## Exercise 3: Argument Matching

**Scenario:** Verify that a method is called with specific arguments using argument matchers.

**Example Code:**

java

CopyEdit

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.processData("input1");

verify(mockApi).sendData(eq("input1"));

}

}

## Exercise 4: Handling Void Methods

**Scenario:** Stub a void method and verify the interaction.

**Example Code:**

java

CopyEdit

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);

// Stubbing a void method to do nothing (this is default, but shown here explicitly)

doNothing().when(mockApi).clearCache();

MyService service = new MyService(mockApi);

service.clearExternalCache();

verify(mockApi).clearCache();

}

}

## Exercise 5: Mocking and Stubbing with Multiple Returns

**Scenario:** Stub the method to return different values on consecutive calls.

**Example Code:**

java

CopyEdit

import static org.mockito.Mockito.\*;import org.junit.jupiter.api.Test;import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testMultipleReturns() {

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

when(mockApi.getData())

.thenReturn("First Call")

.thenReturn("Second Call");

MyService service = new MyService(mockApi);

String first = service.fetchData();

String second = service.fetchData();

assertEquals("First Call", first);

assertEquals("Second Call", second);

}

}

## Exercise 6: Verifying Interaction Order

**Scenario:** Ensure methods are called in a specific order.

**Example Code:**

java

CopyEdit

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.initialize();

service.fetchData();

InOrder inOrder = inOrder(mockApi);

inOrder.verify(mockApi).initializeConnection();

inOrder.verify(mockApi).getData();

}

}

## Exercise 7: Handling Void Methods with Exceptions

**Scenario:** Stub a void method to throw an exception.

**Example Code:**

java

CopyEdit

import static org.mockito.Mockito.\*;import org.junit.jupiter.api.Test;import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVoidMethodWithException() {

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

doThrow(new RuntimeException("Failure"))

.when(mockApi).clearCache();

MyService service = new MyService(mockApi);

try {

service.clearExternalCache();

fail("Expected RuntimeException");

} catch (RuntimeException ex) {

assertEquals("Failure", ex.getMessage());

}

verify(mockApi).clearCache();

}

}