**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.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **SingletonPatternExample**.
2. **Define a Singleton Class:**
   * Create a class named Logger that has a private static instance of itself.
   * Ensure the constructor of Logger is private.
   * Provide a public static method to get the instance of the Logger class.
3. **Implement the Singleton Pattern:**
   * Write code to ensure that the Logger class follows the Singleton design pattern.
4. **Test the Singleton Implementation:**
   * Create a test class to verify that only one instance of Logger is created and used across the application.

**// Solution:**

***// Java Project : Singleton design pattern:***

***// Logger.java***

**public class Logger {**

**//1. Created Class named Logger that has a private static instance of itself.**

**private static Logger *instance*;**

**// Ensured Logger is private.**

**private Logger(){**

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

**}**

**// Provided Public method to return the instance**

**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) {**

**// Getting Instance one**

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

**logger1.log("This is Instance one");**

**// Getting Instance Two**

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

**logger2.log("This is Instance Two");**

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

**System.*out*.println("Both logger1 and logger2 are the same");**

**}else {**

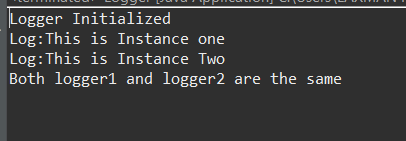
**System.*out*.println("Different instances exist");**

**}**

**}**

**}**

**// OUTPUT:**

****