**TDD using JUnit5 and Mockito**

**Exercise 1: Setting Up JUnit**

**Pom.xml (JUnit setup)**

<?xml version="1.0" encoding="UTF-8"?>  
<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>Junitexample</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>24</maven.compiler.source>  
 <maven.compiler.target>24</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
 <dependencies>  
 <!-- JUnit Dependency -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
</project>

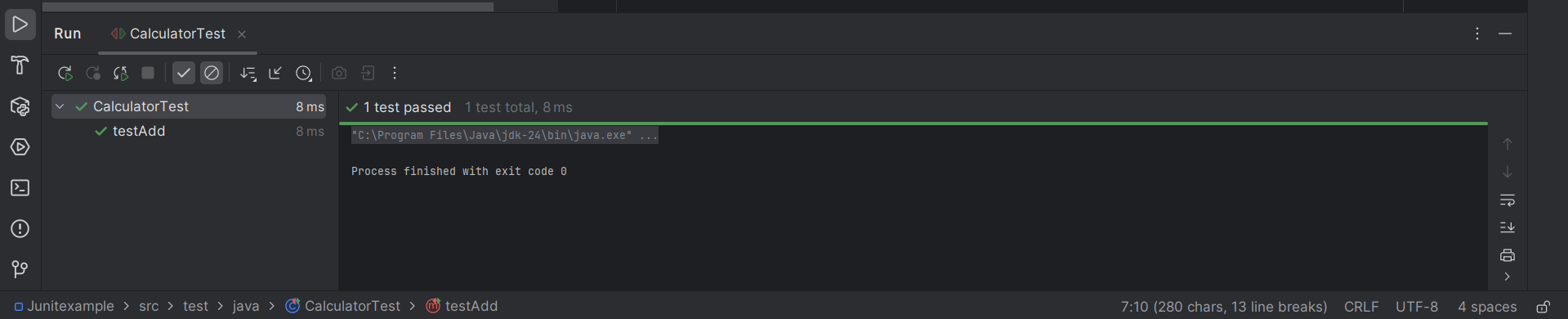
**Calculator.java**

package org.example;  
public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
}

**CalculatorTest.java**

import org.example.Calculator;  
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);  
 }  
}

**Output:**

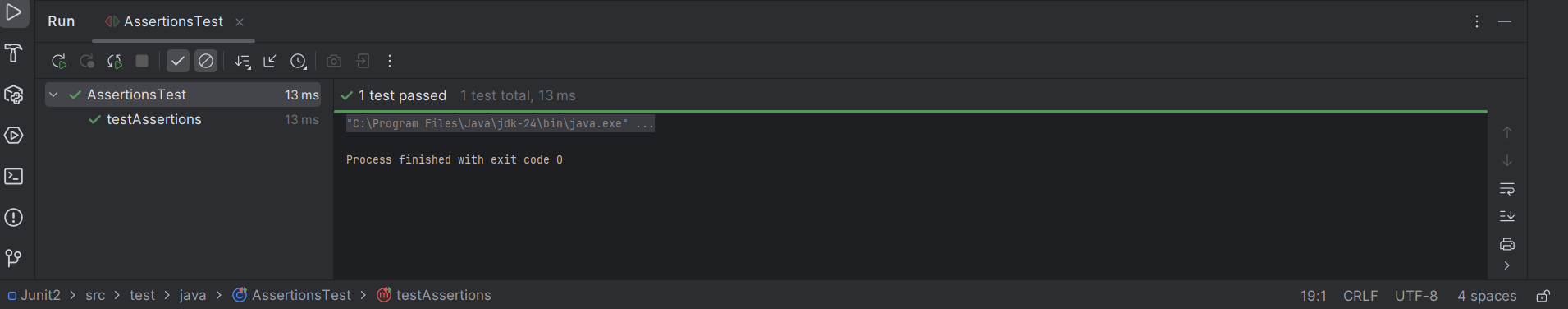
****

**Exercise 3: Assertions in JUnit**

**AssertionsTest.java**

import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class AssertionsTest {  
  
 @Test  
 public void testAssertions() {  
 // Assert that two values are equal  
 *assertEquals*(5, 2 + 3);  
  
 // Assert that a condition is true  
 *assertTrue*(5 > 3);  
  
 // Assert that a condition is false  
 *assertFalse*(5 < 3);  
  
 // Assert that the object is null  
 *assertNull*(null);  
  
 // Assert that the object is not null  
 *assertNotNull*(new Object());  
 }  
}

**Output:**

****

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

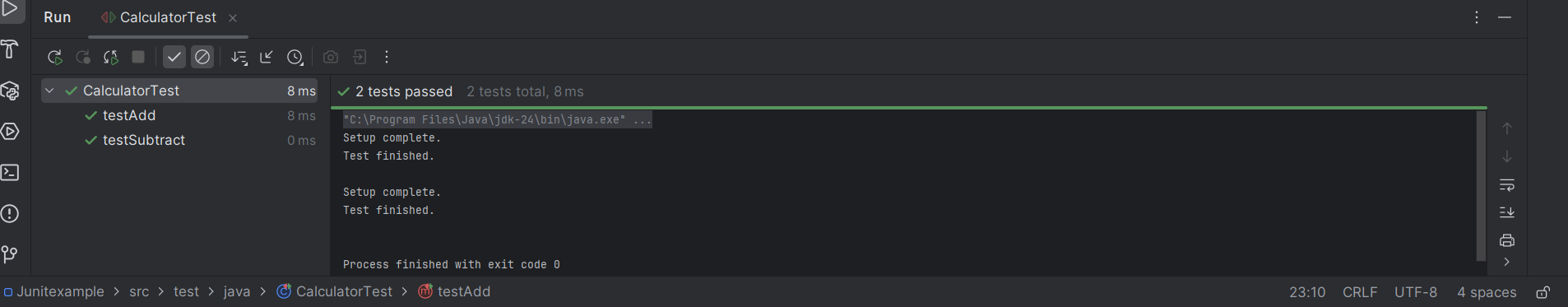
**Calculator.java**

public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
  
 public int subtract(int a, int b) {  
 return a - b;  
 }  
}

**CalculatorTest.java**

import org.junit.Before;  
import org.junit.After;  
import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class CalculatorTest {  
  
 private Calculator calc;  
  
 // Arrange (Setup)  
 @Before  
 public void setUp() {  
 calc = new Calculator(); // Create Calculator before each test  
 System.*out*.println("Setup complete.");  
 }  
  
 // Teardown  
 @After  
 public void tearDown() {  
 System.*out*.println("Test finished.\n");  
 }  
  
 @Test  
 public void testAdd() {  
 // Arrange  
 int a = 5;  
 int b = 3;  
  
 // Act  
 int result = calc.add(a, b);  
  
 // Assert  
 *assertEquals*(8, result);  
 }  
  
 @Test  
 public void testSubtract() {  
 // Arrange  
 int a = 10;  
 int b = 4;  
  
 // Act  
 int result = calc.subtract(a, b);  
  
 // Assert  
 *assertEquals*(6, result);  
 }  
}

**Output:**

****

**Mockito exercises**

**Exercise 1: Mocking and Stubbing**

**Pom.xml**

<dependencies>

<!-- JUnit 5 -->

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</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>

**ExternalApi.java**

package org.example;  
  
public interface ExternalApi {  
 String getData();  
}

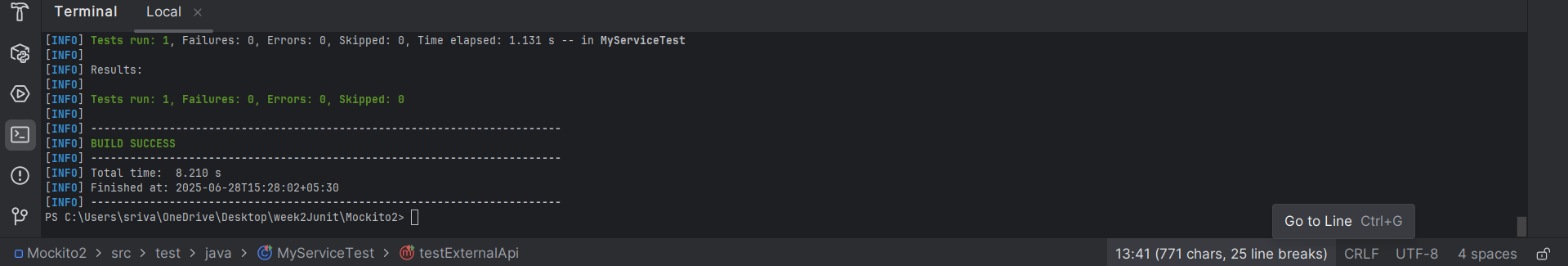
**MyService.java**

package org.example;  
  
public class MyService {  
 private ExternalApi api;  
  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
  
 public String fetchData() {  
 return api.getData();  
 }  
}

**MyServiceTest.java**

import org.example.ExternalApi;  
import org.example.MyService;  
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 a mock object  
 ExternalApi mockApi = Mockito.*mock*(ExternalApi.class);  
  
 // Step 2: Stub the method to return predefined value  
 *when*(mockApi.getData()).thenReturn("Mock Data");  
  
 // Step 3: Inject mock into service and call the method  
 MyService service = new MyService(mockApi);  
 String result = service.fetchData();  
  
 // Step 4: Assert the result  
 *assertEquals*("Mock Data", result);  
 }  
}

**Output:**

****

**Exercise 2: Verifying Interactions**

**ExternalApi.java**

package org.example;  
  
public interface ExternalApi {  
 String getData();  
}

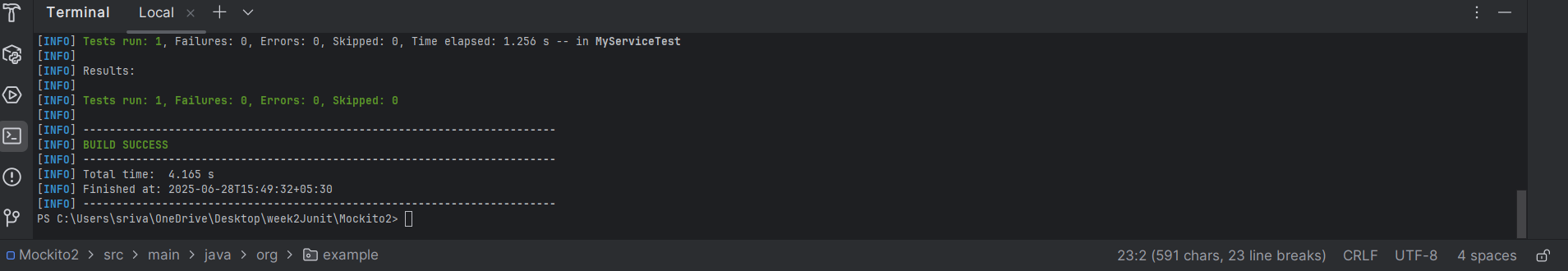
**MySevice.java**

package org.example;  
  
  
public class MyService {  
 private ExternalApi api;  
  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
  
 public void fetchData() {  
 api.getData(); // <- This method should be verified in the test  
 }  
}

**MyServiceTest.java**

import org.example.ExternalApi;  
import org.example.MyService;  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
  
public class MyServiceTest {  
  
 @Test  
 public void testVerifyInteraction() {  
 // Step 1: Create a mock object  
 ExternalApi mockApi = *mock*(ExternalApi.class);  
  
 // Step 2: Inject mock into the service  
 MyService service = new MyService(mockApi);  
  
 // Step 3: Call the method we want to test  
 service.fetchData();  
  
 // Step 4: Verify that getData() was called  
 *verify*(mockApi).getData();  
 }  
}

**Output:**

****