**Exercise 1: Implementing the Singleton Pattern**

**Problem Statement:**

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:**

**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 class SingletonPattern1 {**

**public static void main(String[] args) {**

**Logger logger1 = Logger.getInstance();**

**Logger logger2 = Logger.getInstance();**

**Logger logger3 = Logger.getInstance();**

**logger1.log("This is the 1st log message.");**

**logger2.log("This is the 2nd log message.");**

**logger3.log("This is the 3rd log message.");**

**if (logger1 == logger2 && logger2 == logger3) {**

**System.out.println("All logger instances here are the same. That means Singleton works!");**

**} else {**

**System.out.println("Different instances found. Singleton failed.");**

**}**

**}**

**}**

**Output Screenshot:**

**A screenshot of a computer

AI-generated content may be incorrect.**