Design Pattern and principles

Ex.1 Implementing the singleton pattern.(in C# dotnet)

Solution: Below pictures are the implementation of singleton pattern that is done in the VS code.

>**Logger.cs**

public sealed class Logger

{

    private static readonly Logger \_instance = new Logger();

    private Logger()

    {

        Console.WriteLine("Logger instance created");

    }

    public static Logger Instance => \_instance;

    public void Log(string message)

    {

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

    }

}

>**Program.cs**

class Program

{

    static void Main(string[] args)

    {

        Logger logger1 = Logger.Instance;

        logger1.Log("First log message");

        Logger logger2 = Logger.Instance;

        logger2.Log("Second log message");

        Console.WriteLine($"Same instance? {ReferenceEquals(logger1, logger2)}");

    }

}

>**SingletonPatternExample.csproj**

<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>

    <OutputType>Exe</OutputType>

    <TargetFramework>net9.0</TargetFramework>

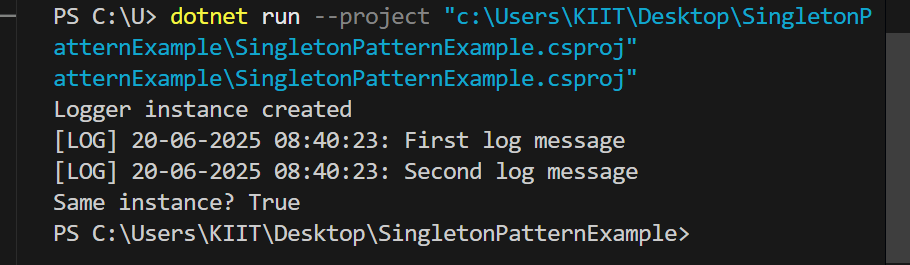
    <ImplicitUsings>enable</ImplicitUsings>

    <Nullable>enable</Nullable>

  </PropertyGroup>

</Project>

**Output:**



**Ex.2 Implementing the Factory method pattern:**

**Solution:**

>**Program.cs**

using System;

class Program

{

    static void Main()

    {

        DocumentFactory factory;

        // Word Document

        factory = new WordDocumentFactory();

        IDocument wordDoc = factory.CreateDocument();

        wordDoc.Open();

        // PDF Document

        factory = new PdfDocumentFactory();

        IDocument pdfDoc = factory.CreateDocument();

        pdfDoc.Open();

        // Excel Document

        factory = new ExcelDocumentFactory();

        IDocument excelDoc = factory.CreateDocument();

        excelDoc.Open();

    }

}

>**DocumentFactory.cs**

public abstract class DocumentFactory

{

    public abstract IDocument CreateDocument();

}

>**IDocument.cs**

public interface IDocument

{

    void Open();

}

>**WordDocument.cs**

public class WordDocument : IDocument

{

    public void Open()

    {

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

    }

}

**>WordDocumentFactory.cs**

public class WordDocumentFactory : DocumentFactory

{

    public override IDocument CreateDocument()

    {

        return new WordDocument();

    }

}

**>PdfDocument.cs**

public class PdfDocument : IDocument

{

    public void Open()

    {

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

    }

}

>PdfDocumentFactory.cs

// PdfDocumentFactory.cs

public class PdfDocumentFactory : DocumentFactory

{

    public override IDocument CreateDocument()

    {

        return new PdfDocument();

    }

}

**>ExcelDocument.cs**

**// ExcelDocument.cs**

**public class ExcelDocument : IDocument**

**{**

**public void Open()**

**{**

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

**}**

**}**

**>ExcelDocumentFactory.cs**

// ExcelDocumentFactory.cs

public class ExcelDocumentFactory : DocumentFactory

{

    public override IDocument CreateDocument()

    {

        return new ExcelDocument();

    }

}

**>factorymethodpattern.csproj**

**<Project Sdk="Microsoft.NET.Sdk">**

**<PropertyGroup>**

**<OutputType>Exe</OutputType>**

**<TargetFramework>net9.0</TargetFramework>**

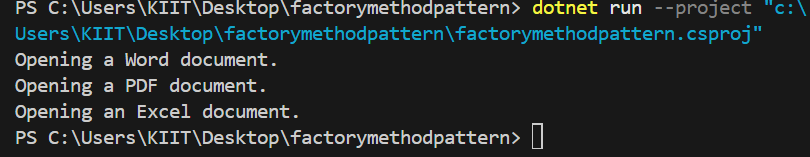
**<ImplicitUsings>enable</ImplicitUsings>**

**<Nullable>enable</Nullable>**

**</PropertyGroup>**

**</Project>**

**Output of EX.2**

****