**Java Mokito and SL4J Testing**

**✅ 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.

**✔️ Answer:**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

// Assume this is the external API interface

interface ExternalApi {

    String getData();

}

// Service that depends on the External API

class MyService {

    private final ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

}

// Test class using Mockito

public class MyServiceTest {

    @Test

    public void testExternalApi() {

        // Step 1: Create mock object

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

        // Step 2: Stub the method

        when(mockApi.getData()).thenReturn("Mock Data");

        // Step 3: Use the mock in the service

        MyService service = new MyService(mockApi);

        String result = service.fetchData();

        // Step 4: Assert result

        assertEquals("Mock Data", result);

    }

}

**✅ 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.

**✔️ Answer:**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

// External API interface

interface ExternalApi {

    String getData();

}

// Service class that uses the external API

class MyService {

    private final ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

}

// Test class

public class MyServiceTest {

    @Test

    public void testVerifyInteraction() {

        // Step 1: Create mock

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

        // Step 2: Create service and call method

        MyService service = new MyService(mockApi);

        service.fetchData();

        // Step 3: Verify interaction

        verify(mockApi).getData();

    }

}

**✅ Exercise 3: Logging Error Messages and Warning Levels**

**Task:**  
Write a Java application that demonstrates logging error messages and warning levels using SLF4J.

**✔️ Step 1: Add SLF4J and Logback to pom.xml**

<dependencies>

    <dependency>

        <groupId>org.slf4j</groupId>

        <artifactId>slf4j-api</artifactId>

        <version>1.7.30</version>

    </dependency>

    <dependency>

        <groupId>ch.qos.logback</groupId>

        <artifactId>logback-classic</artifactId>

        <version>1.2.3</version>

    </dependency>

</dependencies>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

✔️ Step 2: Java Class Using SLF4J

java

CopyEdit

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class LoggingExample {

    private static final Logger logger = LoggerFactory.getLogger(LoggingExample.class);

    public static void main(String[] args) {

        logger.error("This is an error message");

        logger.warn("This is a warning message");

    }

}

Top of Form

Bottom of Form