**SOLUTION WEEK 2**

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

public class MyService {

       private final ExternalApi externalApi;

    public MyService(ExternalApi externalApi) {

        this.externalApi = externalApi;

    }

public String fetchData() {

  return externalApi.getData();

    }

}

interface ExternalApi {

    String getData();

}

import org.mockito.Mockito;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.when;

public class MyServiceTest {

    @Test

public void testExternalApi() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

when(mockApi.getData()).thenReturn("Mock Data");

MyService service = new MyService(mockApi);

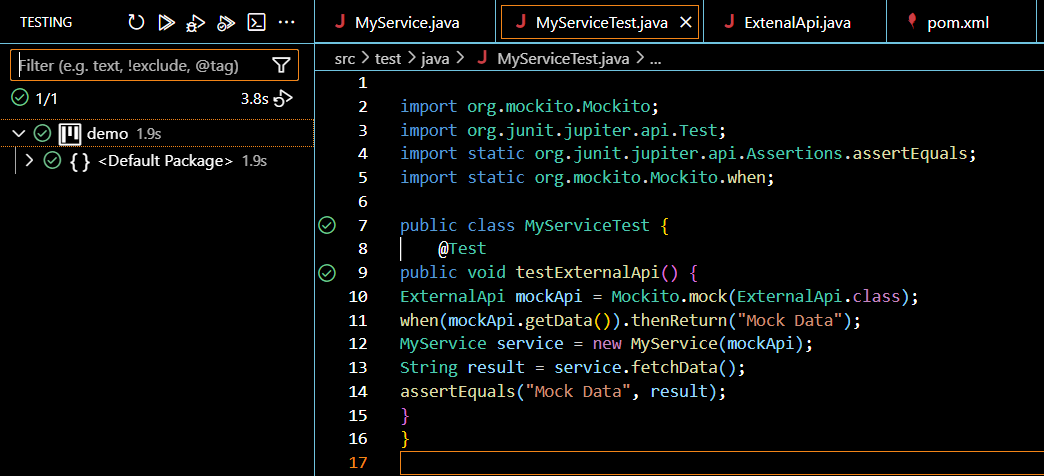
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

OUTPUT



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

public class MyService {

       private final ExternalApi externalApi;

    public MyService(ExternalApi externalApi) {

        this.externalApi = externalApi;

    }

    public String fetchData() {

        return externalApi.getData();

    }

}

interface ExternalApi {

    String getData();

}

import org.mockito.Mockito;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.mock;

import static org.mockito.Mockito.verify;

import static org.mockito.Mockito.when;

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);

}

@Test

    public void testVerifyInteraction(){

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.fetchData();

        verify(mockApi).getData();

    }

}

OUTPUT

