**Exercise 1: Implementing the Singleton Pattern**

**PROGRAM:**

**using System;**

**using System.Threading.Tasks;**

**namespace SingletonPatternExample**

**{**

**public sealed class Logger**

**{**

**private static readonly Lazy<Logger> \_instance =**

**new Lazy<Logger>(() => new Logger());**

**private Logger()**

**{**

**Console.WriteLine("Logger is up and running!");**

**}**

**public static Logger Instance => \_instance.Value;**

**public void Log(string message)**

**{**

**Console.WriteLine($"[{DateTime.Now:HH:mm:ss}] {message}");**

**}**

**}**

**class Program**

**{**

**static void Main()**

**{**

**Logger.Instance.Log("Hey there!");**

**Logger.Instance.Log("Logging still works.");**

**// Quick test: are both instances the same?**

**Console.WriteLine(**

**ReferenceEquals(Logger.Instance, Logger.Instance)**

**? "✅ Same Logger instance"**

**: "❌ Different instances?!");**

**Parallel.For(0, 4, i =>**

**{**

**Logger.Instance.Log($"Parallel message {i}");**

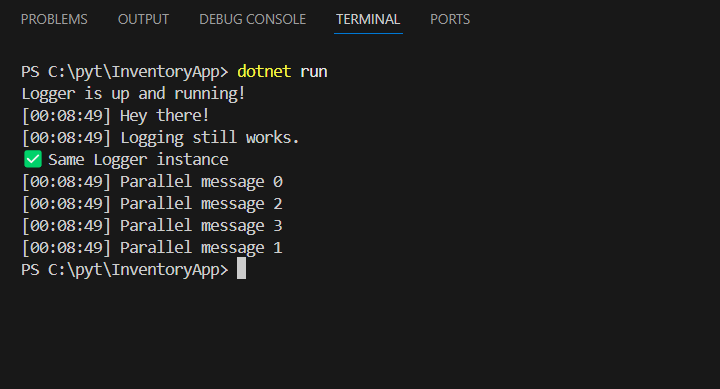
**});**

**}**

**}**

**}**

**OUTPUT:**

****

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

**PROGRAM:**

**using System;**

**namespace FactoryMethodPatternExample**

**{**

**// Define the common interface for all documents**

**public interface IDocument**

**{**

**void Open(); // Every document type needs an Open() method**

**}**

**// Concrete implementations of documents**

**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...");**

**}**

**}**

**// Abstract factory – knows how to create documents**

**public abstract class DocumentFactory**

**{**

**public abstract IDocument CreateDocument();**

**}**

**// Concrete factories for each document type**

**public class WordFactory : DocumentFactory**

**{**

**public override IDocument CreateDocument()**

**{**

**return new WordDocument();**

**}**

**}**

**public class PdfFactory : DocumentFactory**

**{**

**public override IDocument CreateDocument()**

**{**

**return new PdfDocument();**

**}**

**}**

**public class ExcelFactory : DocumentFactory**

**{**

**public override IDocument CreateDocument()**

**{**

**return new ExcelDocument();**

**}**

**}**

**class Program**

**{**

**static void Main()**

**{**

**Console.WriteLine("Document Factory Demo");**

**Console.WriteLine("---------------------\n");**

**// Use the factories to create and open documents**

**UseFactory(new WordFactory());**

**UseFactory(new PdfFactory());**

**UseFactory(new ExcelFactory());**

**}**

**// A helper method to reduce duplicate code**

**static void UseFactory(DocumentFactory factory)**

**{**

**IDocument doc = factory.CreateDocument();**

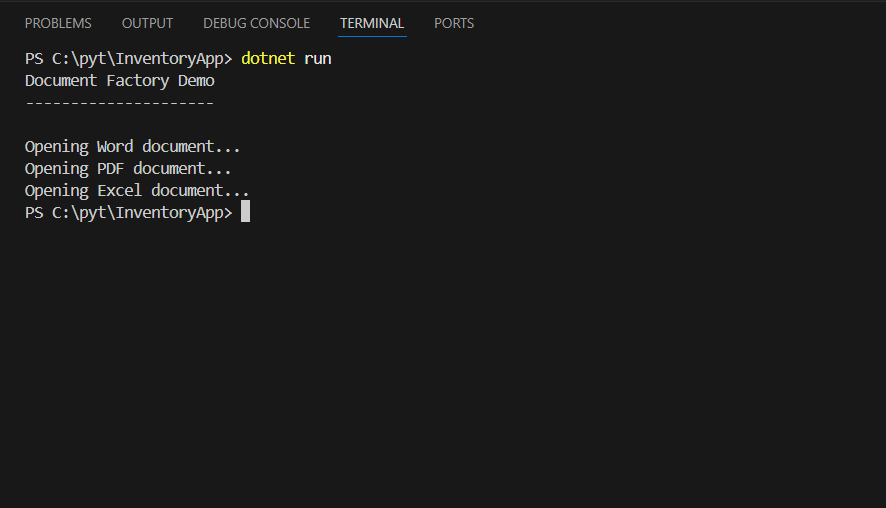
**doc.Open();**

**}**

**}**

**}**

**OUTPUT:**

****