**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.

**Solution Code:**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

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

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);

}

}

**Exercise 2: Verifying Interactions**

**Scenario:**  
You need to ensure that a method is called with specific arguments.

**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();

}

}

**Exercise 3: Argument Matching**

**Scenario:**  
You need to verify that a method is called with specific arguments.

**Solution Code:**

import static org.mockito.Mockito.\*;

import static org.mockito.ArgumentMatchers.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testArgumentMatching() {

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

MyService service = new MyService(mockApi);

service.process("test");

verify(mockApi).process(eq("test"));

}

}

**Exercise 4: Handling Void Methods**

**Scenario:**  
You need to test a void method that performs some action.

**Solution Code:**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVoidMethod() {

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

doNothing().when(mockApi).log(anyString());

MyService service = new MyService(mockApi);

service.logMessage("Hello");

verify(mockApi).log("Hello");

}

}

**Exercise 5: Mocking and Stubbing with Multiple Returns**

**Scenario:**  
You need to test a service that depends on an external API with multiple return values.

**Solution Code:**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

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

public class MyServiceTest {

@Test

public void testMultipleReturns() {

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

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

MyService service = new MyService(mockApi);

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

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

}

}

**Exercise 6: Verifying Interaction Order**

**Scenario:**  
You need to ensure that methods are called in a specific order.

**Solution Code:**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.InOrder;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testInteractionOrder() {

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

MyService service = new MyService(mockApi);

service.doStepOne();

service.doStepTwo();

InOrder inOrder = inOrder(mockApi);

inOrder.verify(mockApi).stepOne();

inOrder.verify(mockApi).stepTwo();

}

}

**Exercise 7: Handling Void Methods with Exceptions**

**Scenario:**  
You need to test a void method that throws an exception.

**Solution Code:**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVoidMethodException() {

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

doThrow(new RuntimeException("Error")).when(mockApi).log(anyString());

MyService service = new MyService(mockApi);

try {

service.logMessage("fail");

} catch (Exception e) {

System.out.println("Caught expected exception: " + e.getMessage());

}

verify(mockApi).log("fail");

}

}