**Exercise 1: Implementing the Singleton Pattern**

using System;

public sealed class Logger

{

    private static readonly Logger instance = new Logger();

    // Private constructor

    private Logger()

    {

    }

    public static Logger Instance

    {

        get { return instance; }

    }

    public void Log(string message)

    {

        Console.WriteLine($"Log: {message}");

    }

}

public class SingletonTest

{

    public static void Main(string[] args)

    {

        Logger logger1 = Logger.Instance;

        Logger logger2 = Logger.Instance;

        if (object.ReferenceEquals(logger1, logger2))

        {

            Console.WriteLine("Logger is a singleton. Both variables hold the same instance.");

        }

        else

        {

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

        }

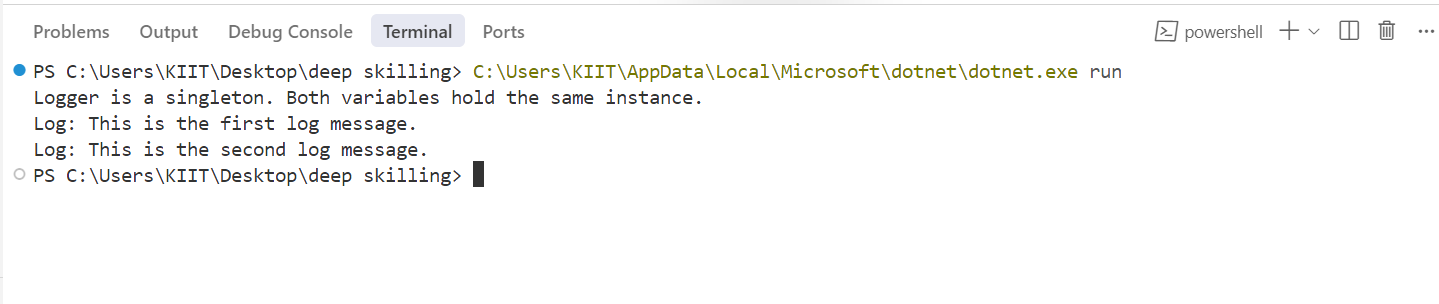
        logger1.Log("This is the first log message.");

        logger2.Log("This is the second log message.");

    }

}

**OUTPUT**

****

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

using System;

public interface IDocument

{

    void Open();

    void Close();

    void Save();

}

public class WordDocument : IDocument

{

    public void Open() => Console.WriteLine("Opening Word document...");

    public void Close() => Console.WriteLine("Closing Word document...");

    public void Save() => Console.WriteLine("Saving Word document...");

}

public class PdfDocument : IDocument

{

    public void Open() => Console.WriteLine("Opening PDF document...");

    public void Close() => Console.WriteLine("Closing PDF document...");

    public void Save() => Console.WriteLine("Saving PDF document...");

}

public class ExcelDocument : IDocument

{

    public void Open() => Console.WriteLine("Opening Excel document...");

    public void Close() => Console.WriteLine("Closing Excel document...");

    public void Save() => Console.WriteLine("Saving Excel document...");

}

public abstract class DocumentFactory

{

    public abstract IDocument CreateDocument();

}

// 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();

    }

}

public class FactoryMethodTest

{

    public static void Main(string[] args)

    {

        Console.WriteLine("Creating document from Word Factory:");

        DocumentFactory wordFactory = new WordDocumentFactory();

        IDocument wordDoc = wordFactory.CreateDocument();

        wordDoc.Open();

        wordDoc.Save();

        wordDoc.Close();

        Console.WriteLine("Creating document from PDF Factory:");

        DocumentFactory pdfFactory = new PdfDocumentFactory();

        IDocument pdfDoc = pdfFactory.CreateDocument();

        pdfDoc.Open();

        pdfDoc.Save();

        pdfDoc.Close();

        Console.WriteLine("Creating document from Excel Factory:");

        DocumentFactory excelFactory = new ExcelDocumentFactory();

        IDocument excelDoc = excelFactory.CreateDocument();

        excelDoc.Open();

        excelDoc.Save();

        excelDoc.Close();

    }

}

**OUTPUT**

