**Mockito Hands-On Exercises**

**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.

**SOLUTION:**

1. **ExternalApi.java**

package com.example;

public class ExternalApi {

String getData() {

return null;

}

}

1. **MyService.java:**

package com.example;

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

1. **MyServiceTest.java:**package com.example;

import static org.junit.Assert.*assertEquals*;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

*@Test*

public void testExternalApi() {

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

*when*(mockApi.getData()).thenReturn("Mock Data");

MyService service = new MyService(mockApi);

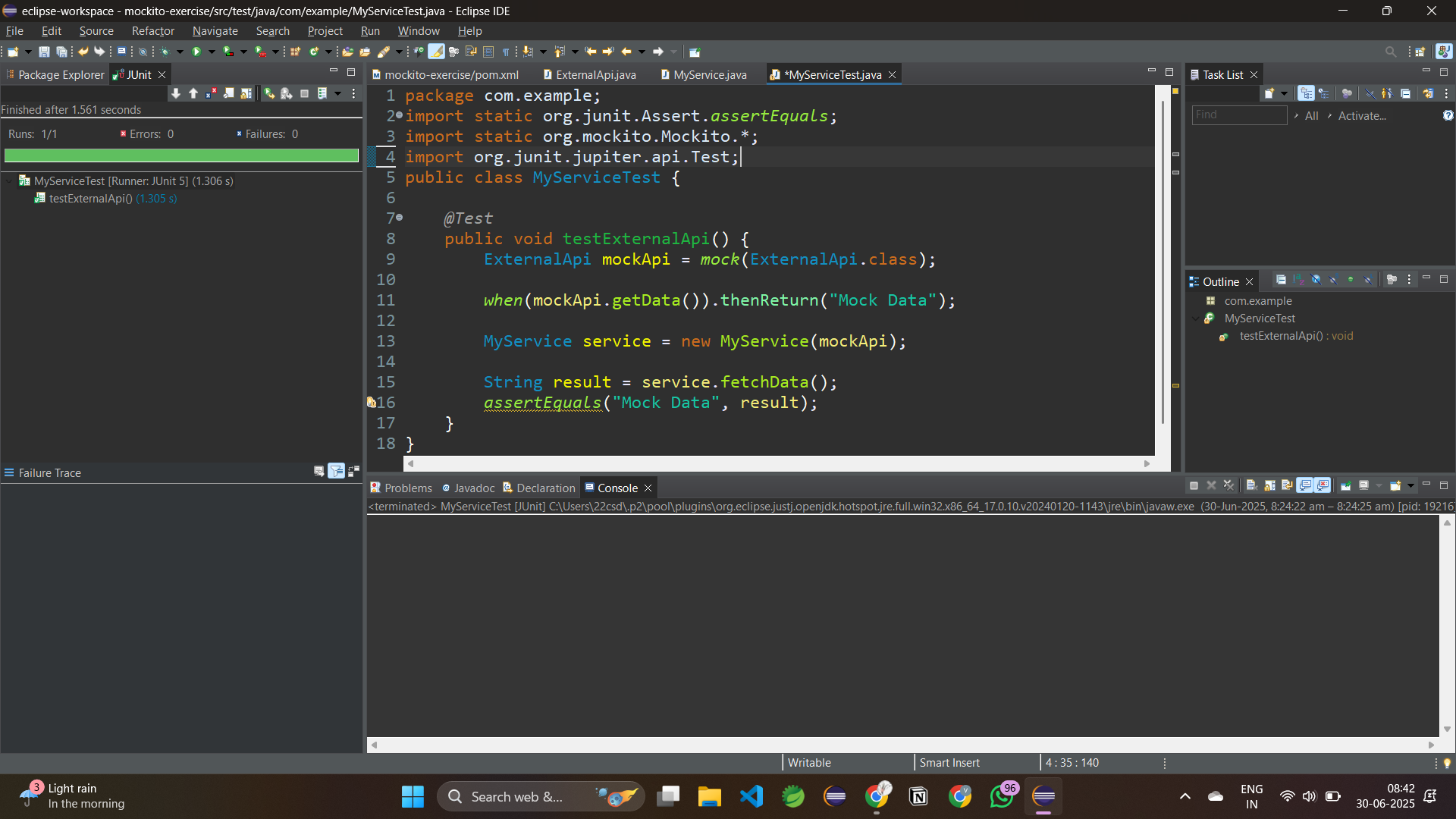
String result = service.fetchData();

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

}

}

**OUTPUT:**



**Exercise 2: Verifying Interactions**

**Scenario:**

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

**Steps:**

1. Create a mock object.

2. Call the method with specific arguments.

3. Verify the interaction.

**SOLUTION:**

1. **ExternalApi.java:**

package com.example;

public class ExternalApi {

String getData() {

return null;

}

}

1. **MyService.java:**

package com.example;

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData(); //Interaction to verify

}

}

1. **MyServiceTest.java:**

package com.example;

import static org.mockito.Mockito.*times*;

import static org.mockito.Mockito.*verify*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

*@Test*

public void testVerifyInteraction() {

//Create mock

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

//Inject into service

MyService service = new MyService(mockApi);

//Call method

service.fetchData();

//service.fetchData();

//Verify method was called

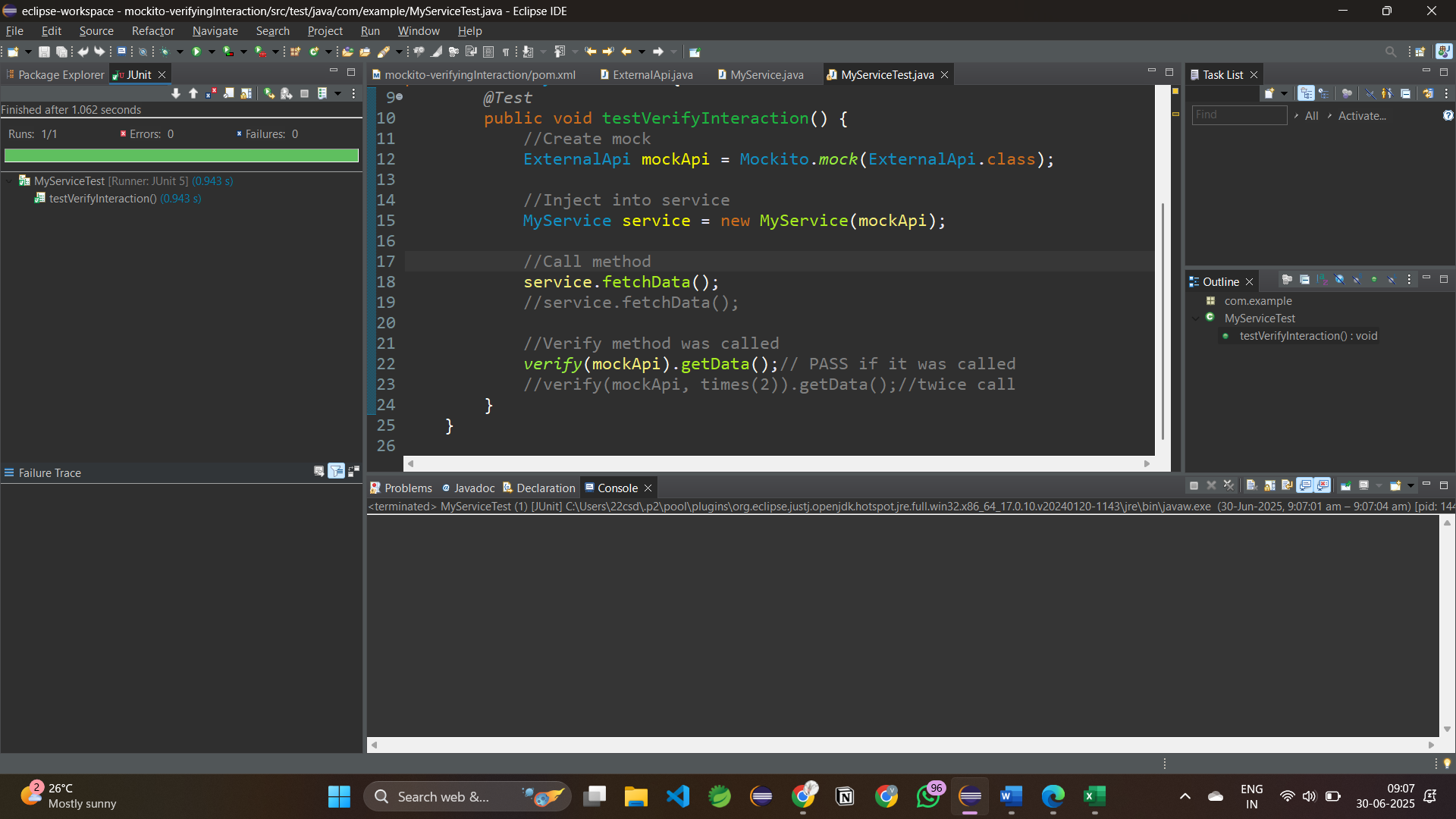
*verify*(mockApi).getData();// PASS if it was called

//verify(mockApi, times(2)).getData();//twice call

}

}

**OUTPUT:**



**Logging using SLF4J**

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

**Task:** Write a Java application that demonstrates logging error messages and warning levels using SLF4J.

**SOLUTION:**

1. Create a **Maven** Project :

groupId: com.example

artifactId: slf4j-logging

1. Add dependencies in **pom.xml** file:

<dependencies>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

</dependencies>

1. Create a **Java** Class:

package com.example;

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");

}

}

**OUTPUT:**

