**Exercise 1: Implementing the Singleton Pattern**

// Filename :- Logger.java

public class Logger {

    private static Logger instance;

    private Logger() {

        System.out.println("Logger initialized. Ready to capture logs.");

    }

    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("[INFO] " + java.time.LocalDateTime.now() + " - " + message);

    }

    public void warn(String message) {

        System.out.println("[WARN] " + java.time.LocalDateTime.now() + " - " + message);

    }

    public void error(String message) {

        System.out.println("[ERROR] " + java.time.LocalDateTime.now() + " - " + message);

    }

}

// TestLogger.java

// Class to test Singleton Logger functionality

public class TestLogger {

    public static void main(String[] args) {

        System.out.println("Application started...\n");

        Logger logger1 = Logger.getInstance();

        Logger logger2 = Logger.getInstance();

        logger1.log("System initialized successfully.");

        logger2.warn("Memory usage is reaching the threshold.");

        logger1.error("Failed to connect to the database.");

        if (logger1 == logger2) {

            System.out.println("\n Singleton test passed: Both references point to the same Logger instance.");

        } else {

            System.out.println("\n Singleton test failed: Logger instances are different.");

        }

        System.out.println("\nApplication terminated.");

    }

}

Output :-

