**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>

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");

}

}

**Program:**

**pom.xml**

<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>

**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");

}

}

**logback.xml**

<configuration>

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

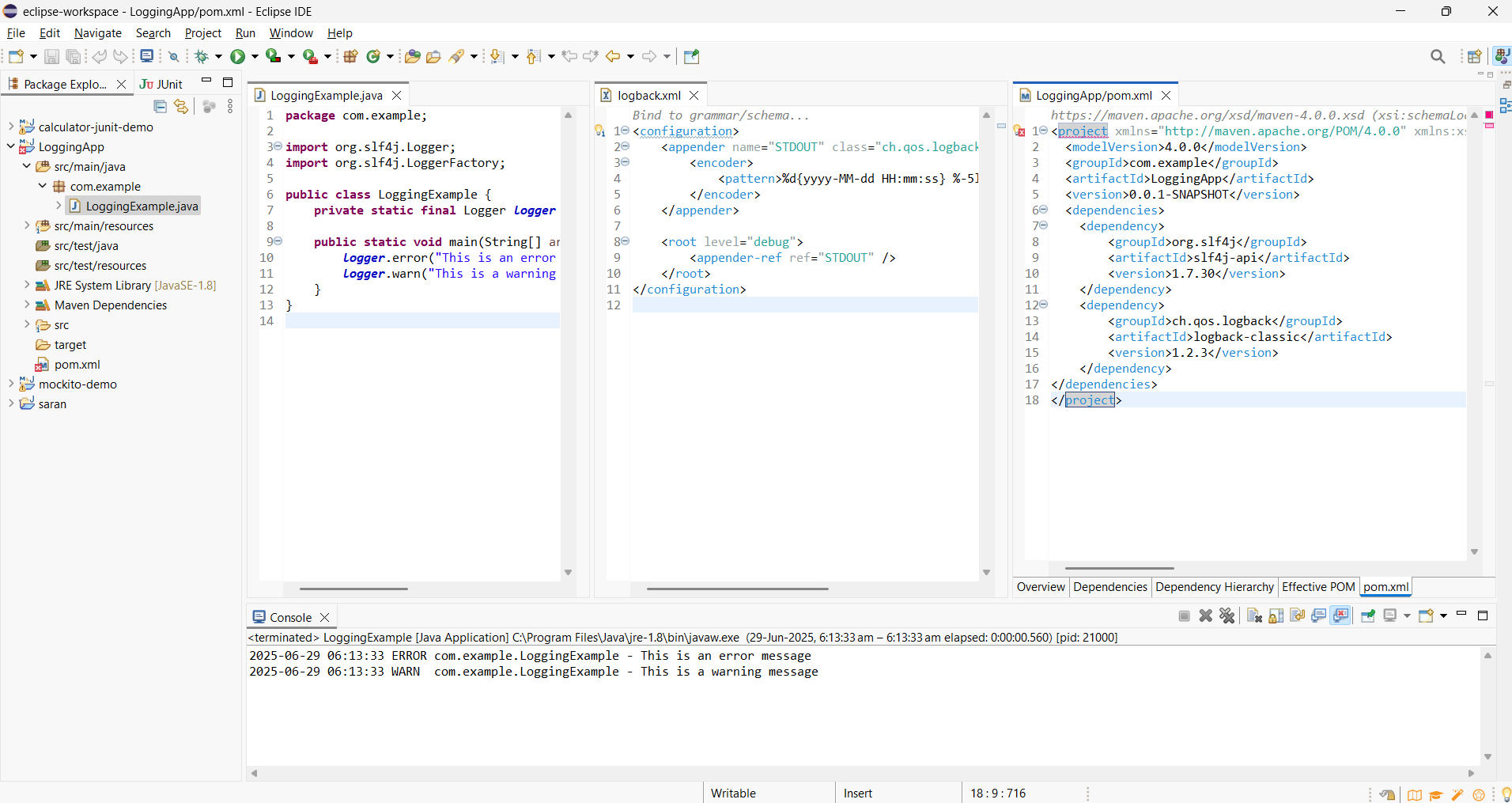
<root level="debug">

<appender-ref ref="STDOUT" />

</root>

</configuration>

**Output:**

****

**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:

Write code for this.

**Program:**

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>LoggingApp</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<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>

</dependencies>

</project>

**ParameterizedLoggingExample.java**

package com.example;

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 = "Poorani";

int userAge = 25;

double accountBalance = 1530.75;

*logger*.info("User {} has logged in", userName);

*logger*.debug("User {} is {} years old", userName, userAge);

*logger*.warn("User {} has a low account balance: ${}", userName, accountBalance);

*logger*.error("An error occurred while processing data for user: {}", userName);

}

}

**Logback.xml**

<configuration>

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

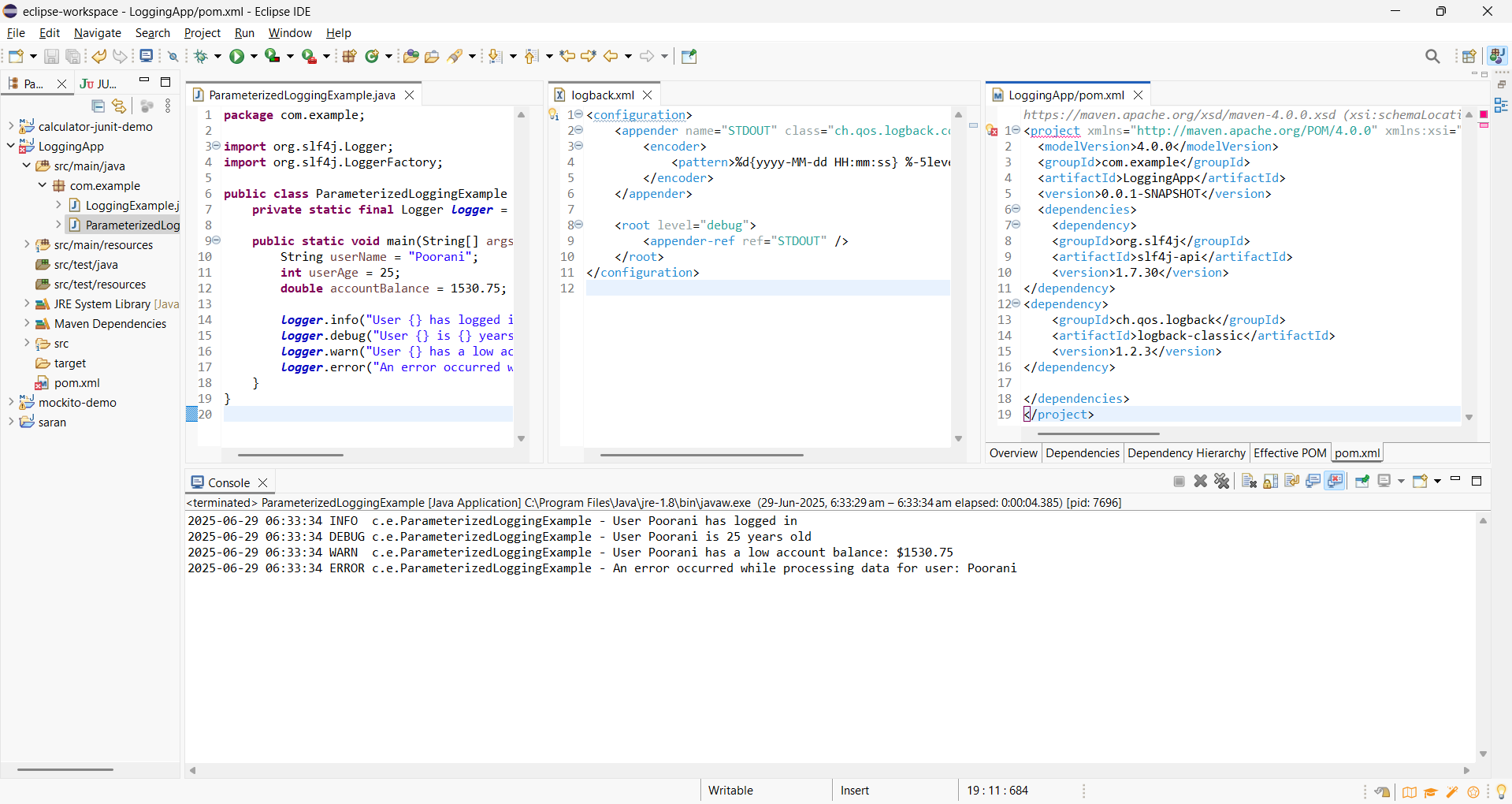
<root level="debug">

<appender-ref ref="STDOUT" />

</root>

</configuration>

**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:

Write code for this.

**Program:**

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>LoggingApp</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<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>

</dependencies>

</project>

**MultiAppenderExample.java**

package com.example;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class MultiAppenderExample {

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

public static void main(String[] args) {

*logger*.debug("Debug message - useful for developers");

*logger*.info("Info message - general information");

*logger*.warn("Warning message - something unusual");

*logger*.error("Error message - something failed");

}

}

**logback.xml**

<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>logs/app.log</file> <!-- this will create logs/app.log file -->

<append>true</append>

<encoder>

<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

<root level="debug">

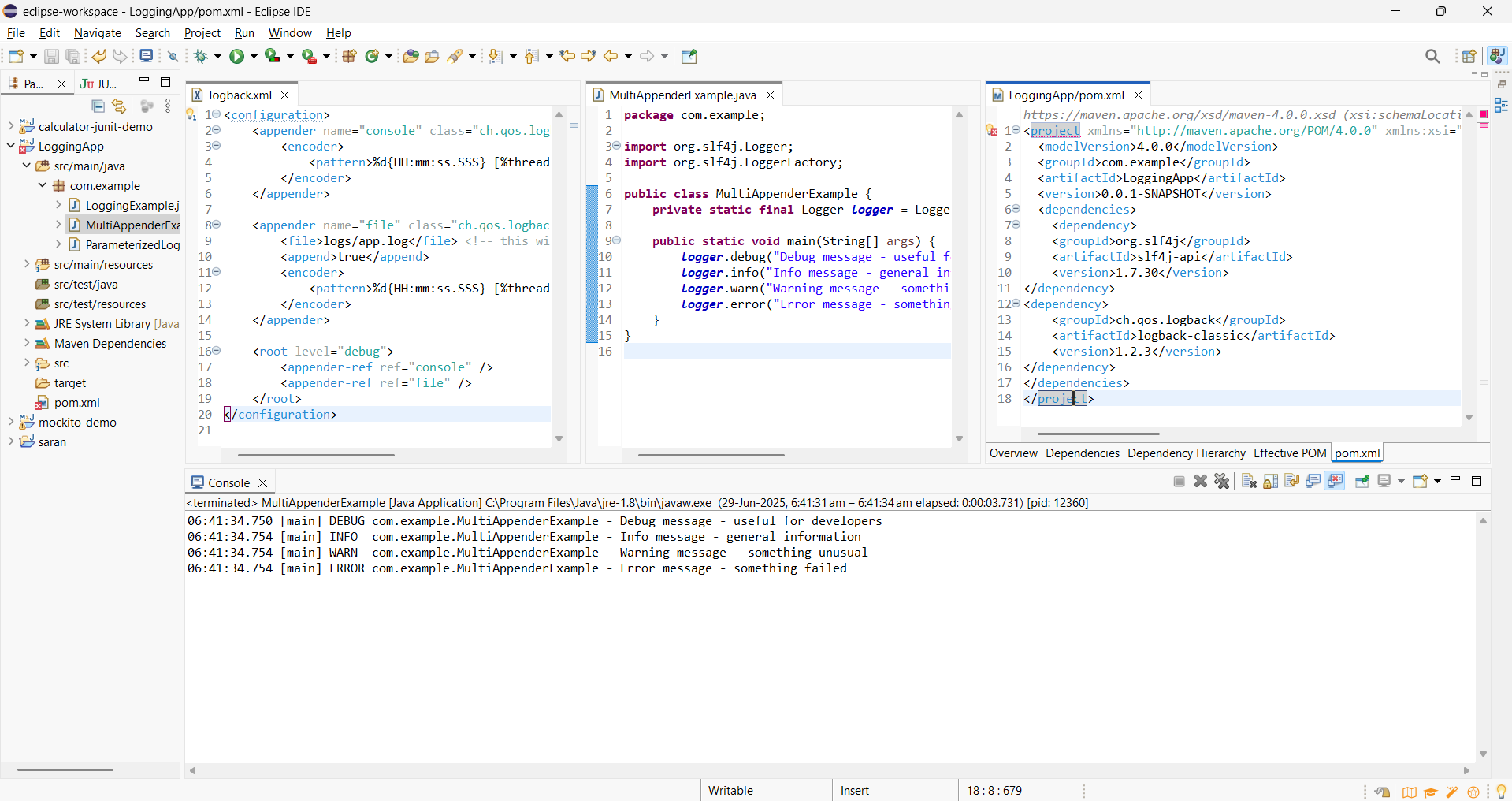
<appender-ref ref="console" />

<appender-ref ref="file" />

</root>

</configuration>

**Output:**

****