**Exercise2: Verifying Interactions with Mockito**

Objective: To ensure that a specific method is called with expected arguments using Mockito’s verify() method.

Prerequisites for the development of Mockito:

Java project configured with:

* Junit 5
* Mockito

Required Dependencies to add to pom.xml:

|  |
| --- |
| <!-- JUnit Jupiter -->  <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> |

Interaction Verification:

Mockito’s verify() method checks that a mock object’s method was called as expected, optionally with specific arguments.

Step1: External API Interface:

ExternalApi.java:

|  |
| --- |
| package harshita;  public interface ExternalApi {  String getData ();  } |

Step2: Services that uses the API:

|  |
| --- |
| package harshita;  public class MyService {  private ExternalApi externalApi;  public MyService (ExternalApi externalApi) {  this.externalApi = externalApi;  }  public String fetchData () {  return externalApi.getData(); // delegate call  }  } |

Step3: Test Class to Verify Interaction

|  |
| --- |
| package harshita;  import org.junit.jupiter.api.Test;  import org.mockito.Mockito;  import static org.mockito.Mockito.\*;  public class MyServiceTest {  @Test  public void testVerifyInteraction() {  // Step 1: Create a mock object  ExternalApi mockApi = Mockito.mock(ExternalApi.class);  // Step 2: Use mock in service  MyService service = new MyService(mockApi);  // Step 3: Call method  service.fetchData();  // Step 4: Verify interaction  verify(mockApi).getData();  }  } |

