**Logging using SLF4J (Contains both Mandatory and Additional Exercises)**

# **Exercise 1: Logging Error Messages and Warning Levels (Mandatory Exercise)**

**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:**

Instead of adding to pom.xml, you’ve completed this step by manually downloading and adding the following JAR files to your project:

Included Dependencies (JAR Files):

* slf4j-api-1.7.30.jar
* logback-classic-1.2.3.jar
* logback-core-1.2.3.jar

1. **Create a Java class that uses SLF4J for logging:**

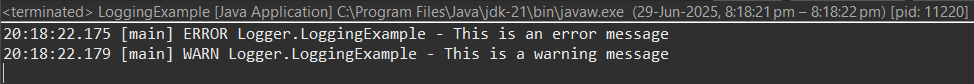
import org.slf4j.Logger; import org.slf4j.LoggerFactory;

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");

} }

# **Output:**



# **Exercise 2: Parameterized Logging (Additional Exercise)**

**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:**

Instead of adding to pom.xml, you’ve completed this step by manually downloading and adding the following JAR files to your project:

Included Dependencies (JAR Files):

* slf4j-api-1.7.30.jar
* logback-classic-1.2.3.jar
* logback-core-1.2.3.jar

1. **Create a Java class that uses SLF4J for parameterized logging:**

**ParameterizedLoggingExample.java**

package Logger;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class ParameterizedLoggingExample {

private static final Logger logger = LoggerFactory.getLogger(ParameterizedLoggingExample.class);

public static void main(String[] args) {

String username = "sonasri";

int loginCount = 5;

logger.info("User {} has logged in {} times.", username, loginCount);

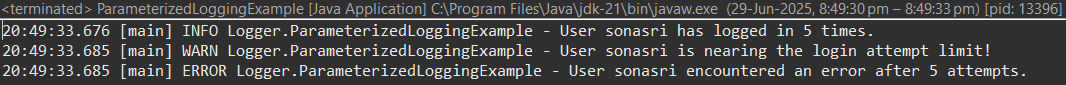
logger.warn("User {} is nearing the login attempt limit!", username);

logger.error("User {} encountered an error after {} attempts.", username, loginCount);

}

}

**Output:**



# **Exercise 3: Using Different Appenders (Additional Exercise)**

**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:**

Instead of adding to pom.xml, you’ve completed this step by manually downloading and adding the following JAR files to your project:

Included Dependencies (JAR Files):

* slf4j-api-1.7.30.jar
* logback-classic-1.2.3.jar
* logback-core-1.2.3.jar

1. **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:**

**MultiAppenderLogger.java**

package Logger;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class MultiAppenderLogger {

private static final Logger logger = LoggerFactory.getLogger(MultiAppenderLogger.class);

public static void main(String[] args) {

logger.debug("This is a DEBUG message");

logger.info("This is an INFO message");

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

logger.error("This is an ERROR message");

}

}

**Output:**

