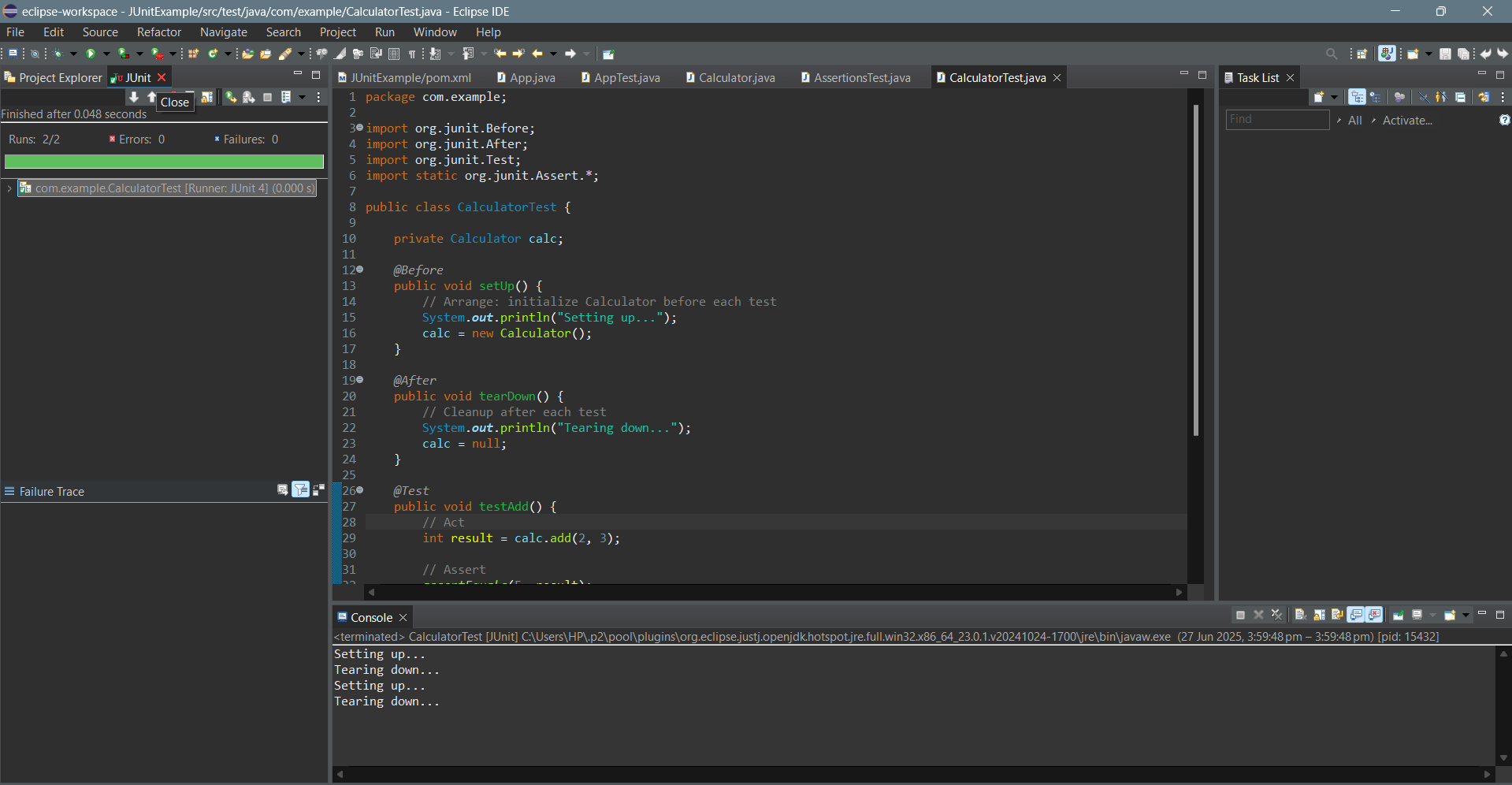
public void testMultiply() {

int result = calc.multiply(3, 4);

assertEquals(12, result);

}

}

****

**Mockito Hands-On Exercises**

**Exercise 1: Mocking and Stubbing**

**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.Assert.\*;

import org.junit.Test;

import org.mockito.Mockito;

public class MyServiceTest {

*@Test*

public void testExternalApi() {

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

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

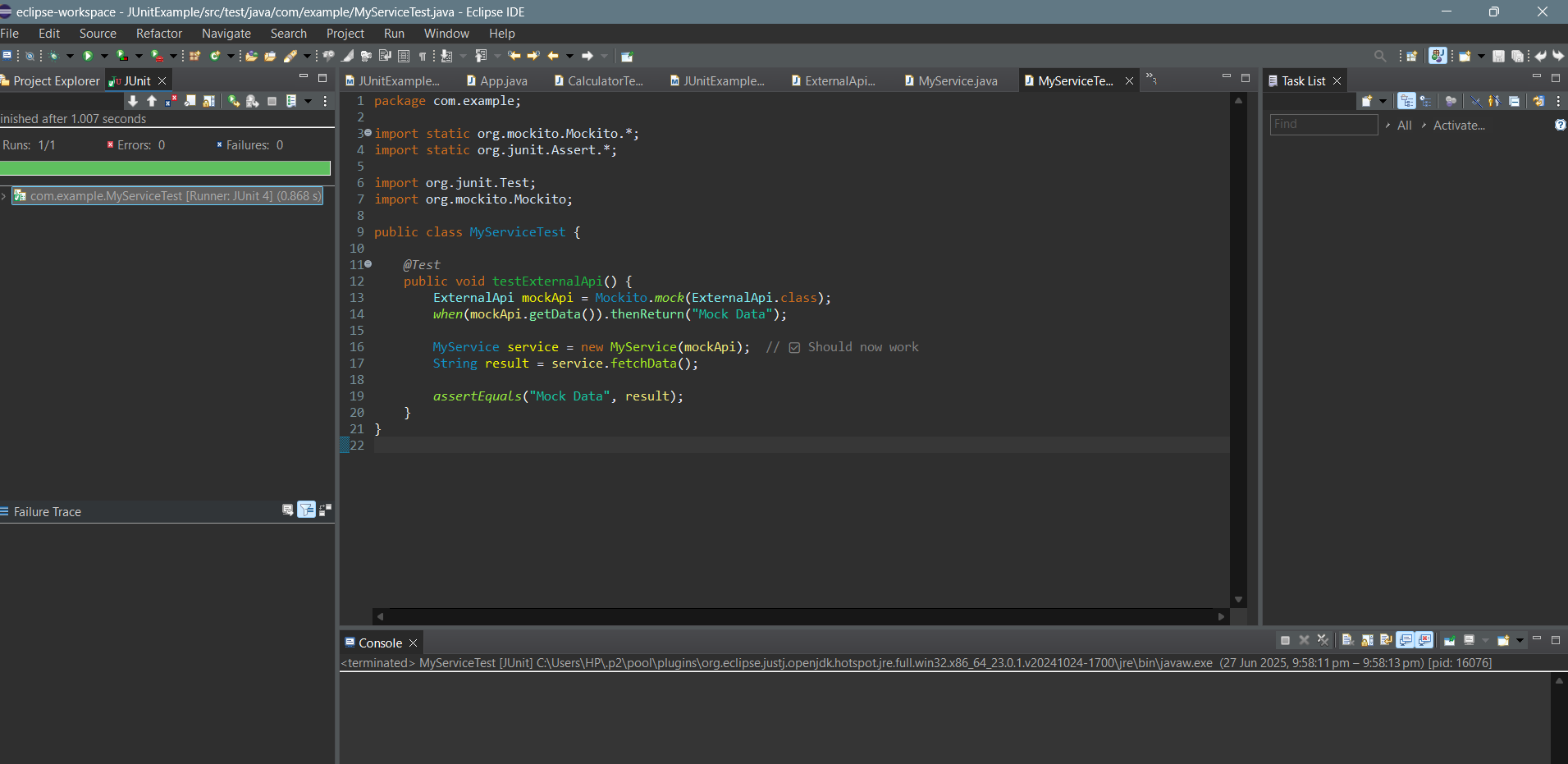
MyService service = new MyService(mockApi); // ✅ Should now work

String result = service.fetchData();

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

}

}



**Exercise 2: Verifying Interactions**

**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(); // This should be verified in test

}

}

**MyServiceTest.java:**

package com.example;

import static org.mockito.Mockito.\*;

import org.junit.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();

}

}

