*MOCKITO 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.***

# *SOLUTION:*

*Step 1:Add the Dependency in pom.xml.*

# *CODE:*

*<project xmlns="*[*http://maven.apache.org/POM/4.0.0"*](http://maven.apache.org/POM/4.0.0) *xmlns:xsi="*[*http://www.w3.org/2001/XMLSchema-instance*](http://www.w3.org/2001/XMLSchema-instance)*" xsi:schemaLocation="*[*http://maven.apache.org/POM/4.0.0*](http://maven.apache.org/POM/4.0.0) *https://maven.apache.org/xsd/maven-*

*4.0.0.xsd">*

*<modelVersion>4.0.0</modelVersion>*

*<groupId>TDDMockito</groupId>*

*<artifactId>mockito</artifactId>*

*<version>0.0.1-SNAPSHOT</version>*

*<properties>*

*<maven.compiler.source>11</maven.compiler.source>*

*<maven.compiler.target>11</maven.compiler.target>*

*<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>*

*<junit.jupiter.version>5.10.0</junit.jupiter.version>*

*<mockito.version>5.11.0</mockito.version>*

*<surefire.version>3.2.3</surefire.version> <!-- Updated surefire version -->*

*</properties>*

*<dependencies>*

*<!-- JUnit 5 -->*

*<dependency>*

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

*<artifactId>junit-jupiter</artifactId>*

*<version>${junit.jupiter.version}</version>*

*<scope>test</scope>*

*</dependency>*

*<!-- Mockito Core -->*

*<dependency>*

*<groupId>org.mockito</groupId>*

*<artifactId>mockito-core</artifactId>*

*<version>${mockito.version}</version>*

*<scope>test</scope>*

*</dependency>*

*<!-- Mockito JUnit 5 integration -->*

*<dependency>*

*<groupId>org.mockito</groupId>*

*<artifactId>mockito-junit-jupiter</artifactId>*

*<version>${mockito.version}</version>*

*<scope>test</scope>*

*</dependency>*

*<!-- Add ByteBuddy dependency for Mockito's modern mocking -->*

*<dependency>*

*<groupId>net.bytebuddy</groupId>*

*<artifactId>byte-buddy</artifactId>*

*<version>1.14.9</version>*

*<scope>test</scope>*

*</dependency>*

*</dependencies>*

*<build>*

*<plugins>*

*<!-- Updated Maven Surefire Plugin configuration -->*

*<plugin>*

*<groupId>org.apache.maven.plugins</groupId>*

*<artifactId>maven-surefire-plugin</artifactId>*

*<version>${surefire.version}</version>*

*<configuration>*

*<argLine>*

*--add-opens java.base/java.lang=ALL-UNNAMED*

*--add-opens java.base/java.util=ALL-UNNAMED*

*</argLine>*

*</configuration>*

*</plugin>*

*<!-- Add Maven Compiler Plugin for explicit configuration -->*

*<plugin>*

*<groupId>org.apache.maven.plugins</groupId>*

*<artifactId>maven-compiler-plugin</artifactId>*

*<version>3.11.0</version>*

*<configuration>*

*<release>11</release>*

*</configuration>*

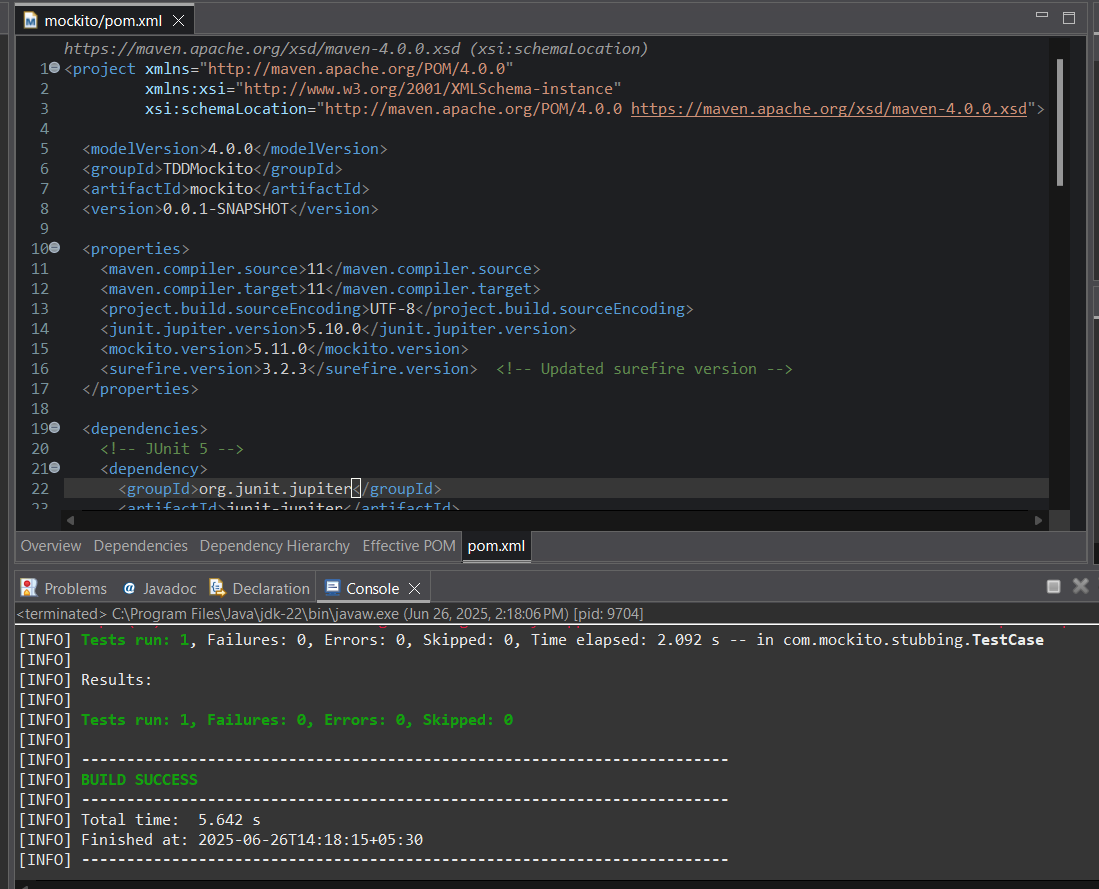
*</plugin>*

*</plugins>*

*</build>*

*</project>*

***OUTPUT:***

******

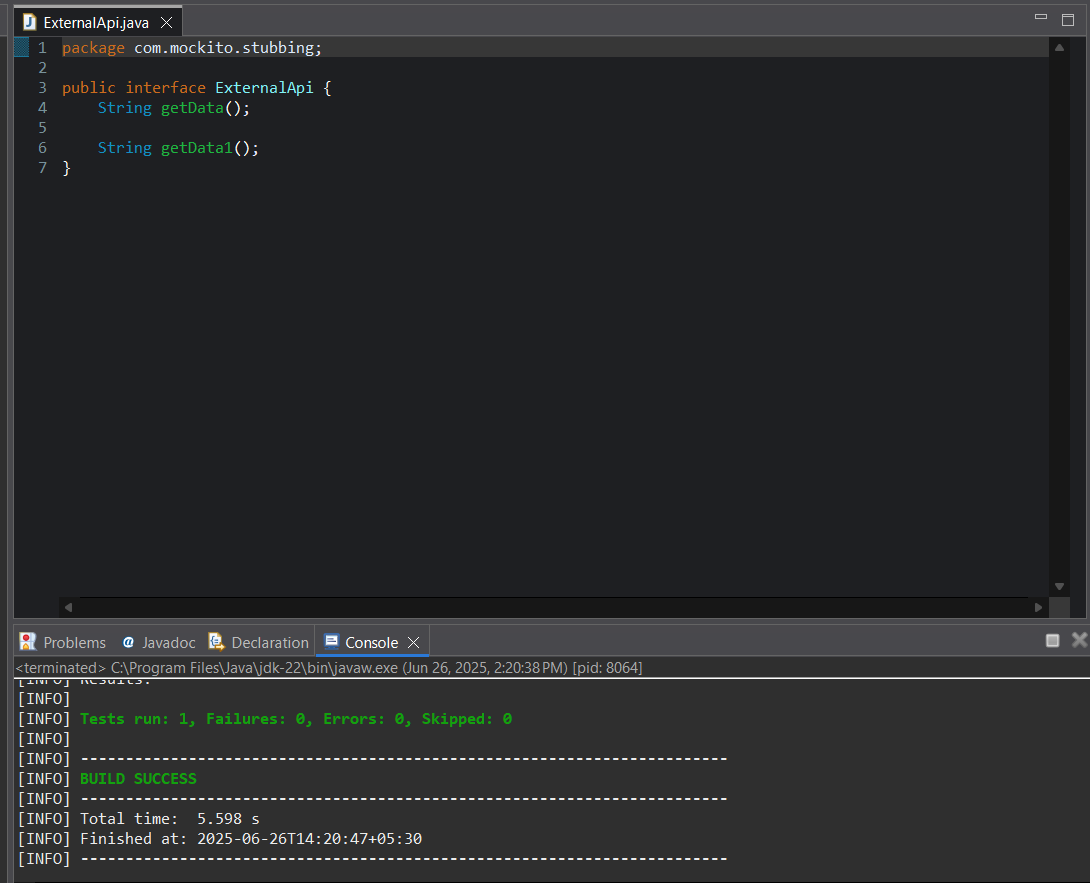
*Step 2:Create a mock object for the external API.*

***CODE (for ExternalApi.java):*** *package com.mockito.stubbing; public interface ExternalApi {*

*String getData(); String getData1();*

*}*

***OUTPUT(for ExternalApi.java):***

******

***CODE(for MyService.java):***

*package com.mockito.stubbing;*

*// Service that uses ExternalApi public class MyService {*

*private ExternalApi api;*

*public MyService(ExternalApi api) { this.api = api;*

*}*

*public String fetchData() { return api.getData();*

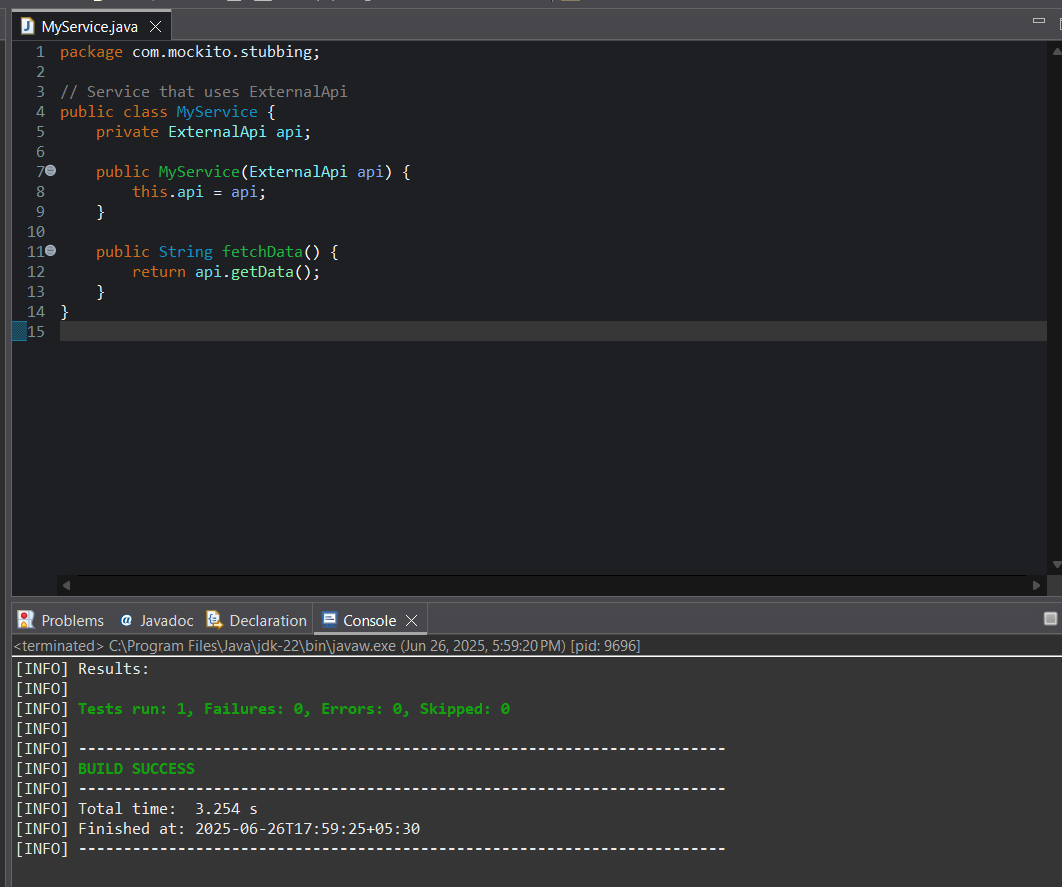
*}*

*}*

*return api.getData();*

*}*

*}*

***OUTPUT(for MyService.java):***

*Step 3&4: Stub the methods to return predefined values. Write a test case that uses the mock object.*

***CODE(for TestCase.java):***

*package com.mockito.stubbing;*

*// Needed for assertEquals*

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

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

*// Test class*

*public class TestCase {*

*@Test*

*public void testExternalApi() {*

*// Create a mock for the ExternalApi*

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

*// Stub the getData() method*

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

*// Inject mock into MyService*

*MyService service = new MyService(mockApi);*

*// Call the method under test String result = service.fetchData();*

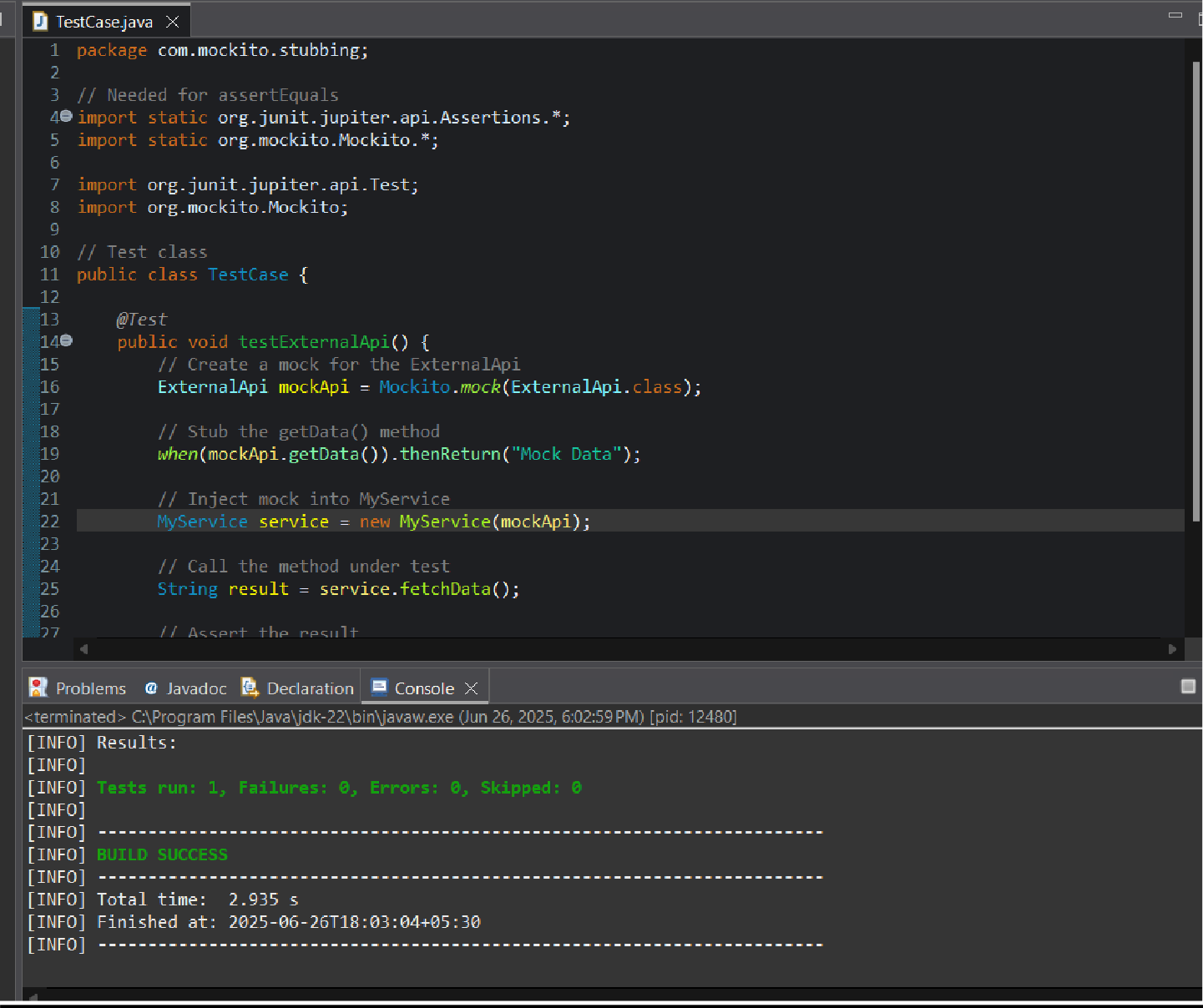
*// Assert the result*

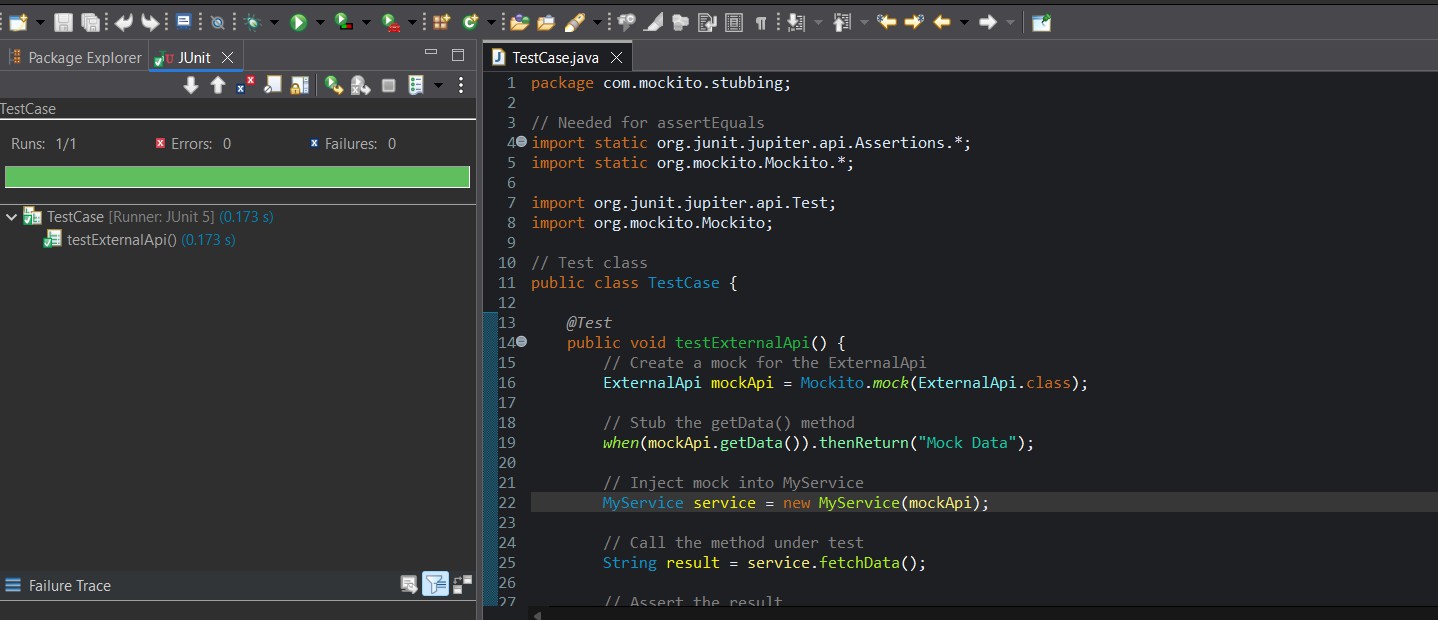
*assertEquals("Mock Data", result);*

*}*

*}*

***OUTPUT(for TestCase.java): Maven test:***

**

***Junit test:***

***Exercise 6: Verifying Interaction Order***

***Scenario: You need to ensure that methods are called in a specific order.***

***SOLUTION:***

*Step 1:Add the Dependency in pom.xml.*

# *CODE:*

*<project xmlns="*[*http://maven.apache.org/POM/4.0.0"*](http://maven.apache.org/POM/4.0.0) *xmlns:xsi="*[*http://www.w3.org/2001/XMLSchema-instance*](http://www.w3.org/2001/XMLSchema-instance)*" xsi:schemaLocation="*[*http://maven.apache.org/POM/4.0.0*](http://maven.apache.org/POM/4.0.0) *https://maven.apache.org/xsd/maven-*

*4.0.0.xsd">*

*<modelVersion>4.0.0</modelVersion>*

*<groupId>Verify</groupId>*

*<artifactId>VerifyingInteractions</artifactId>*

*<version>0.0.1-SNAPSHOT</version>*

*<dependencies>*

*<!-- JUnit 5 (Jupiter) for testing -->*

*<dependency>*

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

*<artifactId>junit-jupiter</artifactId>*

*<version>5.10.0</version>*

*<scope>test</scope>*

*</dependency>*

*<!-- Mockito core library -->*

*<dependency>*

*<groupId>org.mockito</groupId>*

*<artifactId>mockito-core</artifactId>*

*<version>5.11.0</version>*

*<scope>test</scope>*

*</dependency>*

*</dependencies>*

*<build>*

*<plugins>*

*<!-- Maven Surefire Plugin to run JUnit 5 tests -->*

*<plugin>*

*<groupId>org.apache.maven.plugins</groupId>*

*<artifactId>maven-surefire-plugin</artifactId>*

*<version>3.2.5</version>*

*<configuration>*

*<useModulePath>false</useModulePath>*

*</configuration>*

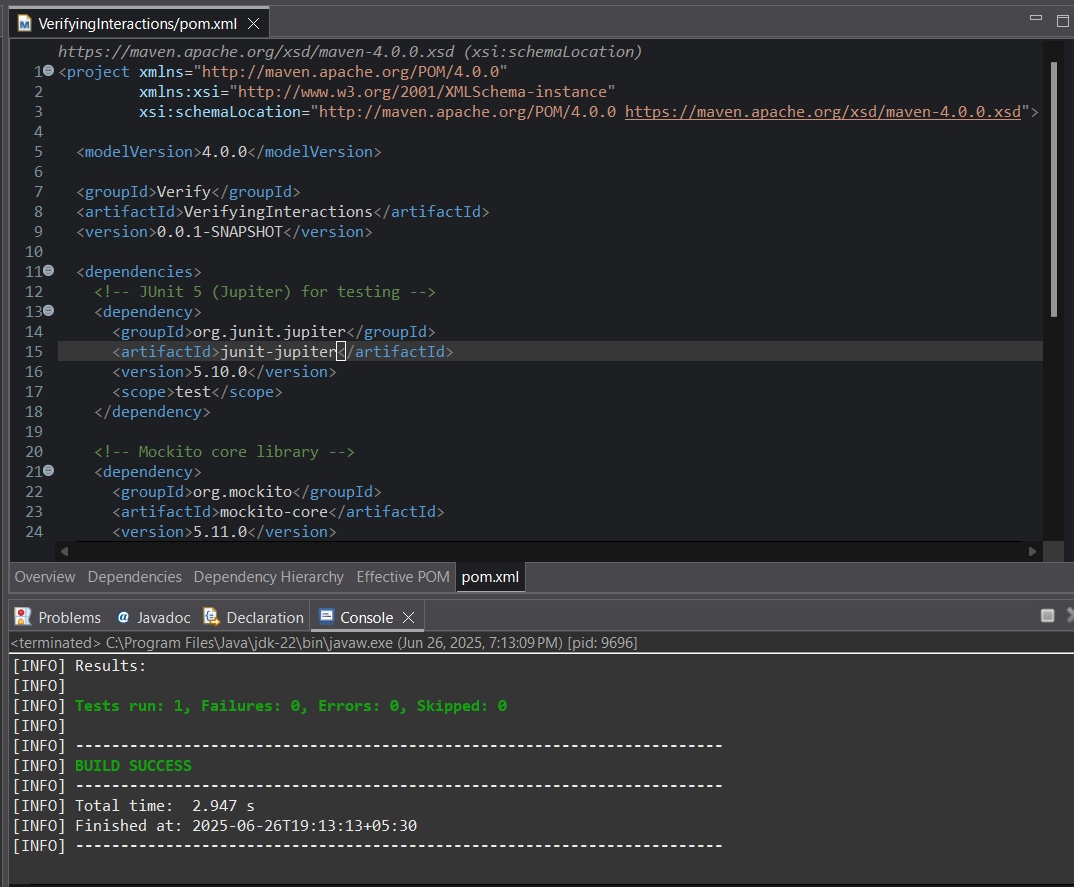
*</plugin>*

*</plugins>*

*</build>*

*</project>*

# *OUTPUT:*

******

*Step 2:Create a mock object.*

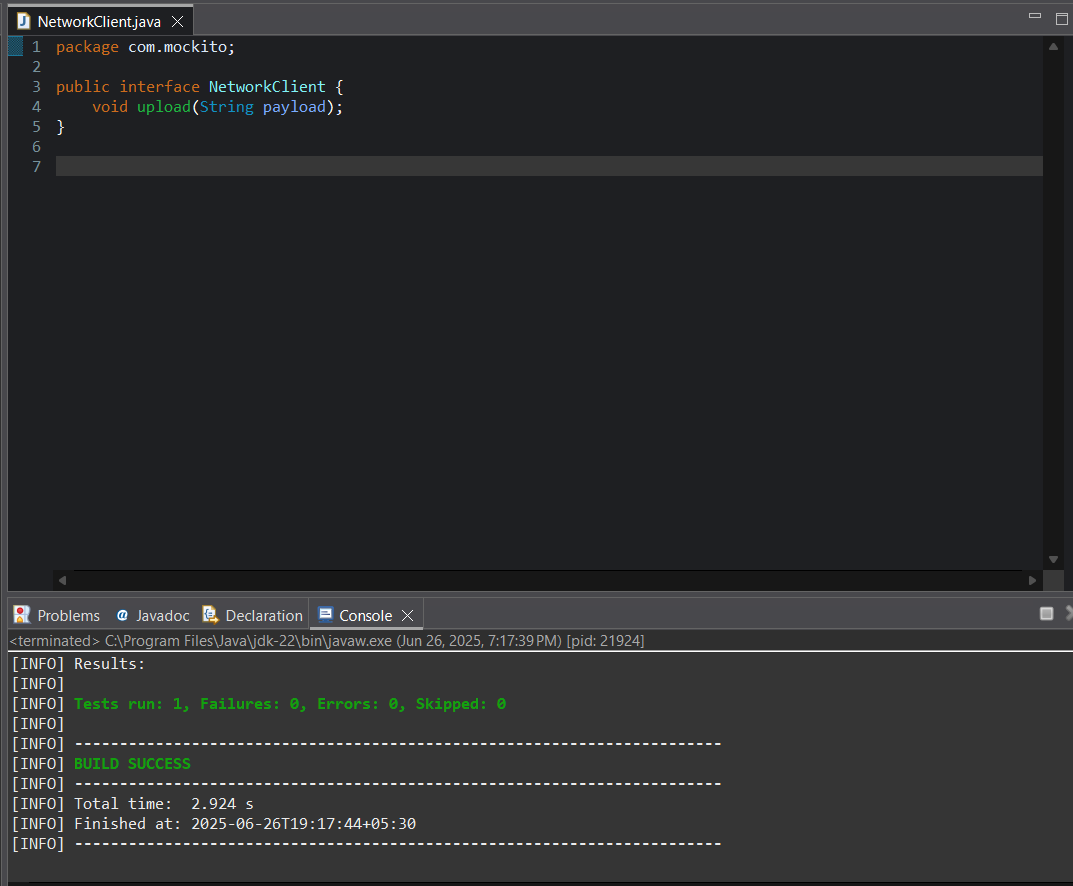
***CODE(for NetworkClient.java):***

*package com.mockito;*

*public interface NetworkClient { void upload(String payload);*

*}*

***OUTPUT(for NetworkClient.java):***

******

***CODE(for DataUploader.java):***

*package com.mockito; public class DataUploader {*

*private final NetworkClient client;*

*public DataUploader(NetworkClient client) { this.client = client;*

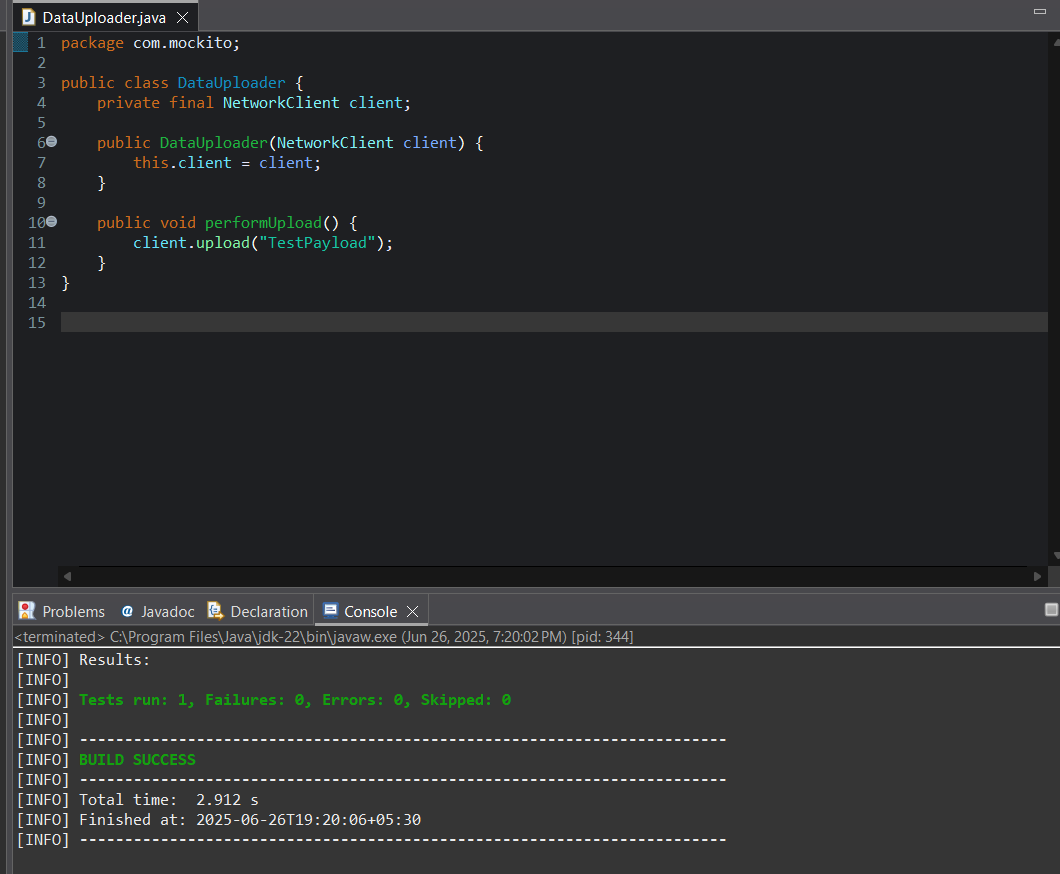
*}*

*public void performUpload() { client.upload("TestPayload");*

*}*

*}*

***OUTPUT(for DataUploader.java):***

******

*Step 3&4: Call the methods in a specific order.Verify the interaction order.*

***CODE(for TestDataUploader.java):***

*package com.mockito;*

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

*public class TestDataUploader {*

*@Test*

*public void testUploadCalledWithCorrectArgument() {*

*// Step 1: Create mock*

*NetworkClient mockClient = Mockito.mock(NetworkClient.class);*

*// Step 2: Call method with specific argument DataUploader uploader = new DataUploader(mockClient); uploader.performUpload();*

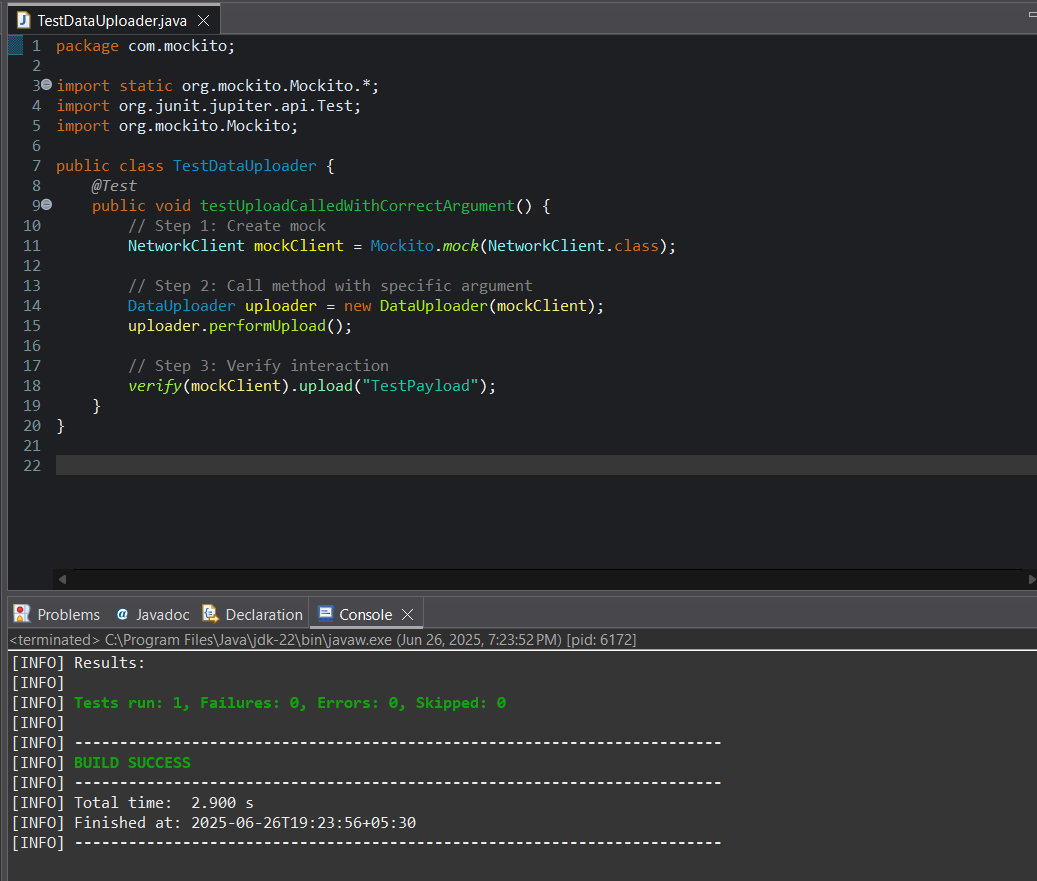
*// Step 3: Verify interaction*

*verify(mockClient).upload("TestPayload");*

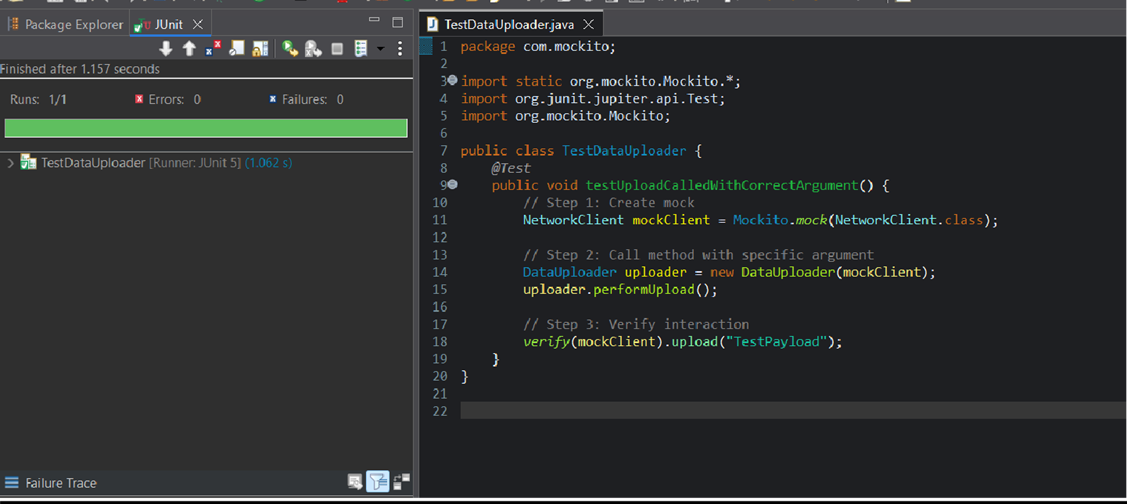
*}*

*}*

# *OUTPUT:*

***Maven Test:***

***Junit Test:***

**