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);

} }

SOLUTION:

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.extension.ExtendWith;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.junit.jupiter.MockitoExtension;

import static org.mockito.Mockito.\*;

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

@ExtendWith(MockitoExtension.class)

class MyServiceTest {

@Mock

ExternalApi externalApi;

@InjectMocks

MyService myService;

@Test

void fetchDataShouldReturnMockValue() {

when(externalApi.getData()).thenReturn("Mock Response");

String result = myService.fetchData();

assertEquals("Mock Response", result);

}

}public interface ExternalApi {

String getData();

}

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

Requirements:

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

Console Output:

Test passed: fetchDataShouldReturnMockValue

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();

} }

SOLUTION:

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.extension.ExtendWith;

import org.mockito.Mock;

import org.mockito.InjectMocks;

import org.mockito.junit.jupiter.MockitoExtension;

import static org.mockito.Mockito.verify;

@ExtendWith(MockitoExtension.class)

class MyServiceTest {

@Mock

ExternalApi externalApi;

@InjectMocks

MyService myService;

@Test

void shouldCallExternalApiGetData() {

myService.fetchData();

verify(externalApi).getData();

}

}

public interface ExternalApi {

String getData();

}

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

OUTPUT:

Tests run: 1, Failures: 0, Errors: 0