**WEEK - 2**

**Mockito Exercises**

**Mandatory**

**Exercise 1: Mocking and Stubbing**

**ExternalAPI.java**

package com.example;

public interface ExternalApi {

String getData();

}

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

package com.example;

import static org.mockito.Mockito.\*;

import static org.junit.Assert.\*;

import org.junit.Test;

public class MyServiceTest {

@Test

public void testExternalApi() {

ExternalApi mockApi = mock(ExternalApi.class);

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

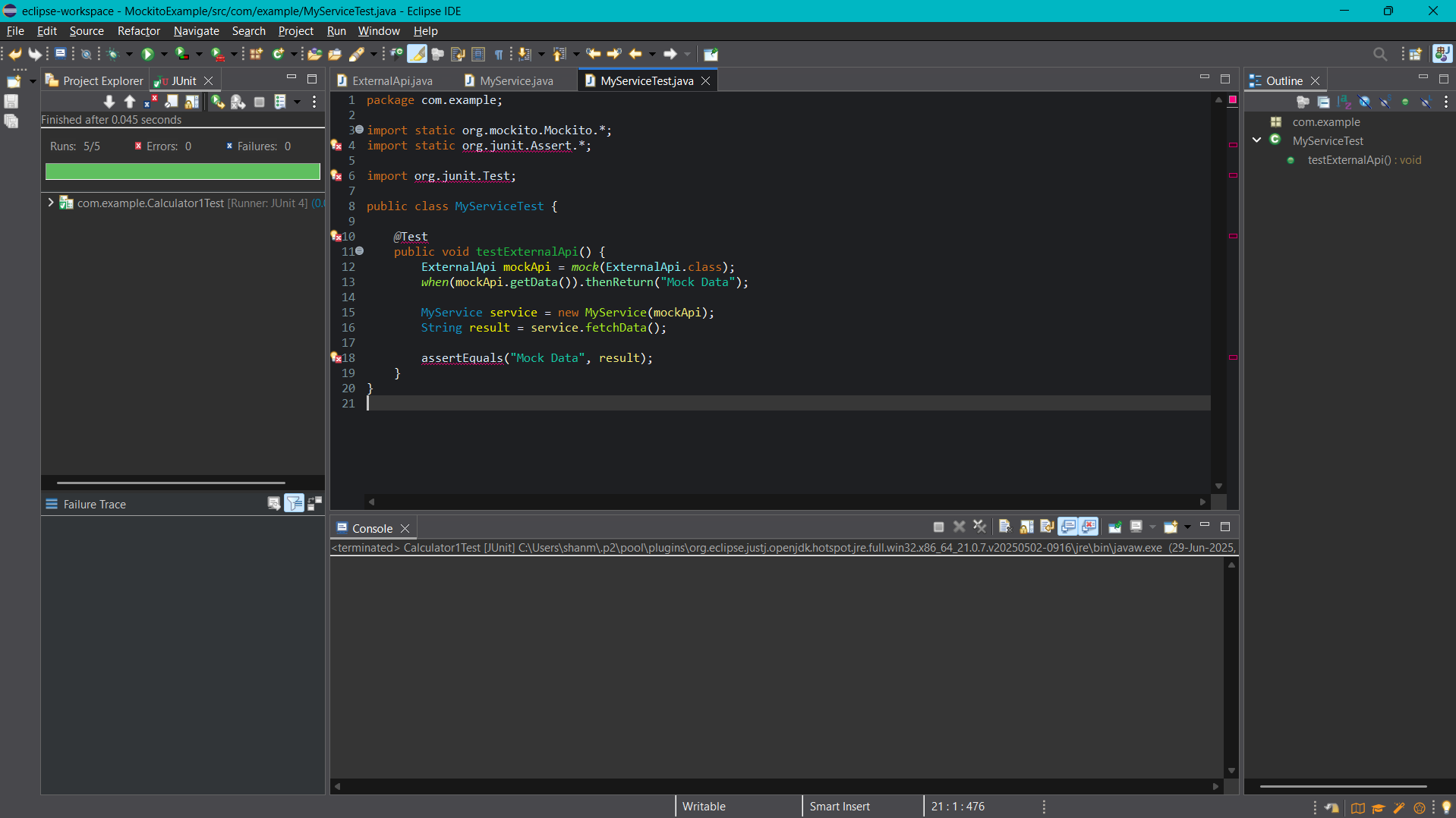
MyService service = new MyService(mockApi);

String result = service.fetchData();

assertEquals("Mock Data", result);

}

}



**Exercise 2: Verifying Interactions**

**ExternalAPI.java**

package com.example;

public interface ExternalAPI {

String getData();

}

**MyService.java**

package com.example;

public class MyService {

private ExternalAPI api;

public MyService(ExternalAPI api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

package com.example;

import static org.mockito.Mockito.mock;

import static org.mockito.Mockito.verify;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalAPI mockApi = mock(ExternalAPI.class);

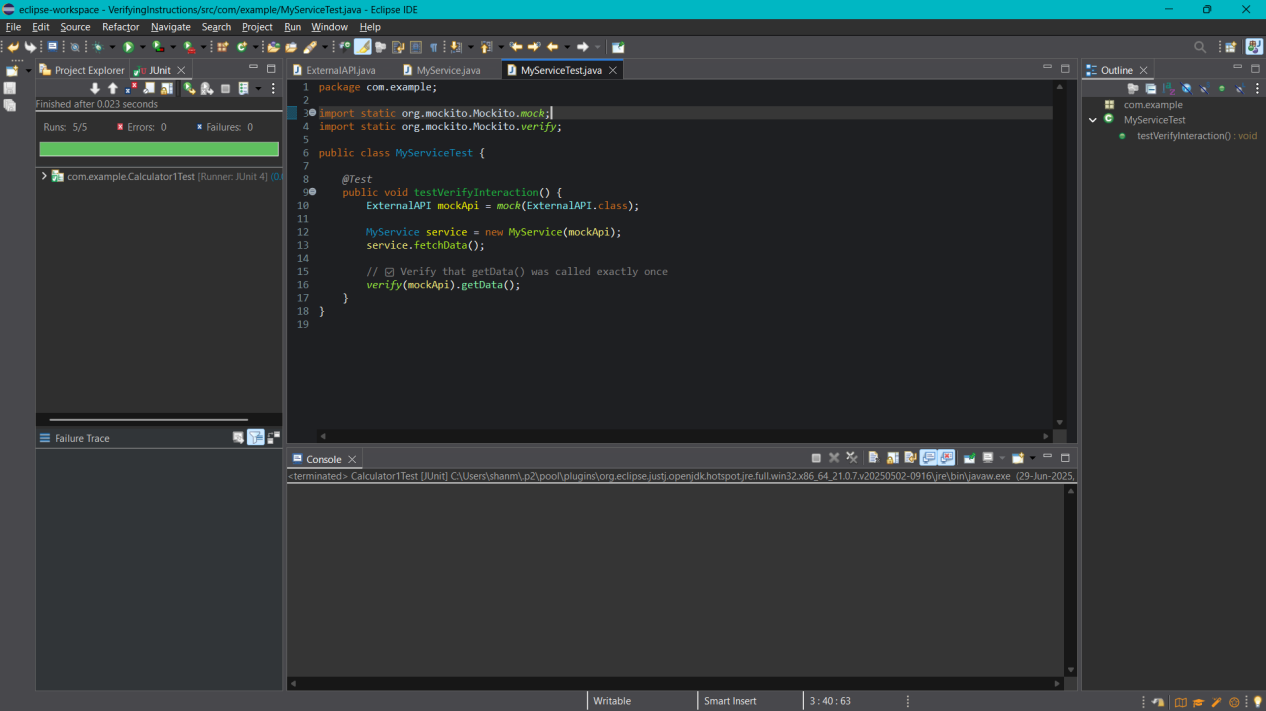
MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi).getData();

}

}



**Additional**

**Exercise 3: Argument Matching**

package com.example;

import static org.mockito.Mockito.\*;

import java.util.List;

import org.junit.Test;

import org.mockito.ArgumentMatchers;

public class ArgumentMatching {

@Test

public void testArgumentMatcher() {

@SuppressWarnings("unchecked")

List<String> mockedList = mock(List.class);

mockedList.get(0);

mockedList.get(1);

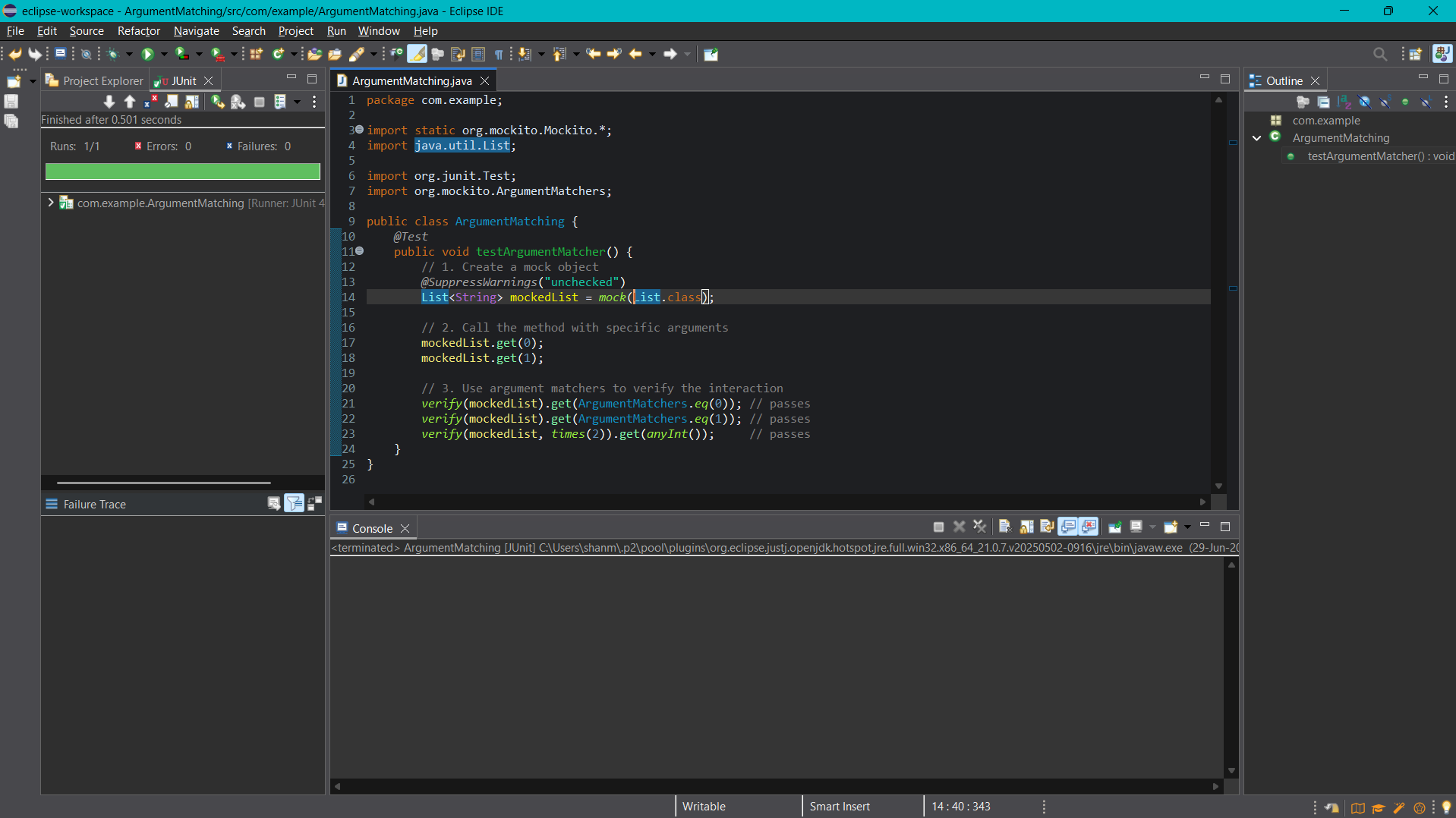
verify(mockedList).get(ArgumentMatchers.eq(0)); // passes

verify(mockedList).get(ArgumentMatchers.eq(1)); // passes

verify(mockedList, times(2)).get(anyInt()); // passes

}

}



**Exercise 4: Handling Void Methods**

package com.example;

import static org.mockito.Mockito.\*;

import org.junit.Test;

public class VoidMethods {

public interface Notifier {

void sendNotification(String message);

}

@Test

public void testVoidMethod() {

Notifier mockNotifier = mock(Notifier.class);

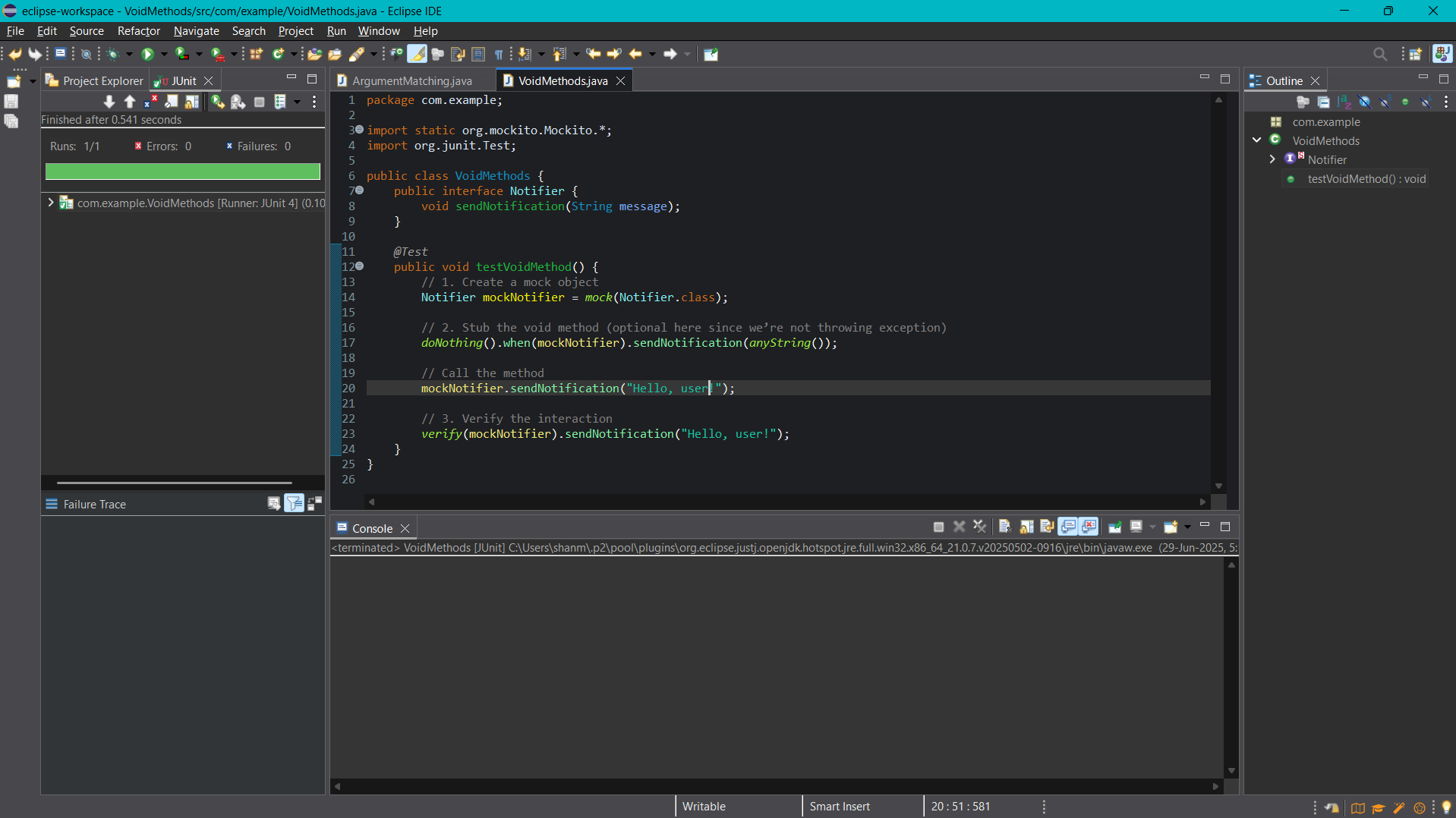
doNothing().when(mockNotifier).sendNotification(anyString());

mockNotifier.sendNotification("Hello, user!");

verify(mockNotifier).sendNotification("Hello, user!");

}

}



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

package com.example;

import static org.mockito.Mockito.\*;

import static org.junit.Assert.\*;

import org.junit.Test;

public class MultipleReturns {

public interface ExternalAPI {

double getExchangeRate();

}

@Test

public void testMultipleReturns() {

ExternalAPI mockApi = mock(ExternalAPI.class);

when(mockApi.getExchangeRate()).thenReturn(74.5, 75.0, 75.5);

assertEquals(74.5, mockApi.getExchangeRate(), 0.0);

assertEquals(75.0, mockApi.getExchangeRate(), 0.0);

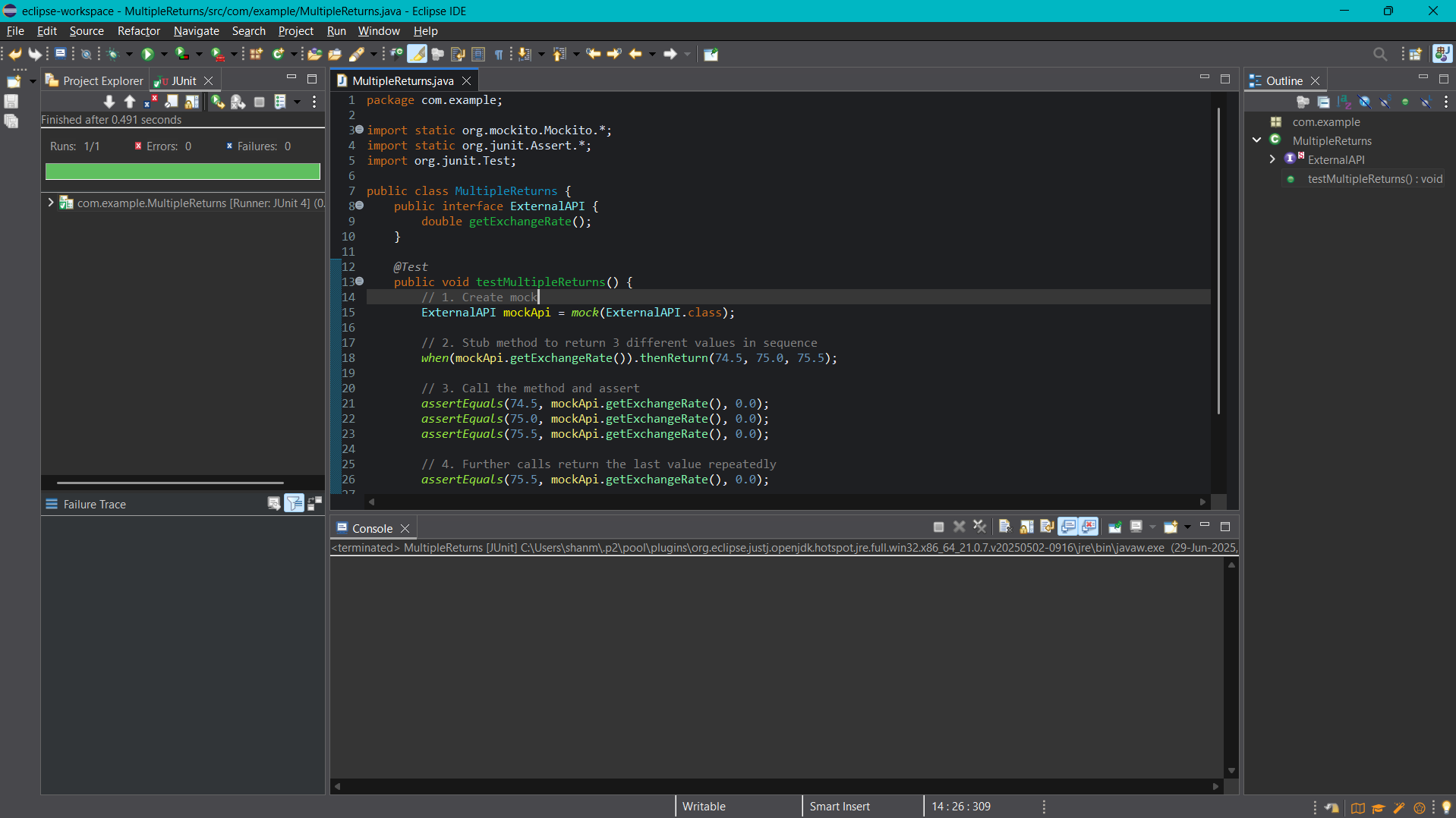
assertEquals(75.5, mockApi.getExchangeRate(), 0.0);

assertEquals(75.5, mockApi.getExchangeRate(), 0.0);

verify(mockApi, times(4)).getExchangeRate();

}

}



**Exercise 6: Verifying Interaction Order**

package com.example;

import static org.mockito.Mockito.\*;

import org.junit.Test;

import org.mockito.InOrder;

public class InteractionOrder {

public interface Service {

void stepOne();

void stepTwo();

void stepThree();

}

@Test

public void testMethodCallOrder() {

Service mockService = mock(Service.class);

mockService.stepOne();

mockService.stepTwo();

mockService.stepThree();

InOrder inOrder = inOrder(mockService);

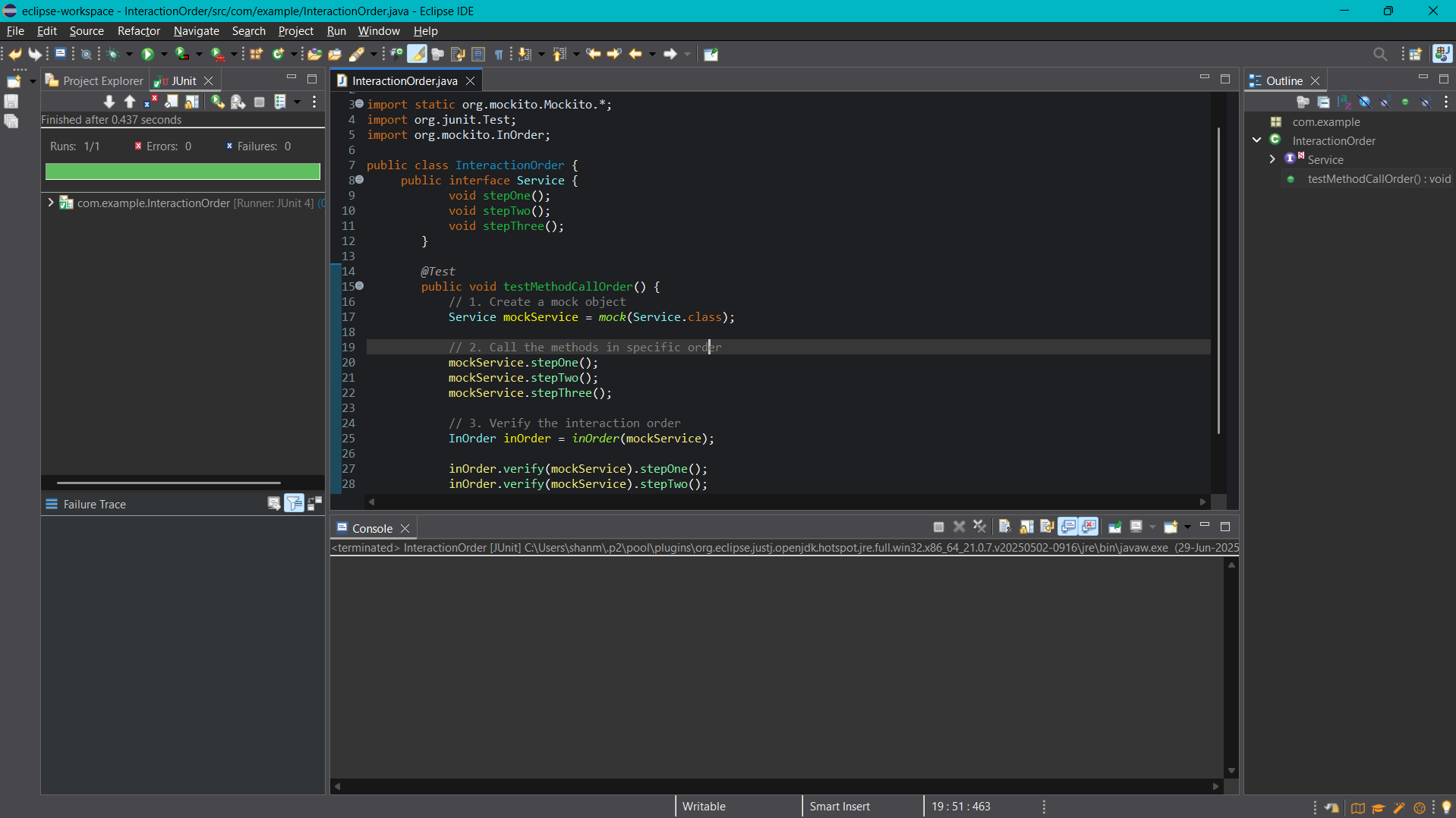
inOrder.verify(mockService).stepOne();

inOrder.verify(mockService).stepTwo();

inOrder.verify(mockService).stepThree();

}

}



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

package com.example;

import static org.mockito.Mockito.\*;

import org.junit.Test;

public class VoidMethodException {

public interface FileService {

void deleteFile(String filename);

}

@Test(expected = RuntimeException.class)

public void testVoidMethodThrowsException() {

FileService mockService = mock(FileService.class);

doThrow(new RuntimeException("File not found"))

.when(mockService).deleteFile("missing.txt");

mockService.deleteFile("missing.txt");

verify(mockService).deleteFile("missing.txt");

}

}

