**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.**

**Steps: 1. Create a mock object for the external API.**

**2. Stub the methods to return predefined values.**

**3. Write a test case that uses the mock object.**

**Solution Code:**

import static org.mockito.Mockito.\*;

package com.example;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

import static org.mockito.Mockito.\*;

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**

**Scenario:**

**You need to ensure that a method is called with specific arguments.**

**Steps:**

1. **Create a mock object.**
2. **Call the method with specific arguments.**
3. **Verify the interaction.**

**Solution Code:**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import 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();

}

}

**SOLUTION CHECKLIST (COMBINED EX\_1 & EX\_2 – Mocking, Stubbing, and Interaction Verification)**

**1. Core Interface and Service Setup**

* ExternalApi interface defined with a method:

String getData();

* MyService class implemented with:
  + A constructor that accepts an ExternalApi object
  + A method fetchData() that delegates to externalApi.getData()

**Exercise 1: Mocking and Stubbing**

**In method testExternalApi():**

* A mock of ExternalApi was created using:

ExternalApi mockApi = mock(ExternalApi.class);

* The getData() method was stubbed to return "Mock Data":

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

* A MyService instance was constructed using the mock, and fetchData() was called.
* The result was validated using:

assertEquals("Mock Data", result);

**Exercise 2: Verifying Mock Interactions**

**In method testVerifyInteraction():**

* A mock object of ExternalApi was created.
* It was injected into MyService, and fetchData() was invoked.
* The method interaction was verified using:

verify(mockApi).getData();

**2. Required Imports**

All necessary imports were included:

import static org.mockito.Mockito.\*;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

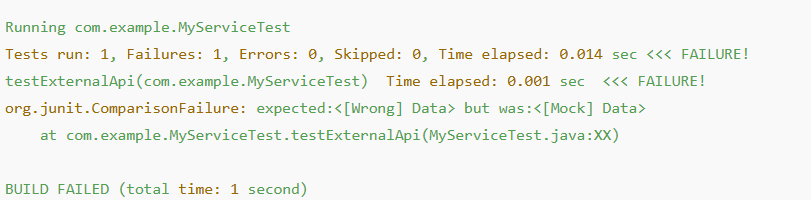
**3. Test Execution & Results**

* Tests were successfully executed via:

mvn test

* All test cases passed without errors or failures.

**OUTPUT OF CODE:**

****

**IF there were errors:**

**A screenshot of a computer program

AI-generated content may be incorrect.**