**File: Mockito\_Hands\_On\_Exercises**

**Exercise 1: Mocking and Stubbing**

**Steps:**

1. Create a mock object for the external API.
2. Stub its methods to return predefined values.
3. Write a test case that uses the mock.

Example:

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

public class MyServiceTest {

@Test

public void testExternalApi() {

ExternalApi mockApi = 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**

**Steps:**

1. Create a mock object.
2. Call the method with specific arguments.
3. Verify the interaction.

Example:

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();

}

}

**Exercise 3: Argument Matching**

**Steps:**

1. Create a mock object.
2. Call the method with arguments.
3. Use argument matchers to verify.

Example:

import static org.mockito.Mockito.\*;

import static org.mockito.ArgumentMatchers.\*;

import org.junit.jupiter.api.Test;

public class ArgumentMatcherTest {

@Test

public void testArgumentMatcher() {

ExternalApi mockApi = mock(ExternalApi.class);

mockApi.sendData("Test Message");

verify(mockApi).sendData(anyString());

}

}

**Exercise 4: Handling Void Methods**

**Steps:**

1. Create a mock object.
2. Stub the void method.
3. Verify the interaction.

Example:

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class VoidMethodTest {

@Test

public void testVoidMethod() {

Logger mockLogger = mock(Logger.class);

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

mockLogger.log("Testing void method");

verify(mockLogger).log("Testing void method");

}

}

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

**Steps:**

1. Create a mock object.
2. Stub methods to return different values consecutively.
3. Write a test case.

Example:

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

public class MultipleReturnValuesTest {

@Test

public void testMultipleReturns() {

ExternalApi mockApi = mock(ExternalApi.class);

when(mockApi.getData())

.thenReturn("First Call")

.thenReturn("Second Call");

assertEquals("First Call", mockApi.getData());

assertEquals("Second Call", mockApi.getData());

}

}

**Exercise 6: Verifying Interaction Order**

**Steps:**

1. Create a mock object.
2. Call methods in specific order.
3. Verify the order.

Example:

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.InOrder;

public class InteractionOrderTest {

@Test

public void testOrder() {

ExternalApi mockApi = mock(ExternalApi.class);

mockApi.connect();

mockApi.getData();

mockApi.disconnect();

InOrder inOrder = inOrder(mockApi);

inOrder.verify(mockApi).connect();

inOrder.verify(mockApi).getData();

inOrder.verify(mockApi).disconnect();

}

}

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

**Steps:**

1. Create a mock object.
2. Stub the void method to throw an exception.
3. Verify the interaction.

Example:

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class VoidMethodExceptionTest {

@Test

public void testVoidMethodException() {

Logger mockLogger = mock(Logger.class);

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

try {

mockLogger.log("Test");

} catch (RuntimeException e) {

assert(e.getMessage().equals("Error"));

}

}

}