**Exercise 1: Mocking and Stubbing**

**Scenario:**  
The objective is to test a service class (MyService) that interacts with an external API. Rather than invoking the actual API, Mockito is used to mock the external dependency and stub its behavior to return controlled data during testing.

**Objective:**  
To isolate the MyService class by mocking its dependency (ExternalApi) using Mockito and validate that the service behaves as expected based on predefined mock responses.

**Steps:**

1. Defined the ExternalApi interface representing the external service.
2. Used Mockito.mock() to create a mock implementation of ExternalApi.
3. Stubbed the method getData() to return a fixed string value.
4. Passed the mock to MyService through its constructor.
5. Validated the output using a JUnit assertion to ensure expected behavior.

**Code:**

package com.example;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

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);

}

}

**Supporting Classes**

package com.example;

public interface ExternalApi {

String getData();

}

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

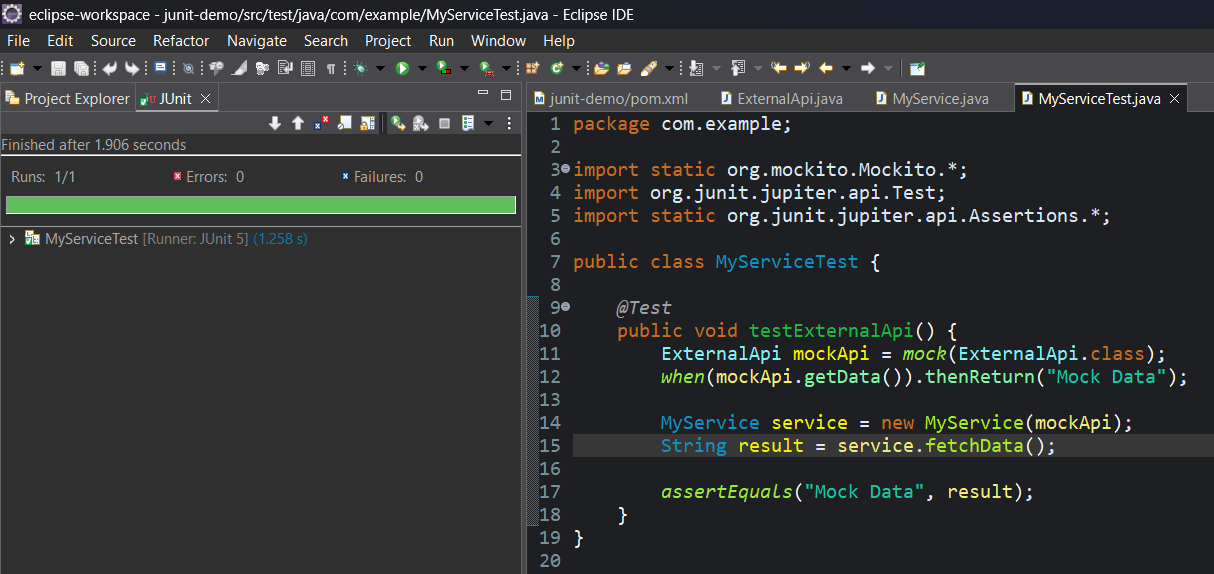
public String fetchData() {

return api.getData();

}

}

**Output:**



**Exercise 2: Verifying Interactions**

**Scenario:**  
This exercise focuses on validating that a specific method is invoked with the expected parameters. Instead of only verifying the return value, it ensures that interactions between components occur as intended.

**Objective:**  
To confirm that a method on a mocked dependency is called during the execution of the service logic, using Mockito.verify() for interaction verification.

**Steps:**

1. A mock object of the ExternalApi interface was created using Mockito.
2. The MyService class was instantiated with the mock dependency.
3. The service method fetchData() was called.
4. Mockito’s verify() method was used to confirm that getData() was invoked on the mock.

**Code:**

package com.example;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

*@Test*

public void testVerifyInteraction() {

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

MyService service = new MyService(mockApi);

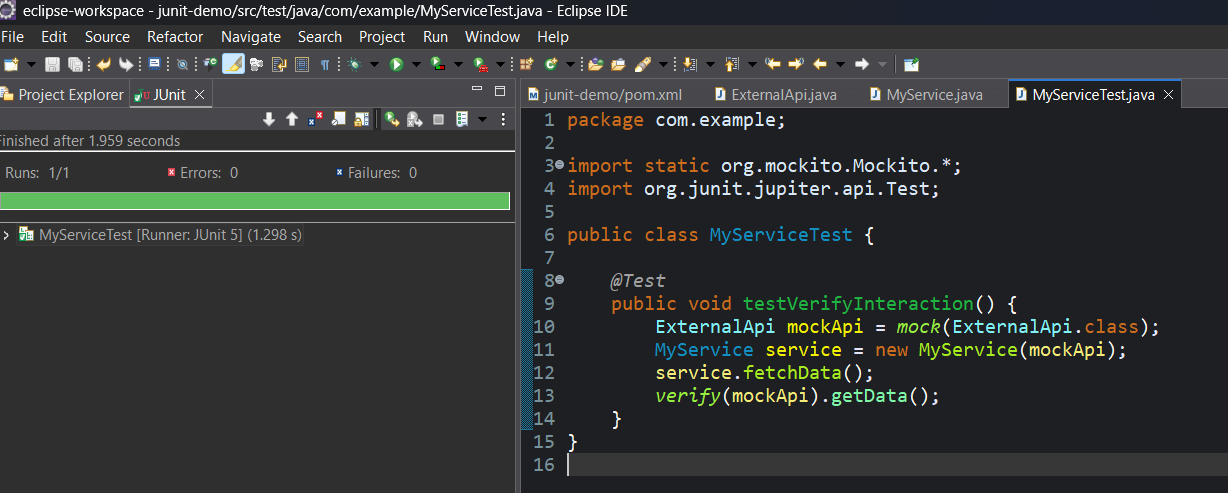
service.fetchData();

*verify*(mockApi).getData();

}

}

**Output:**

****