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

**Objective:**  
To demonstrate the integration of SLF4J with Logback for logging error and warning messages in a Java application, using Maven for dependency management.

**Scenario:**  
A Java-based application requires a lightweight and efficient logging mechanism. This exercise focuses on implementing SLF4J (Simple Logging Facade for Java) with Logback as the logging backend. The aim is to generate error and warning logs that are output to the console.

**Implementation Details**

**1. Maven Dependencies Configuration**

To enable SLF4J with Logback, the following dependencies were added to the pom.xml file:

<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>

These dependencies ensure that:

* slf4j-api provides the logging abstraction.
* logback-classic serves as the SLF4J-compatible logging implementation.

**2. Java Class Implementation**

A simple Java class was created to demonstrate logging at the error and warning levels:

package com.example;

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");

}

}

* LoggerFactory.getLogger() is used to instantiate the logger associated with the current class.
* logger.error() logs messages at the **ERROR** level, typically used for critical failures.
* logger.warn() logs messages at the **WARNING** level, used to highlight potential issues.

**3. Output**

