**Exercise 1: Mocking and Stubbing**

**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>junit-demo</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter</artifactId>  
 <version>5.10.0</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.mockito</groupId>  
 <artifactId>mockito-core</artifactId>  
 <version>5.12.0</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>3.0.0</version>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**ExternalApi.java**

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

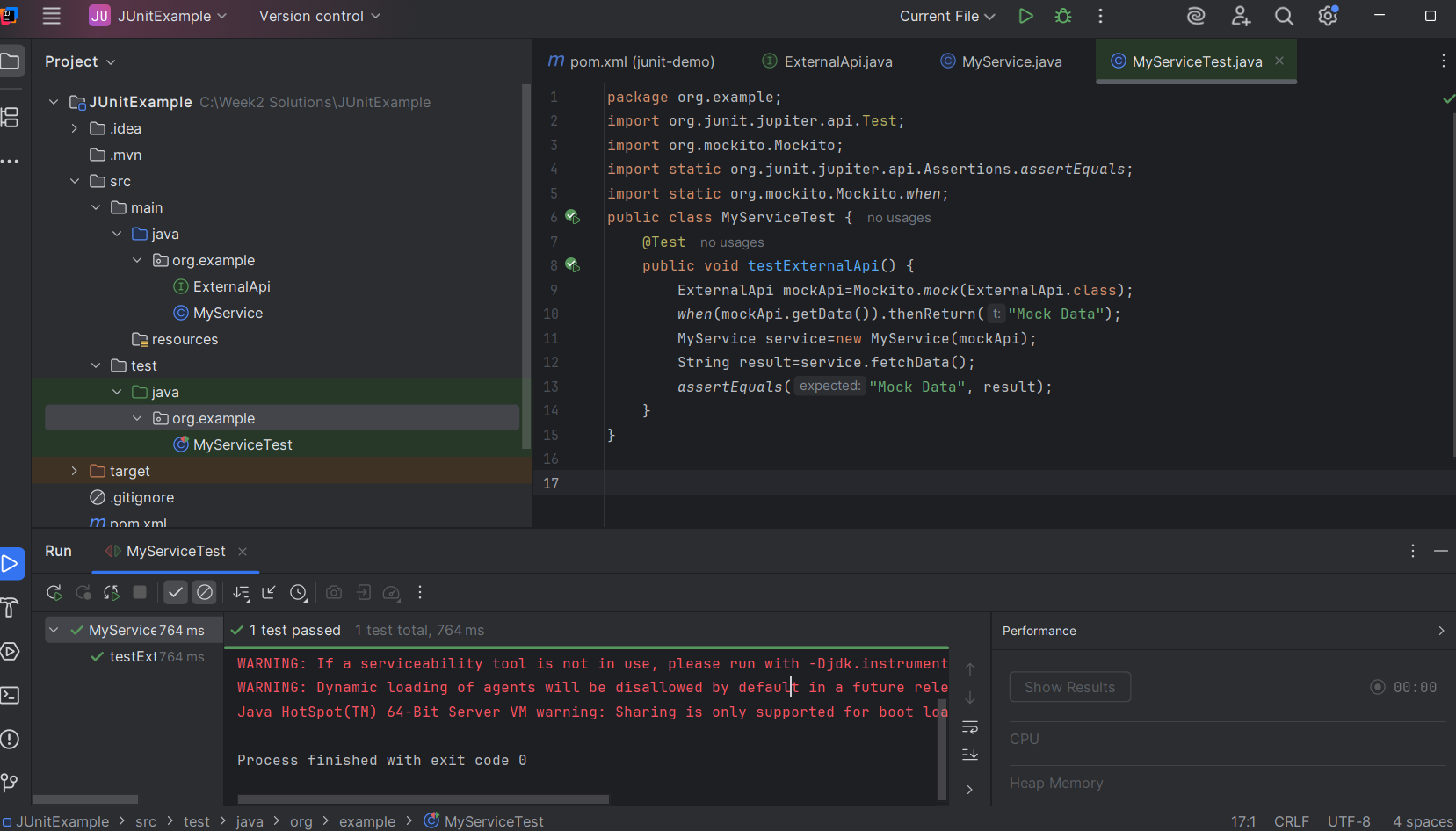
**MyService.java**

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

**MyServiceTest.java**

package org.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**

****

**Exercise 2: Verifying Interactions**

**ExternalApi.java**

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

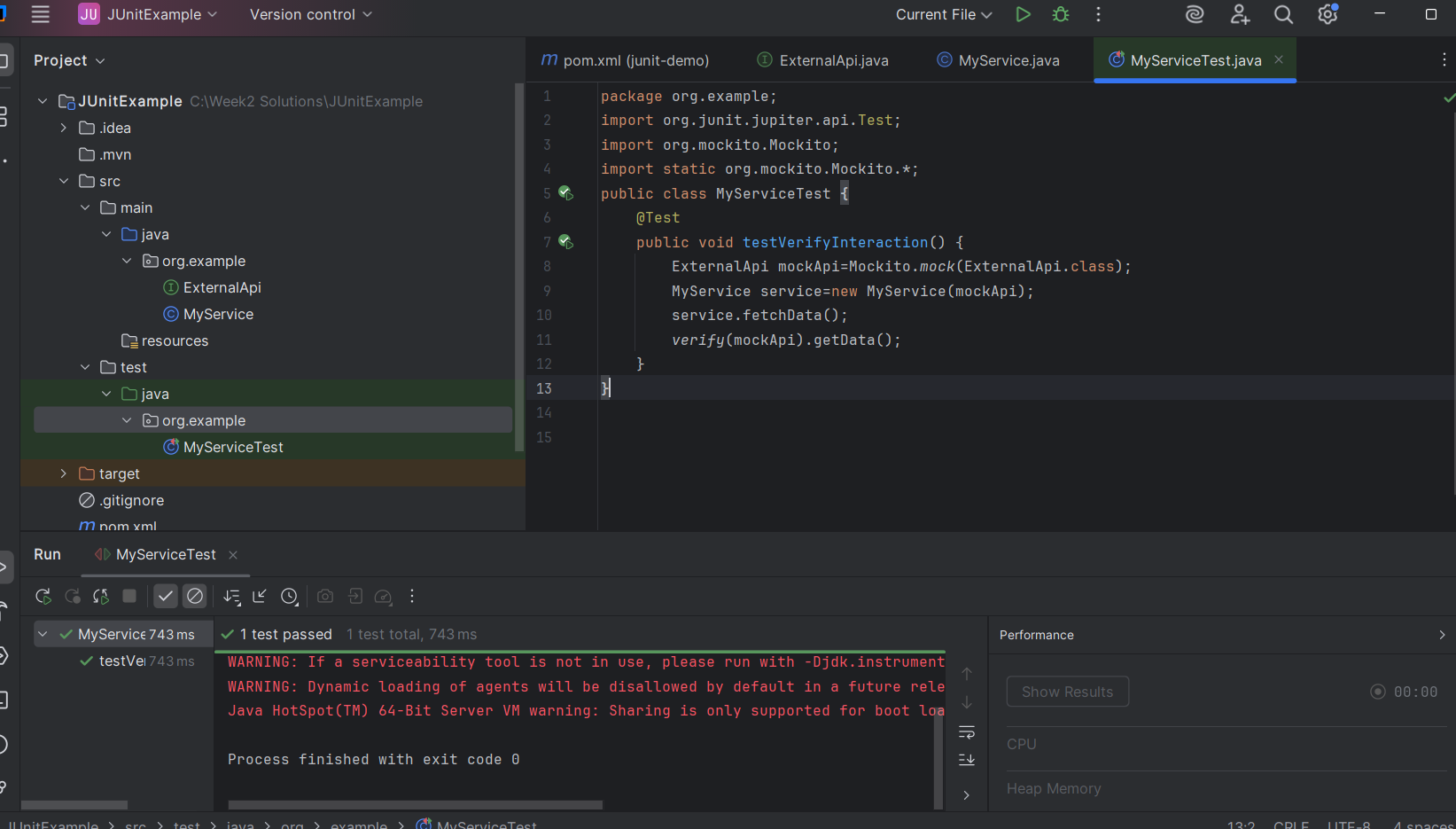
**MyService.java**

package org.example;  
public class MyService {  
 private final ExternalApi externalApi;  
 public MyService(ExternalApi externalApi) {  
 this.externalApi=externalApi;  
 }  
 public String fetchData() {  
 return externalApi.getData();  
 }  
}

**MyServiceTest.java**

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

****

**Exercise 3: Argument Matching**

**ExternalApi.java**

package org.example;  
public interface ExternalApi {  
 String getDataById(int id); // Method with argument  
}

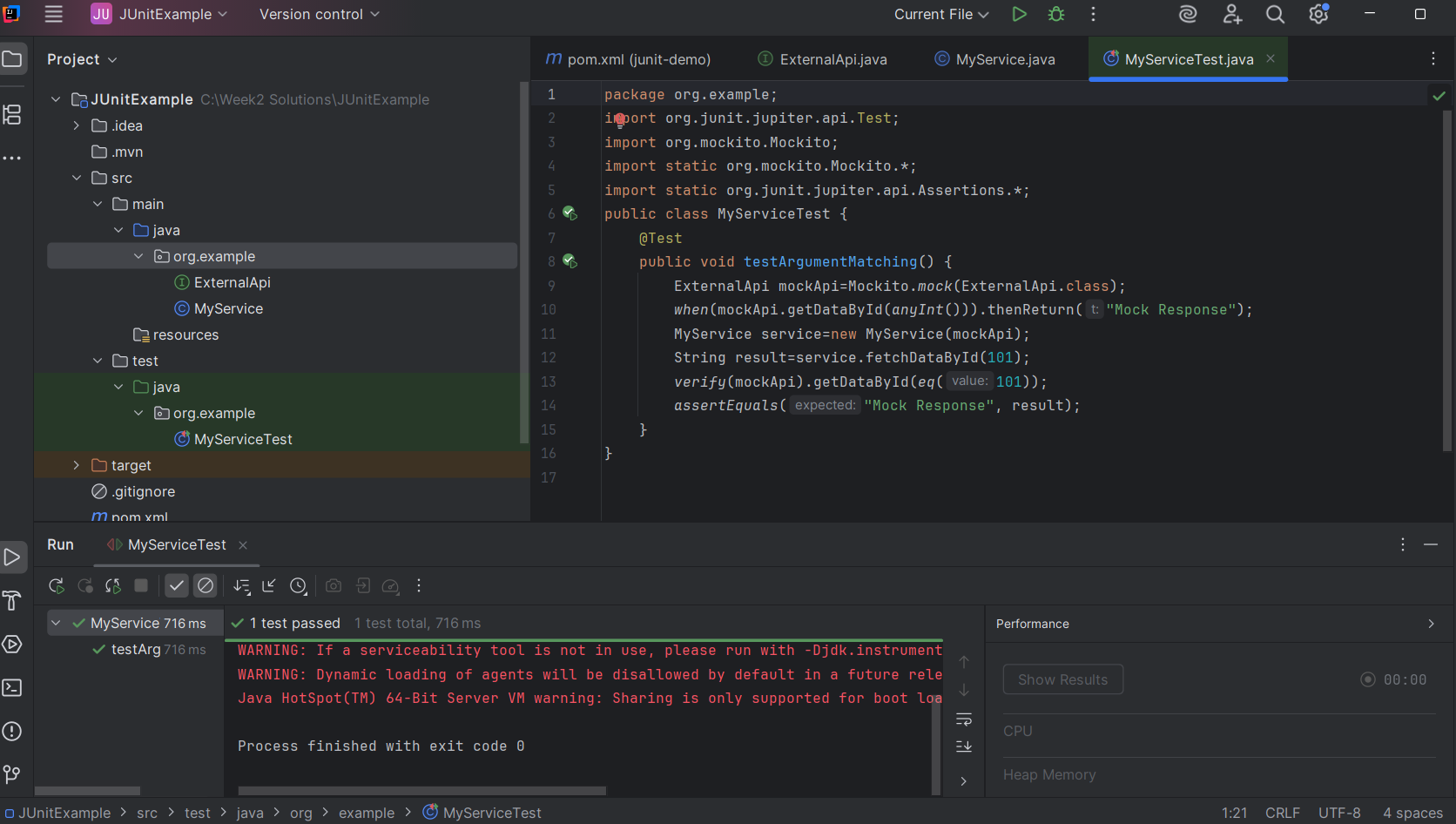
**Myservice.java**

package org.example;  
public class MyService {  
 private final ExternalApi externalApi;  
 public MyService(ExternalApi externalApi) {  
 this.externalApi=externalApi;  
 }  
 public String fetchDataById(int id) {  
 return externalApi.getDataById(id);  
 }  
}

**MyServiceTest.java**

package org.example;  
import org.junit.jupiter.api.Test;  
import org.mockito.Mockito;  
import static org.mockito.Mockito.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
public class MyServiceTest {  
 @Test  
 public void testArgumentMatching() {  
 ExternalApi mockApi=Mockito.*mock*(ExternalApi.class);  
 *when*(mockApi.getDataById(*anyInt*())).thenReturn("Mock Response");  
 MyService service=new MyService(mockApi);  
 String result=service.fetchDataById(101);  
 *verify*(mockApi).getDataById(*eq*(101));  
 *assertEquals*("Mock Response", result);  
 }  
}

**Output**

****

**Exercise 4: Handling Void Methods**

**ExternalLogger.java**

package org.example;  
public interface ExternalLogger {  
 void log(String message);  
}

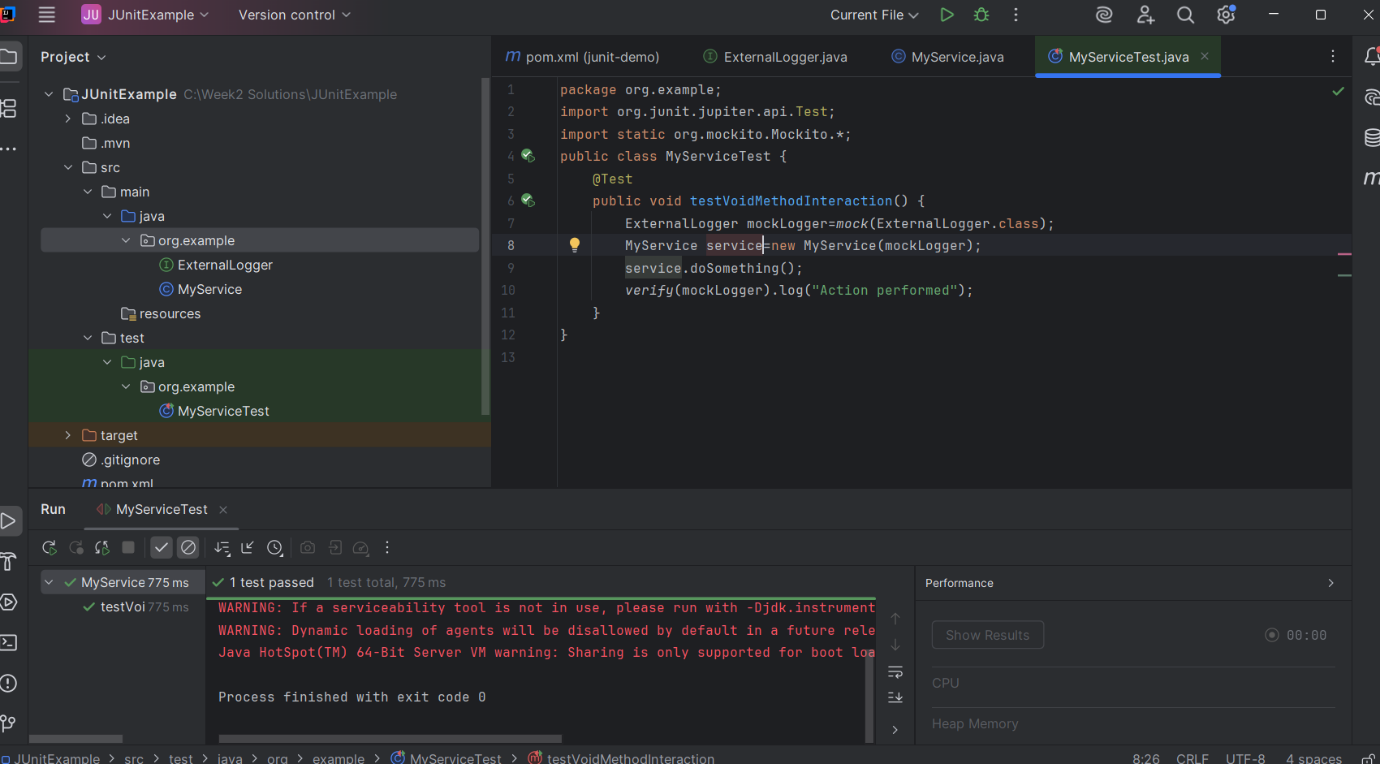
**MyService.java**

package org.example;  
public class MyService {  
 private ExternalLogger logger;  
 public MyService(ExternalLogger logger) {  
 this.logger=logger;  
 }  
 public void doSomething() {  
 logger.log("Action performed");  
 }  
}

**MyServiceTest.java**

package org.example;  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
public class MyServiceTest {  
 @Test  
 public void testVoidMethodInteraction() {  
 ExternalLogger mockLogger=*mock*(ExternalLogger.class);  
 MyService service=new MyService(mockLogger);  
 service.doSomething();  
 *verify*(mockLogger).log("Action performed");  
 }  
}

**Output**

****

**Exercise 5: Mocking and Stubbing with Multiple Returns**

**ExternalApi.java**

package org.example;  
public interface ExternalApi {  
 String getStatus();  
}

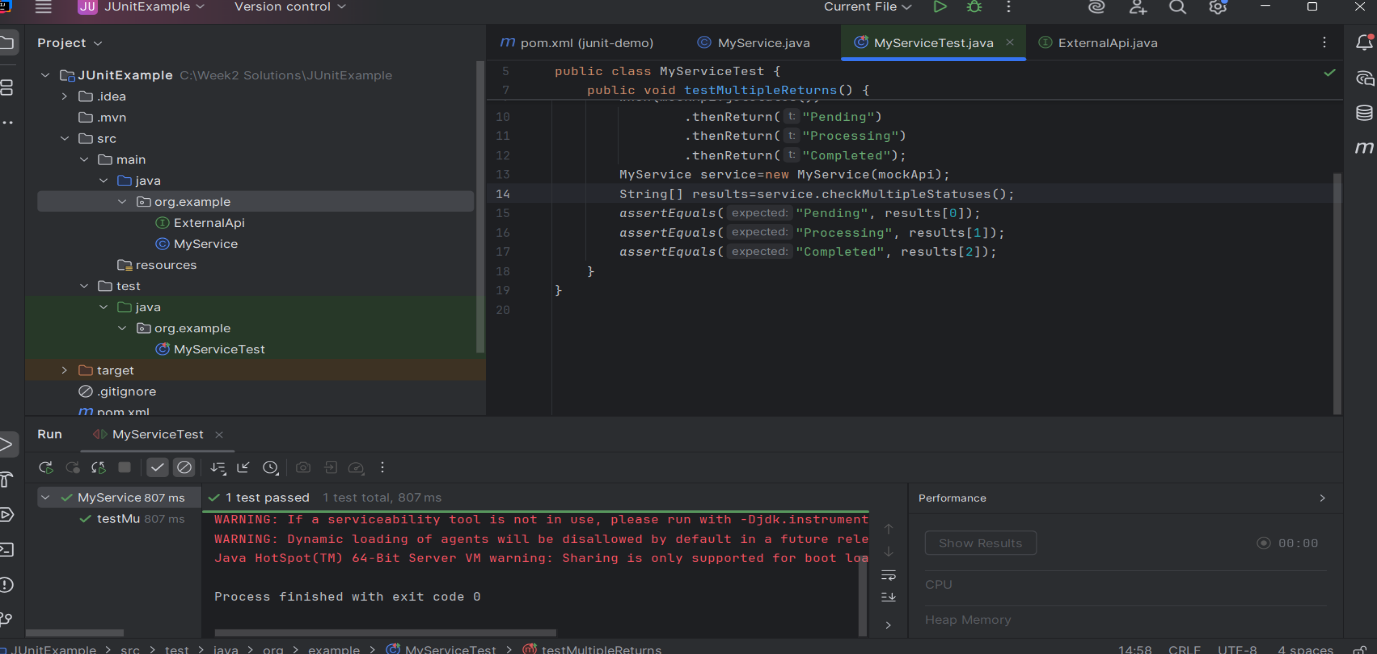
**MyService.java**

package org.example;  
  
public class MyService {  
 private ExternalApi api;  
 public MyService(ExternalApi api) {  
 this.api=api;  
 }  
 public String[] checkMultipleStatuses() {  
 return new String[] {  
 api.getStatus(),  
 api.getStatus(),  
 api.getStatus()  
 };  
 }  
}

**MyServiceTest.java**

package org.example;  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
public class MyServiceTest {  
 @Test  
 public void testMultipleReturns() {  
 ExternalApi mockApi=*mock*(ExternalApi.class);  
 *when*(mockApi.getStatus())  
 .thenReturn("Pending")  
 .thenReturn("Processing")  
 .thenReturn("Completed");  
 MyService service=new MyService(mockApi);  
 String[] results=service.checkMultipleStatuses();  
 *assertEquals*("Pending", results[0]);  
 *assertEquals*("Processing", results[1]);  
 *assertEquals*("Completed", results[2]);  
 }  
}

**Output**

****

**Exercise 6: Verifying Interaction Order**

**ExternalApi.java**

package org.example;  
public interface ExternalApi  
{  
 void login();  
 void fetchData();  
 void logout();  
}

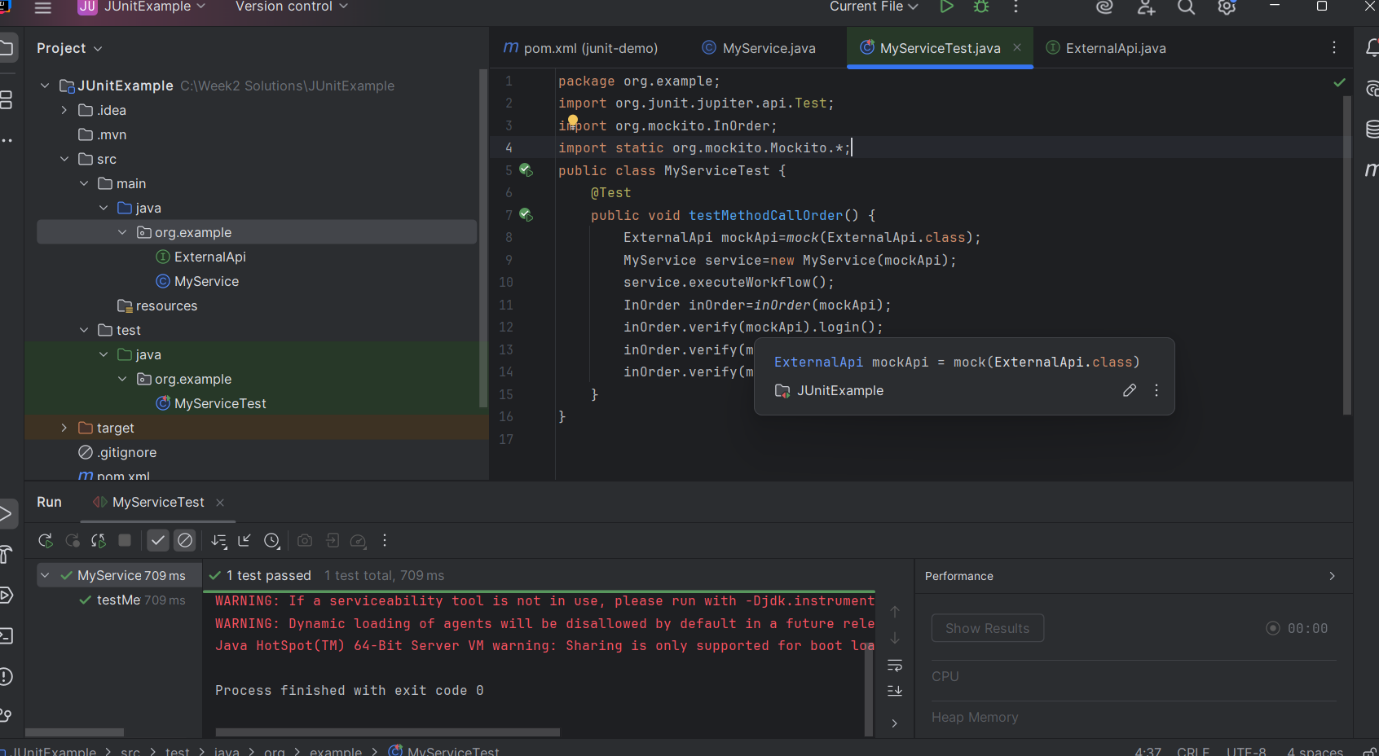
**MyService.java**

package org.example;  
public class MyService {  
 private ExternalApi api;  
 public MyService(ExternalApi api) {  
 this.api=api;  
 }  
 public void executeWorkflow() {  
 api.login();  
 api.fetchData();  
 api.logout();  
 }  
}

**MyServiceTest.java**

package org.example;  
import org.junit.jupiter.api.Test;  
import org.mockito.InOrder;  
import static org.mockito.Mockito.\*;  
public class MyServiceTest {  
 @Test  
 public void testMethodCallOrder() {  
 ExternalApi mockApi=*mock*(ExternalApi.class);  
 MyService service=new MyService(mockApi);  
 service.executeWorkflow();  
 InOrder inOrder=*inOrder*(mockApi);  
 inOrder.verify(mockApi).login();  
 inOrder.verify(mockApi).fetchData();  
 inOrder.verify(mockApi).logout();  
 }  
}

**Output**



**Exercise 7: Handling Void Methods with Exceptions**

**ExternalApi.java**

package org.example;  
public interface ExternalApi {  
 void deleteData() throws Exception;  
}

**MyService.java**

package org.example;  
public class MyService {  
 private final ExternalApi api;  
 public MyService(ExternalApi api) {  
 this.api=api;  
 }  
 public void performDelete() {  
 try {  
 api.deleteData();  
 } catch (Exception e) {  
 System.*out*.println("Exception caught: "+e.getMessage());  
 }  
 }  
}

**MyServiceTest.java**

package org.example;  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
public class MyServiceTest {  
 @Test  
 public void testDeleteDataThrowsException() throws Exception {  
 ExternalApi mockApi=*mock*(ExternalApi.class);  
 *doThrow*(new RuntimeException("Delete failed")).when(mockApi).deleteData();  
 MyService service=new MyService(mockApi);  
 *assertDoesNotThrow*(service::performDelete);  
 *verify*(mockApi).deleteData();  
 }  
}

**Output**

