**Exercise 1: Setting Up JUnit**

Created Maven Project then added JUnit Dependency into pom.xml.

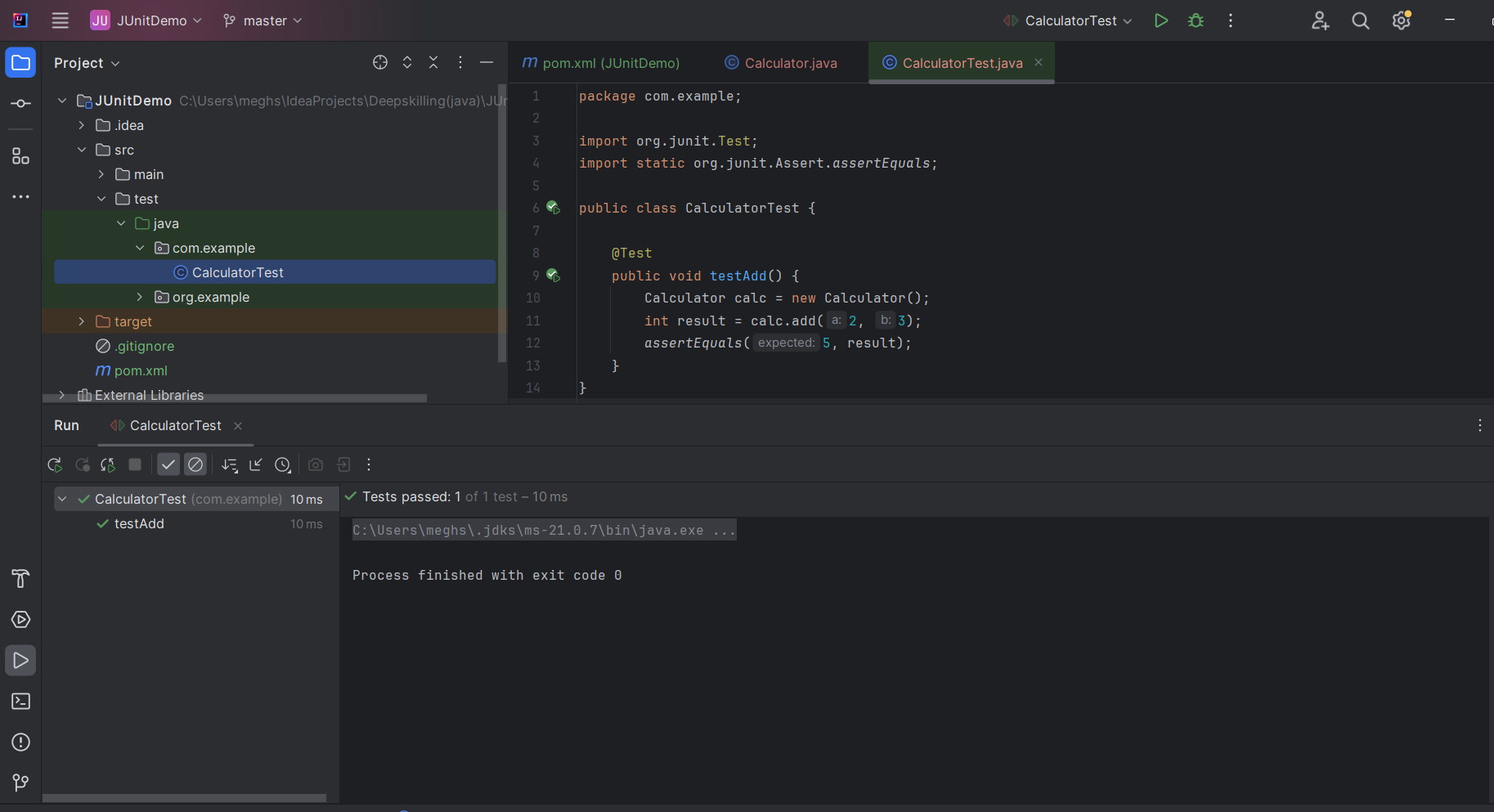
project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>JUnitDemo</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>JUnitDemo</name>  
 <url>http://maven.apache.org</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
</project>

Then created simple class Calculator.

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

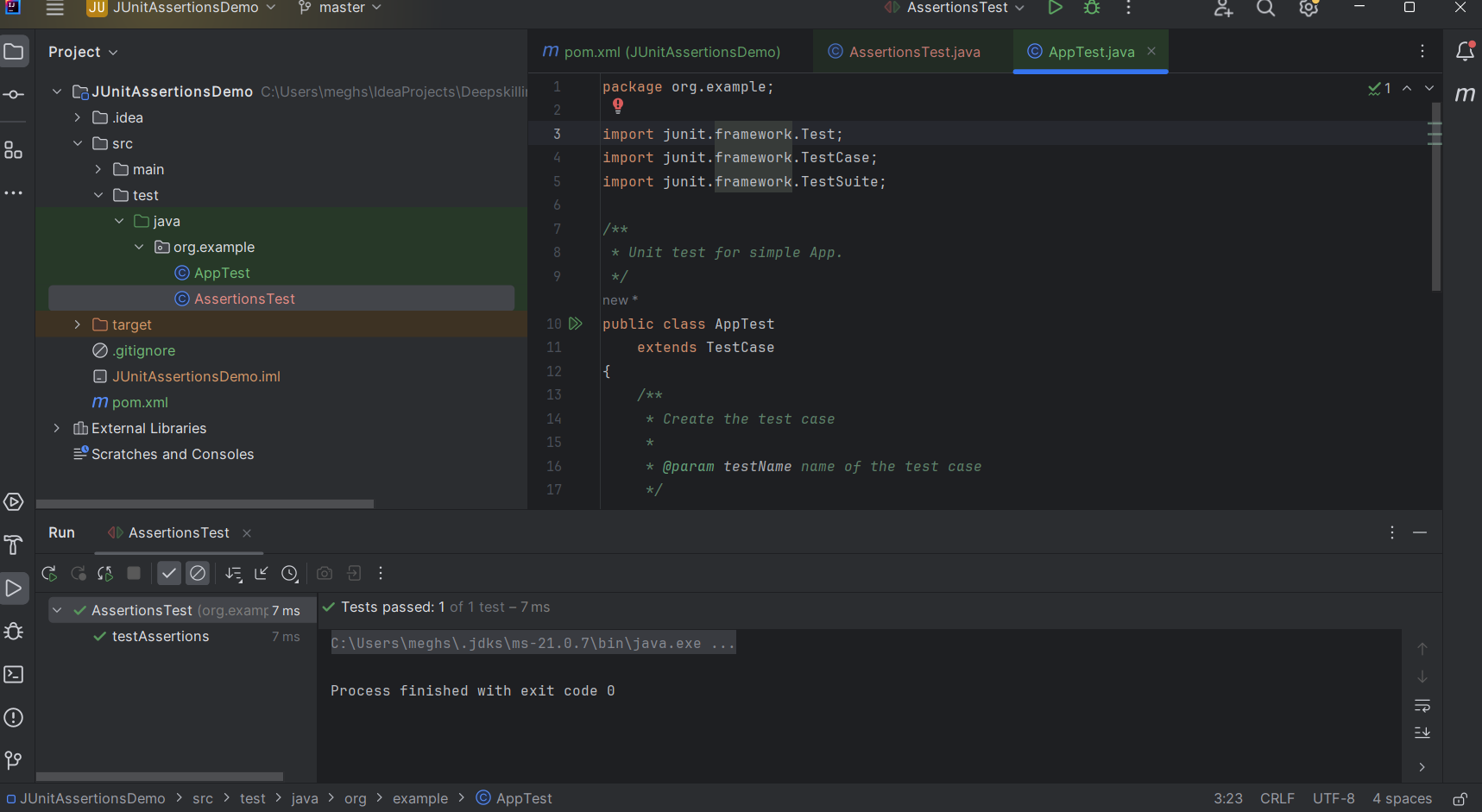
Then created TestClass

package com.example;  
  
import org.junit.Test;  
import static org.junit.Assert.*assertEquals*;  
  
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**

package org.example;  
  
import junit.framework.Test;  
import junit.framework.TestCase;  
import junit.framework.TestSuite;  
  
public class AppTest   
 extends TestCase  
{  
 */\*\*  
 \* Create the test case  
 \*  
 \* @param testName name of the test case  
 \*/* public AppTest( String testName )  
 {  
 super( testName );  
 }  
  
 */\*\*  
 \* @return the suite of tests being tested  
 \*/* public static Test suite()  
 {  
 return new TestSuite( AppTest.class );  
 }  
  
 */\*\*  
 \* Rigourous Test :-)  
 \*/* public void testApp()  
 {  
 *assertTrue*( true );  
 }  
}



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

**Teardown Methods in JUnit**

AAA Pattern?

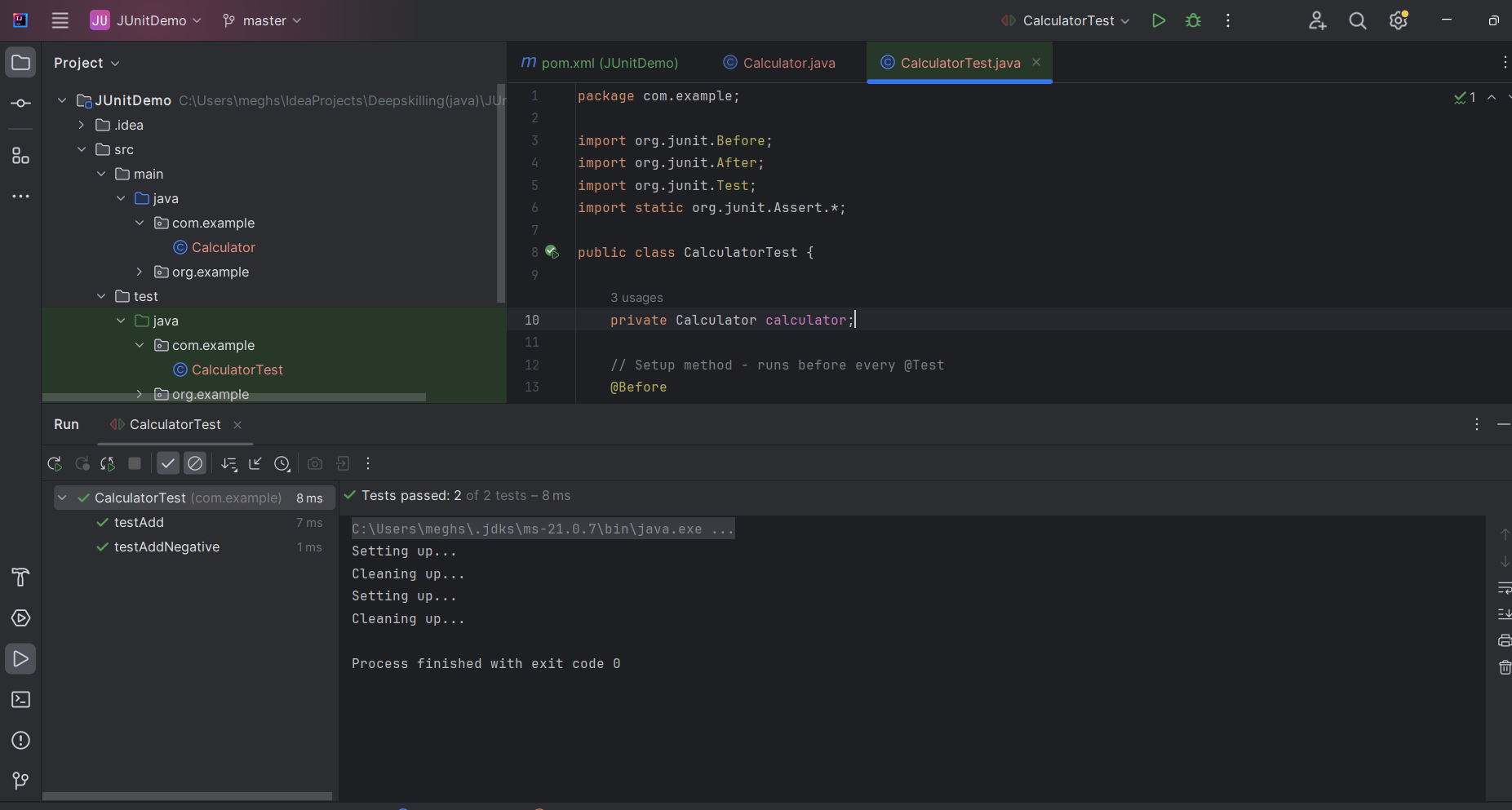
**Arrange:** Set up data and objects  
**Act:** Call the method you want to test  
**Assert:** Verify the result/output

**Calculator class: (code)**

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

**Then I created JUnit test class i.e**

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 calculator;  
  
 // Setup method - runs before every @Test  
 @Before  
 public void setUp() {  
 calculator = new Calculator(); // Arrange  
 System.*out*.println("Setting up...");  
 }  
  
 // Teardown method - runs after every @Test  
 @After  
 public void tearDown() {  
 System.*out*.println("Cleaning up...");  
 }  
  
 @Test  
 public void testAdd() {  
 // Act  
 int result = calculator.add(2, 3);  
  
 // Assert  
 *assertEquals*(5, result);  
 }  
  
 @Test  
 public void testAddNegative() {  
 // Act  
 int result = calculator.add(-2, -3);  
  
 // Assert  
 assertEquals(-5, result);  
 }  
}

****

**Exercise 1: Mocking and Stubbing**

**Pom.xml(MockitoDemo)**

<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>com.example</groupId>  
 <artifactId>MockitoDemo</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>MockitoDemo</name>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <repositories>  
 <repository>  
 <id>central</id>  
 <url>https://repo.maven.apache.org/maven2</url>  
 </repository>  
 </repositories>  
  
 <dependencies>  
 <!-- JUnit 5 -->  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter</artifactId>  
 <version>5.10.2</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- Mockito -->  
 <dependency>  
 <groupId>org.mockito</groupId>  
 <artifactId>mockito-core</artifactId>  
 <version>5.12.0</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
</project>

**Created ExternalApi interface(code)**

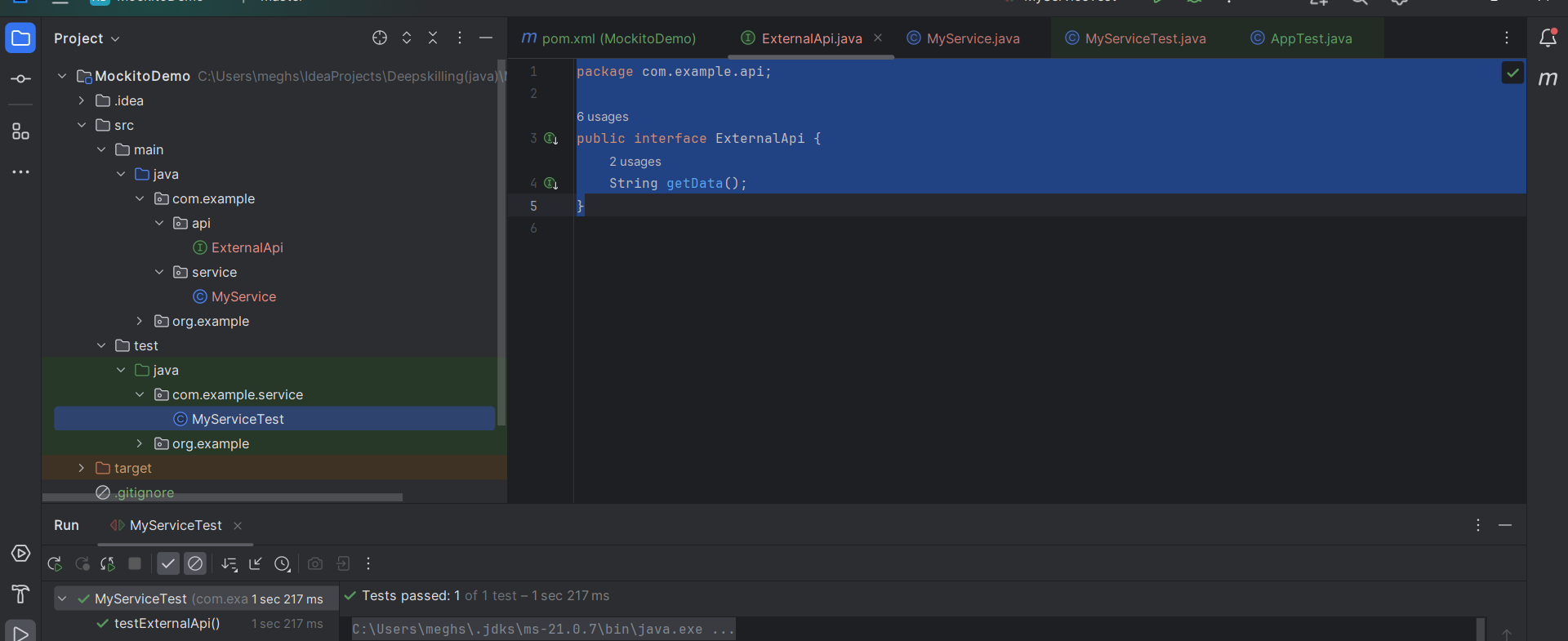
package com.example.api;  
  
public interface ExternalApi {  
 String getData();  
}

**Created MyService class(code)**

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

**Created MyServiceTest class(code)**

package com.example.service;  
  
import com.example.api.ExternalApi;  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class MyServiceTest {  
  
 @Test  
 public void testExternalApi() {  
 // Mock object creation  
 ExternalApi mockApi = *mock*(ExternalApi.class);  
  
 // Stubbing the method  
 *when*(mockApi.getData()).thenReturn("Mock Data");  
  
 // Injecting mock into service  
 MyService service = new MyService(mockApi);  
  
 // Act & Assert  
 String result = service.fetchData();  
 assertEquals("Mock Data", result);  
 }  
}



**Exercise 2: Verifying Interactions**

pom.xml same as used before , in this stage we need to verify the interactions so,

**Created ExternalApi interface(code)**

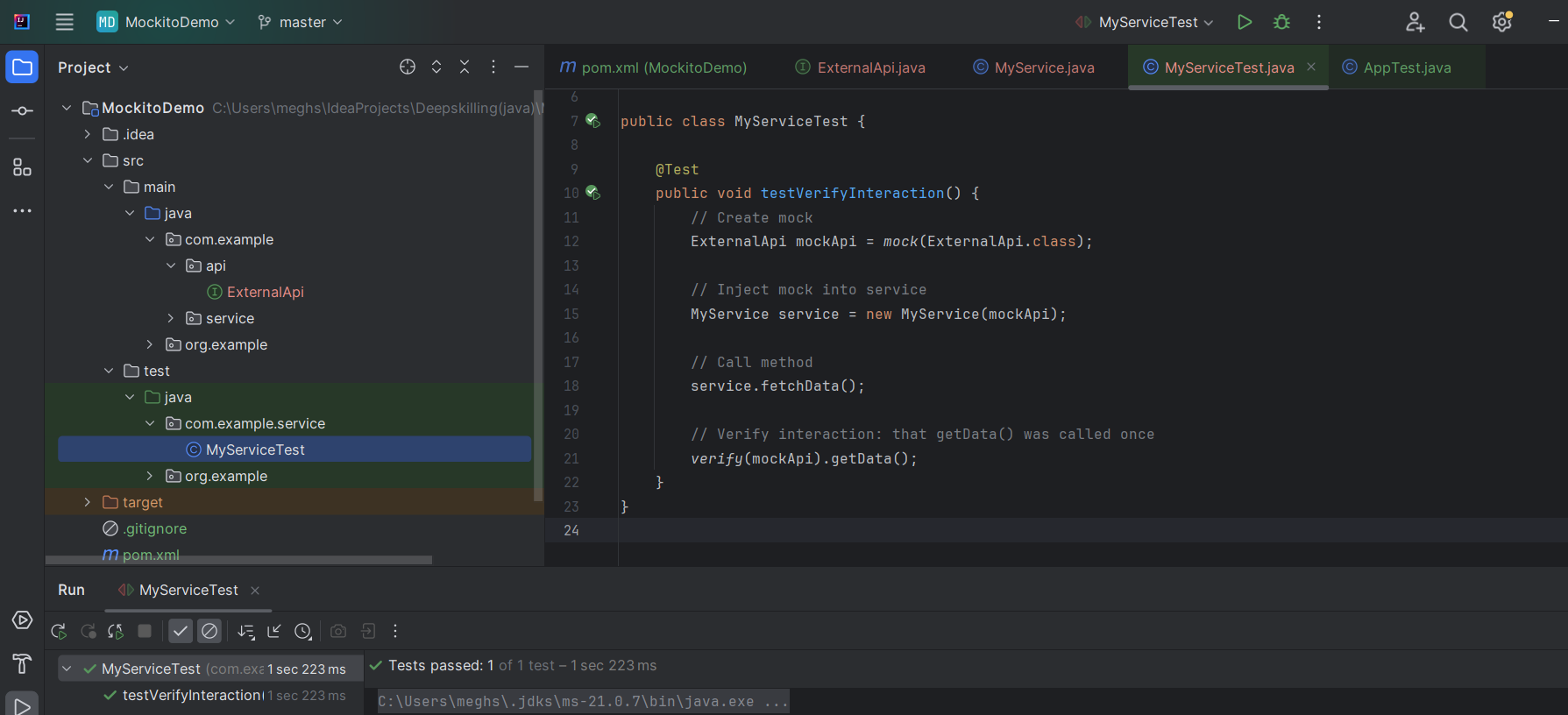
package com.example.api;  
  
public interface ExternalApi {  
 String getData();  
}

**Created MyService class(code)**

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

**Created Test Class MyServiceTest(code)**

package com.example.service;  
  
import com.example.api.ExternalApi;  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
  
public class MyServiceTest {  
  
 @Test  
 public void testVerifyInteraction() {  
 // Create mock  
 ExternalApi mockApi = *mock*(ExternalApi.class);  
  
 // Inject mock into service  
 MyService service = new MyService(mockApi);  
  
 // Call method  
 service.fetchData();  
  
 // Verify interaction: that getData() was called once  
 *verify*(mockApi).getData();  
 }  
}

****

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

package com.example;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
public class LoggingExample {  
 // Create logger instance for this class  
 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");  
 logger.info("This is an info message");  
 logger.debug("This is a debug message");  
 }  
}

