# 2. Define a Singleton Class

Create a Logger class that follows the Singleton pattern:  
- It has a private static instance of itself.  
- Its constructor is private.  
- A public static method GetInstance() is provided to access the single instance.

# 3. Implementation

using System;  
  
public class Logger  
{  
 private static Logger \_instance;  
 private static readonly object \_lock = new object();  
  
 private Logger()  
 {  
 Console.WriteLine("Logger instance created.");  
 }  
  
 public static Logger GetInstance()  
 {  
 if (\_instance == null)  
 {  
 lock (\_lock)  
 {  
 if (\_instance == null)  
 {  
 \_instance = new Logger();  
 }  
 }  
 }  
 return \_instance;  
 }  
  
 public void Log(string message)  
 {  
 Console.WriteLine($"[Log]: {message}");  
 }  
}  
  
class Program  
{  
 static void Main(string[] args)  
 {  
 Console.WriteLine("Testing Singleton Logger...");  
  
 Logger logger1 = Logger.GetInstance();  
 logger1.Log("This is the first log message.");  
  
 Logger logger2 = Logger.GetInstance();  
 logger2.Log("This is the second log message.");  
  
 if (object.ReferenceEquals(logger1, logger2))  
 {  
 Console.WriteLine("\n Both logger instances are the same. Singleton works!");  
 }  
 else  
 {  
 Console.WriteLine("\n Logger instances are different. Singleton failed.");  
 }  
 }  
}

# 4. Test the Singleton Implementation

In the Main method, we create two Logger instances and use them to log messages. We also check if both instances refer to the same object using ReferenceEquals(). If they do, the Singleton pattern is working correctly.

Output:

