**WEEK-02 HANDS ON SOLUTIONS**

**Mockito Hands-On Exercises**

**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();

} }

**EXPLANATION:**

In this exercise, a unit test was constructed to verify that the method getData() from an external dependency is invoked when the fetchData() method of the MyService class is called.

The test was written using the JUnit 5 framework along with Mockito for mocking and interaction verification.

The test class MyServiceTest contains the method testVerifyInteraction() annotated with @Test, signalling JUnit to treat it as a unit test method.

Inside this method, a mock instance of the ExternalApi interface was created using Mockito.mock(ExternalApi.class).

This mock does not execute real logic but records how it is used. An instance of MyService was then created, with the mock API injected via its constructor.

Upon calling service.fetchData(), the mock's getData() method was expected to be triggered internally by the service. Finally, the line verify(mockApi).getData() checks whether getData() was indeed called.

If the method was not called,

Mockito throws an org.mockito.exceptions.verification.WantedButNotInvoked error, causing the test to fail.

The test executed successfully, confirming that the interaction occurred as intended.

