Mockito Hands-On Exercises

Exercise 1: Mocking and Stubbing

Scenario:

You need to test a service that depends on an external API. Use Mockito to mock the external API and stub its methods.

Steps:

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

Solution Code:

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

public class MyServiceTest {

@Test

public void testExternalApi() {

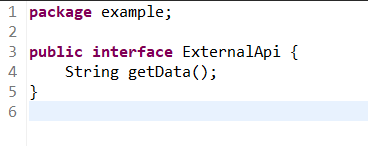
ExternalApi mockApi = Mockito.mock(ExternalApi.class); when(mockApi.getData()).thenReturn("Mock Data");

MyService service = new MyService(mockApi); String result = service.fetchData(); assertEquals("Mock Data", result);

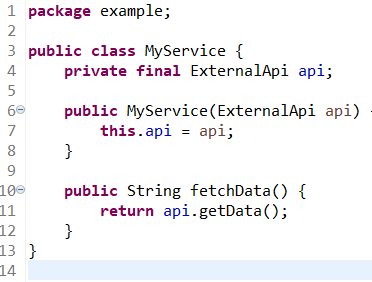
}

}

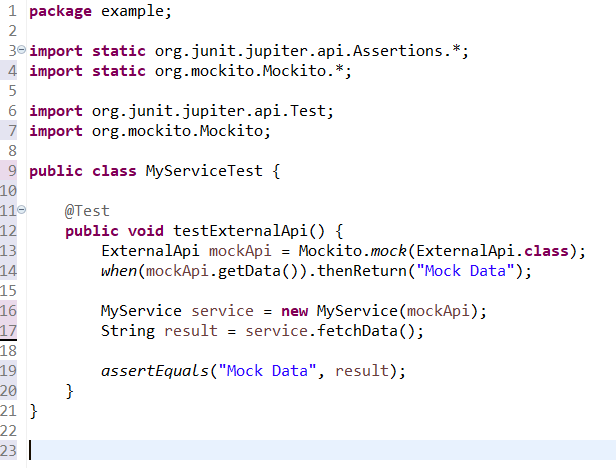
**ExternalApi.java:**

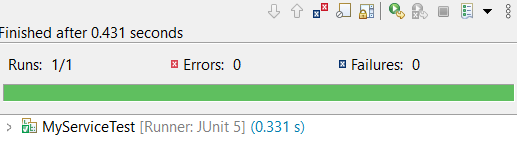


**MyService.java:**



**MyServiceTest.java:**





Exercise 2: Verifying Interactions

Scenario:

You need to ensure that a method is called with specific arguments.

Steps:

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

Solution Code:

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test; import org.mockito.Mockito;

public class MyServiceTest {

@Test

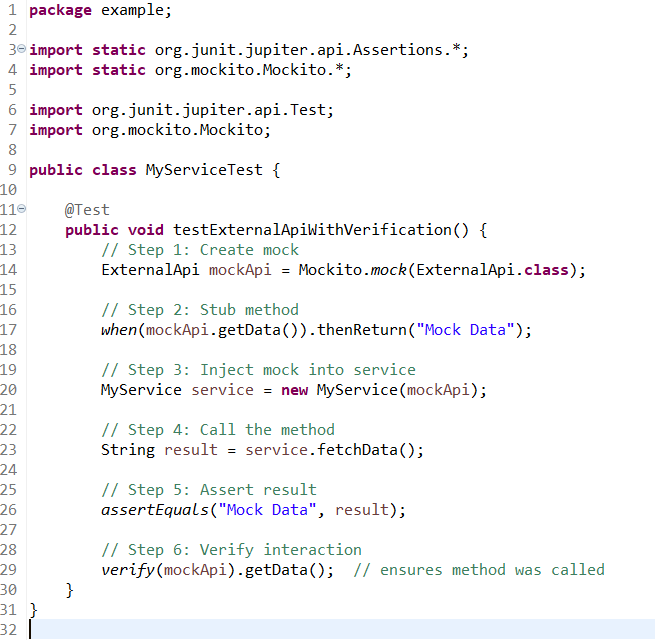
public void testVerifyInteraction() {

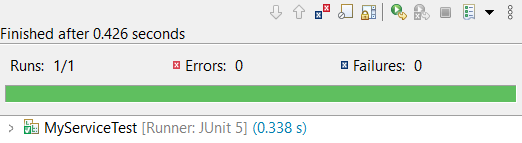
ExternalApi mockApi = Mockito.mock(ExternalApi.class); MyService service = new MyService(mockApi); service.fetchData(); verify(mockApi).getData();

}

}

**MyServiceTest.java:**





Exercise 3: Argument Matching

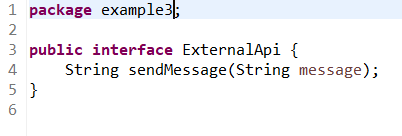
Scenario:

You need to verify that a method is called with specific arguments.

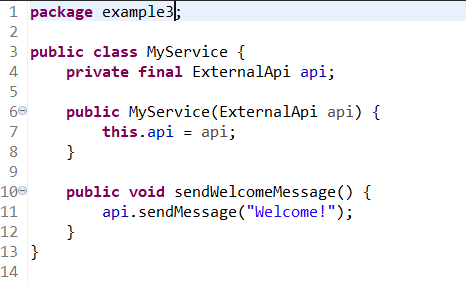
Steps:

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

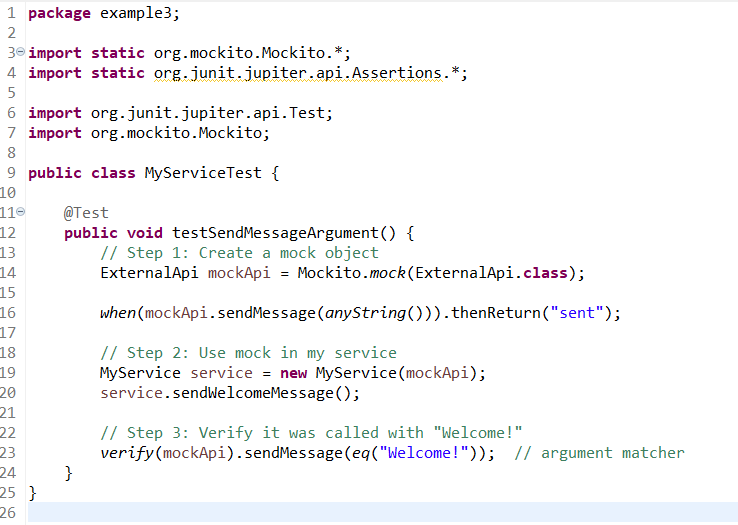
**ExternalApi.java:**

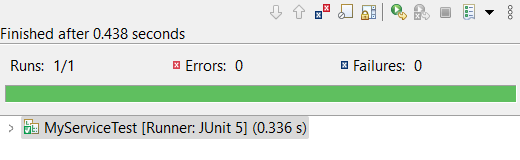


**MyService.java:**



**MyServiceTest.java:**





Exercise 4: Handling Void Methods

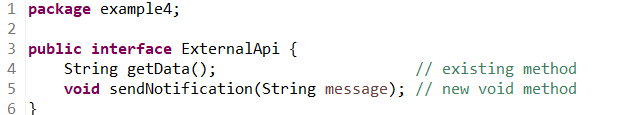
Scenario:

You need to test a void method that performs some action.

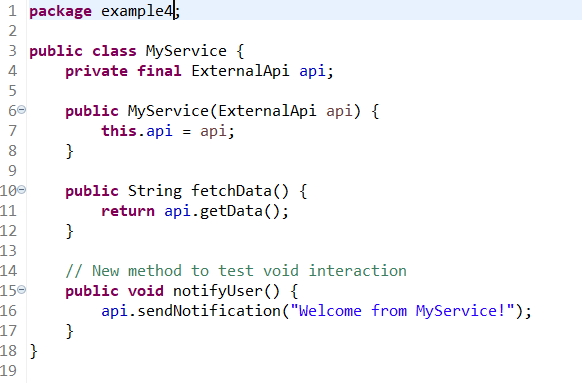
Steps:

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

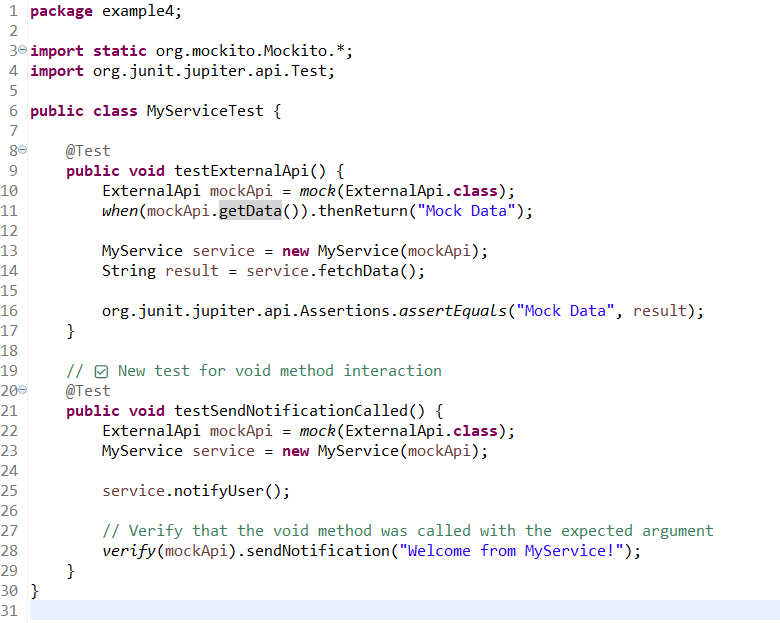
ExternalApi.java:

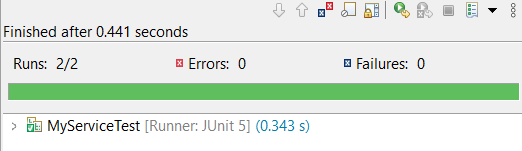


**MyService.java:**



**MyServiceTest.java:**





Exercise 5: Mocking and Stubbing with Multiple Returns

Scenario:

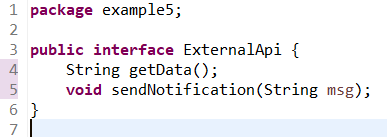
You need to test a service that depends on an external API with multiple return values.

Steps:

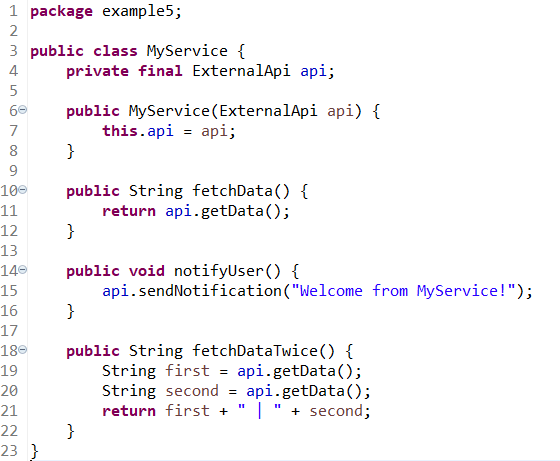
1. Create a mock object for the external API.
2. Stub the methods to return different values on consecutive calls.

Write a test case that uses the mock object.

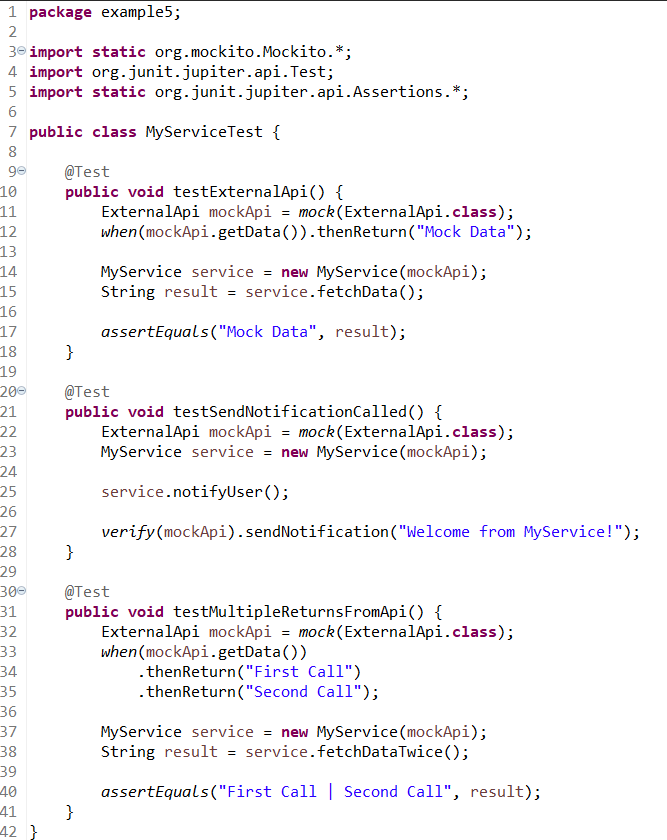
**ExternalApi.java:**

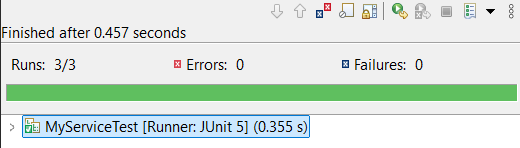


**MyService.java:**



**MyserviceTest.java:**





Exercise 6: Verifying Interaction Order

Scenario:

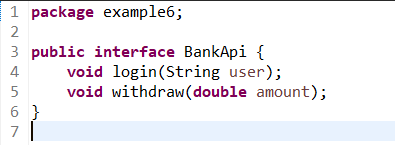
You need to ensure that methods are called in a specific order.

Steps:

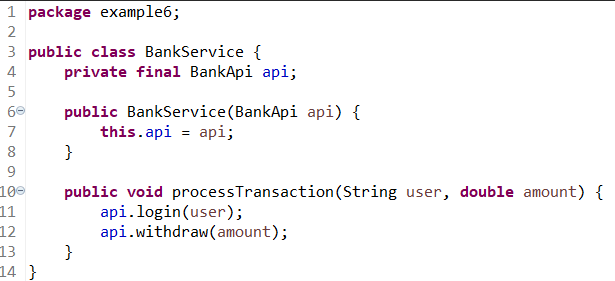
1. Create a mock object.
2. Call the methods in a specific order.

Verify the interaction order.

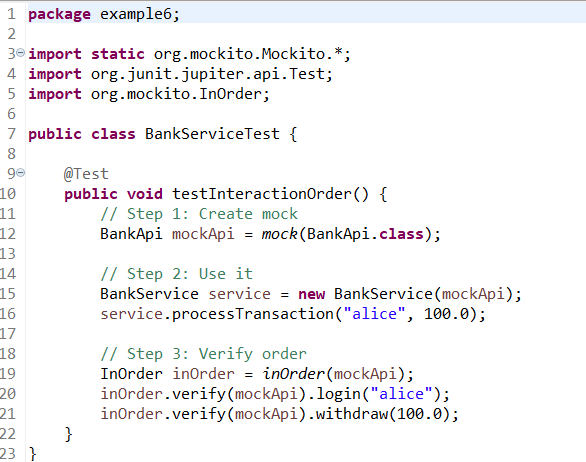
**BankApi.java:**

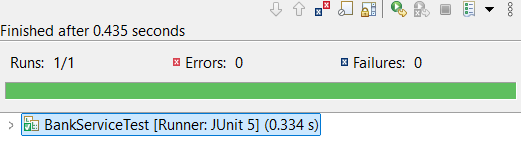


**BankService.java:**



**BankServiceTest.java:**





Exercise 7: Handling Void Methods with Exceptions

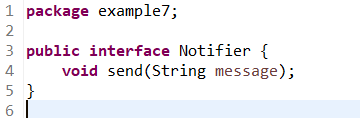
Scenario:

You need to test a void method that throws an exception.

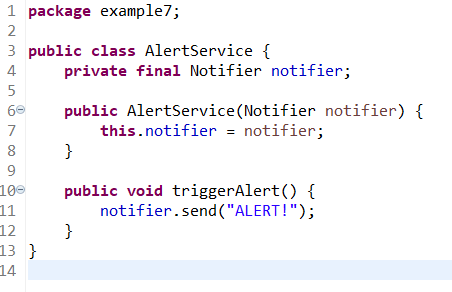
Steps:

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

**Notifier.java:**



**AlertService.java:**



**AlertServiceTest.java:**

