**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:  
Main.java  
**package** com.singleton.example;

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

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

Logger logger1 = Logger.*getInstance*();

Logger logger2 = Logger.*getInstance*();

logger1.log("This is the first log message.");

logger2.log("This is the second log message.");

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

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

} **else** {

System.***out***.println("Logger instances are different (Singleton failed).");

}

}

}

Logger.java  
**package** com.singleton.example;

**public** **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);

}

}

A screenshot of a computer

AI-generated content may be incorrect.