**Exercise 1: Mocking and Stubbing**

In this project, I implemented unit testing for a service that depends on an external API using Mockito and JUnit. The goal was to test the service's behavior independently by mocking the external dependencies.

I started by creating an interface named ExternalApi which defines several methods such as getData, getUser, isConnected, getStatusCode, getExchangeRate, getRetryCount, login, and sendData. This simulates the external API.

Then, I created a service class named MyService that depends on this interface. The service contains methods that internally call the respective methods of the external API.

To test the service class, I created a separate test class MyServiceTest using the JUnit testing framework along with Mockito. In the test class, I created a mock object for the ExternalApi interface and stubbed its methods to return predefined values using the when(...).thenReturn(...) syntax. This ensures that the service methods are tested in isolation without relying on the actual external API implementation.

I wrote separate test methods to verify each functionality of the service, such as fetching data, retrieving user information, checking connection status, verifying status codes, handling exchange rates, retry logic, login validation, and data submission. Additionally, I used verify(...) to ensure that certain methods were invoked with expected arguments.

This approach ensures reliable and repeatable unit tests by isolating external dependencies and focusing only on the service logic.

**Code:**

**ExternalApi.java:**

package com.example;

public interface ExternalApi {

    String getData();

    String getUser(String userId);

    boolean isConnected();

    int getStatusCode();

    double getExchangeRate(String currency);

    int getRetryCount();

    boolean login(String username, String password);

    void sendData(String data);

}

**MyService.java:**

package com.example;

public class MyService {

    private final ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

    public String getUserDetails(String userId) {

        return api.getUser(userId);

    }

    public boolean checkConnection() {

        return api.isConnected();

    }

    public int getStatusCode() {

        return api.getStatusCode();

    }

    public double getExchangeRate(String currency) {

        return api.getExchangeRate(currency);

    }

    public int getRetryCount() {

        return api.getRetryCount();

    }

    public boolean login(String user, String pass) {

        return api.login(user, pass);

    }

    public void send(String data) {

        api.sendData(data);

    }

}

**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 {

    ExternalApi mockApi = mock(ExternalApi.class);

    MyService service = new MyService(mockApi);

    @Test

    public void testGetData() {

        when(mockApi.getData()).thenReturn("Hello, World");

        assertEquals("Hello, World", service.fetchData());

    }

    @Test

    public void testGetUser() {

        when(mockApi.getUser("user1")).thenReturn("User One");

        assertEquals("User One", service.getUserDetails("user1"));

    }

    @Test

    public void testIsConnected() {

        when(mockApi.isConnected()).thenReturn(true);

        assertTrue(service.checkConnection());

    }

    @Test

    public void testGetStatusCode() {

        when(mockApi.getStatusCode()).thenReturn(200);

        assertEquals(200, service.getStatusCode());

    }

    @Test

    public void testGetExchangeRate() {

        when(mockApi.getExchangeRate("USD")).thenReturn(82.3);

        assertEquals(82.3, service.getExchangeRate("USD"));

    }

    @Test

    public void testRetryCount() {

        when(mockApi.getRetryCount()).thenReturn(3);

        assertEquals(3, service.getRetryCount());

    }

    @Test

    public void testLoginSuccess() {

        when(mockApi.login("admin", "pass")).thenReturn(true);

        assertTrue(service.login("admin", "pass"));

    }

    @Test

    public void testSendData() {

        service.send("data123");

        verify(mockApi).sendData("data123");

    }

}

**pom.xml:**

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

         http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>

  <artifactId>mockito-example</artifactId>

  <version>1.0-SNAPSHOT</version>

  <dependencies>

    <!-- JUnit 5 -->

    <dependency>

      <groupId>org.junit.jupiter</groupId>

      <artifactId>junit-jupiter</artifactId>

      <version>5.9.3</version>

      <scope>test</scope>

    </dependency>

    <!-- Mockito -->

    <dependency>

      <groupId>org.mockito</groupId>

      <artifactId>mockito-core</artifactId>

      <version>5.11.0</version>

      <scope>test</scope>

    </dependency>

  </dependencies>

  <build>

    <plugins>

      <!-- Maven Surefire Plugin for running tests -->

      <plugin>

        <groupId>org.apache.maven.plugins</groupId>

        <artifactId>maven-surefire-plugin</artifactId>

        <version>3.0.0</version>

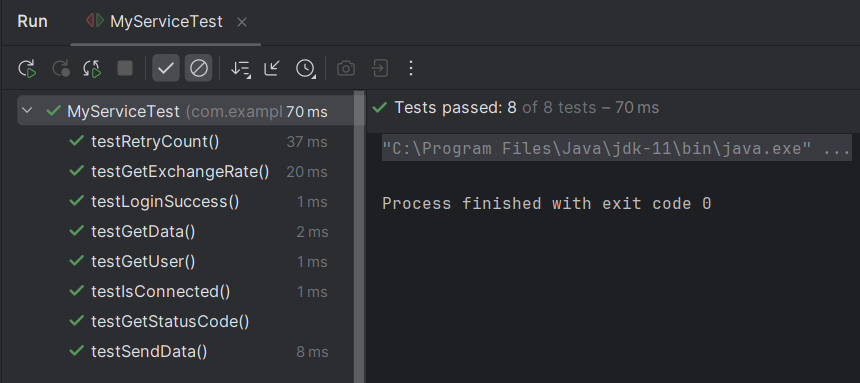
      </plugin>

    </plugins>

  </build>

</project>

**Output:**



**Exercise 2: Verifying Interactions**

In this project, I focused on testing method interactions using Mockito. The objective was to verify that specific methods in an external API were invoked with the correct arguments and the expected number of times.

To achieve this, I created an interface named ExternalApi which simulates the behavior of an external dependency. This interface contains several method declarations, such as getData, getUser, isConnected, getStatusCode, getExchangeRate, getRetryCount, login, sendData, resetPassword, and logout.

Next, I implemented a service class named MyService, which uses an instance of ExternalApi. The service contains wrapper methods that call the respective methods of the external API.

The testing was done using JUnit and Mockito. I created a test class named MyServiceTest where I mocked the ExternalApi interface using Mockito.mock(). I then used the service methods and applied the verify() method to ensure that the corresponding methods in the mock object were invoked as expected.

In total, I wrote ten unit test cases that verify different interactions, such as method calls with specific arguments, number of times a method was called, and ensuring certain methods were not called. This approach ensures that the service behaves correctly in terms of its interaction with external dependencies, without relying on the actual implementation of the external API.

**Code:**

**ExternalApi.java:**

package com.example;

public interface ExternalApi {

    String getData();

    String getUser(String userId);

    boolean isConnected();

    int getStatusCode();

    double getExchangeRate(String currency);

    int getRetryCount();

    boolean login(String user, String pass);

    void sendData(String data);

    void resetPassword(String user);

    void logout();

}

**MyService.java:**

package com.example;

public class MyService {

    private final ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

    public String getUserDetails(String userId) {

        return api.getUser(userId);

    }

    public boolean checkConnection() {

        return api.isConnected();

    }

    public int getStatusCode() {

        return api.getStatusCode();

    }

    public double getExchangeRate(String currency) {

        return api.getExchangeRate(currency);

    }

    public int getRetryCount() {

        return api.getRetryCount();

    }

    public boolean login(String username, String password) {

        return api.login(username, password);

    }

    public void send(String data) {

        api.sendData(data);

    }

    public void resetPassword(String user) {

        api.resetPassword(user);

    }

    public void logout() {

        api.logout();

    }

}

**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 testGetDataCalled() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.fetchData();

        verify(mockApi).getData();

    }

    @Test

    public void testGetUserCalledWithCorrectId() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.getUserDetails("user123");

        verify(mockApi).getUser("user123");

    }

    @Test

    public void testIsConnectedInvoked() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.checkConnection();

        verify(mockApi).isConnected();

    }

    @Test

    public void testStatusCodeCheckedOnce() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.getStatusCode();

        verify(mockApi, times(1)).getStatusCode();

    }

    @Test

    public void testExchangeRateCheckedWithUSD() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.getExchangeRate("USD");

        verify(mockApi).getExchangeRate("USD");

    }

    @Test

    public void testRetryCountCalledOnce() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.getRetryCount();

        verify(mockApi, times(1)).getRetryCount();

    }

    @Test

    public void testLoginCalledWithCorrectCredentials() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.login("admin", "password123");

        verify(mockApi).login("admin", "password123");

    }

    @Test

    public void testSendDataCalledWithPayload() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.send("sampleData");

        verify(mockApi).sendData("sampleData");

    }

    @Test

    public void testResetPasswordNotCalled() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.fetchData();

        verify(mockApi, never()).resetPassword(anyString());

    }

    @Test

    public void testLogoutCalledOnceOnly() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        mockApi.logout();

        verify(mockApi, times(1)).logout();

    }

}

**Output:**

