Logging using SLF4J

Exercise 1: Logging Error Messages and Warning Levels

Task: Write a Java application that demonstrates logging error messages and warning levels using SLF4J.

Step-by-Step Solution:

1. Add SLF4J and Logback dependencies to your `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>
```

2. Create a Java class that uses SLF4J for logging:

```
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");
   }
}
```

pom.xml

```
<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-api</artifactId>
```

```
    <version>1.7.30</version>
</dependency>

<!-- Logback Classic - Actual Logger Implementation -->
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.2.3</version>
</dependency>
```

LoggingExample.java

```
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");
    }
}
```

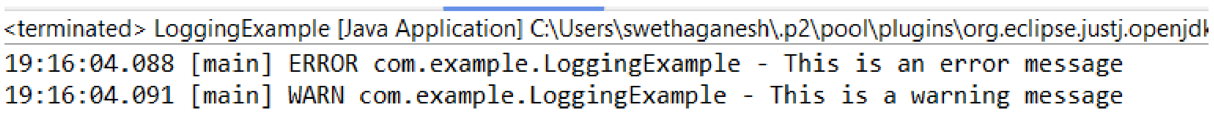
Explanation of Key Concepts:

In this application:

- SLF4J is used as the logging API that provides a uniform interface.
- Logback is used as the logging engine that actually processes and prints logs.
- The LoggerFactory.getLogger() method creates a logger instance for the LoggingExample class.
- We use:
    - logger.error(...) to print high-severity issues.
    - logger.warn(...) to indicate potential problems or warnings.

This separation allows flexibility — developers can switch logging implementations (e.g., from Logback to Log4j) without changing application code.

OUTPUT:



```
<terminated> LoggingExample [Java Application] C:\Users\swethaganesh\.p2\pool\plugins\org.eclipse.justj.openjdk
19:16:04.088 [main] ERROR com.example.LoggingExample - This is an error message
19:16:04.091 [main] WARN com.example.LoggingExample - This is a warning message
```