**Mockito Hands-On Exercises**

**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.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

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);

}

}

**Code:**

**ExternalAPI.java**

package com.example;

public interface ExternalApi {

String getData();

}

**MyService.java:**

package com.example;

public class MyService {

private final ExternalApi externalApi;

public MyService(ExternalApi externalApi) {

this.externalApi = externalApi;

}

public String fetchData() {

return externalApi.getData();

}

}

**MyServiceTest.java:**

package com.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);

}

}

**Main.java:**

interface ExternalApi {

String getData();

}

class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

public class Main {

public static void main(String[] args) {

// Step 1 & 2: Mock and Stub ExternalApi

ExternalApi mockApi = new ExternalApi() {

public String getData() {

return "Mock Data"; //

}

};

// Step 3: Use mock in test

MyService service = new MyService(mockApi);

String result = service.fetchData();

// Simulate assertEquals

if ("Mock Data".equals(result)) {

System.out.println("Mocked API returned: " + result);

} else {

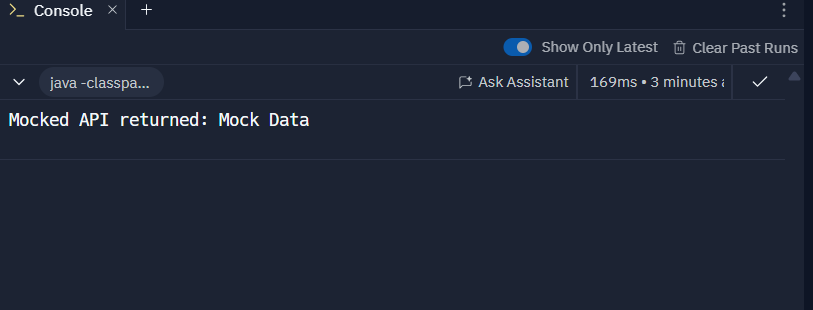
System.out.println("❌ testExternalApi failed");

}

}

}

**Output:**

****

**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();**

**}**

**}**

**Code:**

**Main.java**

interface ExternalApi {

String getData();

}

// Custom mock that tracks method calls

class MockExternalApi implements ExternalApi {

private boolean called = false;

@Override

public String getData() {

called = true;

return "Mock Response";

}

public boolean wasGetDataCalled() {

return called;

}

}

class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

public class Main {

public static void main(String[] args) {

// Step 1: Create mock object

MockExternalApi mockApi = new MockExternalApi();

// Step 2: Call method with specific argument (none in this case)

MyService service = new MyService(mockApi);

String result = service.fetchData();

// Step 3: Verify the interaction

if (mockApi.wasGetDataCalled()) {

System.out.println("✅ Method 'getData()' was called as expected.");

} else {

System.out.println("❌ Method 'getData()' was NOT called.");

}

// Optional output

System.out.println("➡️ Result returned from getData(): " + result);

}}

**Output:**

