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.

**MyService.java:**

package sample;

public class MyService {

private ExternalApi externalApi;

public MyService(ExternalApi externalApi) {

this.externalApi = externalApi;

}

public String fetchData() {

return externalApi.getData();

}

}

**ExternalApi.java:**

package sample;

public interface ExternalApi {

String getData();

}

**MyServiceTest.java:**

package sample;

import static org.mockito.Mockito.\*;

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

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);

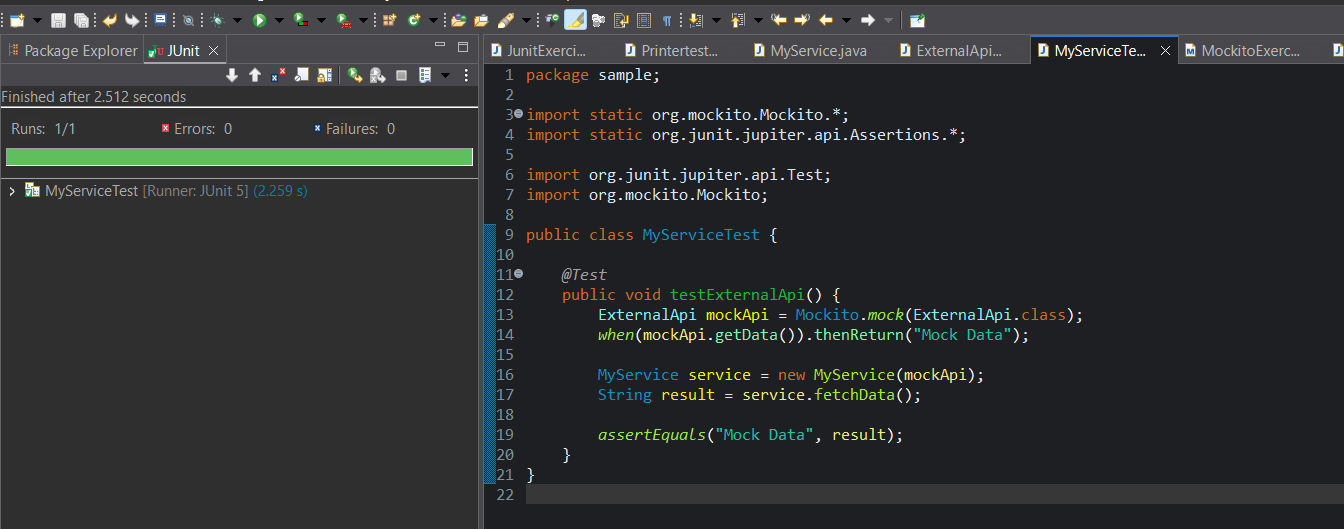
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

## Output:



## Exercise 2: Verifying Interactions:

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

**ExternalApi.java:**

package sample;

public interface ExternalApi {

String getData();

}

**MyService.java:**

package sample;

public class MyService {

private ExternalApi externalApi;

public MyService(ExternalApi externalApi) {

this.externalApi = externalApi;

}

public String fetchData() {

return externalApi.getData();

}

}

**Interaction.java:**

package sample;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class Interaction {

*@Test*

public void verifyGetDataCalledOnce() {

ExternalApi mockApi = Mockito.*mock*(ExternalApi.class);

MyService service = new MyService(mockApi);

service.fetchData();

*verify*(mockApi).getData();

}

}

## Output:

