**Implementing the Singleton Pattern**

**CODE  
  
Program.cs**

namespace SingletonPattern

{

    public sealed class Logger

    {

        private static Logger? \_instance = null;

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

        private Logger()

        {

            Console.WriteLine("Logger initialized.");

        }

        public static Logger Instance

        {

            get

            {

                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()

        {

            Logger logger1 = Logger.Instance;

            logger1.Log("First log message");

            Logger logger2 = Logger.Instance;

            logger2.Log("Second log message");

            if (ReferenceEquals(logger1, logger2))

            {

                Console.WriteLine("Logger is a singleton. Both instances are the same.");

            }

            else

            {

                Console.WriteLine("Logger is NOT a singleton. Instances are different.");

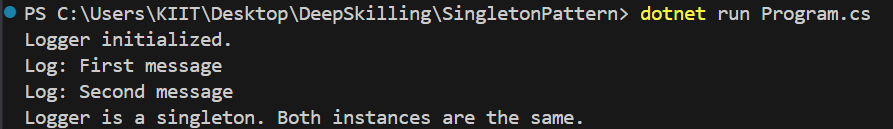
            }

        }

    }

}

**Output**



**Implementing the Factory Method Pattern**

**CODE**

using System;

namespace FactoryMethodPatternExample

{

    // Step 1: Document Interface

    public interface IDocument

    {

        void Open();

}

    // Step 2: Concrete Document Classes

    public class WordDocument : IDocument

    {

        public void Open()

        {

            Console.WriteLine("Opening Word document.");

        }

}

    public class PdfDocument : IDocument

    {

        public void Open()

        {

            Console.WriteLine("Opening PDF document.");

        }

}

    public class ExcelDocument : IDocument

    {

        public void Open()

        {

            Console.WriteLine("Opening Excel document.");

        }

}

    // Step 3: Abstract Factory

    public abstract class DocumentFactory

    {

        public abstract IDocument CreateDocument();

}

    // Step 4: Concrete Factories

    public class WordDocumentFactory : DocumentFactory

    {

        public override IDocument CreateDocument()

        {

            return new WordDocument();

        }

}

    public class PdfDocumentFactory : DocumentFactory

    {

        public override IDocument CreateDocument()

        {

            return new PdfDocument();

        }

}

    public class ExcelDocumentFactory : DocumentFactory

    {

        public override IDocument CreateDocument()

        {

            return new ExcelDocument();

        }

    }

    // Step 5: Main Program to test

    class Program

    {

        static void Main()

        {

            DocumentFactory wordFactory = new WordDocumentFactory();

            IDocument wordDoc = wordFactory.CreateDocument();

            wordDoc.Open();

            DocumentFactory pdfFactory = new PdfDocumentFactory();

            IDocument pdfDoc = pdfFactory.CreateDocument();

            pdfDoc.Open();

            DocumentFactory excelFactory = new ExcelDocumentFactory();

            IDocument excelDoc = excelFactory.CreateDocument();

            excelDoc.Open();

        }

    }

}

**OUTPUT**

