Name: Palanivel M

Superset ID: 6373179

Mail ID: [727822tuec135@skct.edu.in](mailto:727822tuec135@skct.edu.in)

## 

# Week 2 Hands-on Exercise

**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.

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);

}

}

CODE:

ExternalApi.java:

**public** **interface** ExternalApi {

String getUserName(**int** userId);

**int** getUserAge(**int** userId);

String getData();

}

MyService.java:

**public** **class** MyService {

**private** ExternalApi api;

**public** MyService(ExternalApi api) {

**this**.api = api;

}

**public** String fetchData() {

**return** api.getData();

}

**public** String getUserSummary(**int** userId) {

String name = api.getUserName(userId);

**int** age = api.getUserAge(userId);

**return** "User: " + name + ", Age: " + age;

}

}

MyServiceTest.java:

**import** **static** org.mockito.Mockito.\*;

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

**import** org.junit.jupiter.api.Test;

**public** **class** MyServiceTest {

@Test

**public** **void** testGetUserSummary() {

// Create mock for ExternalApi

ExternalApi mockApi = *mock*(ExternalApi.**class**); *when*(mockApi.getData()).thenReturn("Mock Data");

*when*(mockApi.getUserName(101)).thenReturn("Alice");

*when*(mockApi.getUserAge(101)).thenReturn(25);

MyService service = **new** MyService(mockApi);

String summary = service.getUserSummary(101);

String result = service.fetchData();

System.***out***.println("Fetched from mocked API: " + result);

System.***out***.println("User Summary from Mocked API: " + summary);

*assertEquals*("User: Alice, Age: 25", summary);

}

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**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();

}

}

CODE:

ExternalApi.java:

**public** **interface** ExternalApi {

**void** getData(); // No return type for this example

}

MyService.java:

**public** **class** MyService {

**private** ExternalApi api;

**public** MyService(ExternalApi api) {

**this**.api = api;

}

**public** **void** fetchData() {

System.***out***.println("Inside MyService.fetchData()");

api.getData();

}

}

MyServiceTest.java:

**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();

System.***out***.println("Verified: getData() was called on ExternalApi");

}

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.