Design Patterns and Principles:

Exercise 1: Implementing the Singleton Pattern

Code:

**//File: Logger.cs**

using System;

public class Logger

{

    private static Logger onlyLoggerObject = null;

    private Logger()

    {

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

    }

    public static Logger GetLogger()

    {

        if (onlyLoggerObject == null)

        {

            onlyLoggerObject = new Logger();

        }

        return onlyLoggerObject;

    }

        public void Log(string message)

    {

        Console.WriteLine("Log message: " + message);

    }

}

**// File: Program.cs**

using System;

class Program

{

    static void Main(string[] args)

    {

        Logger logger1 = Logger.GetLogger();

        logger1.Log("First log line");

        Logger logger2 = Logger.GetLogger();

        logger2.Log("Second log line");

        if (logger1 == logger2){

            Console.WriteLine("Both loggers are same. Singleton is working.");

        }

        else{

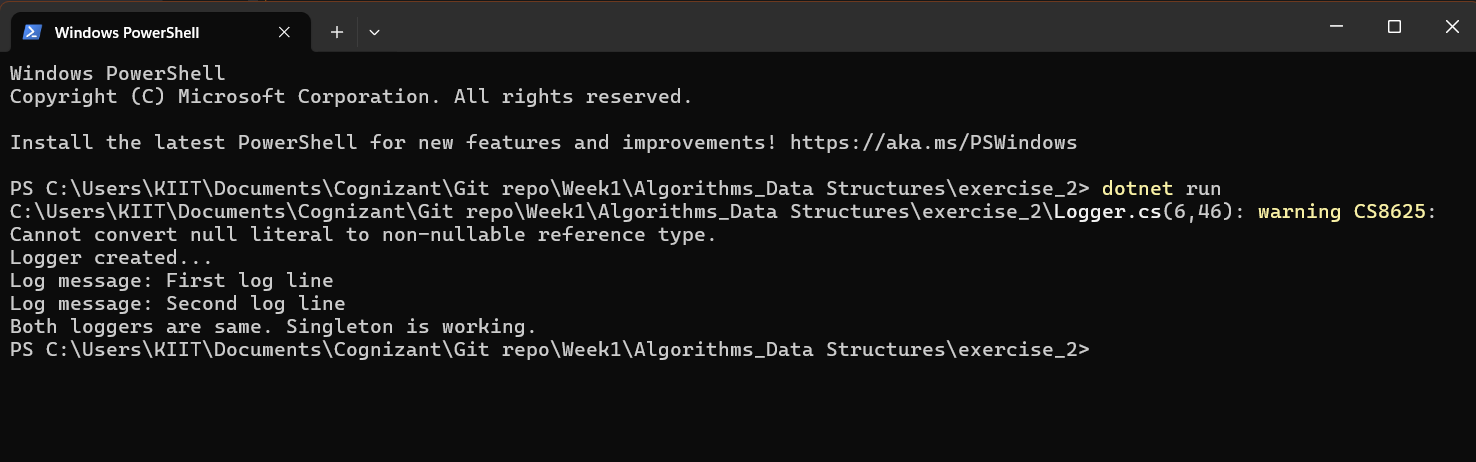
            Console.WriteLine("Loggers are different. Something is wrong.");

        }

    }

}

Output:



Exercise 2: Implementing the Factory Method Pattern

Code:

1. **Interface**

**//File: Document.cs**

public interface Document

{

void Open();

}

**2.Concrete Document classes**

**//File: WordDocument.cs**

using System;

public class WordDocument : Document

{

public void Open()

{

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

}

}

**//File: PdfDocument.cs**

using System;

public class PdfDocument : Document

{

public void Open()

{

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

}

}

**//File: ExcelDocument.cs**

using System;

public class ExcelDocument : Document

{

public void Open()

{

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

}

}

**3.Abstract Factory class**

**//File: DocumentFactory.cs**

public abstract class DocumentFactory

{

public abstract Document CreateDocument();

}

**4.Concrete Factories**

**//File: WordFactory.cs**

public class WordFactory : DocumentFactory

{

public override Document CreateDocument()

{

return new WordDocument();

}

}

**//File: PdfFactory.cs**

public class PdfFactory : DocumentFactory

{

public override Document CreateDocument()

{

return new PdfDocument();

}

}

**//File: ExcelFactory.cs**

public class ExcelFactory : DocumentFactory

{

public override Document CreateDocument()

{

return new ExcelDocument();

}

}

**5.Main Program**

**//File: Program.cs**

using System;

class Program

{

static void Main(string[] args)

{

DocumentFactory wordFactory = new WordFactory();

Document wordDoc = wordFactory.CreateDocument();

wordDoc.Open();

DocumentFactory pdfFactory = new PdfFactory();

Document pdfDoc = pdfFactory.CreateDocument();

pdfDoc.Open();

DocumentFactory excelFactory = new ExcelFactory();

Document excelDoc = excelFactory.CreateDocument();

excelDoc.Open();

}

}

Output:

