**Exercise 1: Setting Up Junit**

**Code**

**Calculator.java**

package org.saswata;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorTest.java**

import org.junit.Test;

import org.saswata.Calculator;

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);

}

}

**Requirements**

Pom.xml

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

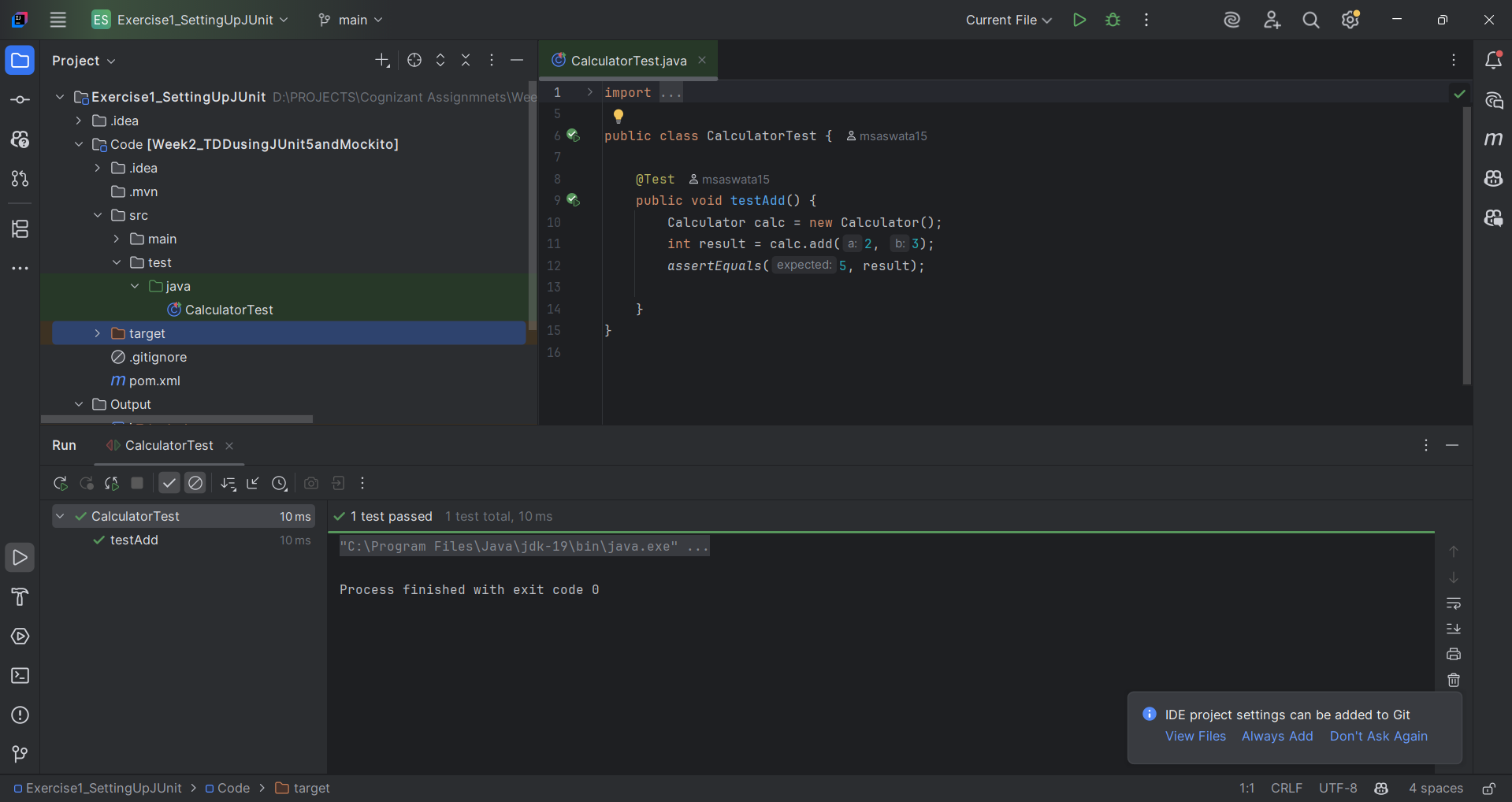
<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

**Output**



**Exercise 3: Assertions in JUnit**

**Code**

import org.junit.jupiter.api.Test;

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

public class AssertionsTest {

@Test

public void testAssertions() {

assertEquals(5, 2 + 3, "2 + 3 should equal 5");

assertArrayEquals(new int[] { 1, 2 }, new int[] { 1, 2 });

assertThrows(ArithmeticException.class, () -> {

int x = 1 / 0;

});

assertAll(

() -> assertTrue(10 > 5),

() -> assertNotNull("JUnit"));

assertTrue(5 > 3, "5 is greater than 3");

assertFalse(5 < 3, "5 is not less than 3");

assertNull(null, "Expected value to be null");

assertNotNull(new Object(), "Expected non-null object");

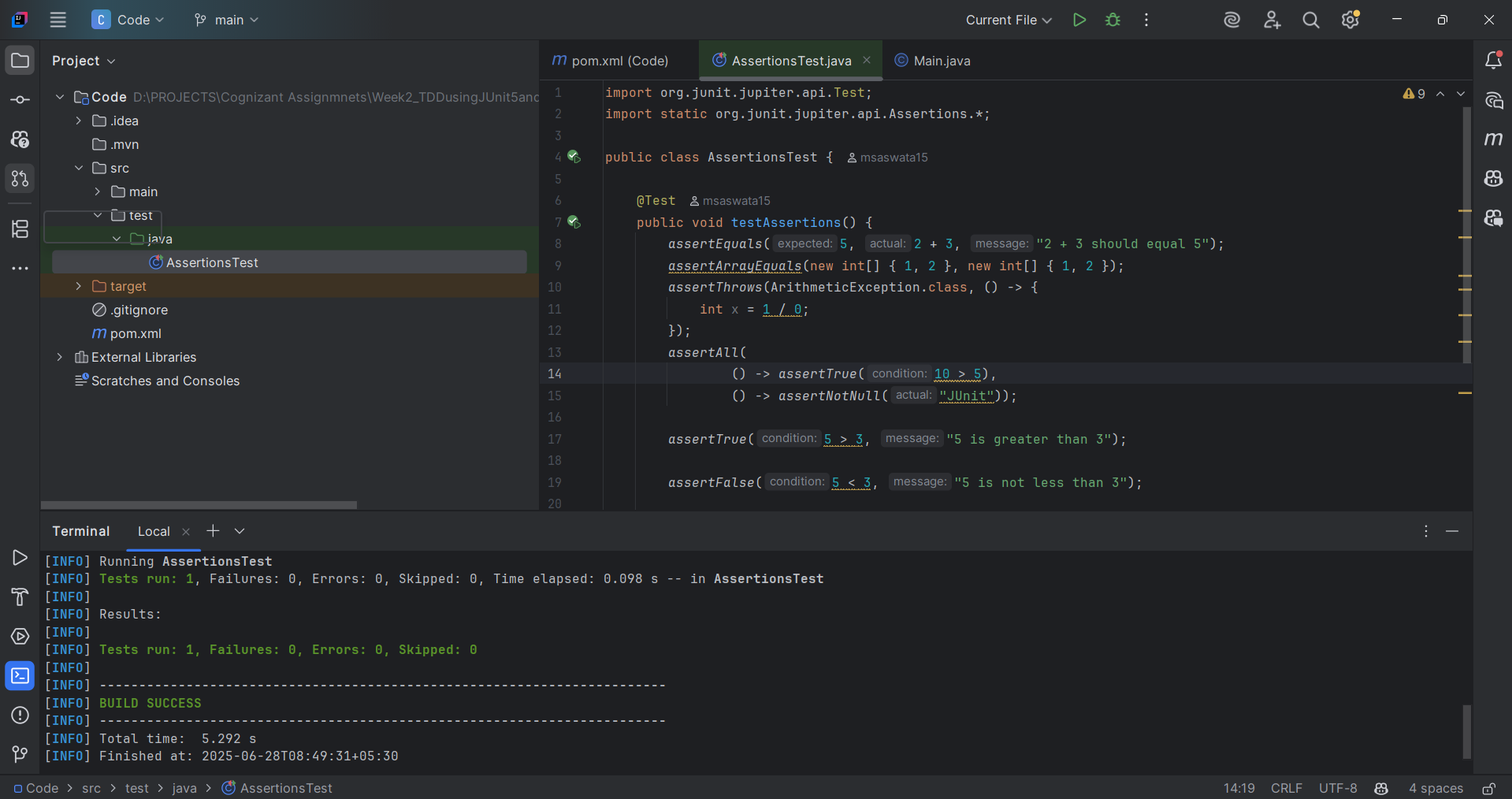
}

}

Pom.xml

<dependencies>  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter</artifactId>  
 <version>5.10.0</version>  
 <scope>test</scope>  
 </dependency>

Output:



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

**Calculator.java**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int multiply(int a, int b) {

return a \* b;

}

public void clearMemory() {

// Empty for now — can log or reset state in future

}

}

**CalculatorTest.java**

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

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

public class CalculatorTest {

private Calculator calculator;

@BeforeEach

void setUp() {

calculator = new Calculator();

System.out.println("Setup complete");

}

@AfterEach

void tearDown() {

calculator.clearMemory();

System.out.println("🧹 Teardown complete");

}

@Test

void testAddition() {

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

assertEquals(5, result, "2 + 3 should equal 5");

}

@Test

void testMultiplication() {

int result = calculator.multiply(4, 5);

assertEquals(20, result, "4 \* 5 should equal 20");

}

}

Pom.xml

<dependencies>

<!-- JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

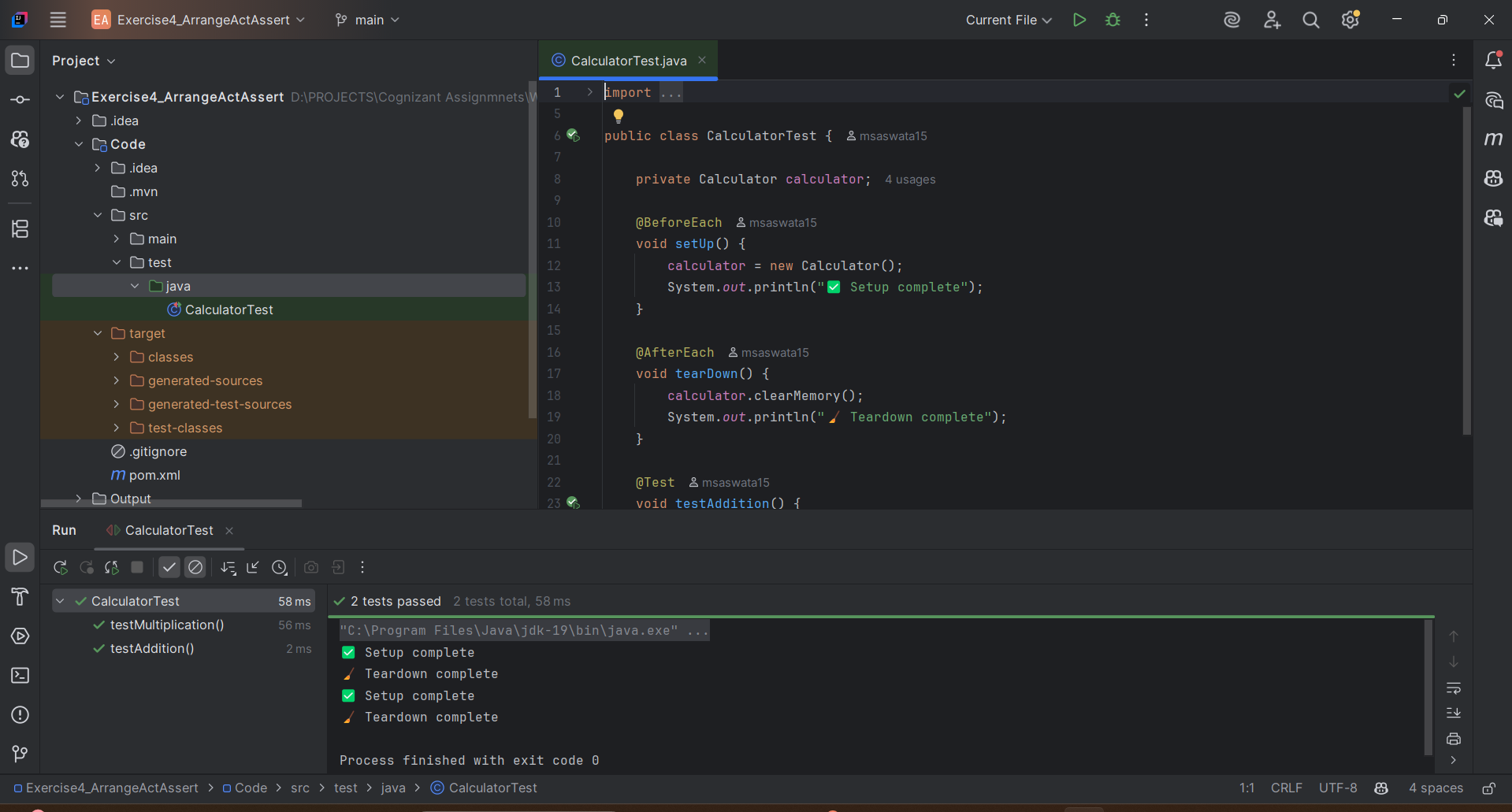
<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

</dependencies>

**Output:** ****

**Exercise 1: Mocking and Stubbing**

**ExternalApi.java**

package org.saswata;

public interface ExternalApi {

String getData();

}

**MyService.java**

package org.saswata;

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData(); // external call

}

}

**MyServiceTest.java**

import org.junit.jupiter.api.Test;

import org.saswata.ExternalApi;

import org.saswata.MyService;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

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

MyService service = new MyService(mockApi); // inject mock

service.fetchData(); // call method

verify(mockApi).getData(); // verify interaction

}

}

Pom.xml

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

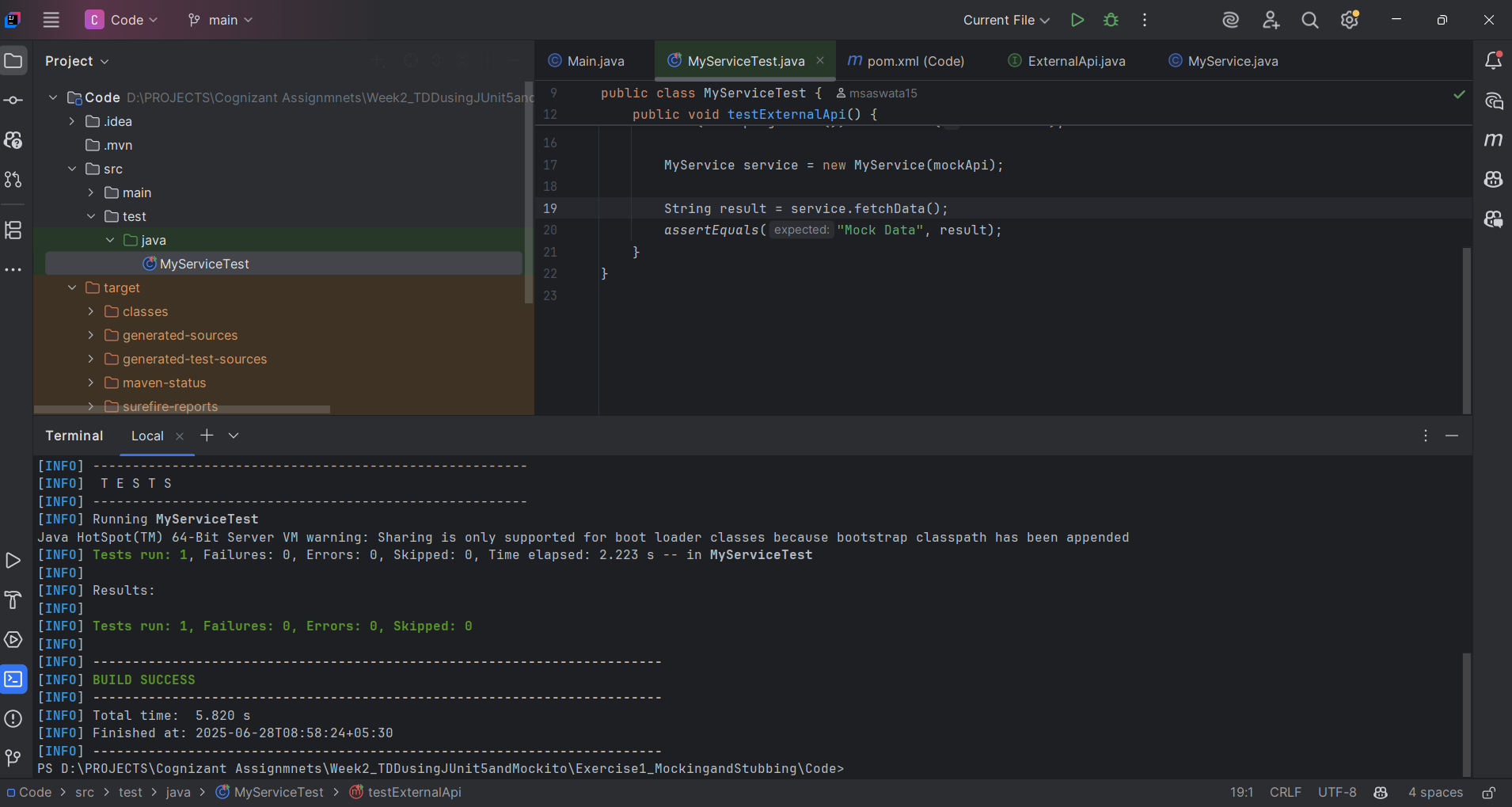
<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

**Output**



**Exercise 2: Verifying Interactions**

**ExternalApi.java**

package org.saswata;

public interface ExternalApi {

String getData();

}

A simple interface representing an external API.

**MyService.java**

package org.saswata;

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData(); // Calls the external API (mocked in tests)

}

}

A service class that depends on the ExternalApi.

**3. MyServiceTest.java**

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

import org.saswata.ExternalApi;

import org.saswata.MyService;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.when;

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);

}

}

This test:

* Mocks the ExternalApi interface.
* Stubs getData() to return "Mock Data".
* Verifies that MyService.fetchData() returns the expected mocked value.

Pom,xml

<dependencies>

<!-- JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.9.2</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

**Output**

