**Module: Logging Using SLF4J**

**Objective**

This assignment focuses on implementing logging in Java using the SLF4J framework along with the Logback backend. Logging plays a crucial role in every application’s lifecycle, enabling developers to capture runtime events, trace errors, and monitor application behavior in a production-safe way. In this task, the aim was to produce meaningful log outputs at error and warning levels.

**Overview**

SLF4J, which stands for Simple Logging Facade for Java, provides a generic API for various logging frameworks. For this implementation, Logback was chosen as the backend due to its popularity and flexibility. The objective was to log both an error message and a warning message in a simple Java application using SLF4J.

**Maven Project Configuration**

The SLF4J API and Logback Classic dependencies were added to the Maven pom.xml file to support the logging system. These dependencies connect the logging interface to an actual implementation.

**pom.xml Configuration**

<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>

Once the dependencies were added, the project was reloaded so that Maven could resolve and download the required libraries.

**Java Implementation**

A simple Java class was created to simulate a situation where two log levels — error and warning — are used. The logger was initialized using LoggerFactory from SLF4J. Inside the main method, two log statements were added — one representing a system error, and the other a potential issue that should be monitored as a warning.

**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");

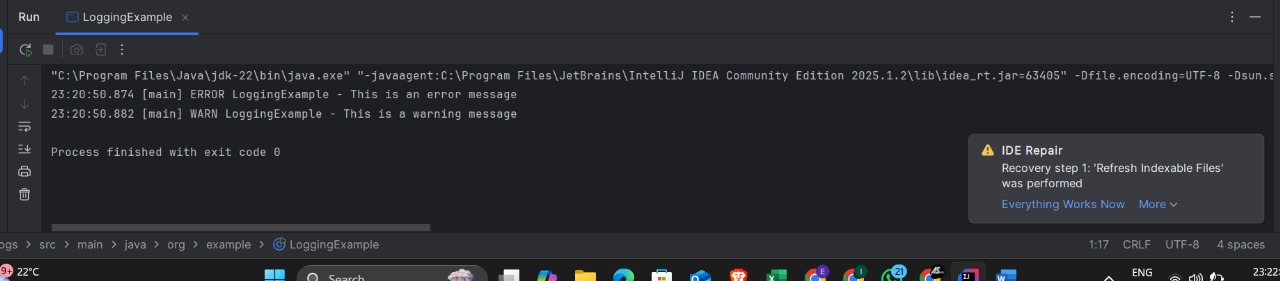
logger.warn("This is a warning message");

}

}

The logger.error() method was used to simulate a high-severity issue, such as system failure or application crash. The logger.warn() method was used to log a less critical issue that still deserves attention, such as invalid configuration or unexpected behavior that doesn't break functionality.

**Output:**



**Conclusion**

This exercise introduced the core concept of logging using SLF4J with Logback as its implementation. It demonstrated how easy it is to integrate and utilize structured logging in Java applications. Logging is an essential part of any software system, and this foundational implementation ensures that messages of different severity levels are correctly captured and can be traced effectively. As applications scale, the same mechanism can be extended to include info, debug, and trace levels for more granular logging control.