**Week 2**

**Exercise 2: Verifying Interactions**

**Code:**

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<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>org.example</groupId>  
 <artifactId>mockit1</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <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>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>3.0.0</version>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**Main.java:**

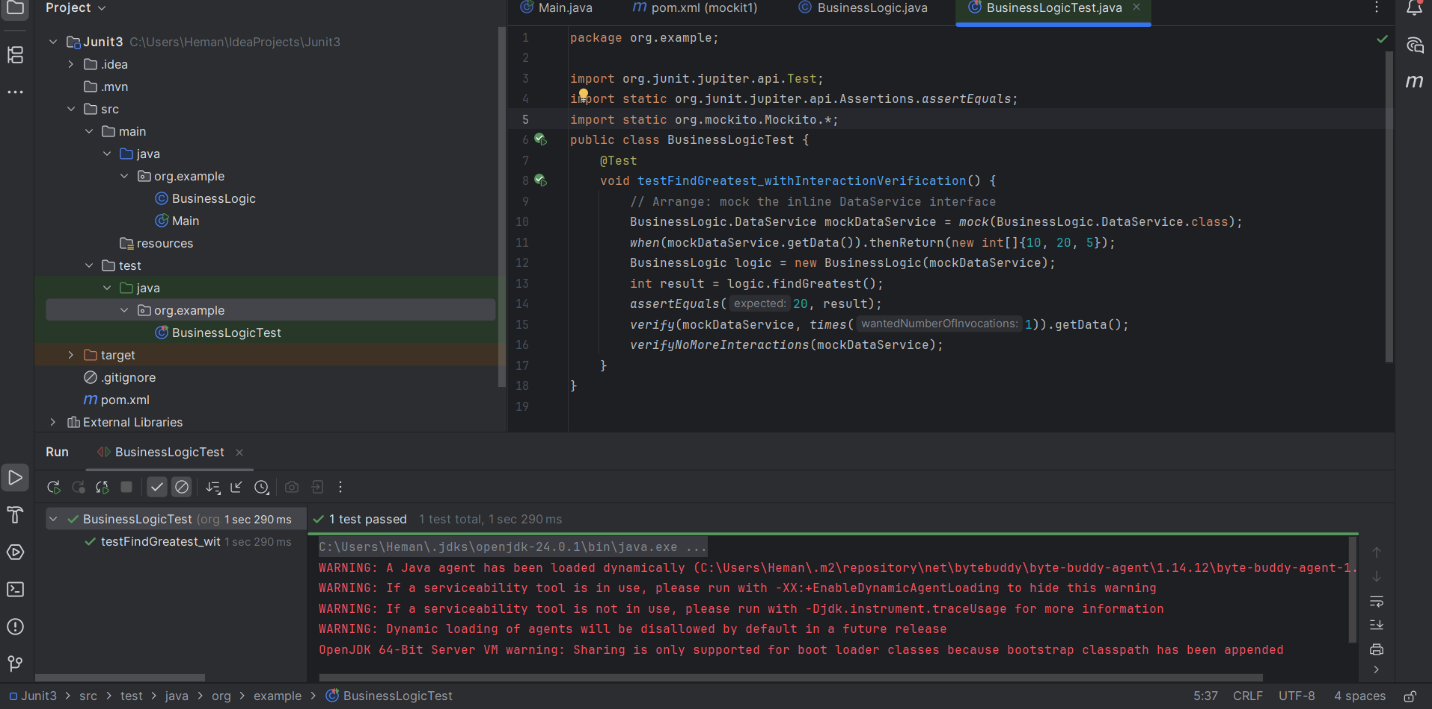
package org.example;  
  
public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("Hello and welcome!");  
 for (int i = 1; i <= 5; i++) {  
 System.*out*.println("i = " + i);  
 }  
 }  
}

**BusinessLogic.java:**

package org.example;  
  
public class BusinessLogic {  
  
 public interface DataService {  
 int[] getData();  
 }  
  
 private final DataService dataService;  
  
 public BusinessLogic(DataService dataService) {  
 this.dataService = dataService;  
 }  
  
 public int findGreatest() {  
 int[] data = dataService.getData();  
 int max = Integer.*MIN\_VALUE*;  
 for (int value : data) {  
 if (value > max) {  
 max = value;  
 }  
 }  
 return max;  
 }  
}

**BusinessLogicTest.java**:

package org.example;  
  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
import static org.mockito.Mockito.\*;  
public class BusinessLogicTest {  
 @Test  
 void testFindGreatest\_withInteractionVerification() {  
 // Arrange: mock the inline DataService interface  
 BusinessLogic.DataService mockDataService = *mock*(BusinessLogic.DataService.class);  
 *when*(mockDataService.getData()).thenReturn(new int[]{10, 20, 5});  
 BusinessLogic logic = new BusinessLogic(mockDataService);  
 int result = logic.findGreatest();  
 *assertEquals*(20, result);  
 *verify*(mockDataService, *times*(1)).getData();  
 *verifyNoMoreInteractions*(mockDataService);  
 }  
}



**Exercise 2: Verifying Interactions**

**Code:**

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<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>mockit2</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <junit.jupiter.version>5.9.3</junit.jupiter.version>  
 <mockito.version>5.2.0</mockito.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>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <!-- Fix for maven-deploy-plugin issue -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-deploy-plugin</artifactId>  
 <version>3.1.1</version>  
 </plugin>  
  
 <!-- JUnit Platform (Surefire Plugin) -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>3.0.0</version>  
 <configuration>  
 <includes>  
 <include>\*\*/\*Test.java</include>  
 </includes>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**Main.java**:

package org.example;  
  
//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or  
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.  
public class Main {  
 public static void main(String[] args) {  
 //TIP Press <shortcut actionId="ShowIntentionActions"/> with your caret at the highlighted text  
 // to see how IntelliJ IDEA suggests fixing it.  
 System.*out*.println("Hello and welcome!");  
  
 for (int i = 1; i <= 5; i++) {  
 //TIP Press <shortcut actionId="Debug"/> to start debugging your code. We have set one <icon src="AllIcons.Debugger.Db\_set\_breakpoint"/> breakpoint  
 // for you, but you can always add more by pressing <shortcut actionId="ToggleLineBreakpoint"/>.  
 System.*out*.printf("i = " + i);  
 }  
 }  
}

**MessageService.java:**

package org.example;  
  
public interface MessageService {  
 void sendMessage(String message);  
}

**UserNotifier.java:**

package org.example;  
  
public class UserNotifier {  
 private final MessageService messageService;  
  
 public UserNotifier(MessageService messageService) {  
 this.messageService = messageService;  
 }  
  
 public void notifyUser(boolean important) {  
 if (important) {  
 messageService.sendMessage("Important Notification!");  
 }  
 }  
}

**UserNotifierTest.java:**  
package org.example;  
  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
  
public class UserNotifierTest {  
  
 @Test  
 public void testNotifyUser\_WhenImportant\_ShouldSendMessage() {  
 MessageService mockService = *mock*(MessageService.class);  
 UserNotifier notifier = new UserNotifier(mockService);  
 notifier.notifyUser(true);  
 *verify*(mockService).sendMessage("Important Notification!");  
 }  
  
 @Test  
 public void testNotifyUser\_WhenNotImportant\_ShouldNotSendMessage() {  
 MessageService mockService = *mock*(MessageService.class);  
 UserNotifier notifier = new UserNotifier(mockService);  
 notifier.notifyUser(false);  
 *verify*(mockService, *never*()).sendMessage(*anyString*());  
 }  
}**Output:** 