**Exercise 1 : Mocking and Stubbing**

This file demonstrates how to use Mockito with JUnit to test a service that depends on an external API.

**🡪 Components**

**- ExternalApi:**

A simple class simulating an external dependency with a `getData()` method.

**- MyService:**

A service class that depends on `ExternalApi`. It has a `fetchData()` method that calls `getData()` from the API.

**- Ex1 (Test Class):**

Contains a JUnit test method `**testExternalApi()**` that:

1. Creates a mock of `**ExternalApi**` using Mockito.

2. Stubs the `**getData()**` method to return `"Mock Data"`.

3. Injects the mock into `**MyService**`.

4. Calls `**fetchData()**` and asserts that the returned value is `"**Mock Data**"`.

**🡪 Purpose**

- **Mocking**: Isolates `**MyService**` from the real `**ExternalApi**` for testing.

- **Stubbing**: Controls the return value of `**getData**()` to test `**MyService**` behavior.

- **Unit Testing:** Ensures `**MyService**` interacts correctly with its dependency.

🡪**Code Snippet :**

import org.junit.Test;

import static org.junit.Assert.\*;

import static org.mockito.Mockito.\*;

class ExternalApi {

    public String getData() {

        return "Real Data";

    }

}

class MyService {

    private ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

}

public class Ex1 {

    @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);

    }

}