Exercise 1: Implementing the Singleton Pattern:

using System;

public class Singleton

{

    private static Singleton? \_instance; // Declare as nullable

    private static readonly object \_lock = new object();

    // Private constructor to prevent instantiation

    private Singleton()

    {

        // Initialization code here

        Console.WriteLine("A new Singleton instance has been created!");

    }

    public static Singleton Instance

    {

        get

        {

            // Double-check locking

            if (\_instance == null)

            {

                lock (\_lock)

                {

                    if (\_instance == null)

                    {

                        \_instance = new Singleton();

                    }

                }

            }

            return \_instance!;

        }

    }

    public void SomeBusinessLogic()

    {

        Console.WriteLine("The Singleton is now executing some important business logic.");

    }

}

class Program

{

    static void Main(string[] args)

    {

        Console.WriteLine("Attempting to access the Singleton instance...");

        // Access the Singleton instance

        Singleton singleton1 = Singleton.Instance;

        singleton1.SomeBusinessLogic();

        // Access the Singleton instance again

        Console.WriteLine("Trying to access the Singleton instance again...");

        Singleton singleton2 = Singleton.Instance;

        singleton2.SomeBusinessLogic();

        // Check if both instances are the same

        Console.WriteLine($"Are both instances the same? {singleton1 == singleton2}");

    }

}

OUTPUT:

