**Exercise -1**

**Implementing the Singleton Pattern**

**Scenario:**

You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named SingletonPatternExample**.**
2. **Define a Singleton Class:**
   * Create a class named Logger that has a private static instance of itself.
   * Ensure the constructor of Logger is private.
   * Provide a public static method to get the instance of the Logger class.
3. **Implement the Singleton Pattern:**
   * Write code to ensure that the Logger class follows the Singleton design pattern.
4. **Test the Singleton Implementation:**
   * Create a test class to verify that only one instance of Logger is created and used across the application.

**Answer:**

* + 1. Created a folder SinglePatternExample.

**Code:**

// file name Logger.java

// Create a class named Logger

package designPattern;

public class Logger {

private static Logger instance;

private Logger(){

System.out.println("Singleton design pattern --- Logger class");

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void login(String msg) {

System.out.println("Log: " + msg);

}

}

// file name LoggerTest.java

import designPattern.Logger;

public class LoggerTest {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance(); // First call

Logger logger2 = Logger.getInstance(); // Second call

logger1.login("First login message");

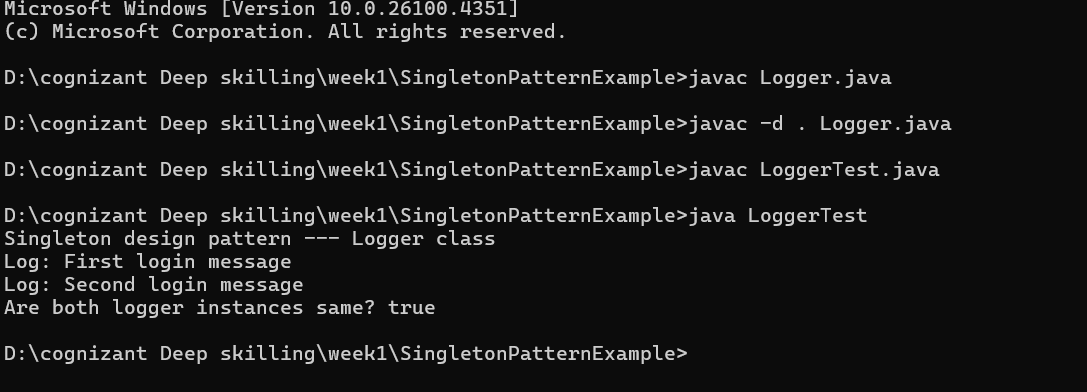
logger2.login("Second login message");

System.out.println("Are both logger instances same? " + (logger1 == logger2));

}

}

**Output:**

****