**WEEK-01**

**HANDS ON EXERCISES SOLUTIONS**

Design patterns and principles

1]**Exercise 1: Implementing the Singleton Pattern**

Scenario:

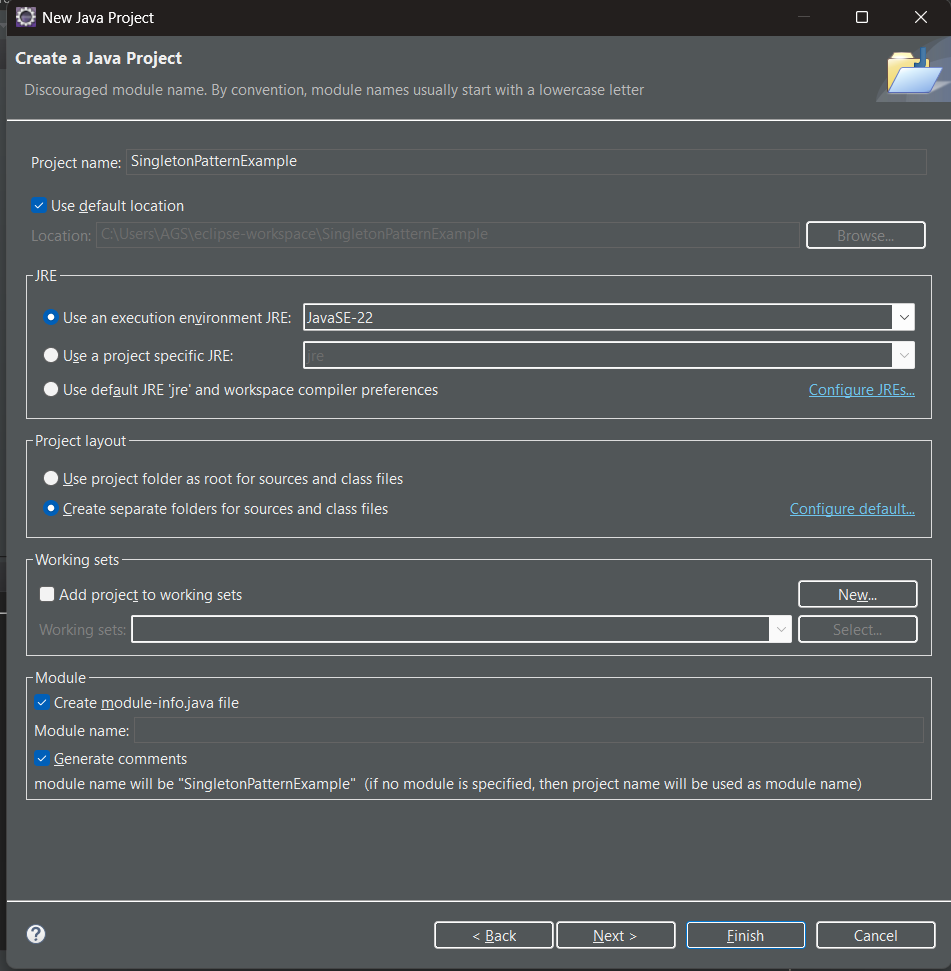
You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

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.

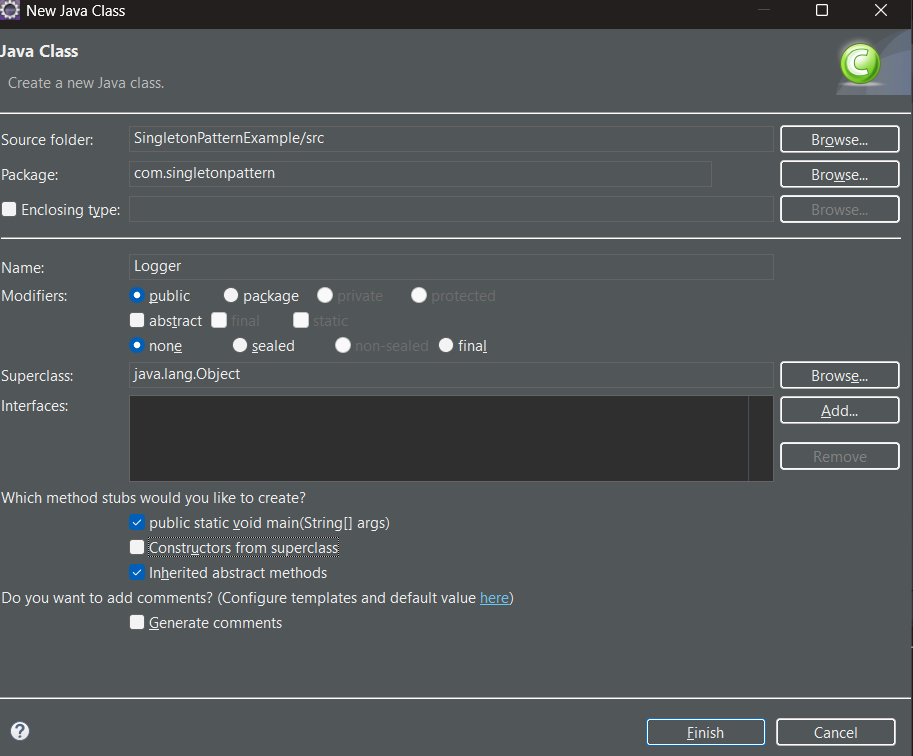
IMPLEMENTATION:

1.Created a new Java project named “SingletonPatternExample”

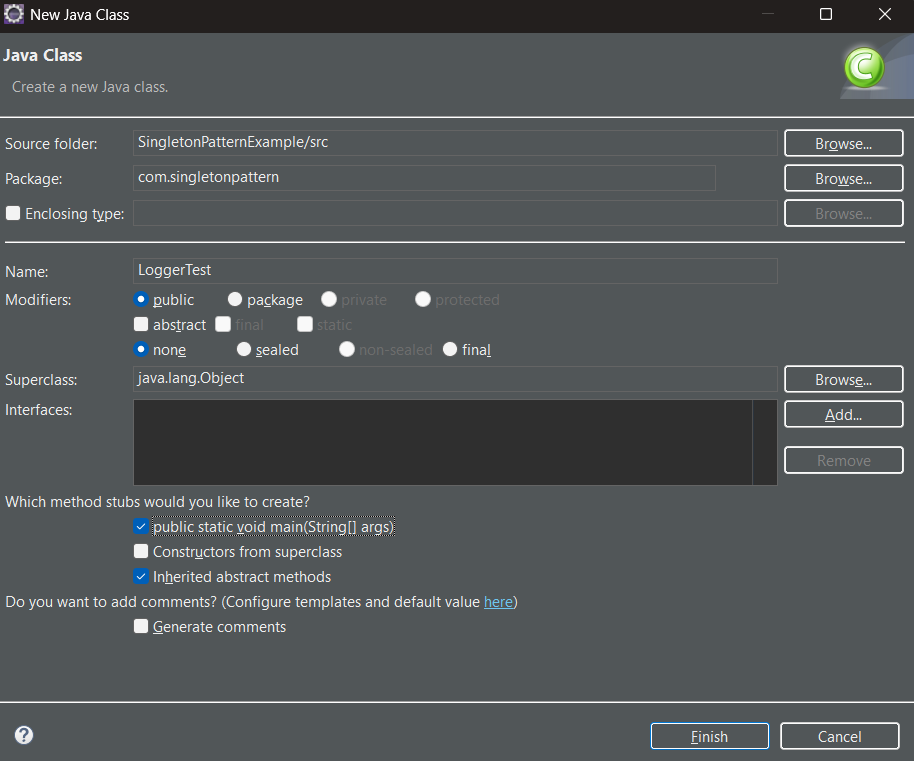




1. Defined a Singleton Class:
   * Created a class named Logger that has a private static instance of itself.
   * Ensured that the constructor of Logger is private.
   * Provided a public static method to get the instance of the Logger class.







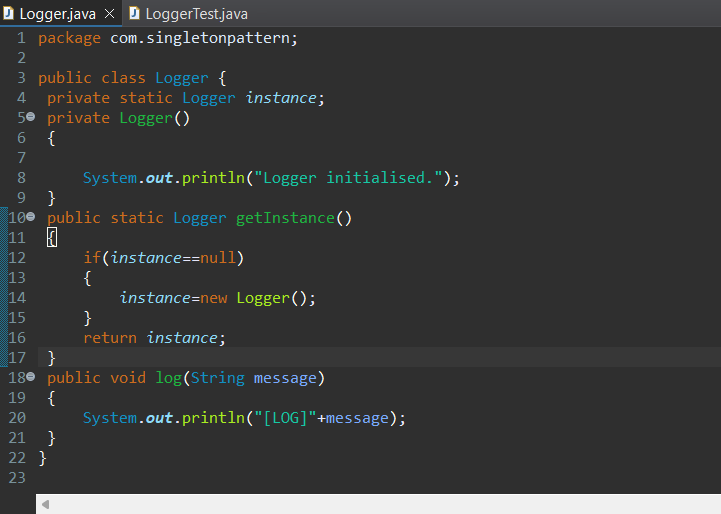


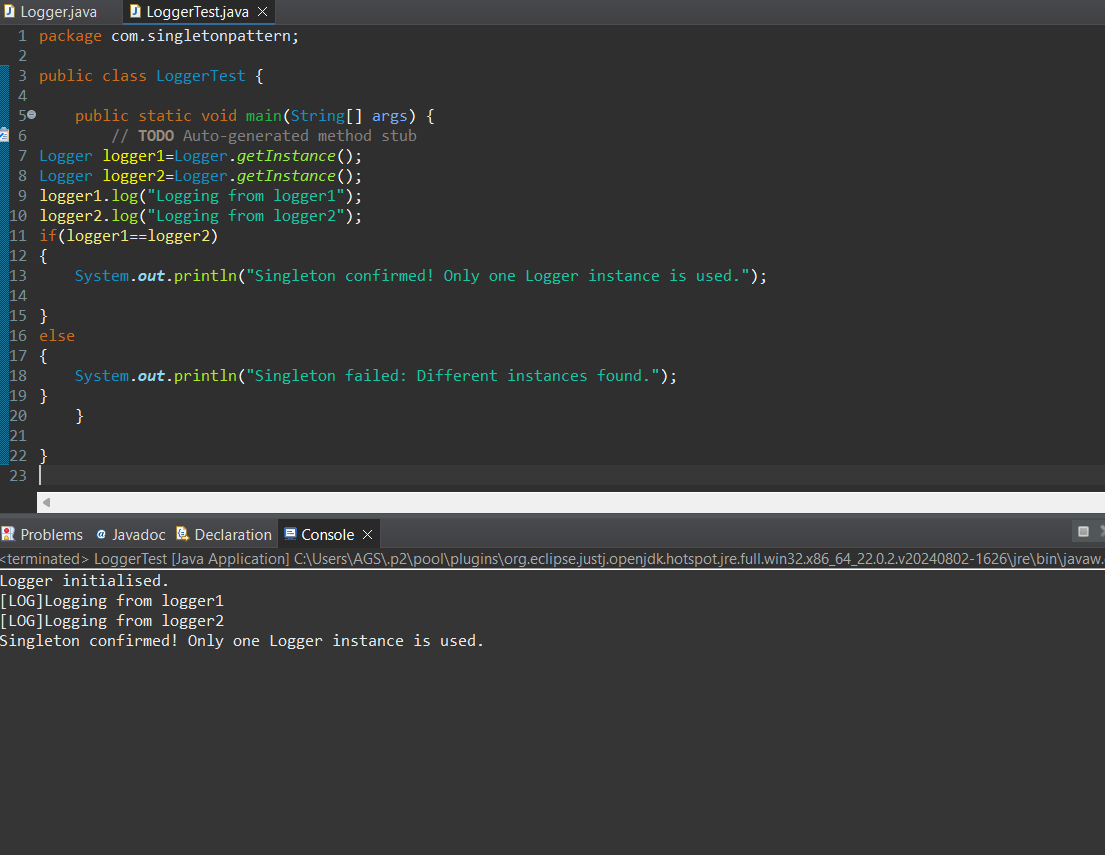
3. To implement the Singleton Pattern, the Logger class was carefully structured to restrict instantiation and provide a single global access point.

This was achieved by declaring a private static instance variable of the Logger class, ensuring that only one instance exists at any time.

The constructor was made private, which prevents any external class from creating new objects using the new keyword. A public static method getInstance() was provided to return the single instance, using lazy initialization—meaning the object is created only when it's first needed.

This setup guarantees that all components of the application use the same Logger instance, fulfilling the core principle of the Singleton Design Pattern.





OUTPUT:

