**Mockito Exercises:**

**Project Structure**

MockitoExercises/

├── src/

│ ├── main/java/com/example/

│ │ └── ExternalApi.java, MyService.java

│ └── test/java/com/example/

│ └── MyServiceTest.java

├── pom.xml

**ExternalApi.java**

package com.example;

public interface ExternalApi {

    String getData();

    void sendData(String data);

}

**MyService.java**

package com.example;

public class MyService {

    private ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

    public void processAndSend(String data) {

        api.sendData(data + "-processed");

    }

}

**MyServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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

import org.mockito.InOrder;

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);

    }

    @Test

    public void testVerifyInteraction() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.fetchData();

        verify(mockApi).getData();

    }

    @Test

    public void testArgumentMatching() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.processAndSend("data");

        verify(mockApi).sendData(contains("data"));

    }

    @Test

    public void testVoidMethodStub() {

        ExternalApi mockApi = mock(ExternalApi.class);

        doNothing().when(mockApi).sendData(anyString());

        MyService service = new MyService(mockApi);

        service.processAndSend("Hello");

        verify(mockApi).sendData("Hello-processed");

    }

    @Test

    public void testMultipleReturnValues() {

        ExternalApi mockApi = mock(ExternalApi.class);

        when(mockApi.getData()).thenReturn("First", "Second");

        MyService service = new MyService(mockApi);

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

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

    }

    @Test

    public void testInteractionOrder() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.fetchData();

        service.processAndSend("X");

        InOrder inOrder = inOrder(mockApi);

        inOrder.verify(mockApi).getData();

        inOrder.verify(mockApi).sendData("X-processed");

    }

    @Test

    public void testVoidMethodThrowsException() {

        ExternalApi mockApi = mock(ExternalApi.class);

        doThrow(new RuntimeException("Error")).when(mockApi).sendData("BadData-processed");

        MyService service = new MyService(mockApi);

        assertThrows(RuntimeException.class, () -> {

            service.processAndSend("BadData");

        });

    }

}

**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>com.example</groupId>

    <artifactId>MockitoExercises</artifactId>

    <version>1.0-SNAPSHOT</version>

    <dependencies>

        <dependency>

            <groupId>org.mockito</groupId>

            <artifactId>mockito-core</artifactId>

            <version>5.10.0</version>

            <scope>test</scope>

        </dependency>

        <dependency>

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

            <artifactId>junit-jupiter</artifactId>

            <version>5.10.0</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>

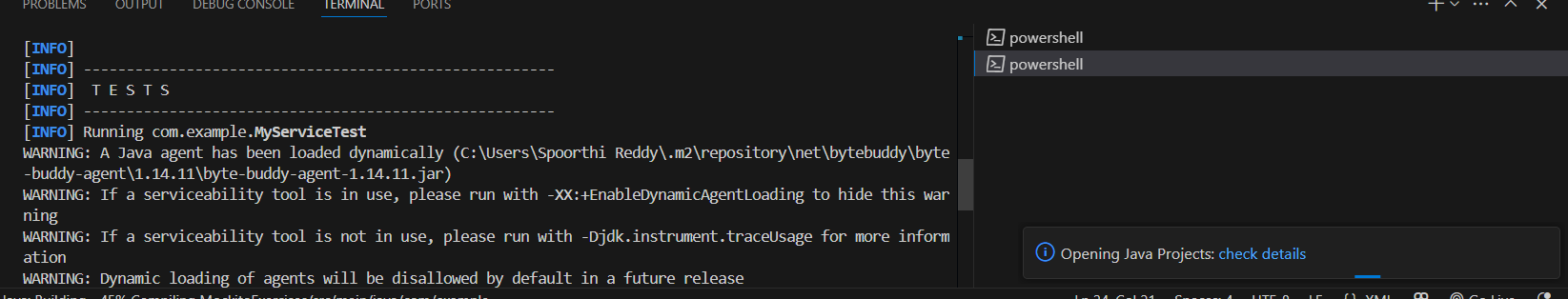
            </plugin>

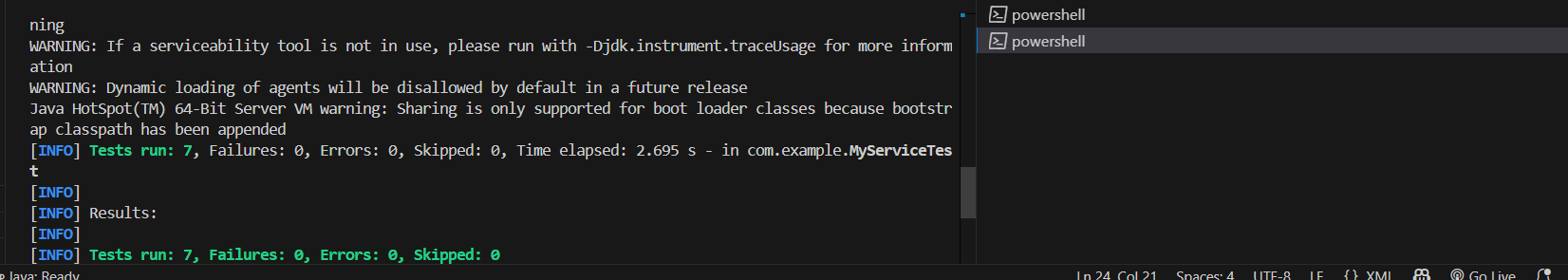
        </plugins>

    </build>

</project>

**OUTPUT:**

****

****

