**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.

**Code:**

public sealed class Logger

{

private static Logger? \_instance;

private Logger()

{

Console.WriteLine("Private constructor created!");

}

public static Logger Instance

{

get

{

if (\_instance == null)

{

\_instance = new Logger();

}

return \_instance;

}

}

public void Log(string message)

{

Console.WriteLine($"Log entry: {message}");

}

}

class Program

{

static void Main()

{

Logger logger1 = Logger.Instance;

logger1.Log("First message");

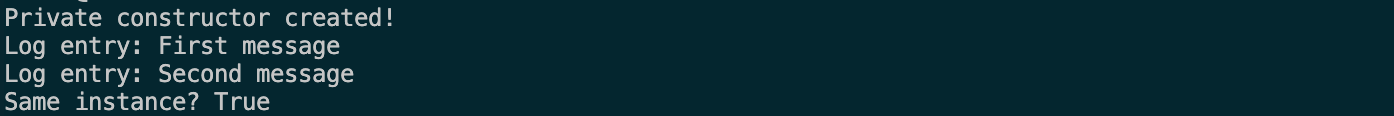
Logger logger2 = Logger.Instance;

logger2.Log("Second message");

Console.WriteLine($"Same instance? {object.ReferenceEquals(logger1, logger2)}");

}

}

**Output:**