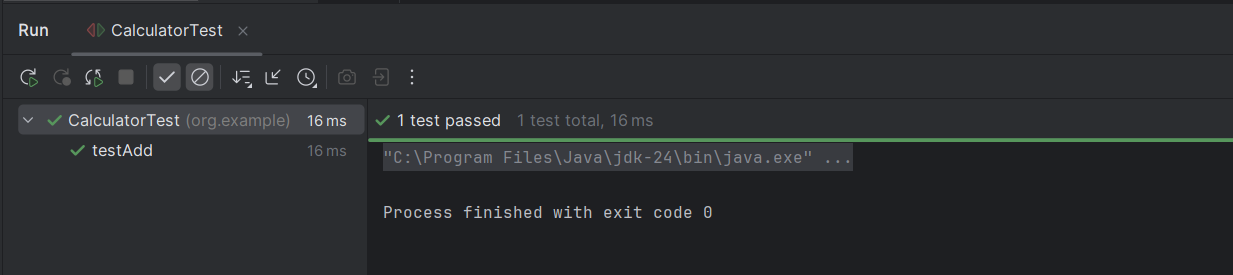
**WEEK – 2**

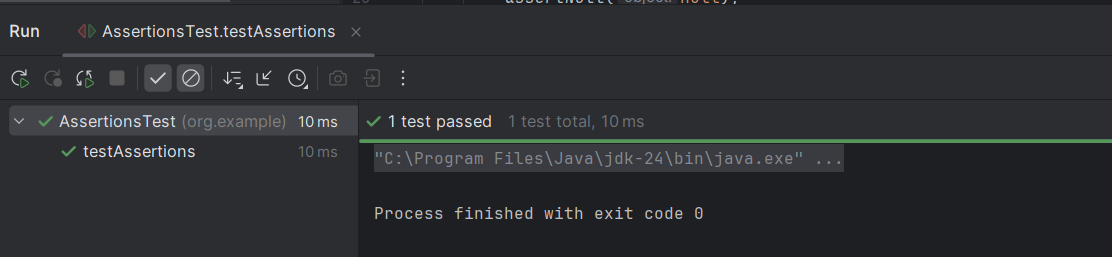
**TDD using JUnit5 and Mockito  
  
Mandatory Hands-on**

**Exercise 1:** Setting Up Junit **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>JUnitex-1</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>JUnitex-1</name>  
 <url>http://maven.apache.org</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
</project>

**Calculator.java**  
package org.example;  
  
public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
}  
  
  
**CalculatorTest.java**  
package org.example;  
  
public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
}  
  


**Exercise 3:** Assertions in Junit  
**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>JUnitex-2</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>JUnitex-2</name>  
 <url>http://maven.apache.org</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
</project>

**Utils.java**package org.example;  
  
public class Utils {  
 public static int add(int a, int b) {  
 return a + b;  
 }  
  
 public static boolean isPositive(int n) {  
 return n > 0;  
 }  
  
 public static Object getNull() {  
 return null;  
 }  
}

**AssertionsTest.java**package org.example;  
  
import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class AssertionsTest {  
  
 @Test  
 public void testAssertions() {  
 // Assert equals  
 *assertEquals*(5, 2 + 3);  
  
 // Assert true  
 *assertTrue*(5 > 3);  
  
 // Assert false  
 *assertFalse*(5 < 3);  
  
 // Assert null  
 *assertNull*(null);  
  
 // Assert not null  
 *assertNotNull*(new Object());  
 }  
}  


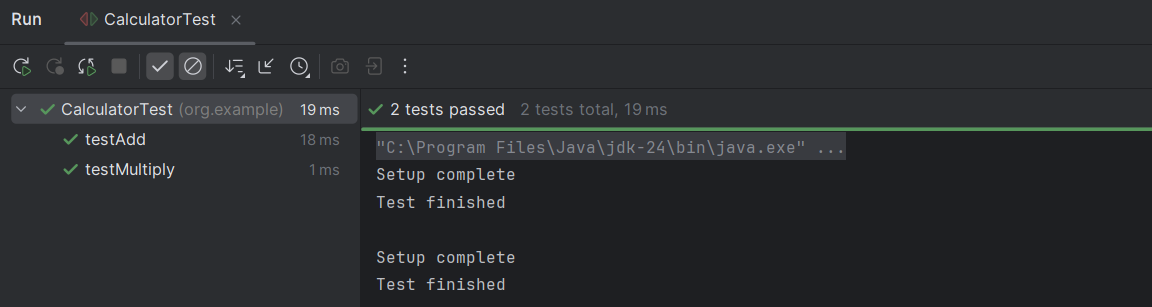
**Exercise 4:** Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup andTeardown Methods in JUnit

**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>JUnitex-3</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>JUnitex-3</name>  
 <url>http://maven.apache.org</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
</project>  
**Calculator.java**

package org.example;  
  
public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
  
 public int multiply(int a, int b) {  
 return a \* b;  
 }  
}

**CalculatorTest.java**

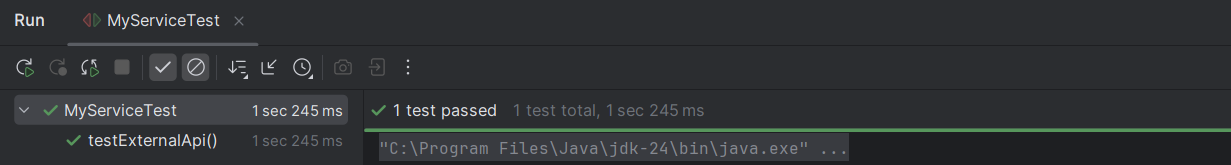
package org.example;  
  
import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;  
  
import static org.junit.Assert.\*;  
  
public class CalculatorTest {  
  
 private Calculator calculator;  
  
 // 🔧 Setup method (runs before each test)  
 @Before  
 public void setUp() {  
 calculator = new Calculator(); // Arrange  
 System.*out*.println("Setup complete");  
 }  
  
 // 🧹 Teardown method (runs after each test)  
 @After  
 public void tearDown() {  
 System.*out*.println("Test finished\n");  
 }  
  
 // ✅ Test using Arrange-Act-Assert  
 @Test  
 public void testAdd() {  
 // Arrange done in @Before  
  
 // Act  
 int result = calculator.add(2, 3);  
  
 // Assert  
 *assertEquals*(5, result);  
 }  
  
 @Test  
 public void testMultiply() {  
 int result = calculator.multiply(3, 4);  
 *assertEquals*(12, result);  
 }  
}

****

**Mockito exercises  
  
Mandatory Hands-on  
  
Exercise 1:** Mocking and Stubbing  
**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>Mockitoex-1</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>Mockitoex-1</name>  
 <url>http://maven.apache.org</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter</artifactId>  
 <version>5.10.0</version>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.mockito</groupId>  
 <artifactId>mockito-core</artifactId>  
 <version>5.11.0</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
</project>

**ExternalApi.java**public interface ExternalApi {  
 String getData();  
}  
**MyService.java**public interface ExternalApi {  
 String getData();  
}  
**MyServiceTest.java**

import static org.mockito.Mockito.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
  
import org.junit.jupiter.api.Test;  
import org.mockito.Mockito;  
  
public class MyServiceTest {  
  
 @Test  
 public void testExternalApi() {  
 // Step 1: Create mock object  
 ExternalApi mockApi = Mockito.*mock*(ExternalApi.class);  
  
 // Step 2: Stub the method  
 *when*(mockApi.getData()).thenReturn("Mock Data");  
  
 // Step 3: Inject mock into service  
 MyService service = new MyService(mockApi);  
  
 // Step 4: Call method and assert  
 String result = service.fetchData();  
 *assertEquals*("Mock Data", result);  
 }  
}

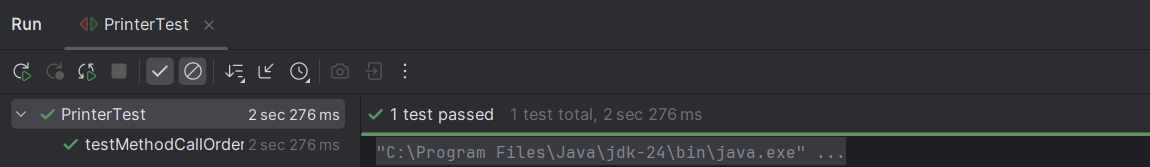


**Exercise 2:** Verifying Interactions

**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>Mockitoex-2</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>Mockitoex-2</name>  
 <url>http://maven.apache.org</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </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>  
 </dependencies>  
</project>

**Printer.java**public interface Printer {  
 void turnOn();  
 void printDocument(String doc);  
 void turnOff();  
}

**PrinterTest.java**import org.junit.jupiter.api.Test;  
import org.mockito.InOrder;  
  
import static org.mockito.Mockito.\*;  
  
public class PrinterTest {  
  
 @Test  
 public void testMethodCallOrder() {  
 // Step 1: Create a mock  
 Printer mockPrinter = *mock*(Printer.class);  
  
 // Step 2: Call methods in a specific order  
 mockPrinter.turnOn();  
 mockPrinter.printDocument("TestDoc.pdf");  
 mockPrinter.turnOff();  
  
 // Step 3: Verify the order  
 InOrder inOrder = *inOrder*(mockPrinter);  
 inOrder.verify(mockPrinter).turnOn();  
 inOrder.verify(mockPrinter).printDocument("TestDoc.pdf");  
 inOrder.verify(mockPrinter).turnOff();  
 }  
}



**SL4J Logging exercises  
  
Exercise 1:** Logging Error Messages and Warning Levels  
  
**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>LoggingExample</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>LoggingExample</name>  
 <url>http://maven.apache.org</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.7.30</version>  
 </dependency>  
 <dependency>  
 <groupId>ch.qos.logback</groupId>  
 <artifactId>logback-classic</artifactId>  
 <version>1.2.3</version>  
 </dependency>  
 </dependencies>  
</project>

**Logback.xml**<configuration>  
 <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">  
 <encoder>  
 <pattern>%d{HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n</pattern>  
 </encoder>  
 </appender>  
  
 <root level="debug">  
 <appender-ref ref="STDOUT"/>  
 </root>  
</configuration>  
  
  
  
  
  
**LoggingExample.java**import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
public class LoggingExample {  
 private static final Logger *logger* = LoggerFactory.*getLogger*(LoggingExample.class);  
  
 public static void main(String[] args) {  
 *logger*.error("This is an error message");  
 *logger*.warn("This is a warning message");  
 }  
}

