**TDD USING JUNIT5 AND MOCKITO**

**Exercise 1: Setting Up JUnit**

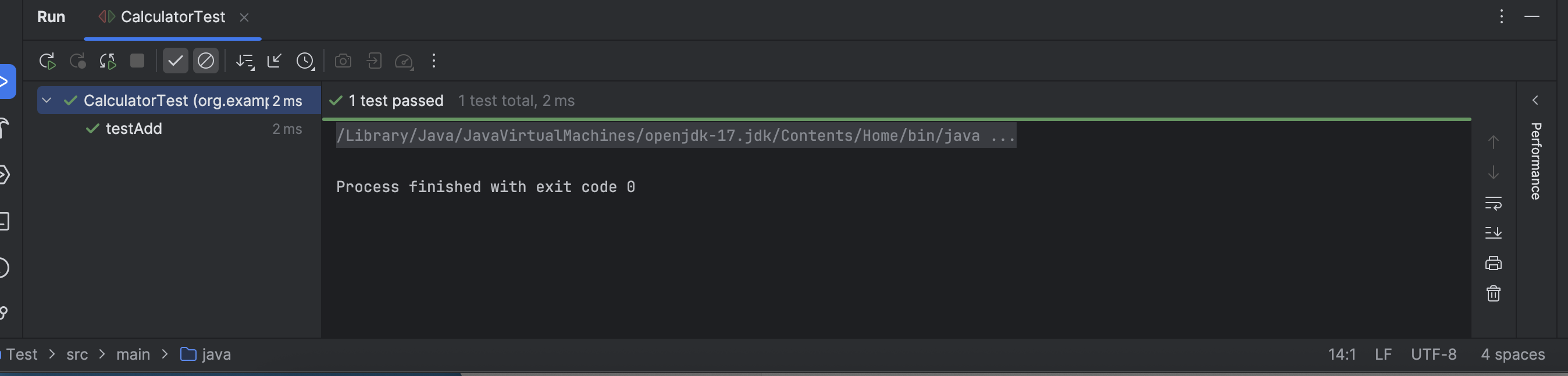
**Calculator.java**

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

**CalculatorTest.java**

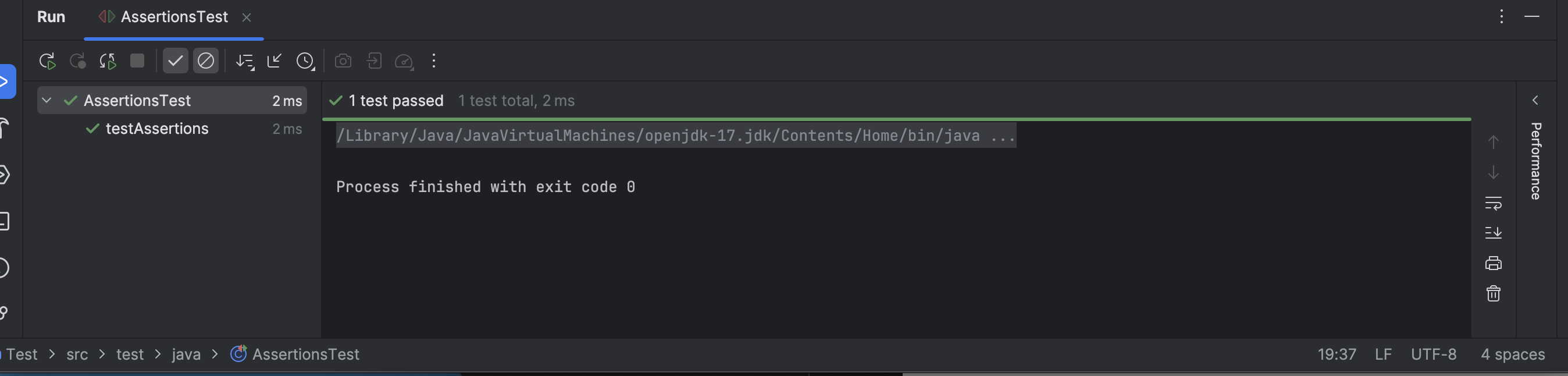
import org.example.Calculator;  
import org.junit.Assert;  
import org.junit.Test;  
  
public class CalculatorTest {  
  
 @Test  
 public void testAdd() {  
 Calculator calc = new Calculator();  
 int result = calc.add(2, 3);  
 Assert.*assertEquals*(5, result);  
 }  
}

**OUTPUT:**

****

**Exercise 3: Assertions in Junit**

**OUTPUT:**

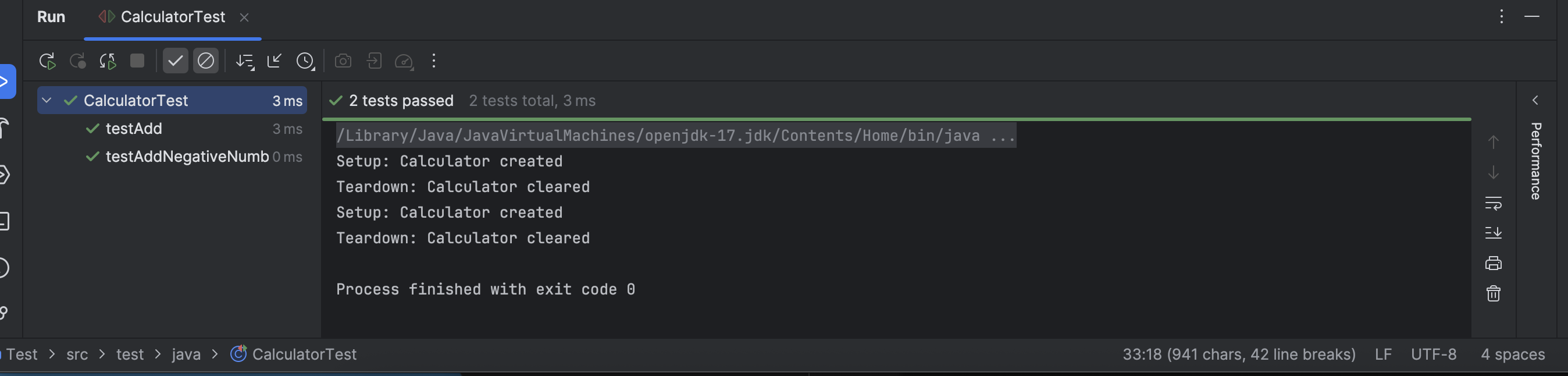
****

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

**CalculatorTest.java**

import org.example.Calculator;  
import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class CalculatorTest {  
  
 private Calculator calculator;  
  
 // 🔧 Setup: runs before every test  
 @Before  
 public void setUp() {  
 calculator = new Calculator(); // Arrange  
 System.*out*.println("Setup: Calculator created");  
 }  
  
 // 🧹 Teardown: runs after every test  
 @After  
 public void tearDown() {  
 calculator = null;  
 System.*out*.println("Teardown: Calculator cleared");  
 }  
  
 // ✅ Test using Arrange-Act-Assert  
 @Test  
 public void testAdd() {  
 // Arrange done in @Before  
  
 // Act  
 int result = calculator.add(2, 3);  
  
 // Assert  
 *assertEquals*(5, result);  
 }  
  
 @Test  
 public void testAddNegativeNumbers() {  
 int result = calculator.add(-2, -3);  
 *assertEquals*(-5, result);  
 }  
}

OUTPUT:



**MOCKITO**

**Exercise 1: Mocking and Stubbing & Exercise 2: Verifying Interactions**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

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

MyService service = new MyService(mockApi);

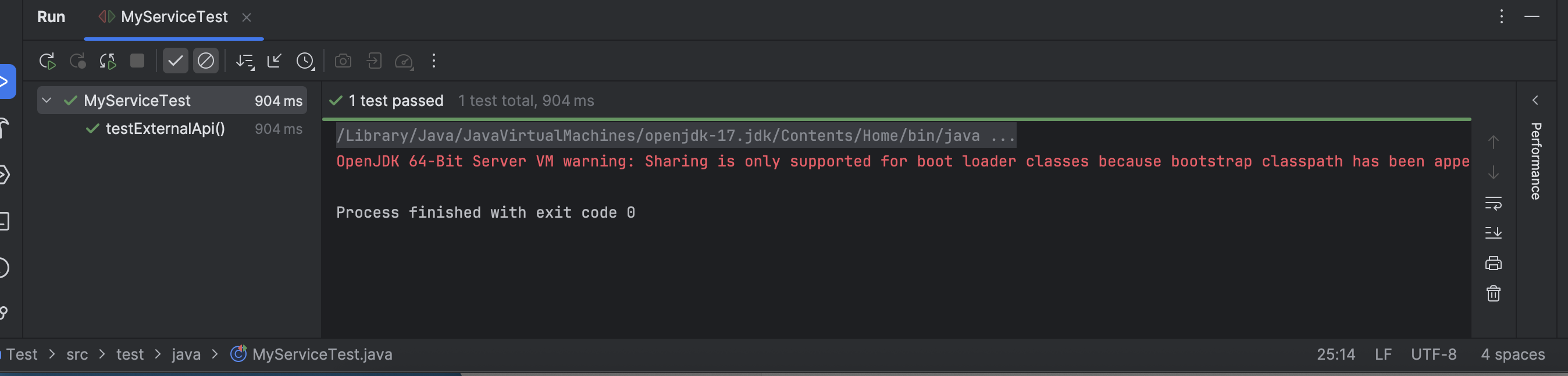
service.fetchData();

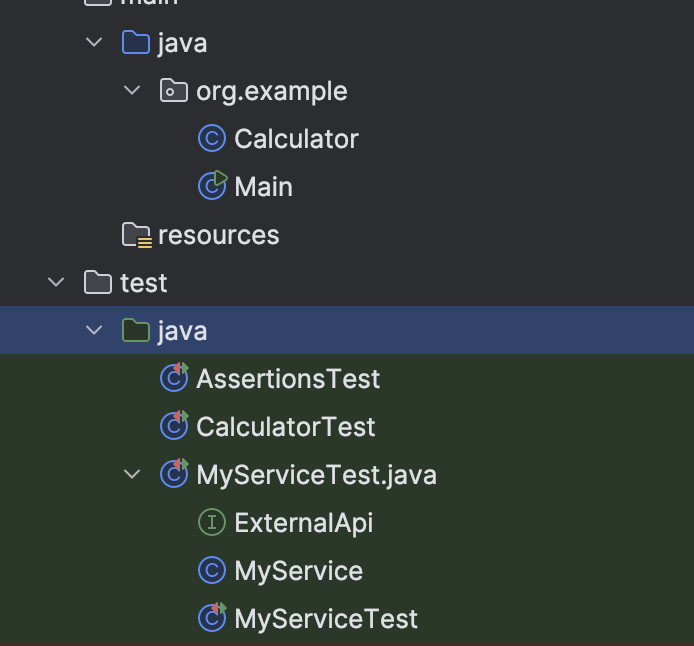
verify(mockApi).getData();

}

}

**OUTPUT:**

****

****

**SLF4J logging framework**

**Exercise 1: Logging Error Messages and Warning Levels**

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class LoggingExample {

private static final Logger logger = LoggerFactory.getLogger(LoggingExample.class);

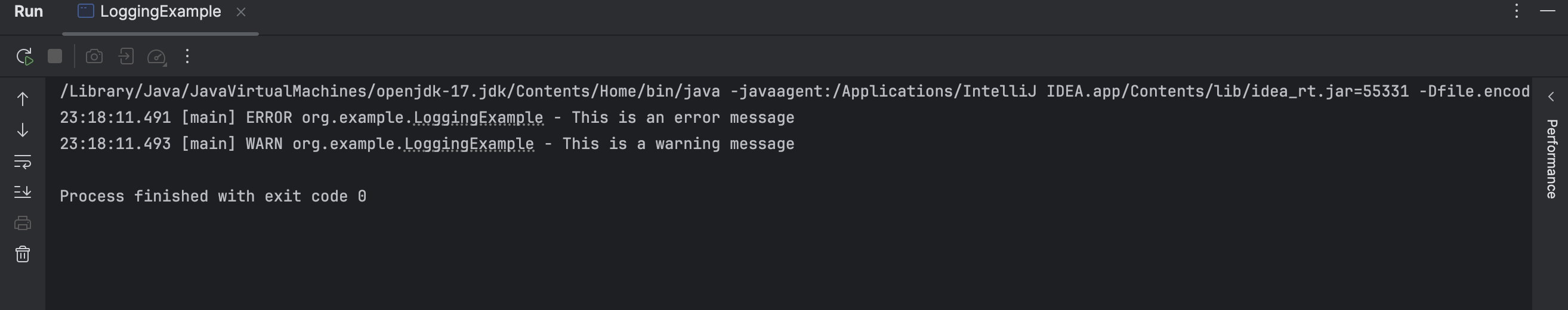
public static void main(String[] args) {

logger.error("This is an error message");

logger.warn("This is a warning message");

}

}

****