TDD using JUnit5 and Mockito

**Candidate Name:** Ayush Singh  
**Superset ID:** 6376485

3. Mockito 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);

}

}

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();

}

}

Explaination

# Mockito Hands-On Exercises

This folder contains unit tests written using \*\*Mockito\*\* to mock dependencies and verify interactions.

✅ Both exercises are implemented within a \*\*single Mockito project\*\* (`MyServiceTest.java`).

✅ Exercise 1: Mocking and Stubbing

- File: `MyServiceTest.java`

- Mocks an `ExternalApi` dependency.

- Uses `when(...).thenReturn(...)` to stub method return values.

- Asserts expected result using `assertEquals`.

✅ Exercise 2: Verifying Interactions

- File: `MyServiceTest.java`

- Uses `verify(mock).method()` to ensure that the mocked method was called after execution.

---

🧩 Project Structure

- `MyService.java`: Main service class that depends on `ExternalApi`.

- `ExternalApi.java`: Interface to be mocked.

- `MyServiceTest.java`: Test class containing both exercises.

---

Requirements

- Mockito (`org.mockito:mockito-core`)

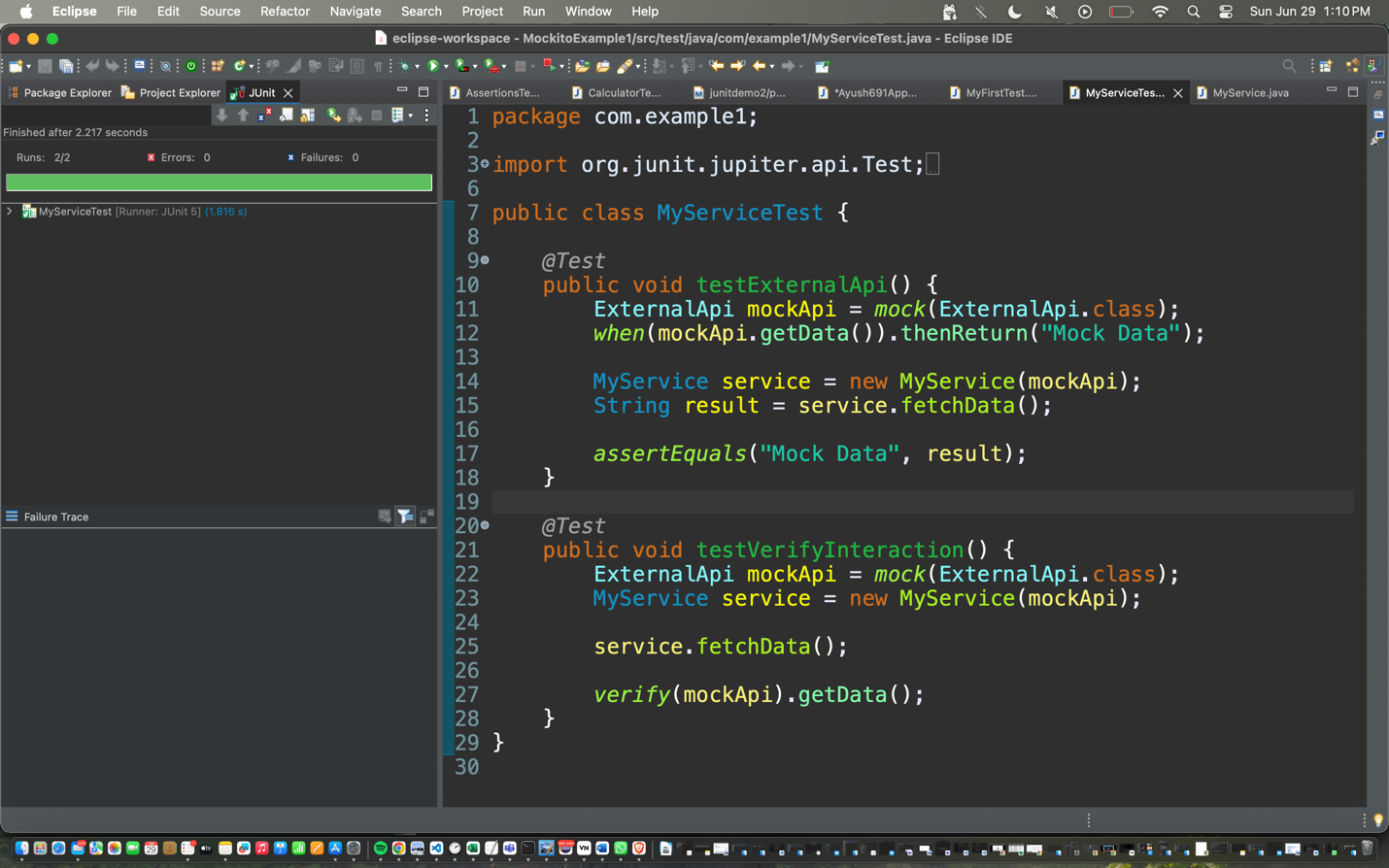
- JUnit 5 (`org.junit.jupiter:junit-jupiter`)

- Maven Surefire Plugin (for running tests)

---

Note

Both exercises are implemented in a single class `MyServiceTest` for simplicity and clarity.

HandsOn:

OUTPUT

