**WEEK 1**

**DESIGN PATTERN AND PRINCIPLES**

**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.

**Code:**

public class Main {

static class Logger {

private static Logger instance;

private Logger() {

System.out.println("Logger Initialized");

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

System.out.println("[LOG]: " + message);

}

}

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

logger1.log("Application started.");

logger2.log("This should use the same logger instance.");

if (logger1 == logger2) {

System.out.println("Both logger instances are the same (Singleton Verified).");

} else {

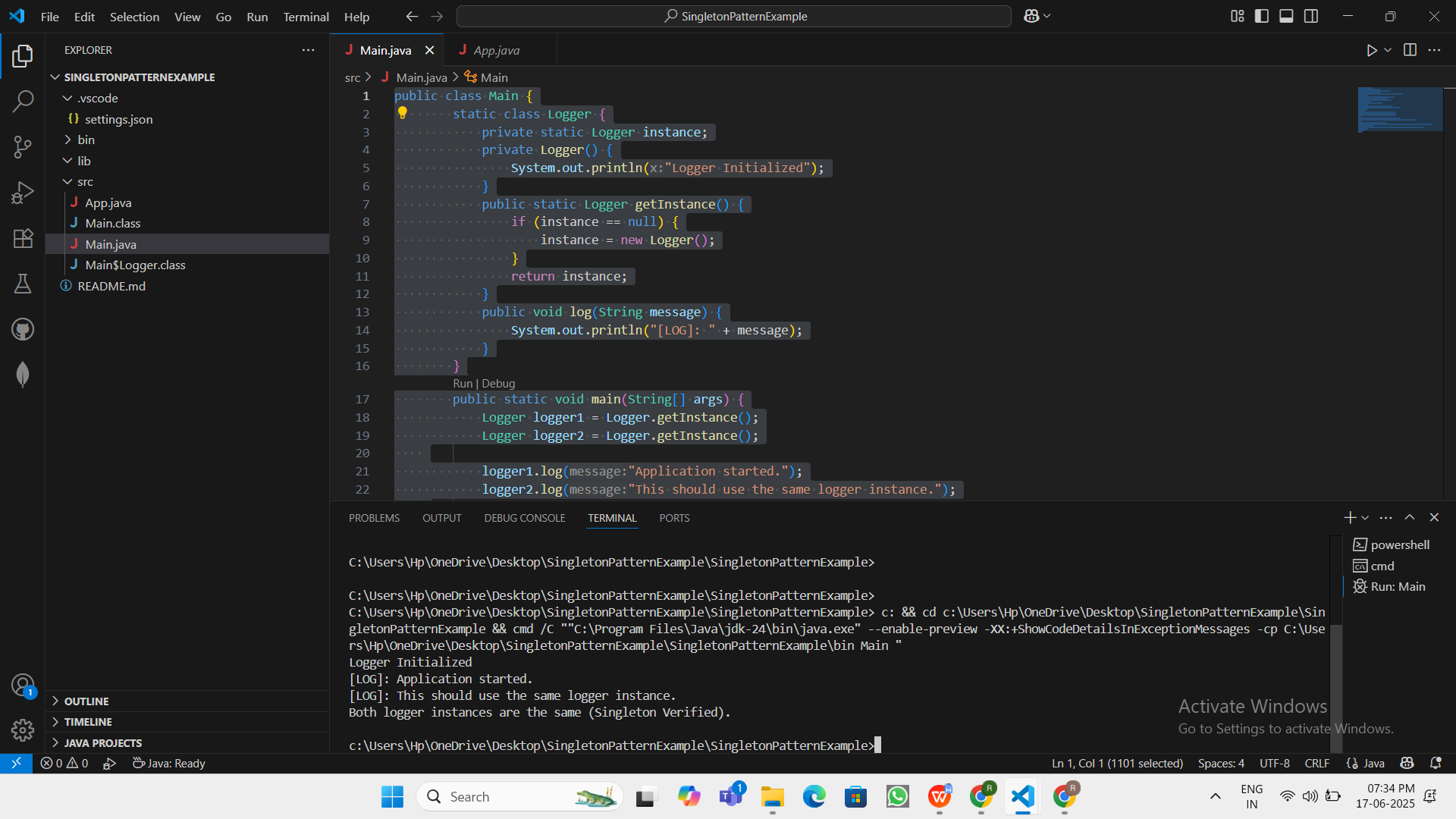
System.out.println("Different instances exist (Singleton Broken).");

}

}

}

**OUTPUT:**



Logger Initialized

[LOG]: Application started.

[LOG]: This should use the same logger instance.

Both logger instances are the same (Singleton Verified).