**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:

o Create a new Java project named SingletonPatternExample.

2. Define a Singleton Class:

o Create a class named Logger that has a private static instance of itself.

o Ensure the constructor of Logger is private.

o Provide a public static method to get the instance of the Logger class.

3. Implement the Singleton Pattern:

o Write code to ensure that the Logger class follows the Singleton design pattern.

4. Test the Singleton Implementation:

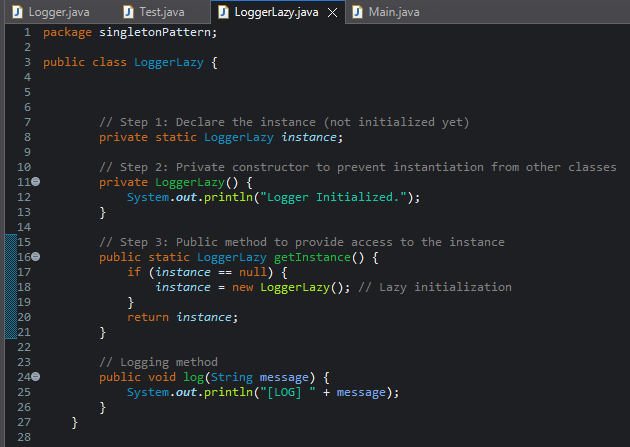
o Create a test class to verify that only one instance of Logger is created and used across the application.

**Solution:**

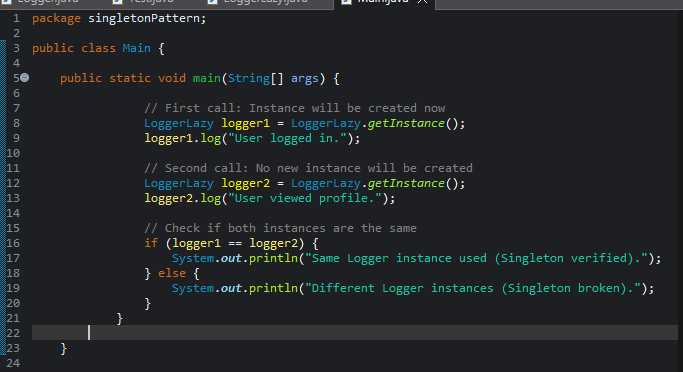
**Lazy Initialization:**

**Code:**

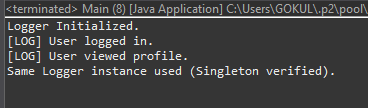
LoggerLazy.java



Main.java

****

**Output:**

****

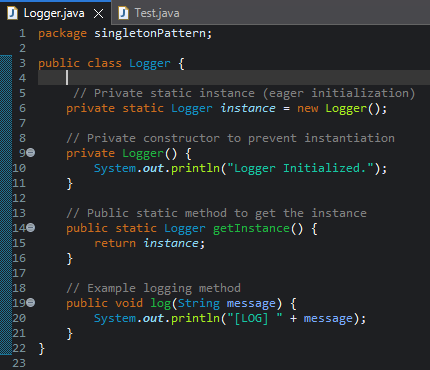
**Singleton Pattern:**A design pattern that restricts the instantiation of a class to one "single" object**.**

**Lazy Initialization:**The Singleton object is not created at the time of class loading.  
Instead, it is created the first time it's requested by the application**.**

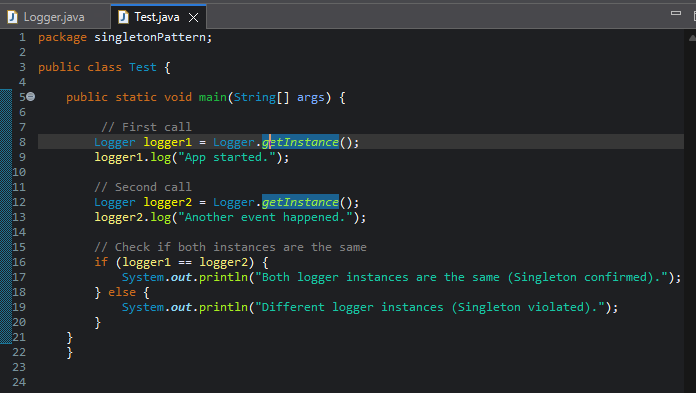
1. **Private Static Variable:**
   * A class-level variable holds the reference to the single instance.
   * It is initialized as null at the beginning.
2. **Private Constructor:**
   * Prevents the creation of instances from outside the class.
   * Ensures that only the class itself can create the instance.
3. **Public Static Method (getInstance):**
   * Checks whether the instance already exists.
   * If not, it creates the instance and returns it.
   * If it already exists, it simply returns the existing one.

**Early Initialization**

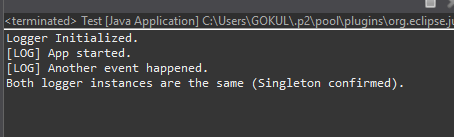
Logger.java

****

Test.java



**Output:**

****

**Eager Initialization:**The Singleton instance is created at the time of class loading, regardless of whether it's used later or not.

Private Static Variable:

* The Singleton instance is created and assigned immediately when declared.

Private Constructor:

* Prevents the creation of new instances from outside the class.

Public Static Method (getInstance):

* Simply returns the pre-created instance.
* No condition checking required.