Mockito Exercises Solutions

Exercise 1: Mocking and Stubbing

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

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

import static org.mockito.Mockito.\*;

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("test", 123);

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

verify(mockApi).process(contains("est"), gt(100));

}

}

Exercise 4: Handling Void Methods

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);

MyService service = new MyService(mockApi);

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

service.update("test");

verify(mockApi).updateData("test");

}

}

Exercise 5: Mocking and Stubbing with Multiple Returns

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.getStatus())

.thenReturn("Pending")

.thenReturn("Processing")

.thenReturn("Completed");

MyService service = new MyService(mockApi);

assertEquals("Pending", service.checkStatus());

assertEquals("Processing", service.checkStatus());

assertEquals("Completed", service.checkStatus());

}

}

Exercise 6: Verifying Interaction Order

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

import org.mockito.InOrder;

public class MyServiceTest {

@Test

public void testInteractionOrder() {

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

MyService service = new MyService(mockApi);

service.completeWorkflow();

InOrder inOrder = inOrder(mockApi);

inOrder.verify(mockApi).start();

inOrder.verify(mockApi).process();

inOrder.verify(mockApi).finish();

}

}

Exercise 7: Handling Void Methods with Exceptions

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 testVoidMethodWithException() {

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

MyService service = new MyService(mockApi);

doThrow(new RuntimeException("Connection failed")).when(mockApi).connect();

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

verify(mockApi).connect();

}

}