**Design Patterns and Principles**

**EXERSICE-1 : IMPLEMENTING 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.

**SOURCE CODE:**

public class Logger {

    private static volatile Logger instance;

    private Logger() {

        // Initialization code here

        System.out.println("Logger 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] " + message);

    }

}

public class Main {

    public static void main(String[] args) {

        Logger logger1 = Logger.getInstance();

        Logger logger2 = Logger.getInstance();

        logger1.log("Starting the application...");

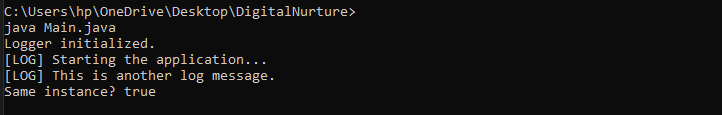
        logger2.log("This is another log message.");

        System.out.println("Same instance? " + (logger1 == logger2)); // true

    }

}

**OUTPUT:**

****