**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.

**Souce Code:**

**Logger.java:**

**public** **class** Logger

{

**private** **static** Logger *ins*;

**private** Logger()

{

System.***out***.println("\nLogger initialized");

}

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

{

**if**(*ins*==**null**)

{

*ins*=**new** Logger();

}

**return** *ins*;

}

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

{

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

}

}

**Main.java:**

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

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

Logger l1=Logger.*getInstance*();

Logger l2=Logger.*getInstance*();

l1.log("Log-1");

l2.log("Log-2");

**if**(l1==l2)

{

System.***out***.println("\nBoth logger1 and logger2 refer to the same instance.");

}

**else**

{

System.***out***.println("\nDifferent instances are existing.");

}

}

}

**Output:**

Logger initialized

LOG:Log-1

LOG:Log-2

Both logger1 and logger2 refer to the same instance.