Week1: Design Principles and Patterns

1. **Exercise 1**

Code:

using System;

namespace SingletonPatternExample

{

    public class Logger

    {

        private static Logger instance;

        private static readonly object lockObj = new object();

        private Logger()

        {

            Console.WriteLine("Logger instance created.");

        }

        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);

        }

    }

}

using System;

namespace SingletonPatternExample

{

    class Program

    {

        static void Main(string[] args)

        {

            Logger logger1 = Logger.GetInstance();

            Logger logger2 = Logger.GetInstance();

            logger1.Log("First log message.");

            logger2.Log("Second log message.");

            if (logger1 == logger2)

                Console.WriteLine("Both logger1 and logger2 refer to the same instance.");

            else

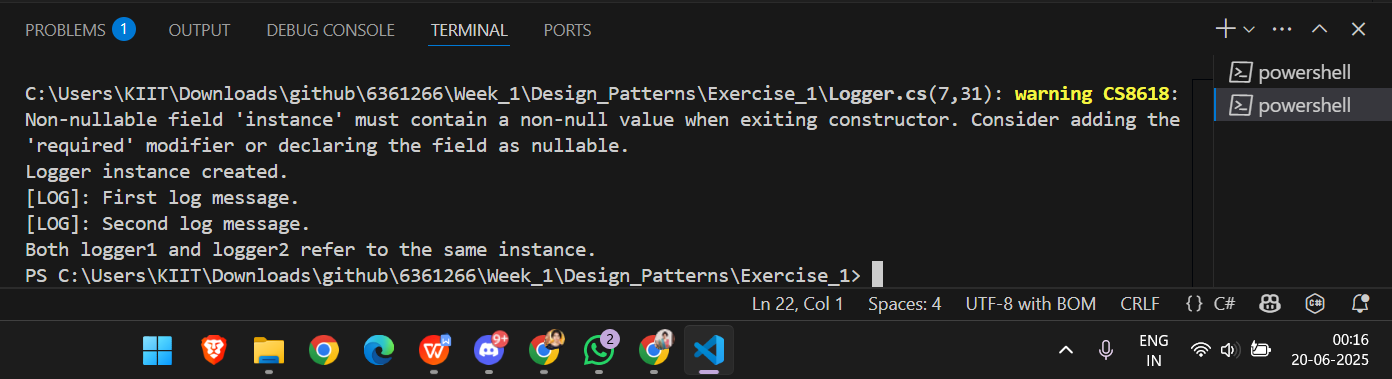
                Console.WriteLine("Different instances exist (not a singleton).");

        }

    }

}

Output:



1. **Exercise 2**

Code:

namespace FactoryMethodPatternExample

{

    public abstract class DocumentFactory

    {

        public abstract IDocument CreateDocument();

    }

}

using System;

namespace FactoryMethodPatternExample

{

    public class ExcelDocument : IDocument

    {

        public void Open()

        {

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

        }

    }

}

namespace FactoryMethodPatternExample

{

    public class ExcelDocumentFactory : DocumentFactory

    {

        public override IDocument CreateDocument()

        {

            return new ExcelDocument();

        }

    }

}

namespace FactoryMethodPatternExample

{

    public interface IDocument

    {

        void Open();

    }

}

using System;

namespace FactoryMethodPatternExample

{

    public class PdfDocument : IDocument

    {

        public void Open()

        {

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

        }

    }

}

namespace FactoryMethodPatternExample

{

    public class PdfDocumentFactory : DocumentFactory

    {

        public override IDocument CreateDocument()

        {

            return new PdfDocument();

        }

    }

}

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();

        }

    }

}

using System;

namespace FactoryMethodPatternExample

{

    public class WordDocument : IDocument

    {

        public void Open()

        {

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

        }

    }

}

namespace FactoryMethodPatternExample

{

    public class WordDocumentFactory : DocumentFactory

    {

        public override IDocument CreateDocument()

        {

            return new WordDocument();

        }

    }

}

Output:

