# Exercise – 1: Mocking and Stubbing

**Scenario:**

You need to test a service that depends on an external API. Use **Mockito** to mock the external API and stub its methods.

**Steps:**

1. **Create a mock object** for the external API
2. **Stub the methods** to return predefined values
3. **Write a test case** that uses the mock object
4. **Create a Java Class for the External API and the Service**

## ExternalApi.java

package com.example;

public interface ExternalApi { String getData();

}

## MyService.java

package com.example;

public class MyService {

private final ExternalApi externalApi;

public MyService(ExternalApi externalApi) { this.externalApi = externalApi;

}

public String fetchData() { return externalApi.getData();

}

}

1. **Add Mockito and JUnit Dependencies to pom.xml**

<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>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

1. **Create the JUnit Test Class using Mocking and Stubbing**

package com.example;

import org.junit.jupiter.api.Test; import org.mockito.Mockito;

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);

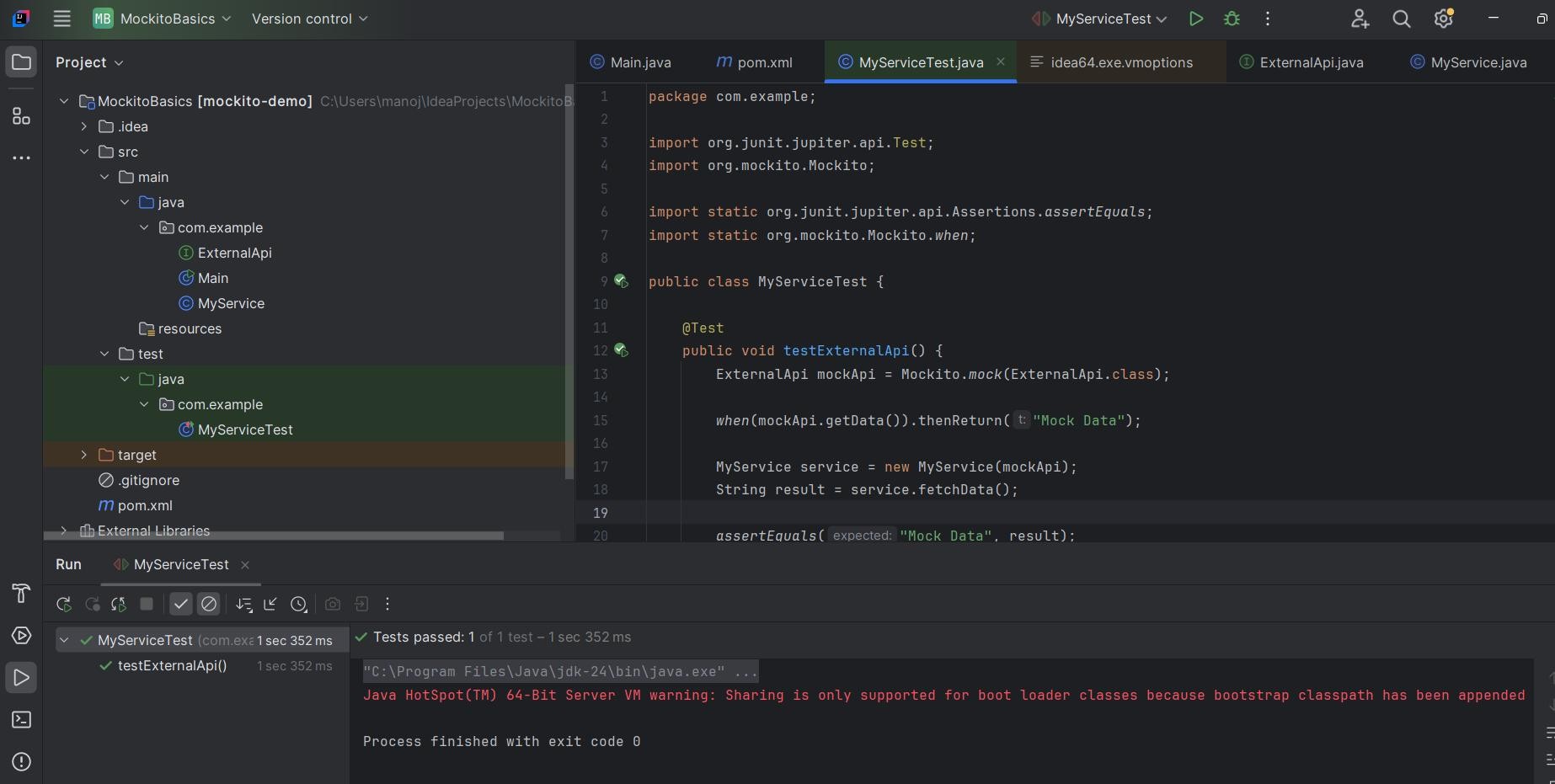
*when*(mockApi.getData()).thenReturn("Mock Data"); MyService service = new MyService(mockApi);

String result = service.fetchData();

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

}

}

**Output:**

**Scenario:**

# Exercise – 2: Verifying Interactions

You need to ensure that a method is called with specific arguments using **Mockito**.

**Steps:**

1. **Create a mock object**
2. **Call the method** with specific arguments
3. **Verify the interaction**
4. **Java Classes Setup**

## ExternalApi.java

package com.example;

public interface ExternalApi { String getData();

}

## MyService.java

package com.example;

public class MyService {

private final ExternalApi externalApi;

public MyService(ExternalApi externalApi) { this.externalApi = externalApi;

}

public String fetchData() { return externalApi.getData();

}

}

1. **Add Mockito and JUnit Dependencies in pom.xml**

<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>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

1. **Test Class for Verifying Interaction**

package com.example;

import org.junit.jupiter.api.Test; import static org.mockito.Mockito.\*;

public class MyServiceInteractionTest { @Test

public void testVerifyInteraction() {

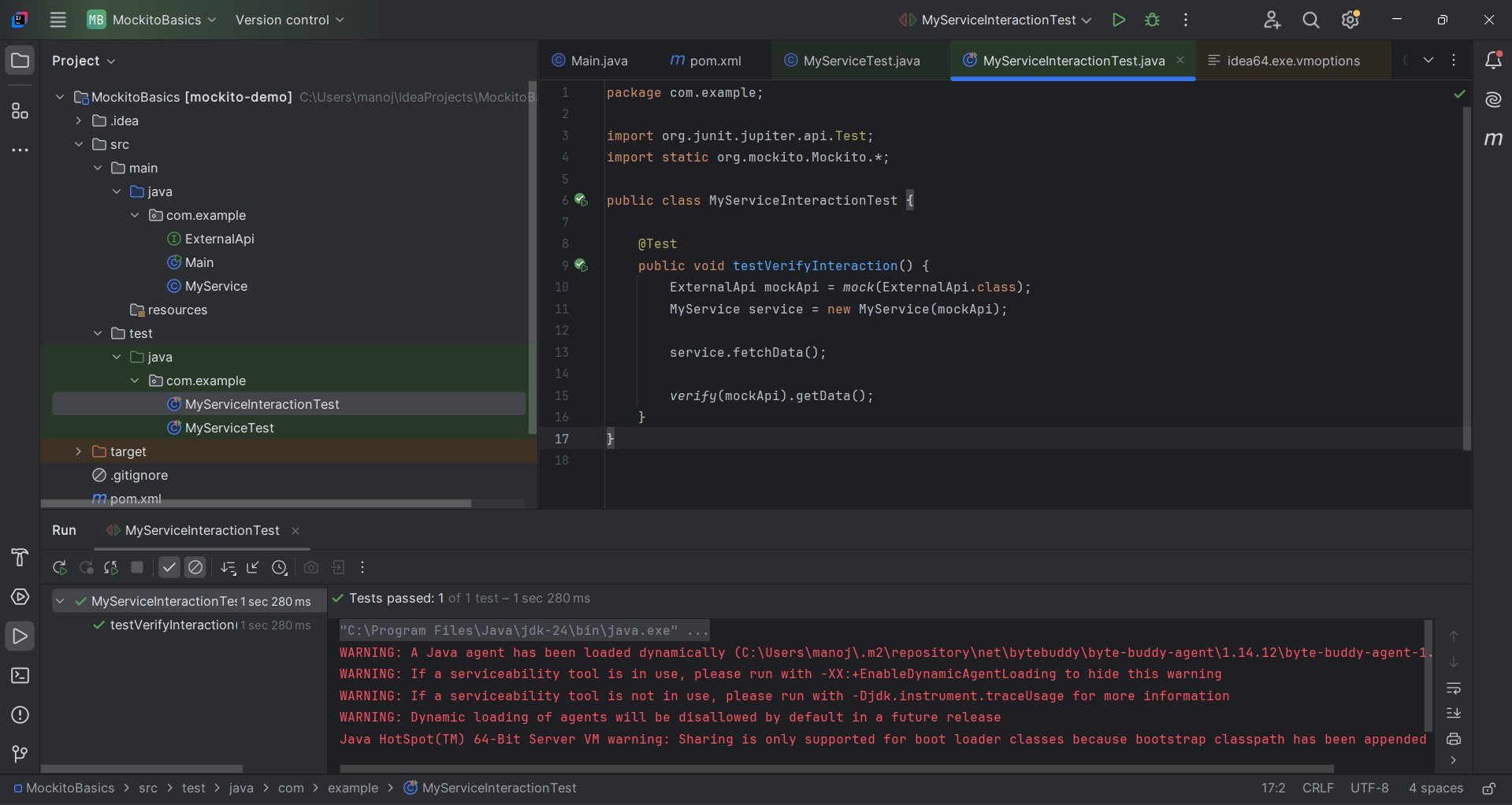
ExternalApi mockApi = *mock*(ExternalApi.class); MyService service = new MyService(mockApi);

service.fetchData();

*verify*(mockApi).getData();

}

}

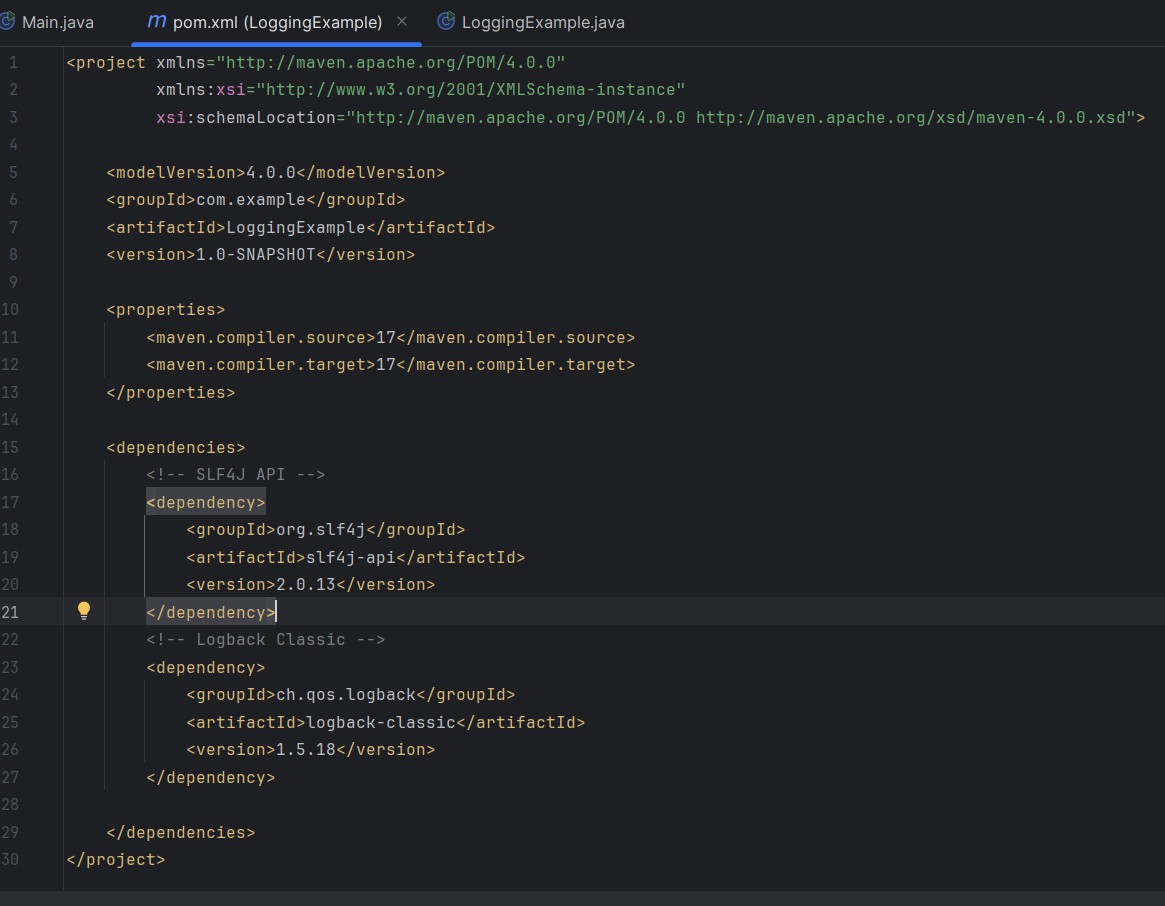
**Output:**

# Exercise – 6

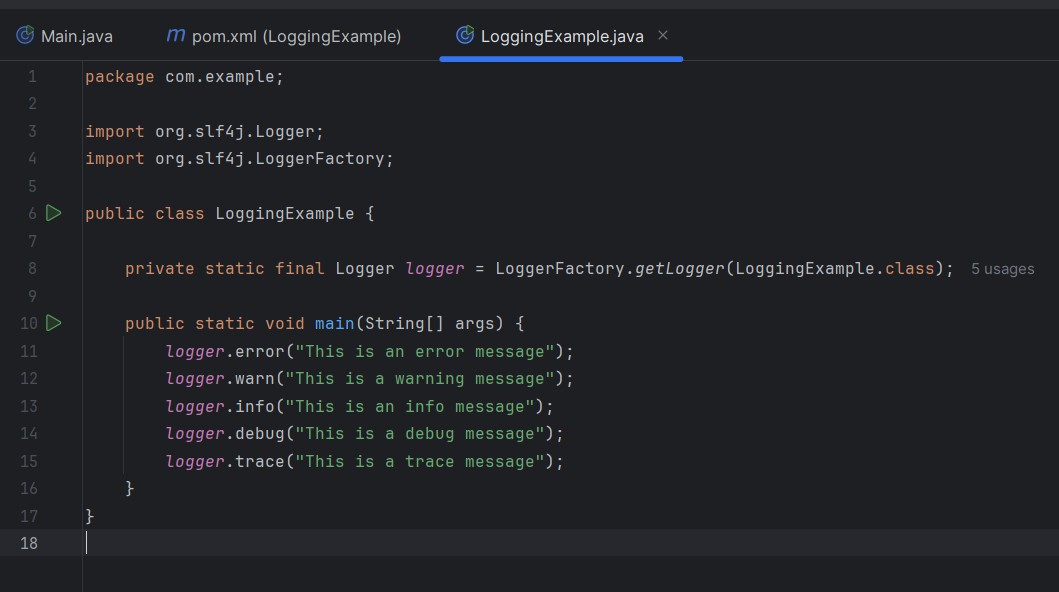
**SLF4J Logging — Logging Error Messages and Warning Levels**

**Scenario:** You need to demonstrate logging in a Java application using the **SLF4J (Simple Logging Facade for Java)** with **Logback** as the implementation backend.

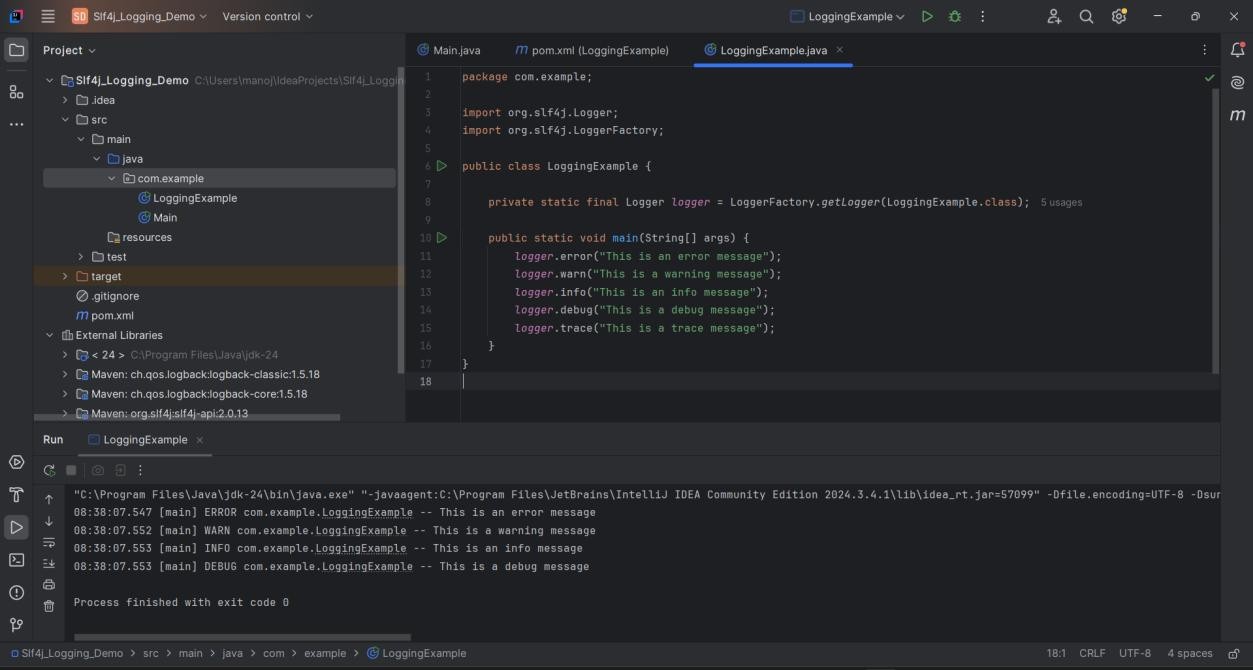
**Step 1: Add SLF4J and Logback dependencies to pom.xml**

****

**Step 2: Create the Logging Class**

****

**Output:**

****