**Common Interfaces & Classes (Used Across Exercises)**

**ExternalApi.java**

public interface ExternalApi {

String getData();

void sendData(String data);

void notifyUser();

}

**MyService.java**

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

public void saveData(String data) {

api.sendData(data);

}

public void notifyUser() {

api.notifyUser();

}

}

**Exercise 1: Mocking and Stubbing**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest1 {

@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**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest2 {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi).getData();

}

}

**Exercise 3: Argument Matching**

import static org.mockito.Mockito.\*;

import static org.mockito.ArgumentMatchers.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest3 {

@Test

public void testArgumentMatching() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.saveData("Hello");

verify(mockApi).sendData(eq("Hello"));

verify(mockApi, never()).sendData(eq("Bye"));

}

}

**Exercise 4: Handling Void Methods**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest4 {

@Test

public void testVoidMethod() {

ExternalApi mockApi = mock(ExternalApi.class);

doNothing().when(mockApi).notifyUser();

MyService service = new MyService(mockApi);

service.notifyUser();

verify(mockApi).notifyUser();

}

}

**Exercise 5: Multiple Returns**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

public class MyServiceTest5 {

@Test

public void testMultipleReturns() {

ExternalApi mockApi = mock(ExternalApi.class);

when(mockApi.getData()).thenReturn("First", "Second", "Third");

MyService service = new MyService(mockApi);

assertEquals("First", service.fetchData());

assertEquals("Second", service.fetchData());

assertEquals("Third", service.fetchData());

}

}

**Exercise 6: Verifying Interaction Order**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.InOrder;

public class MyServiceTest6 {

@Test

public void testInteractionOrder() {

ExternalApi mockApi = mock(ExternalApi.class);

mockApi.sendData("Step 1");

mockApi.getData();

mockApi.notifyUser();

InOrder inOrder = inOrder(mockApi);

inOrder.verify(mockApi).sendData("Step 1");

inOrder.verify(mockApi).getData();

inOrder.verify(mockApi).notifyUser();

}

}

**Exercise 7: Void Method Throws Exception**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

public class MyServiceTest7 {

@Test

public void testVoidMethodThrowsException() {

ExternalApi mockApi = mock(ExternalApi.class);

doThrow(new RuntimeException("Failed")).when(mockApi).notifyUser();

MyService service = new MyService(mockApi);

assertThrows(RuntimeException.class, () -> service.notifyUser());

}

}