**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.

**Implementation:**

**Code:**

**Logger.java**

**package** myproject.singleton;

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

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

**private** Logger() {

System.***out***.println("Logger instance created.");

}

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

**return** *instance*;

}

**public** **void** log(String message) {

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

}

}

**Main.java:**

**package** myproject.singleton;

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

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

Logger lg1 = Logger.*getInstance*();

Logger lg2 = Logger.*getInstance*();

lg1.log("This is a log message from logger1.");

lg2.log("This is a log message from logger2.");

**if** (lg1 == lg2) {

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

} **else** {

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

}

}

}

**Output:**

