**Week 2 : Mockito Hands-On Exercises**

**Exercise 1: Mocking and Stubbing**

**CODE :**

**Excercise1Test.java**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

public class Exercise1Test {

@Test

public void testExternalApi() {

ExternalApi mockApi = mock(ExternalApi.class);

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

MyService service = new MyService(mockApi);

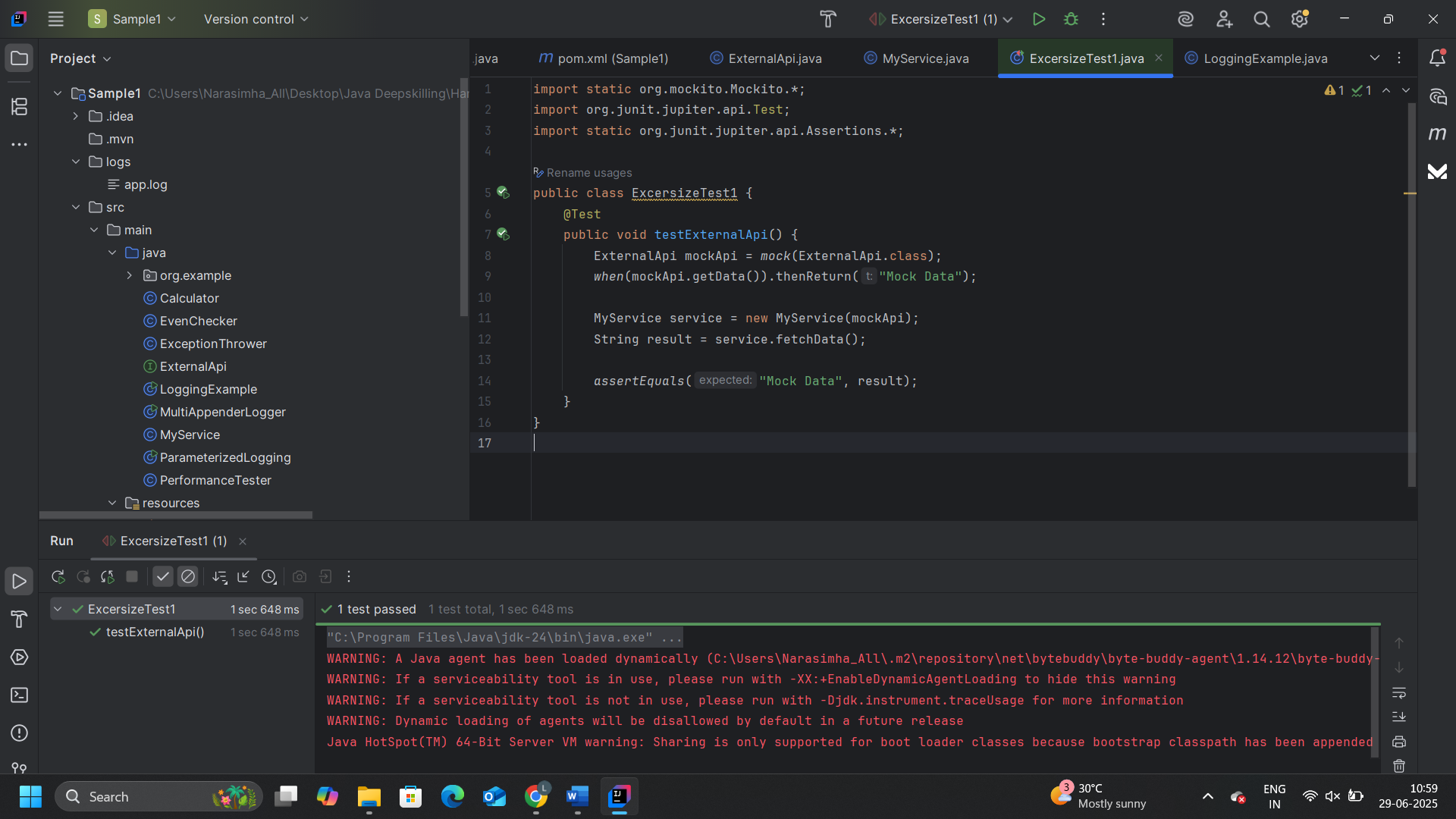
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**Output :**



**Exercise 2: Verifying Interactions**

**CODE :**

**Exercise2Test.java**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class Exercise2Test {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

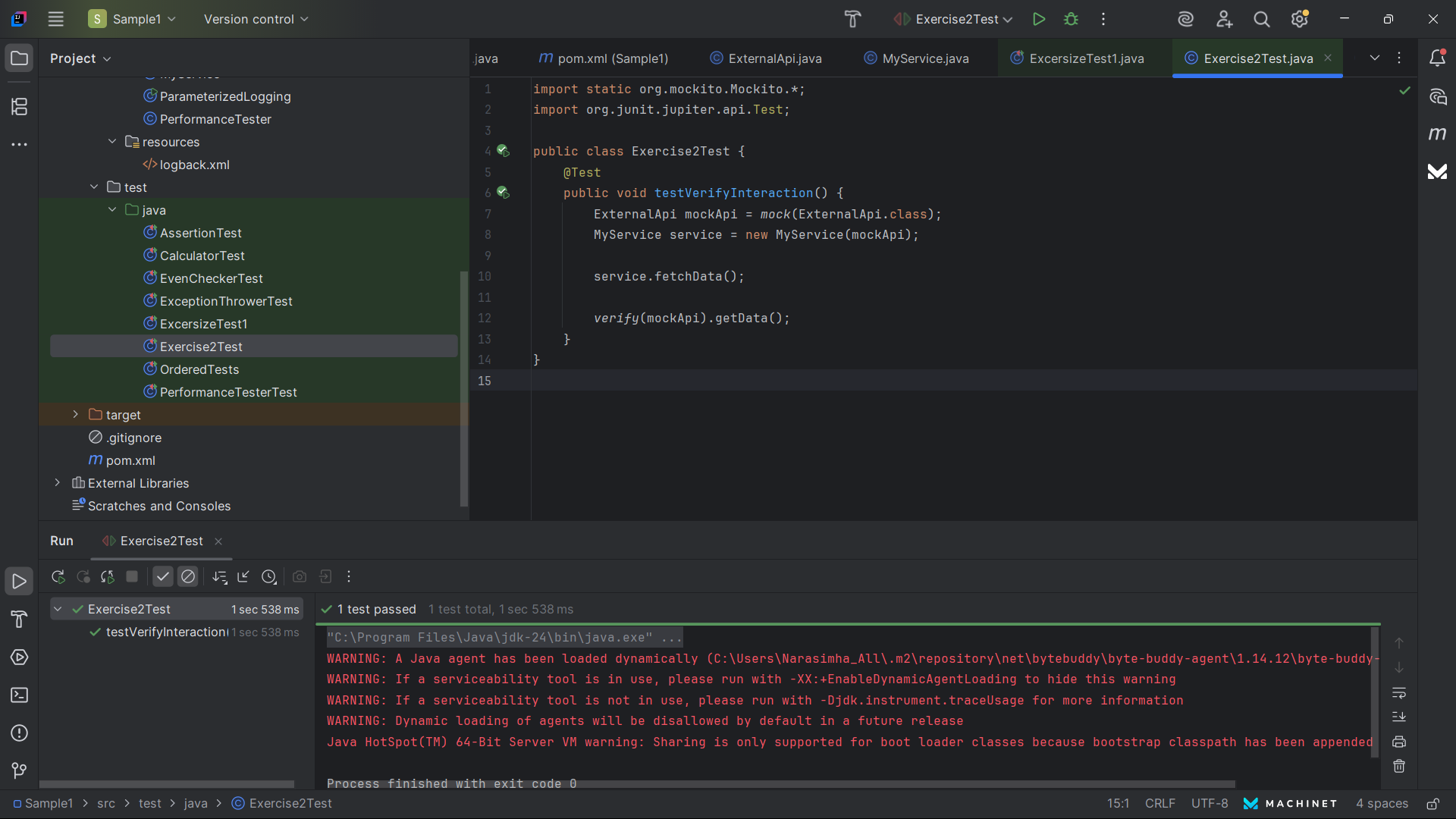
service.fetchData();

verify(mockApi).getData();

}

}

**Output :**



**Exercise 3: Argument Matching**

**CODE :**

**Exercise3Test.java**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class Exercise3Test {

@Test

public void testArgumentMatching() {

ExternalApi mockApi = mock(ExternalApi.class);

when(mockApi.getDataWithParam(anyString())).thenReturn("Mocked Result");

MyService service = new MyService(mockApi);

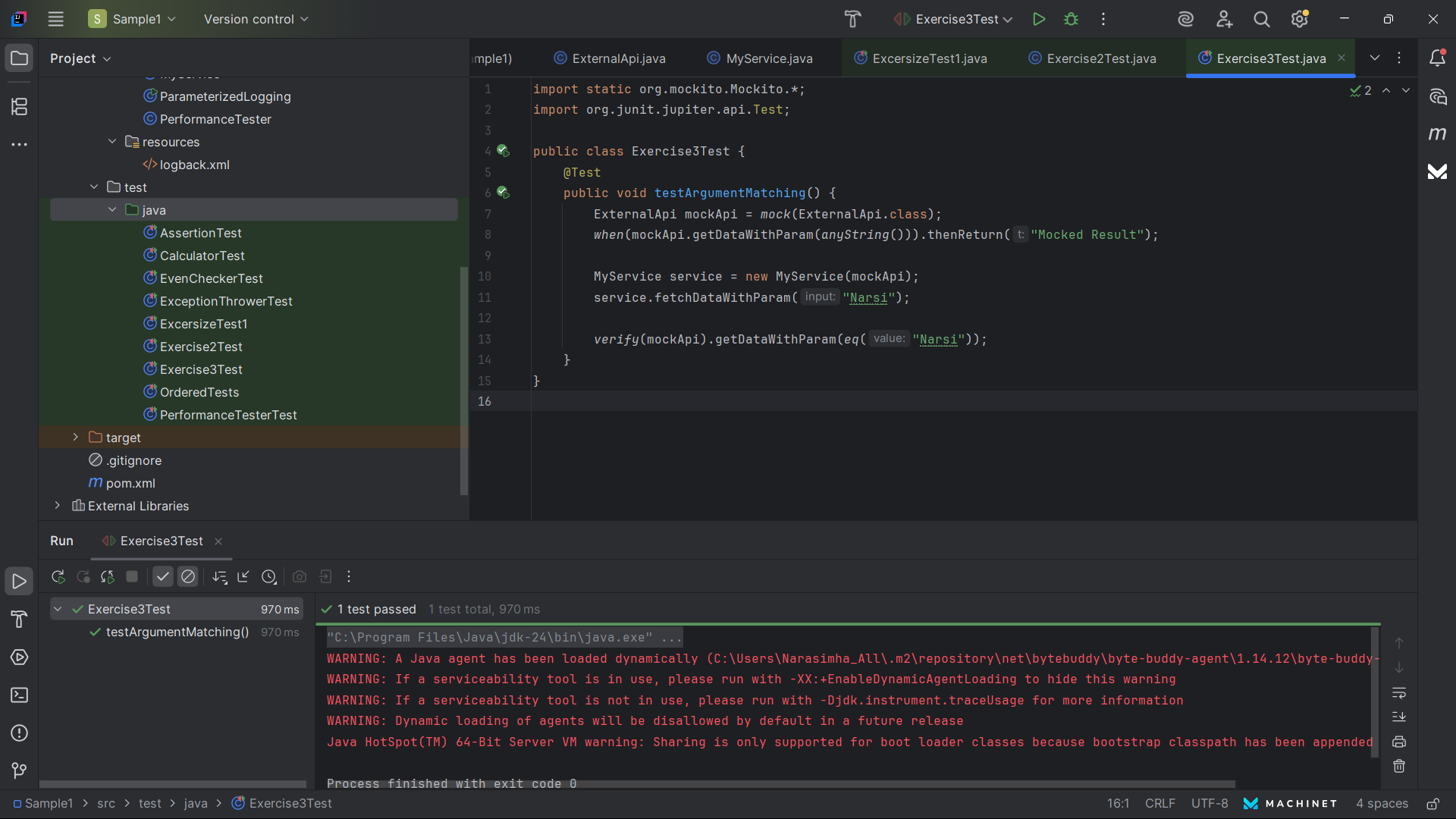
service.fetchDataWithParam("Narsi");

verify(mockApi).getDataWithParam(eq("Narsi"));

}

}

**Output :**



**Exercise 4: Handling Void Methods**

**CODE :**

**Exercise4Test.java**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class Exercise4Test {

@Test

public void testVoidMethod() {

ExternalApi mockApi = mock(ExternalApi.class);

doNothing().when(mockApi).sendNotification();

MyService service = new MyService(mockApi);

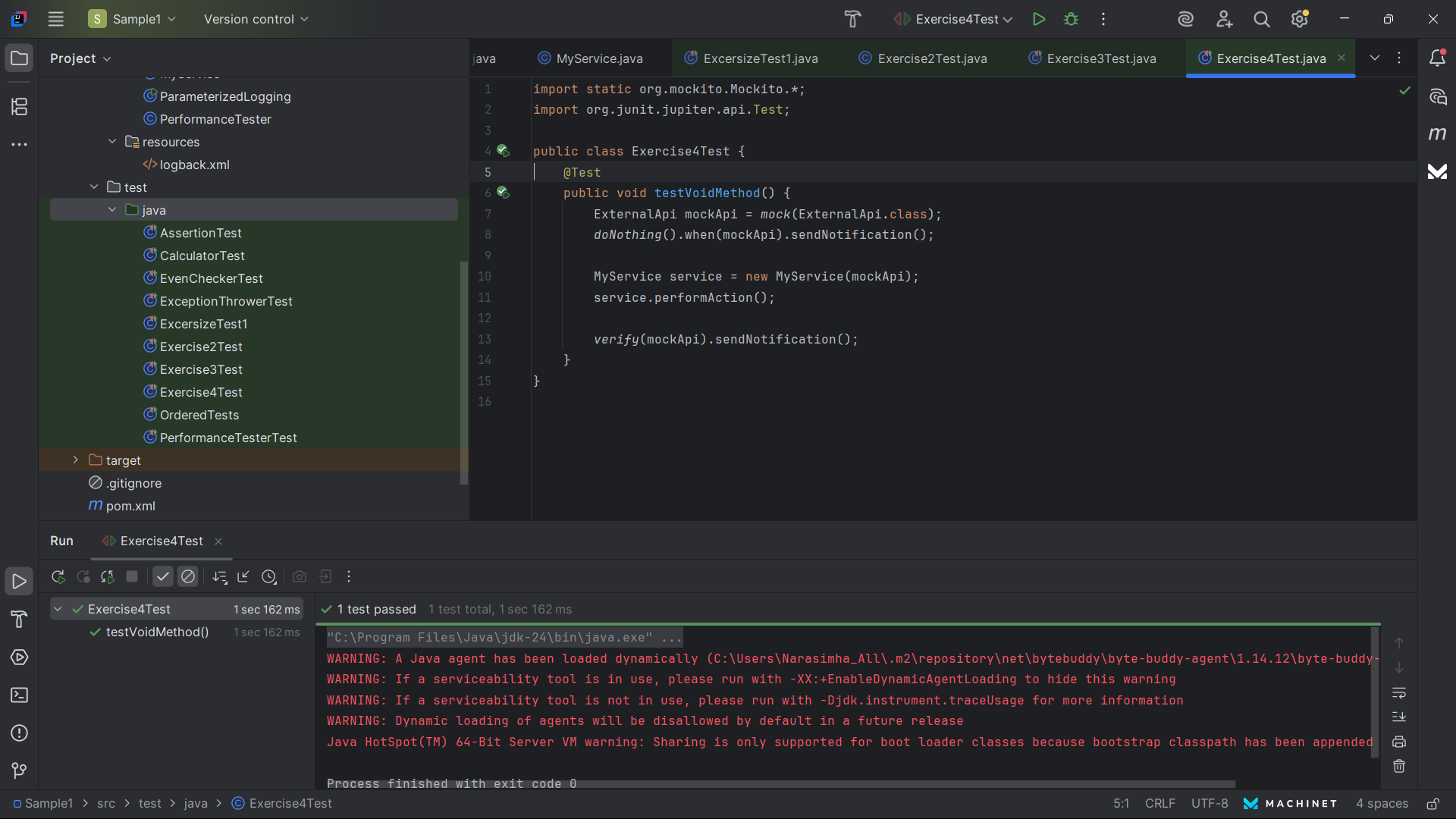
service.performAction();

verify(mockApi).sendNotification();

}

}

**Output :**



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

**CODE :**

**Exercise5Test.java**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

public class Exercise5Test {

@Test

public void testMultipleReturns() {

ExternalApi mockApi = mock(ExternalApi.class);

when(mockApi.getData())

.thenReturn("First")

.thenReturn("Second");

MyService service = new MyService(mockApi);

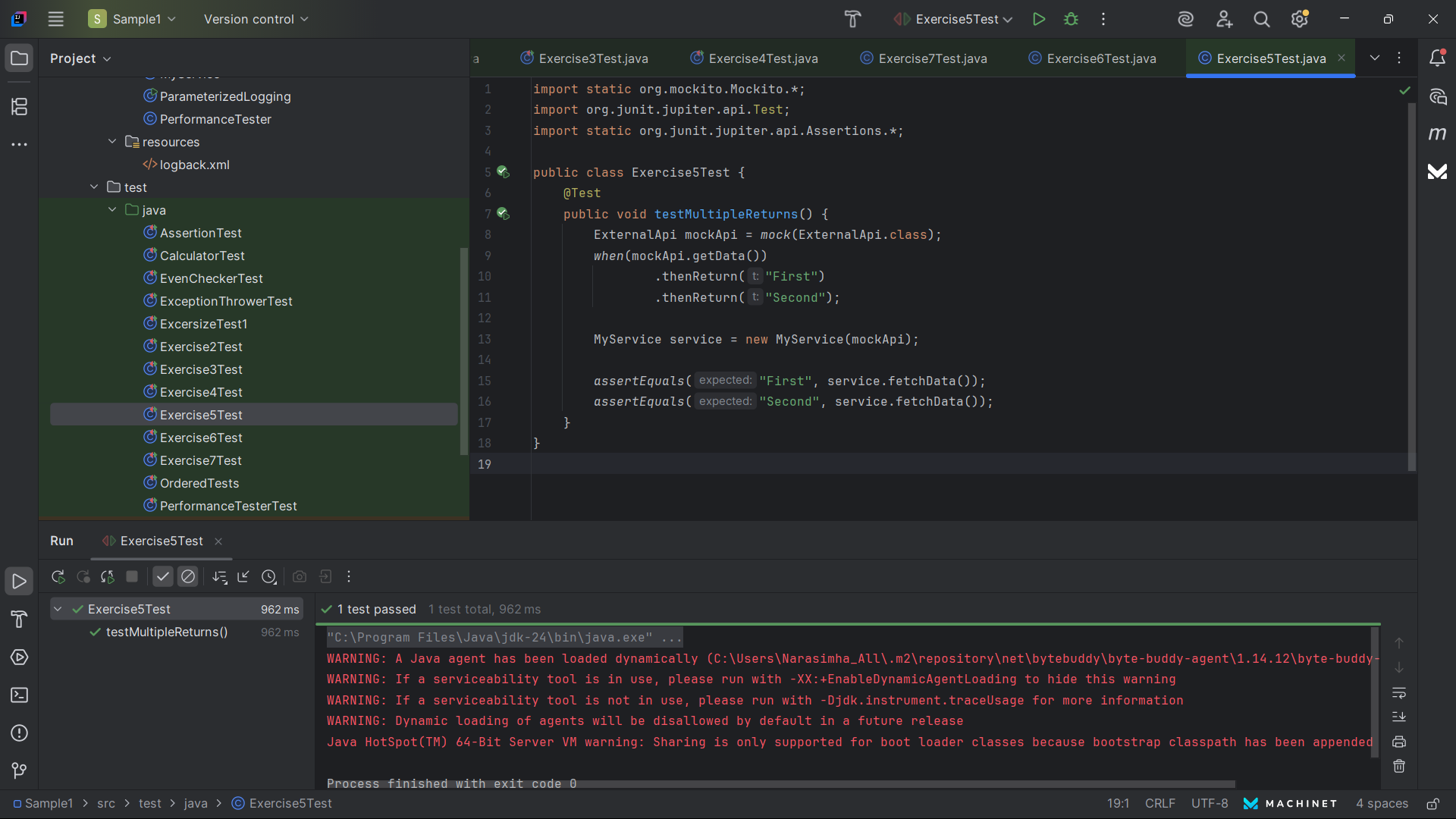
assertEquals("First", service.fetchData());

assertEquals("Second", service.fetchData());

}

}

**Output :**



**Exercise 6: Verifying Interaction Order**

**CODE :**

**Exercise6Test.java**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.InOrder;

public class Exercise6Test {

@Test

public void testInteractionOrder() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.callMethodsInOrder();

InOrder inOrder = inOrder(mockApi);

inOrder.verify(mockApi).start();

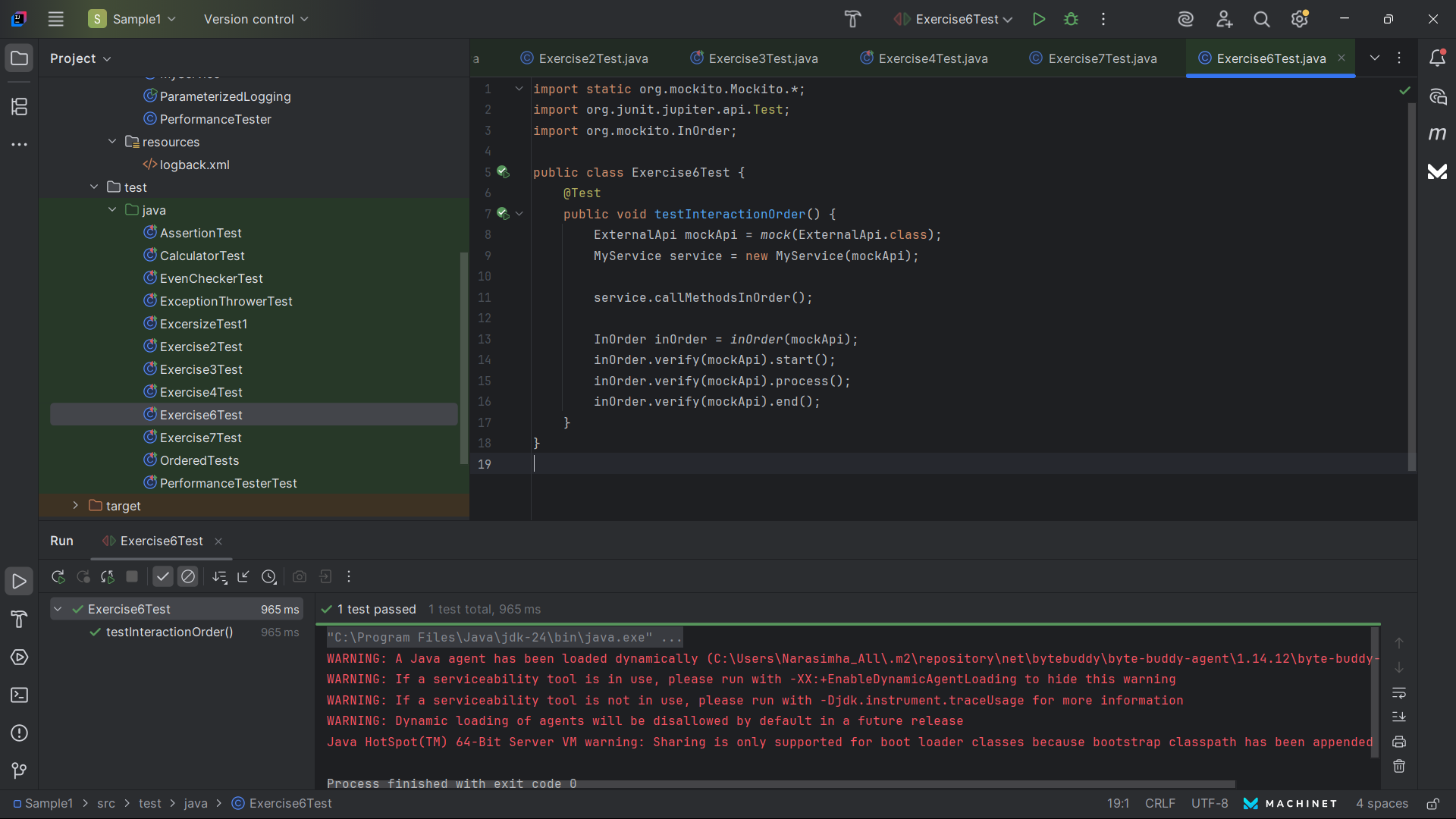
inOrder.verify(mockApi).process();

inOrder.verify(mockApi).end();

}

}

**Output :**



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

**CODE :**

**Exercise7Test.java**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

public class Exercise7Test {

@Test

public void testVoidMethodException() {

ExternalApi mockApi = mock(ExternalApi.class);

doThrow(new RuntimeException("Error!"))

.when(mockApi).sendNotification();

MyService service = new MyService(mockApi);

assertThrows(RuntimeException.class, () -> {

service.performAction();

});

verify(mockApi).sendNotification();

}

}

**Output :**

