**Design Patterns 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:

**Logger.java**

**public** **class** Logger {

// private static instance for object

**private** **static** Logger *instance*;

// Step 2: private constructor to stop create a object in multiple times

**private** Logger() {

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

}

// Step 3: public static method to get instance

**public** **static** Logger getInstance() {

**if** (*instance* == **null**) {

*instance* = **new** Logger();

}

**return** *instance*;

}

}

**Main.java**

**public** **class** Main {

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

Logger logger1 = Logger.*getInstance*();

Logger logger2 = Logger.*getInstance*();

System.***out***.println(logger1==logger2);

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

System.***out***.println("THIS FOLLOWS SINGLETON PATTERN");

}

**else** {

System.***out***.println("OTHER CLASS");

}

}

}

**OUTPUT**:

