Mockito Hands-On Exercises

### Exercise 1: Mocking and Stubbing

ExternalApi.java

package com.example; public interface ExternalApi {

String getData();

}

## MyService.java

package com.example; public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) { this.api = api;

}

public String fetchData() { return api.getData();

}

}

#### MyServiceTest.java

package com.example;

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

import static org.junit.jupiter.api.Assertions.\*; public class MyServiceTest {

@Test

public void testExternalApi() {

// Step 1: Create mock

ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Stub method when(mockApi.getData()).thenReturn("Mock Data");

// Step 3: Inject mock and test

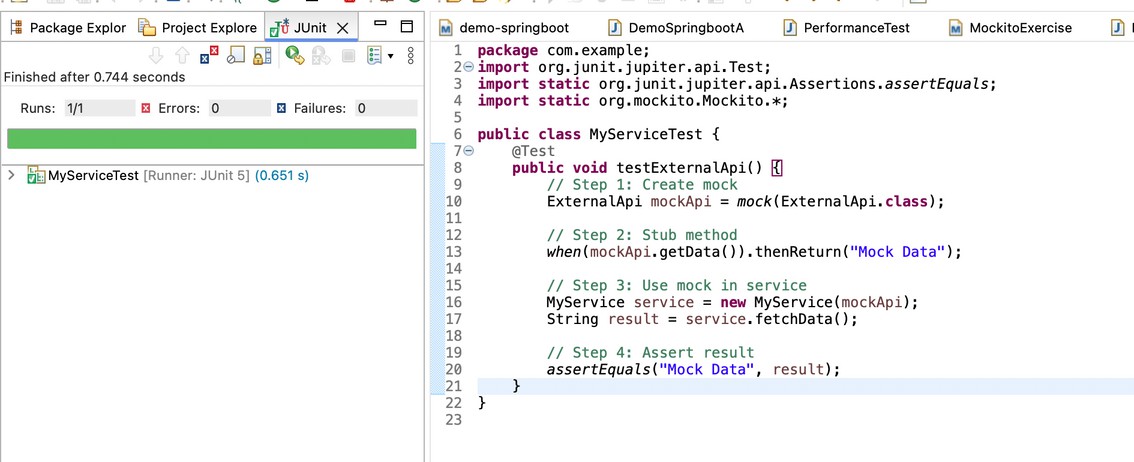
MyService service = new MyService(mockApi); String result = service.fetchData();

// Step 4: Assert assertEquals("Mock Data", result);

}

}

# output



Exercise 2: Verifying Interactions

#### ExternalApi.java

package com.example.MockitoEx2;

public interface ExternalApi { String getData();

}

#### MyService.java

package com.example.MockitoEx2;

public class MyService { private ExternalApi externalApi;

public MyService(ExternalApi externalApi) {

this.externalApi = externalApi;

}

public String fetchData() {

return externalApi.getData();

}

}

#### MyServiceTest.java

package com.example.MockitoEx2; import static org.mockito.Mockito.\*; import org.junit.jupiter.api.Test; public class MyServiceTest {

@Test

public void testVerifyInteraction() {

// Step 1: Create a mock object for ExternalApi ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Inject mock into MyService MyService service = new MyService(mockApi);

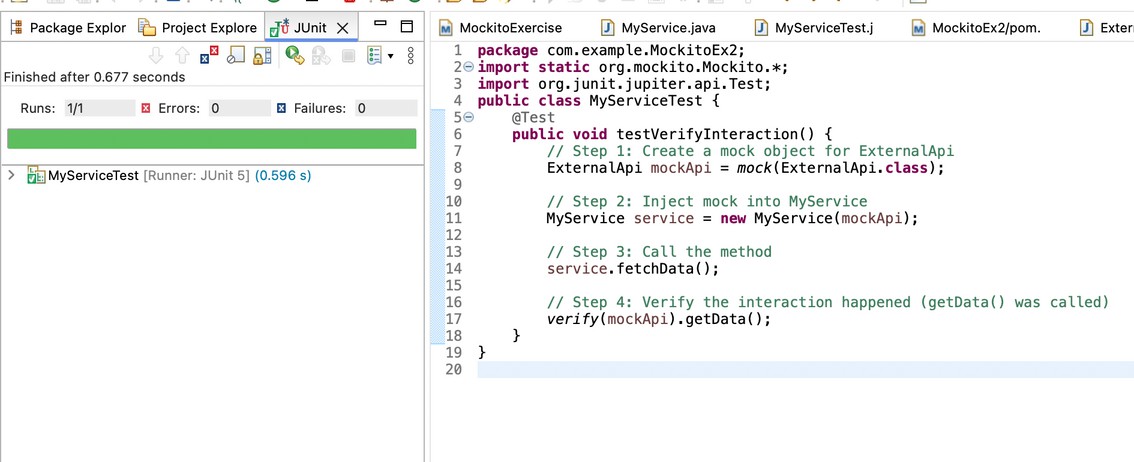
// Step 3: Call the method service.fetchData();

// Step 4: Verify the interaction happened (getData() was called) verify(mockApi).getData();

}

}

# output



### Exercise 3: Argument Matching

##### ExternalApi.java

package com.example.MockitoEx3;

public interface ExternalApi {

void sendData(String data);

}

#### MyService.java

package com.example.MockitoEx3;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public void process(String input) {

// Simulate processing

String processed = "Processed: " + input; api.sendData(processed);

}

}

#### MyServiceTest.java

package com.example.MockitoEx3; import org.junit.jupiter.api.Test; import org.mockito.Mockito;

import static org.mockito.Mockito.\*;

import static org.mockito.ArgumentMatchers.\*; public class MyServiceTest {

@Test

public void testArgumentMatching() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class); MyService service = new MyService(mockApi);

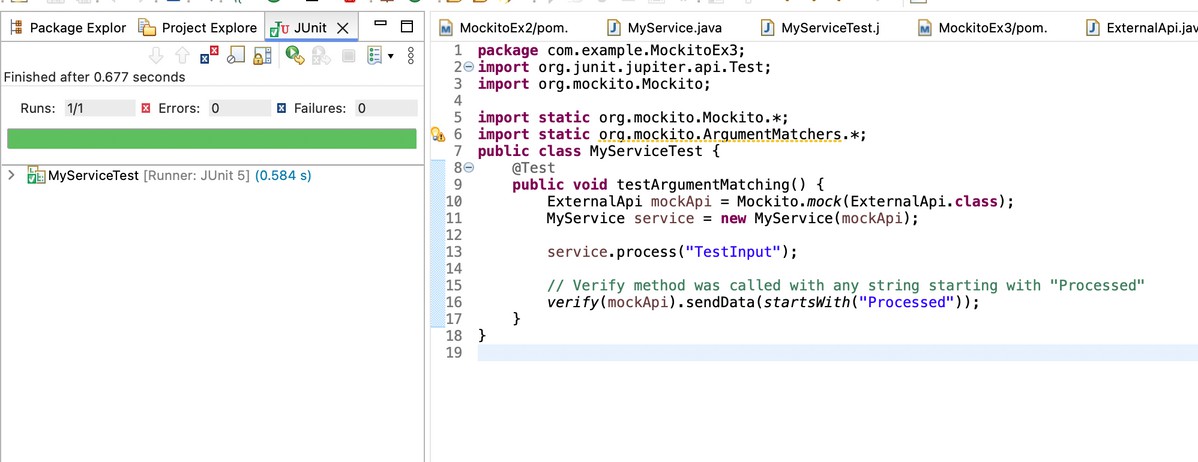
service.process("TestInput");

// Verify method was called with any string starting with "Processed" verify(mockApi).sendData(startsWith("Processed"));

}

}

# output



Exercise 4: Handling Void Methods

#### ExternalApi.java

package com.example.MockitoEx4;

public interface ExternalApi {

void log(String message);

}

##### MyService.java

package com.example.MockitoEx4;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public void process() {

String message = "Important Action"; api.log(message); // call the void method

}

}

## MyServiceTest.java

package com.example.MockitoEx4; import static org.mockito.Mockito.\*; import org.junit.jupiter.api.Test; import org.mockito.Mockito;

public class MyServiceTest { @Test

public void testVoidMethodInteraction() {

// Step 1: Create mock

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

// Step 2: Stub the void method (optional unless throwing exception) doNothing().when(mockApi).log(anyString());

// Step 3: Use mock in service

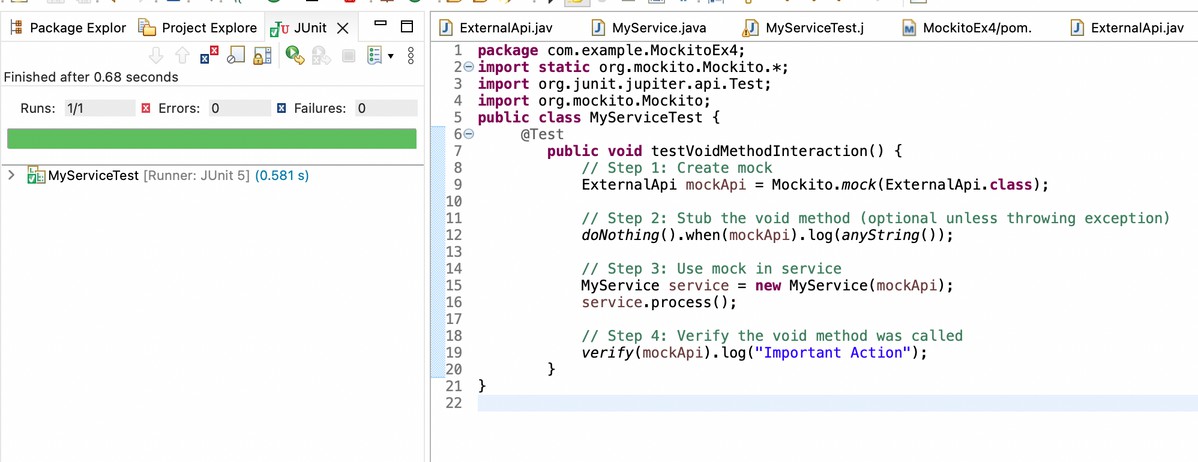
MyService service = new MyService(mockApi); service.process();

// Step 4: Verify the void method was called verify(mockApi).log("Important Action");

}

}

# output



Exercise 5: Mocking and Stubbing with Multiple Returns

##### ExternalAp.java

package com.example.MockitoEx5;

public interface ExternalApi { String getData();

}

##### MyService.java

package com.example.MockitoEx5;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchDataTwice() {

return api.getData() + " | " + api.getData();

}

}

##### MyServiceTest.java

package com.example.MockitoEx5; import org.junit.jupiter.api.Test;

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

import static org.mockito.Mockito.\*; public class MyServiceTest {

@Test

public void testMultipleReturns() {

ExternalApi mockApi = mock(ExternalApi.class);

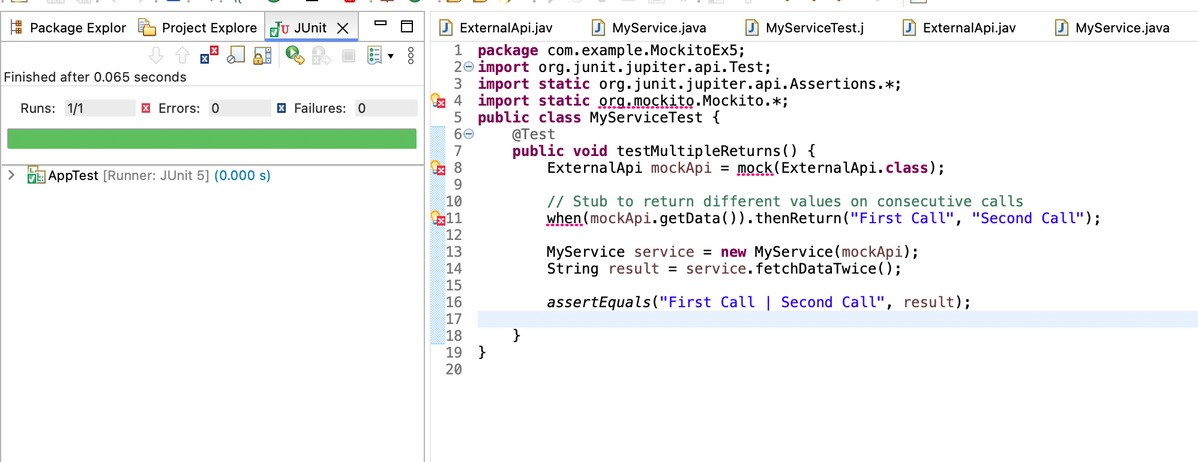
// Stub to return different values on consecutive calls when(mockApi.getData()).thenReturn("First Call", "Second Call"); MyService service = new MyService(mockApi);

String result = service.fetchDataTwice(); assertEquals("First Call | Second Call", result);

}

}

# output



Exercise 6: Verifying Interaction Order

#### ExternalApi.java

package com.example.MockitoEx6;

public interface ExternalApi {

void start(); void process(); void end();

}

##### MyService.java

package com.example.MockitoEx6;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public void doWork() { api.start();

api.process(); api.end();

}

}

#### MyServiceTest .java

package com.example.MockitoEx6; import org.junit.jupiter.api.Test; import static org.mockito.Mockito.\*; import org.mockito.InOrder;

public class MyServiceTest { @Test

public void testInteractionOrder() {

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

MyService service = new MyService(mockApi); service.doWork();

// Verify method calls in order

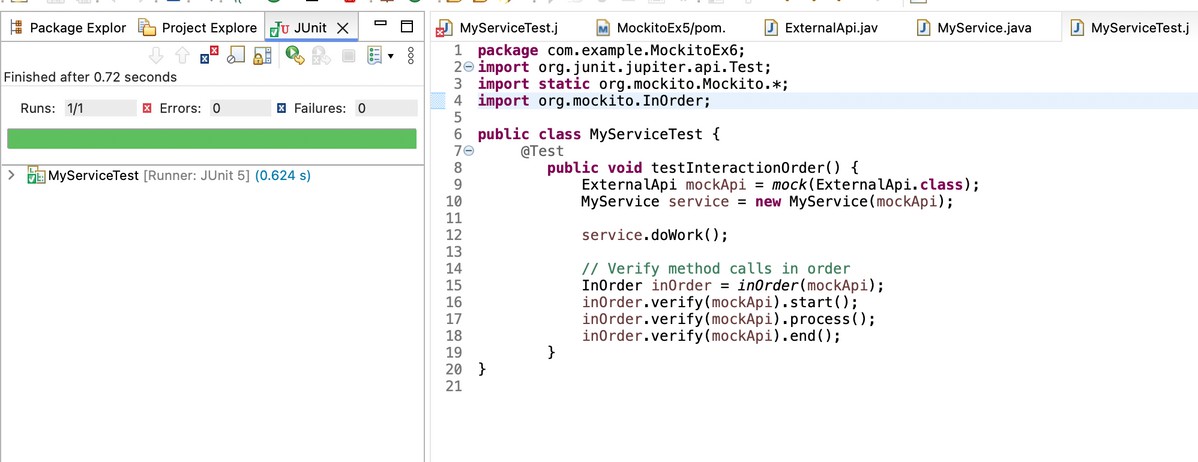
InOrder inOrder = *inOrder*(mockApi); inOrder.verify(mockApi).start();

inOrder.verify(mockApi).process(); inOrder.verify(mockApi).end();

}

}

# output



Exercise 7: Handling Void Methods with Exceptions

## ExternalApi .java

package com.example.MockitoEx7;

public interface ExternalApi {

void riskyOperation() throws Exception;

}

#### MyService.java

package com.example.MockitoEx7;

public class MyService {

private flnal ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public void perform() throws Exception { api.riskyOperation();

}

}

### MyServiceTest.java

package com.example.MockitoEx7; 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() throws Exception {

// Step 1: Create mock

ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Stub the void method to throw exception

doThrow(new Exception("Risky operation failed")).when(mockApi).riskyOperation(); MyService service = new MyService(mockApi);

// Step 3: Verify exception is thrown when perform() is called

Exception exception = assertThrows(Exception.class, () -> { service.perform();

});

assertEquals("Risky operation failed", exception.getMessage());

// Step 4: Verify the interaction

verify(mockApi).riskyOperation();

}

}

output

