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:

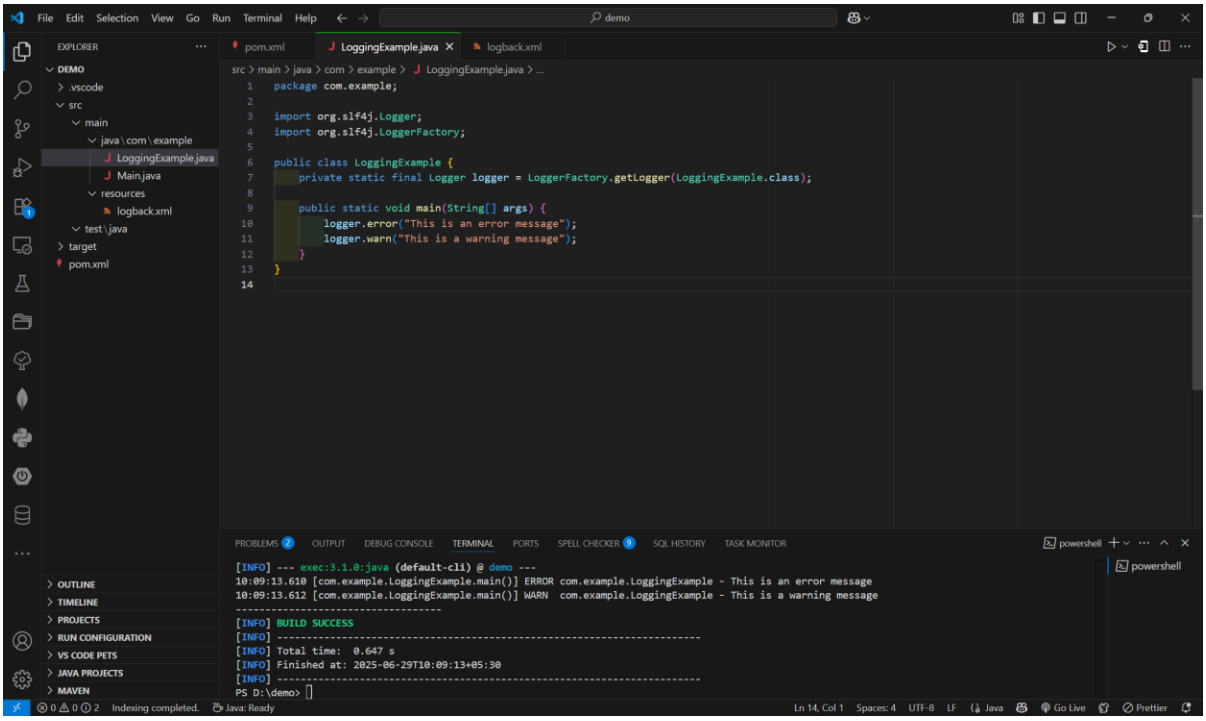
1. Add SLF4J and Logback dependencies to your `pom.xml` file:

```
<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");
  }
}
```

```
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec:3.1.0:java (default-cli) @ demo ---
10:09:13.610 [com.example.LoggingExample.main()] ERROR com.example.LoggingExample - This is an error message
10:09:13.612 [com.example.LoggingExample.main()] WARN  com.example.LoggingExample - This is a warning message
------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.647 s
```



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:

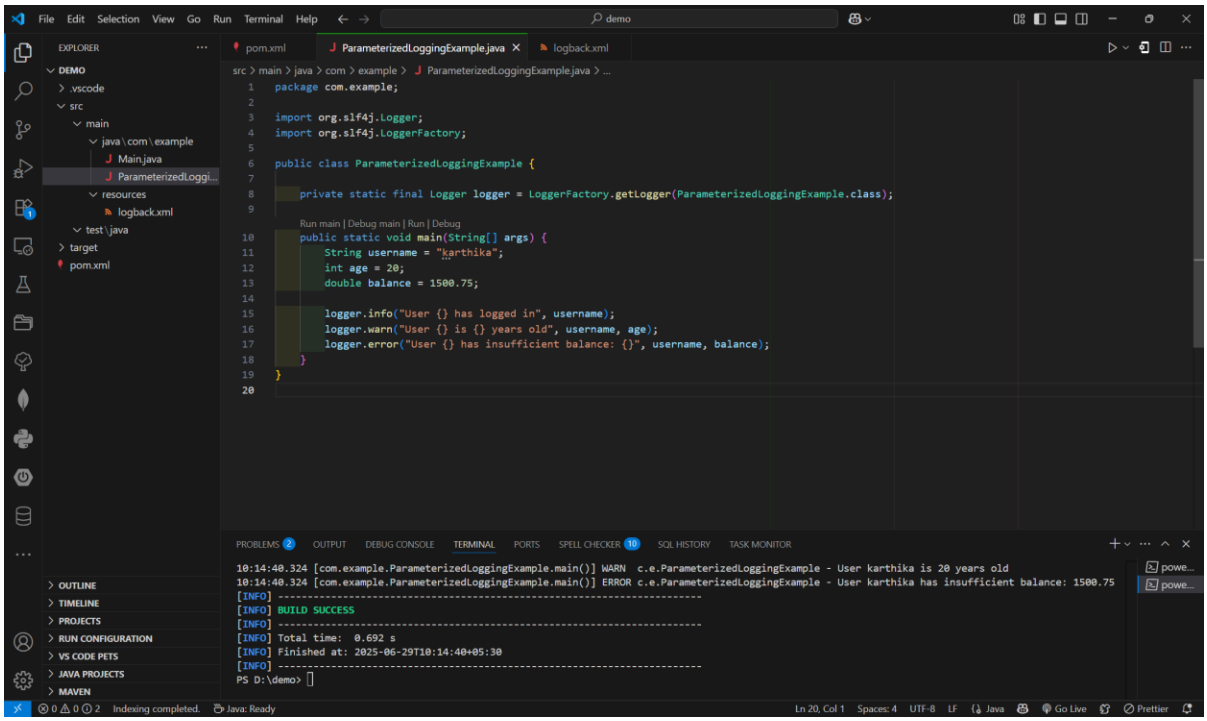
```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>1.7.30</version>
</dependency>
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>1.2.3</version>
</dependency>
```

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

```
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 = "karthika";
        int age = 20;
        double balance = 1500.75;

        logger.info("User {} has logged in", username);
        logger.warn("User {} is {} years old", username, age);
        logger.error("User {} has insufficient balance: {}", username, balance);
    }
}
```

```
[INFO] --- exec:3.1.0:java (default-cli) @ demo ---
10:14:40.321 [com.example.ParameterizedLoggingExample.main()] INFO  c.e.ParameterizedLoggingExample - User karthika has logged in
10:14:40.324 [com.example.ParameterizedLoggingExample.main()] WARN  c.e.ParameterizedLoggingExample - User karthika is 20 years old
10:14:40.324 [com.example.ParameterizedLoggingExample.main()] ERROR c.e.ParameterizedLoggingExample - User karthika has insufficient balance: 1500.75
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.692 s
[INFO] Finished at: 2025-06-29T10:14:40+05:30
[INFO] ------------------------------------------------------------------------
```



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:

```
package com.example;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class AppenderExample {
```

```
   private static final Logger logger =
LoggerFactory.getLogger(AppenderExample.cla
ss);
   public static void main(String[] args) {
      logger.info("This is an info message.");
      logger.warn("This is a warning message.");
      logger.error("This is an error message.");
   }
}
```

```
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec:3.1.0:java (default-cli) @ demo ---
10:19:18.437 [com.example.AppenderExample.main()] INFO  com.example.AppenderExample - This is an info message.
10:19:18.438 [com.example.AppenderExample.main()] WARN  com.example.AppenderExample - This is a warning message.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.598 s
```