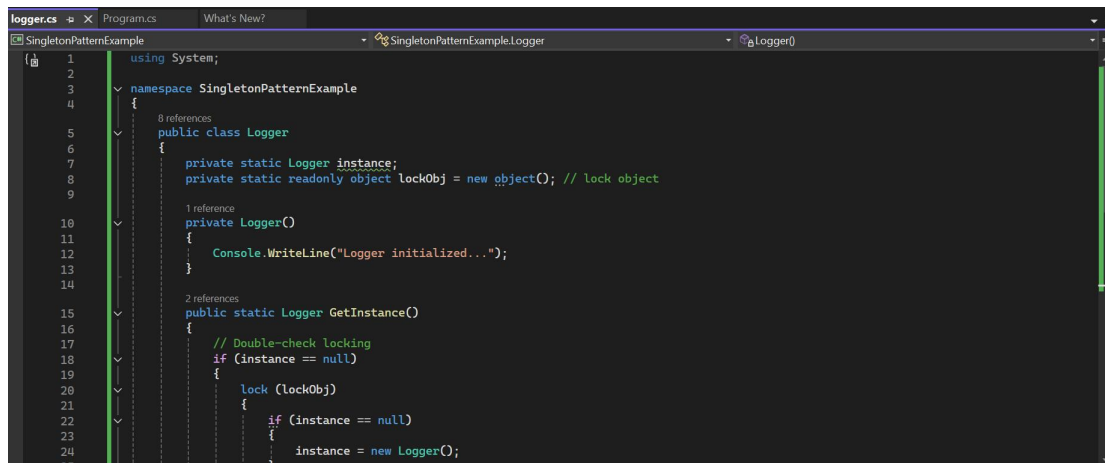


Exercise 1: Implementing the Singleton Pattern

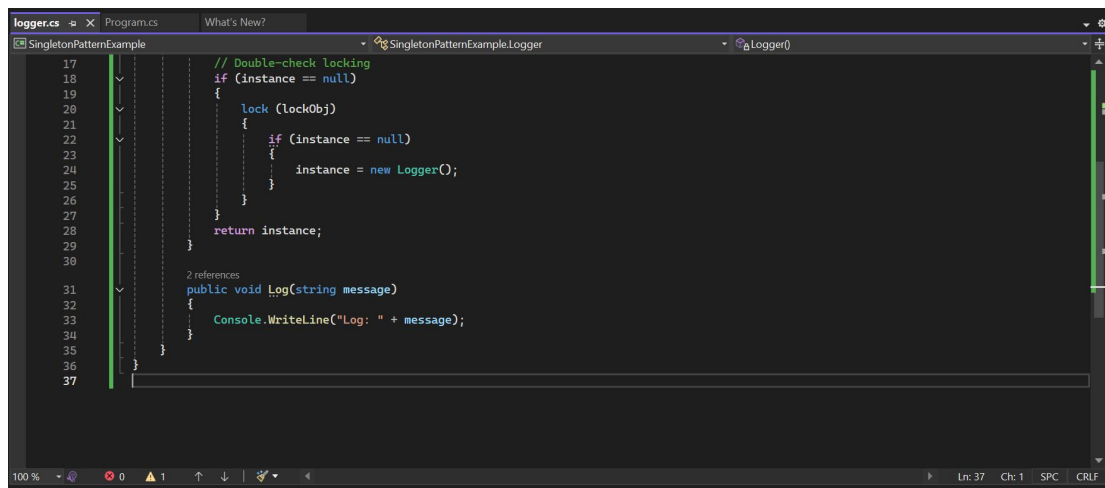
Scenario:

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

Logger.cs:

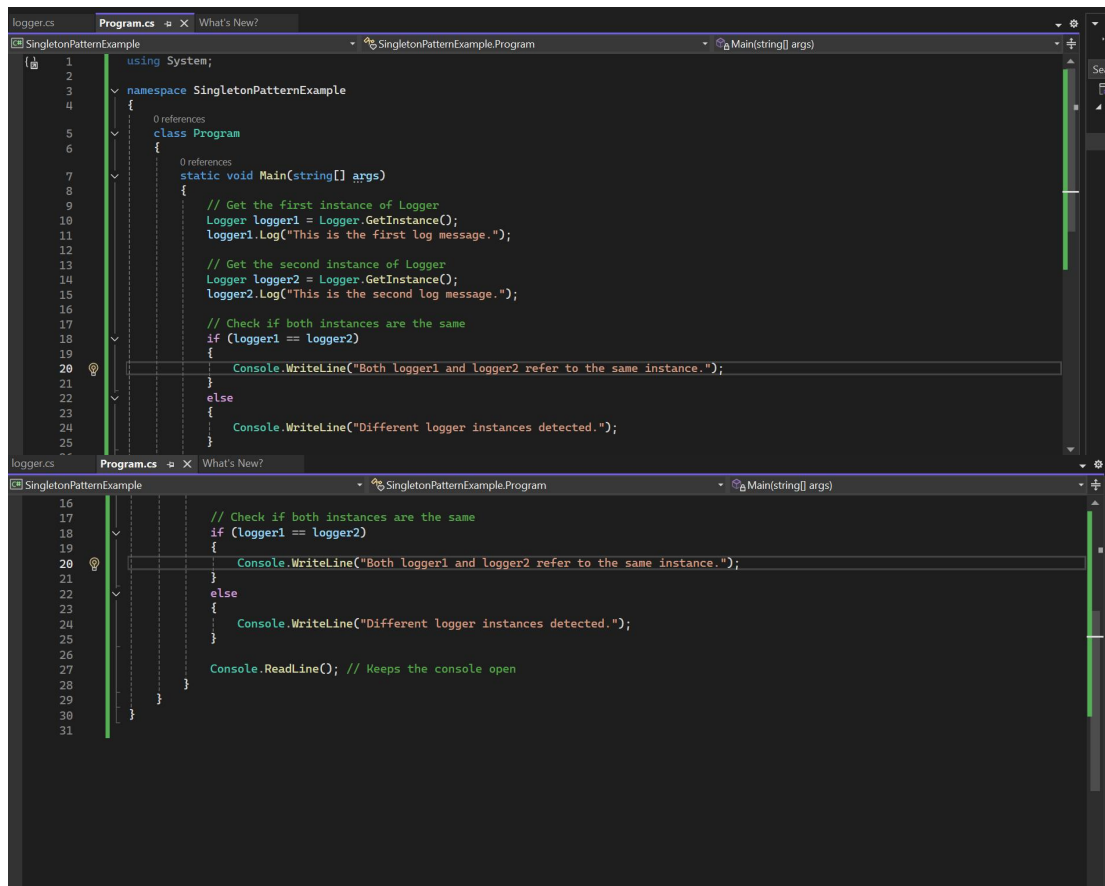


```
1 using System;
2
3 namespace SingletonPatternExample
4 {
5     8 references
6     public class Logger
7     {
8         private static Logger instance;
9         private static readonly object lockObj = new object(); // lock object
10
11     1 reference
12     private Logger()
13     {
14         Console.WriteLine("Logger initialized...");
15     }
16
17     2 references
18     public static Logger GetInstance()
19     {
20         // Double-check locking
21         if (instance == null)
22         {
23             lock (lockObj)
24             {
25                 if (instance == null)
26                 {
27                     instance = new Logger();
28                 }
29             }
30         }
31         return instance;
32     }
33 }
34
35 2 references
36 public void Log(string message)
37 {
38     Console.WriteLine("Log: " + message);
39 }
```



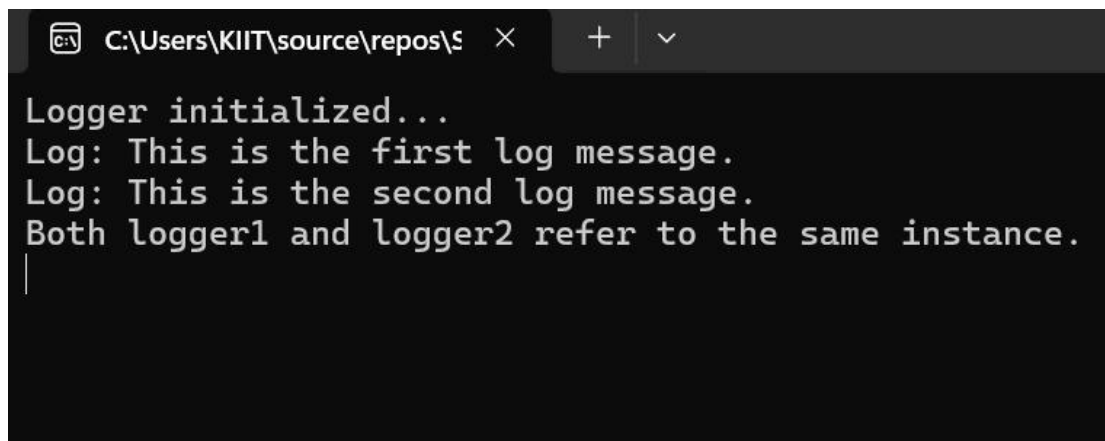
```
17 // Double-check locking
18 if (instance == null)
19 {
20     lock (lockObj)
21     {
22         if (instance == null)
23         {
24             instance = new Logger();
25         }
26     }
27 }
28 return instance;
29
30
31 2 references
32 public void Log(string message)
33 {
34     Console.WriteLine("Log: " + message);
35 }
36
37 }
```

Program.cs



```
1 using System;
2
3 namespace SingletonPatternExample
4 {
5     0 references
6     class Program
7     {
8         0 references
9         static void Main(string[] args)
10         {
11             // Get the first instance of Logger
12             Logger logger1 = Logger.GetInstance();
13             logger1.Log("This is the first log message.");
14
15             // Get the second instance of Logger
16             Logger logger2 = Logger.GetInstance();
17             logger2.Log("This is the second log message.");
18
19             // Check if both instances are the same
20             if (logger1 == logger2)
21             {
22                 Console.WriteLine("Both logger1 and logger2 refer to the same instance.");
23             }
24             else
25             {
26                 Console.WriteLine("Different logger instances detected.");
27             }
28         }
29     }
30
31     16
32     // Check if both instances are the same
33     if (logger1 == logger2)
34     {
35         Console.WriteLine("Both logger1 and logger2 refer to the same instance.");
36     }
37     else
38     {
39         Console.WriteLine("Different logger instances detected.");
40     }
41
42     Console.ReadLine(); // Keeps the console open
43 }
```

Output:



```
C:\Users\KIIT\source\repos\S X + v
Logger initialized...
Log: This is the first log message.
Log: This is the second log message.
Both logger1 and logger2 refer to the same instance.
|
```