**Exercise 1: Implementing the Singleton Pattern**

**OBJECTIVE:**

To design a logging utility class that restricts instantiation to a single object across the application's lifecycle, ensuring consistent logging operations.

**CODE:**

public class Singleton {

public static class Logger {

private static volatile Logger instance;

private Logger() {

System.out.println("Logger successfully initialized.");

}

public static Logger getInstance() {

if (instance == null) {

synchronized (Logger.class) {

if (instance == null) {

instance = new Logger();

}

}

}

return instance;

}

public void log(String message) {

System.out.println("Log Entry: " + message);

}

}

public static void main(String[] args) {

Logger loggerA = Logger.getInstance();

loggerA.log("Initializing logging process...");

Logger loggerB = Logger.getInstance();

loggerB.log("Continuing with logging...");

if (loggerA == loggerB) {

System.out.println(" loggerA and loggerB refer to the same Logger instance.");

} else {

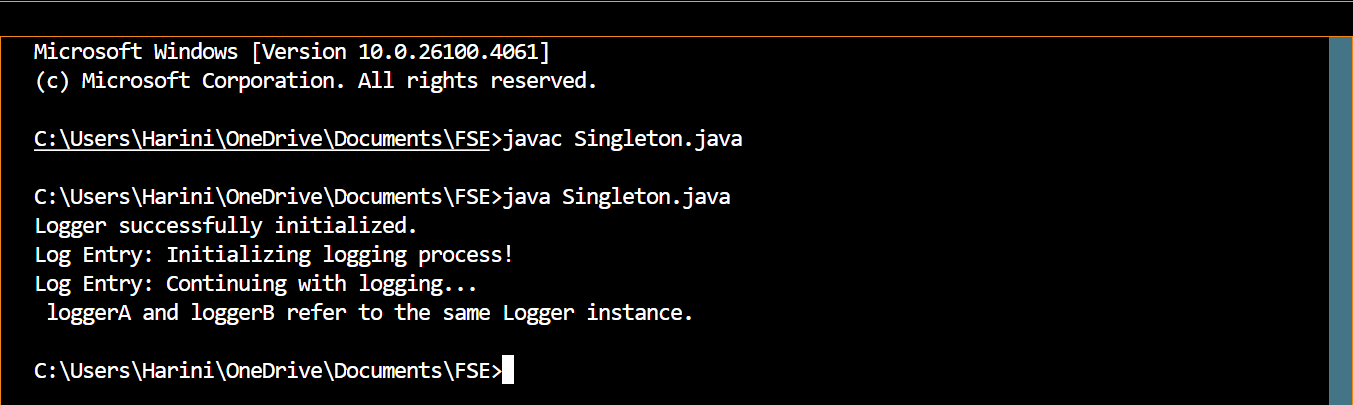
System.out.println(" loggerA and loggerB refer to different instances.");

}

}

}

**OUTPUT:**

****