

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>
```

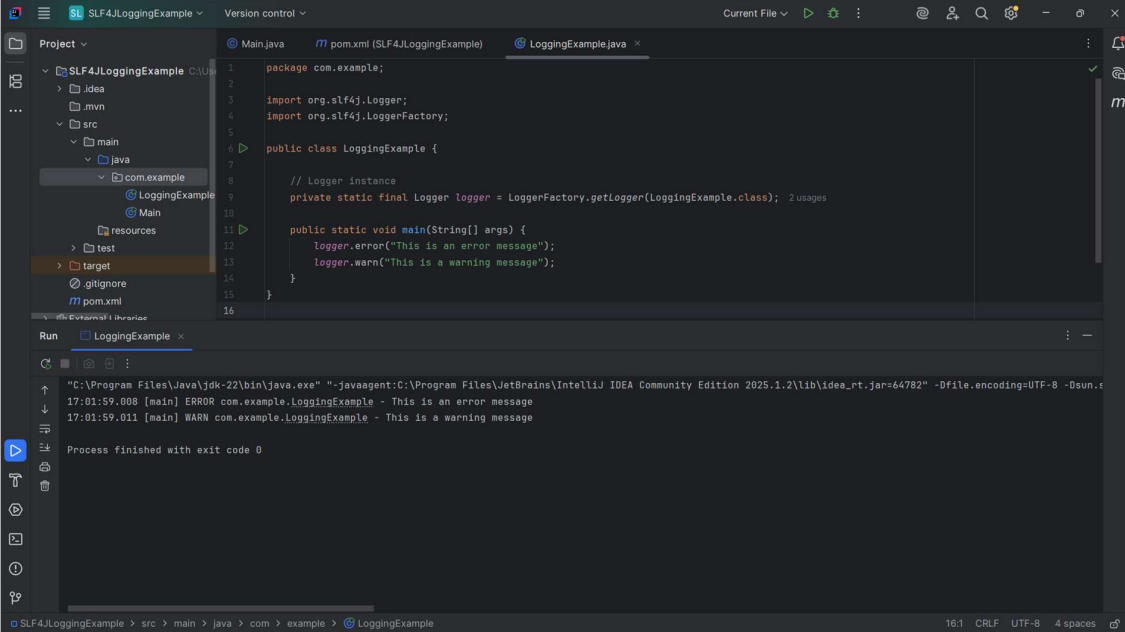
2. Create a Java class that uses SLF4J for logging:

```
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"); } }
```

Ouput:



The screenshot displays the IntelliJ IDEA IDE interface. The top toolbar shows the 'Run' button (a green play icon). The left sidebar contains the 'Project' view, showing a directory structure with 'src/main/java/com/example' selected. The main editor window shows the 'LoggingExample.java' file with the following code:

```
1 package com.example;  
2  
3 import org.slf4j.Logger;  
4 import org.slf4j.LoggerFactory;  
5  
6 public class LoggingExample {  
7  
8     // Logger instance  
9     private static final Logger logger = LoggerFactory.getLogger(LoggingExample.class); 2 usages  
10  
11     public static void main(String[] args) {  
12         logger.error("This is an error message");  
13         logger.warn("This is a warning message");  
14     }  
15 }  
16
```

Below the editor, the 'Run' tab is active, showing the output of the program:

```
Run LoggingExample  
C:\Program Files\Java\jdk-22\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.2\lib\idea_rt.jar=64782" -Dfile.encoding=UTF-8 -Dsun.java2d.crispEdges=true  
17:01:59.008 [main] ERROR com.example.LoggingExample - This is an error message  
17:01:59.011 [main] WARN com.example.LoggingExample - This is a warning message  
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8 and the line length is 161.

Exercise 2: Parameterized Logging Task:

Write a Java application that demonstrates parameterized logging using SLF4J.

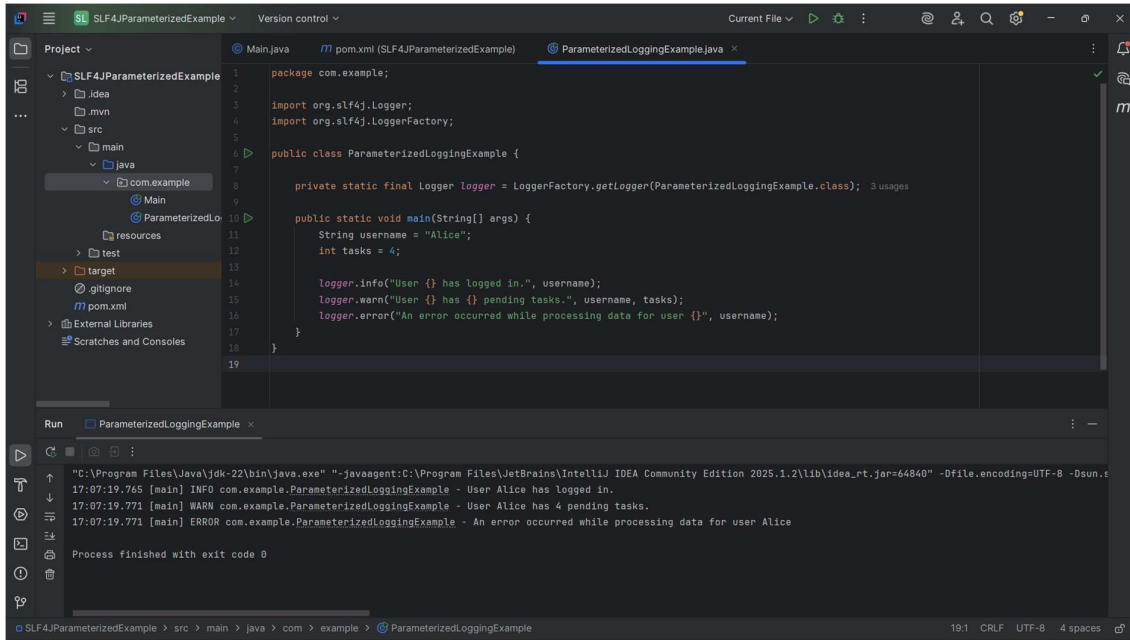
Step-by-Step Solution:

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 parameterized logging:

Output:



Exercise 3: Using Different Appenders Task:

Write a Java application that demonstrates using different appenders with 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 `logback.xml` configuration file to define different appenders:

```
<configuration>
<appender name="console"
class="ch.qos.logback.core.ConsoleAppender">
<encoder>
<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} -
%msg%n</pattern>
```

```
</encoder>
</appender>
<appender name="file"
class="ch.qos.logback.core.FileAppender">
<file>app.log</file>
<encoder>
<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} -
%msg%n</pattern>
</encoder>
</appender>
<root level="debug">
<appender-ref ref="console" />
<appender-ref ref="file" />
</root>
</configuration>
```

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

Output:

