Name-Sangram Mandal

Superset Id-6363848

Week-1(Handson-Exercise)

1. **Design principles & Patterns**

**Exercise 1: Implementing the Singleton Pattern**

Code:-

In Logger.cs :-

using System;

namespace SingletonPattern

{

    class Logger

    {

        private static Logger instance;

        private static readonly object lockObj = new object();

        private Logger() { }

        public static Logger GetInstance()

        {

            if (instance == null)

            {

                lock (lockObj)

                {

                    if (instance == null)

                    {

                        instance = new Logger();

                    }

                }

            }

            return instance;

        }

          public void Log(string message)

        {

            Console.WriteLine($"[LOG] {message}");

        }

    }

}

In Program.cs :-

using System;

namespace SingletonPattern

{

    class Program

    {

        static void Main(string[] args)

        {

            Logger firstLogger = Logger.GetInstance();

            Logger secondLogger = Logger.GetInstance();

            firstLogger.Log("Initialized logging mechanism.");

            secondLogger.Log("Performed another logging attempt.");

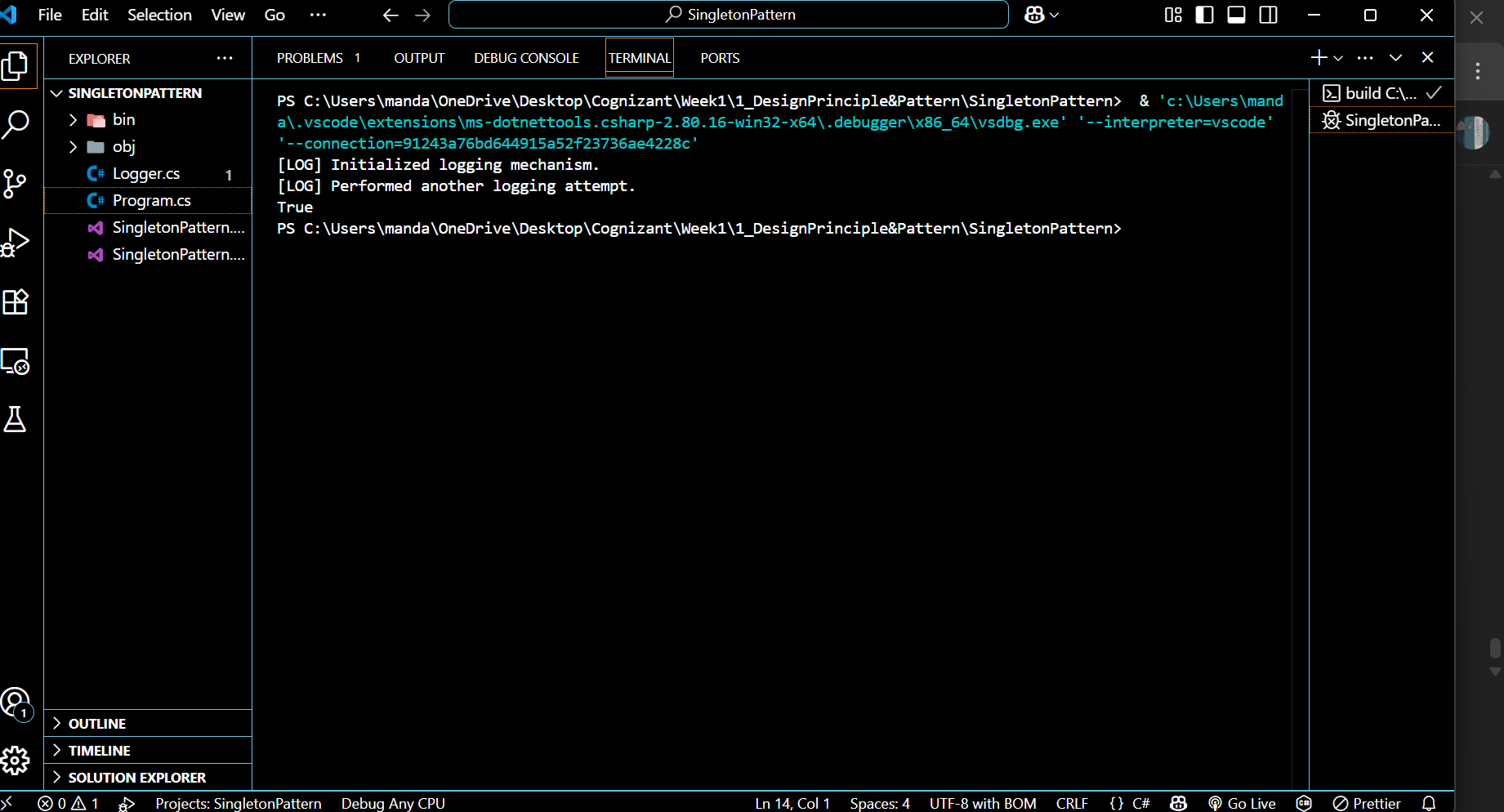
            Console.WriteLine(object.ReferenceEquals(firstLogger, secondLogger));

        }

    }

}

Output :-



**Exercise 2: Implementing the Factory Method Pattern**

**Code:-**

IDocument.cs :-

namespace FactoryMethodPatternExample

{

    public interface IDocument

    {

        void Open();

    }

}

DocumentFactory.cs :-

namespace FactoryMethodPatternExample

{

    public abstract class DocumentFactory

    {

        public abstract IDocument CreateDocument();

    }

}

WordDocument.cs :-

using System;

namespace FactoryMethodPatternExample

{

    public class WordDocument : IDocument

    {

        public void Open()

        {

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

        }

    }

}

PdfDocument.cs :-

using System;

namespace FactoryMethodPatternExample

{

    public class WordDocument : IDocument

    {

        public void Open()

        {

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

        }

}

}

ExcelDocument.cs :-

using System;

namespace FactoryMethodPatternExample

{

    public class ExcelDocument : IDocument

    {

        public void Open()

        {

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

        }

    }

}

WordDocumentFactory.cs :-

namespace FactoryMethodPatternExample

{

    public class WordDocumentFactory : DocumentFactory

    {

        public override IDocument CreateDocument()

        {

            return new WordDocument();

        }

    }

}

PdfDocumentFactory.cs :-

namespace FactoryMethodPatternExample

{

    public class PdfDocumentFactory : DocumentFactory

    {

        public override IDocument CreateDocument()

        {

            return new PdfDocument();

        }

    }

}

ExcelDocumentFactory.cs :-

namespace FactoryMethodPatternExample

{

    public class ExcelDocumentFactory : DocumentFactory

    {

        public override IDocument CreateDocument()

        {

            return new ExcelDocument();

        }

    }

}

Program.cs :-

using System;

namespace FactoryMethodPatternExample

{

    class Program

    {

        static void Main(string[] args)

        {

            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 :-

