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.

ExternalAPI.java

package com.mockito.stubbing;

public interface ExternalApi {

String getData();

String getData1();

}

MYService.java

package com.mockito.stubbing;

// Service that uses ExternalApi

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

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

<modelVersion>4.0.0</modelVersion>

<groupId>TDDMockito</groupId>

<artifactId>mockito</artifactId>

<version>0.0.1-SNAPSHOT</version>

<properties>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<junit.jupiter.version>5.10.0</junit.jupiter.version>

<mockito.version>5.11.0</mockito.version>

<surefire.version>3.2.3</surefire.version> <!-- Updated surefire version -->

</properties>

<dependencies>

<!-- JUnit 5 -->

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>${junit.jupiter.version}</version>

<scope>test</scope>

</dependency>

<!-- Mockito Core -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>${mockito.version}</version>

<scope>test</scope>

</dependency>

<!-- Mockito JUnit 5 integration -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-junit-jupiter</artifactId>

<version>${mockito.version}</version>

<scope>test</scope>

</dependency>

<!-- Add ByteBuddy dependency for Mockito's modern mocking -->

<dependency>

<groupId>net.bytebuddy</groupId>

<artifactId>byte-buddy</artifactId>

<version>1.14.9</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Updated Maven Surefire Plugin configuration -->

<plugin>

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

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

<version>${surefire.version}</version>

<configuration>

<argLine>

--add-opens java.base/java.lang=ALL-UNNAMED

--add-opens java.base/java.util=ALL-UNNAMED

</argLine>

</configuration>

</plugin>

<!-- Add Maven Compiler Plugin for explicit configuration -->

<plugin>

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

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<release>11</release>

</configuration>

</plugin>

</plugins>

</build>

</project>

TestCase.java

package com.mockito.stubbing;

// Needed for assertEquals

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

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

// Test class

public class TestCase {

@Test

public void testExternalApi() {

// Create a mock for the ExternalApi

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

// Stub the getData() method

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

// Inject mock into MyService

MyService service = new MyService(mockApi);

// Call the method under test

String result = service.fetchData();

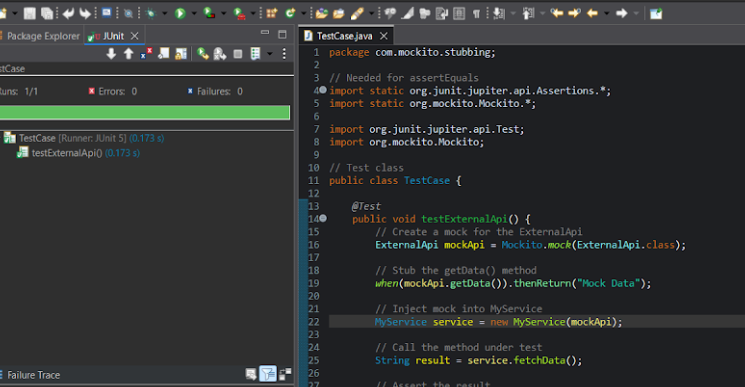
// Assert the result

assertEquals("Mock Data", result);

}

}

Output

A screen shot of a computer

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.