**Exercise 1: Mocking and Stubbing**

**Scenario:** Test a service that depends on an external API.

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:** Ensure a method is called with specific arguments.

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:** Use argument matchers to verify interactions.

import static org.mockito.Mockito.\*;

import static org.mockito.ArgumentMatchers.anyString;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testArgumentMatching() {

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

when(mockApi.sendData(anyString())).thenReturn(true);

MyService service = new MyService(mockApi);

boolean result = service.send("data");

assertTrue(result);

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

}

}

**Exercise 4: Handling Void Methods**

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

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).logInfo();

MyService service = new MyService(mockApi);

service.log();

verify(mockApi).logInfo();

}

}

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

**Scenario:** Return different values on consecutive calls.

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")

.thenReturn("Second");

MyService service = new MyService(mockApi);

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

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

}

}

**Exercise 6: Verifying Interaction Order**

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

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.step1();

service.step2();

InOrder inOrder = inOrder(mockApi);

inOrder.verify(mockApi).step1();

inOrder.verify(mockApi).step2();

}

}

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

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

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

public class MyServiceTest {

@Test

public void testVoidMethodThrowsException() {

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

doThrow(new RuntimeException("Error")).when(mockApi).logError();

MyService service = new MyService(mockApi);

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

verify(mockApi).logError();

}

}