**Digital Nurture 4.0 Deep Skilling - Java FSE**  
**WEEK –2 Hands-on Exercises**  
**Module 4 – Test driven development and Logging framework**

1. **Logging using SLF4J**

**1. Exercise 1: Logging Error Messages and Warning Levels**

**Task:** Write a Java application that demonstrates logging error messages and warning levels

using SLF4J

**Solution:**

**SLF4J:**

SLF4J stands for Simple Logging Facade for Java. It is a logging abstraction that provides a simple and unified API for various logging frameworks such as Logback, Log4j, and java.util.logging.

Rather than tying your application to a specific logging implementation, SLF4J lets you plug in your preferred backend at runtime.

**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>com.example1</groupId>

<artifactId>logging-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>Logging Demo</name>

<properties>

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

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

<junit.version>5.7.1</junit.version>

</properties>

<dependencies>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>2.0.12</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.4.11</version>

</dependency>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>3.0.0-M5</version>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>central</id>

<name>Maven Central Repository</name>

<url>https://repo.maven.apache.org/maven2</url>

<releases>

<enabled>true</enabled>

</releases>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

</repositories>

</project>

**LoggingExample.java**

package com.example1.logging\_demo;

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.atInfo().log("Application started");

try {

simulateProcess();

} catch (Exception e) {

logger.atError().log(" Exception occurred: {}", e.getMessage());

}

logger.atWarn().log(" This is a warning message");

logger.atDebug().log(" This is a debug message (visible only if log level is DEBUG)");

logger.atInfo().log(" Application finished");

}

private static void simulateProcess() {

logger.atInfo().log(" Simulating some process...");

int a = 10;

int b = 0;

if (b == 0) {

logger.atWarn().log(" Divider is zero, skipping division to avoid ArithmeticException");

} else {

int result = a / b;

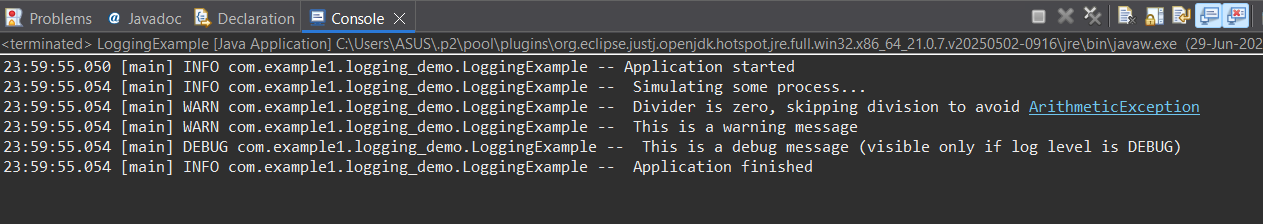
logger.atInfo().log("Division result: {}", result);

}

}

}

**Output:**



**Explanation:**

1. Add SLF4J API and Logback implementation as dependencies in the pom.xml file.
2. Create a Java class and import Logger and LoggerFactory from SLF4J.
3. Create a static logger instance using LoggerFactory.getLogger().
4. Use logger.error() to log an error-level message.
5. Use logger.warn() to log a warning-level message.