**Exercise 2: Verifying Interactions**

**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();**

**}**

**}**

**CODE:**

**DataService.java:**

package org.example;  
  
public interface DataService {  
 String getData();  
}

**MyService.java:**

package org.example;  
public class MyService {  
 private final DataService dataService;  
  
 public MyService(DataService dataService) {  
 this.dataService = dataService;  
 }  
 public String fetchData() {  
 return dataService.getData();  
 }  
}

**MyServiceTest.java:**

package org.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 testVerifyInteraction() {  
 System.*out*.println("=== Test Started ===");  
 DataService mockDataService = *mock*(DataService.class);  
 System.*out*.println("Mock object created");  
 *when*(mockDataService.getData()).thenReturn("Mock Data");  
 System.*out*.println("Stubbed getData() to return 'Mock Data'");  
 MyService service = new MyService(mockDataService);  
 System.*out*.println("Service initialized with mock");  
 String result = service.fetchData();  
 System.*out*.println("fetchData() returned: " + result);  
 *assertEquals*("Mock Data", result);  
 *verify*(mockDataService).getData();  
 System.*out*.println("Verified: getData() was called");  
 System.*out*.println("=== Test Completed Successfully ===");  
 }  
}

**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>org.example</groupId>  
 <artifactId>mockito-demo</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>21</maven.compiler.source>  
 <maven.compiler.target>21</maven.compiler.target>  
 </properties>  
  
 <dependencies>  
 <!-- JUnit 5 -->  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter</artifactId>  
 <version>5.10.0</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- Mockito -->  
 <dependency>  
 <groupId>org.mockito</groupId>  
 <artifactId>mockito-core</artifactId>  
 <version>5.11.0</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- ByteBuddy Agent -->  
 <dependency>  
 <groupId>net.bytebuddy</groupId>  
 <artifactId>byte-buddy-agent</artifactId>  
 <version>1.14.12</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <!-- Maven Surefire Plugin to run tests with VM options -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>3.1.2</version>  
 <configuration>  
 <argLine>  
 -javaagent:${user.home}/.m2/repository/net/bytebuddy/byte-buddy-agent/1.14.12/byte-buddy-agent-1.14.12.jar  
 -Dnet.bytebuddy.agent.prevent=true  
 -Djdk.instrument.traceUsage=false  
 </argLine>  
 <useModulePath>false</useModulePath> <!-- required for Java 21 -->  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**OUTPUT:**

