**WEEK – 1 Assignments**

**Design principles & Patterns**

**Mandatory hands-on :-**

**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.

**Codes :-**

**File names :**

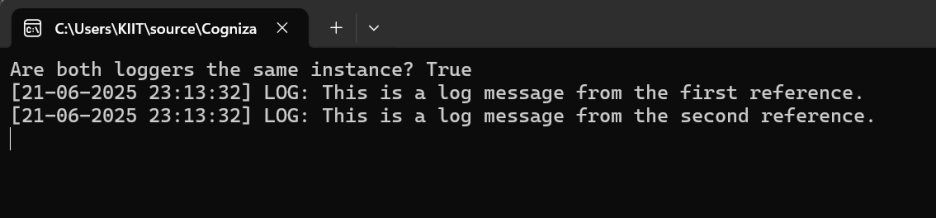
**Logger.cs :**

| public class Logger {  private static Logger? \_instance;  private static readonly object \_lock = new object();   private Logger() { }   public static Logger Instance  {  get  {  if (\_instance == null)  {  lock (\_lock)  {  if (\_instance == null)  \_instance = new Logger();  }  }  return \_instance;  }  }   // Example log method  public void Log(string message)  {  Console.WriteLine($"[{DateTime.Now}] LOG: {message}");  } } |
| --- |

**Program.cs :**

| using System;  class Program {  static void Main()  {  Logger firstLogger = Logger.Instance;  Logger secondLogger = Logger.Instance;   Console.WriteLine($"Are both loggers the same instance? {ReferenceEquals(firstLogger, secondLogger)}");   firstLogger.Log("This is a log message from the first reference.");  secondLogger.Log("This is a log message from the second reference.");  Console.ReadLine();  } } |
| --- |

**Output :-**

****