Design Pattern and Principles

Exercise 1: Implementing Singleton Pattern

Code:

MyLogger.cs

```
using System;

public class MyLogger
{
    private static MyLogger _instance;
    private static readonly object _lock = new object();

    private MyLogger()
    {
        Console.WriteLine("Logger instance created.");
    }

    public static MyLogger GetInstance()
    {
        if (_instance == null)
        {
            lock (_lock)
            {
                if (_instance == null)
                {
                    _instance = new MyLogger();
                }
            }
        }

        return _instance;
    }

    public void Log(string message)
```

```
    public void Log(string message)
    {
        Console.WriteLine("[LOG]: " + message);
    }
}
```

Program.cs

```
using System;

class Program
{
    static void Main(string[] args)
    {
        MyLogger logger1 = MyLogger.GetInstance();
        logger1.Log("Application started.");

        MyLogger logger2 = MyLogger.GetInstance();
        logger2.Log("Another message logged.");

        if (logger1 == logger2)
        {
            Console.WriteLine("logger1 and logger2 are the same instance.");
        }
        else
        {
            Console.WriteLine("Different instances! Singleton failed.");
        }
    }
}
```

SingletonPatternExample.csproj

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

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>
```

Output:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powershell
Logger instance created.
[LOG]: Application started.
[LOG]: Another message logged.
logger1 and logger2 are the same instance.
PS C:\Users\KIIT\Desktop\DotNet FSE\Engineering concepts\Design patterns and principles\SingletonPatternExample>

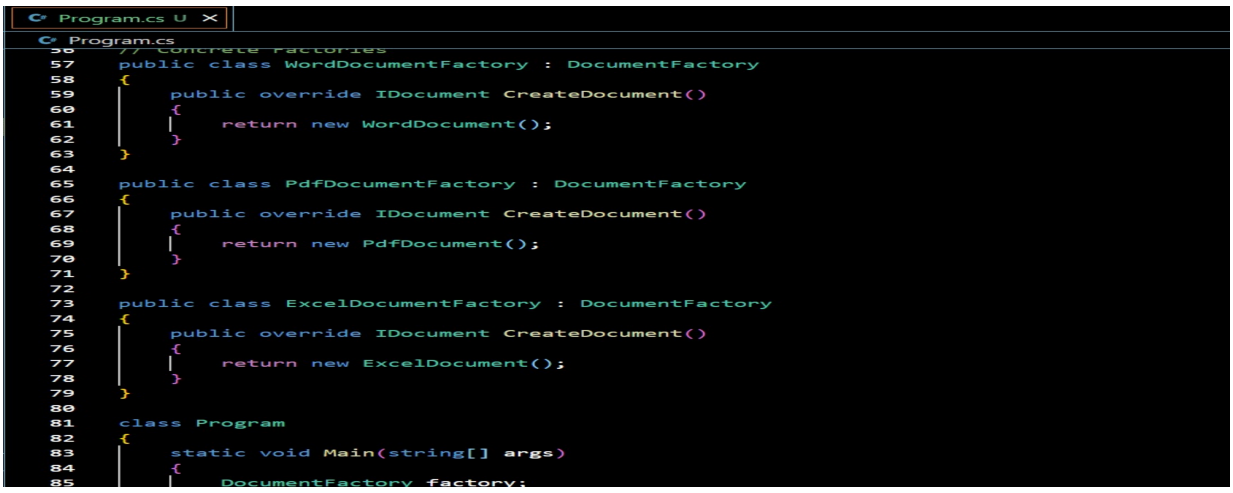
Design Pattern and Principles

Exercise 2: Implementing the Factory Method Pattern

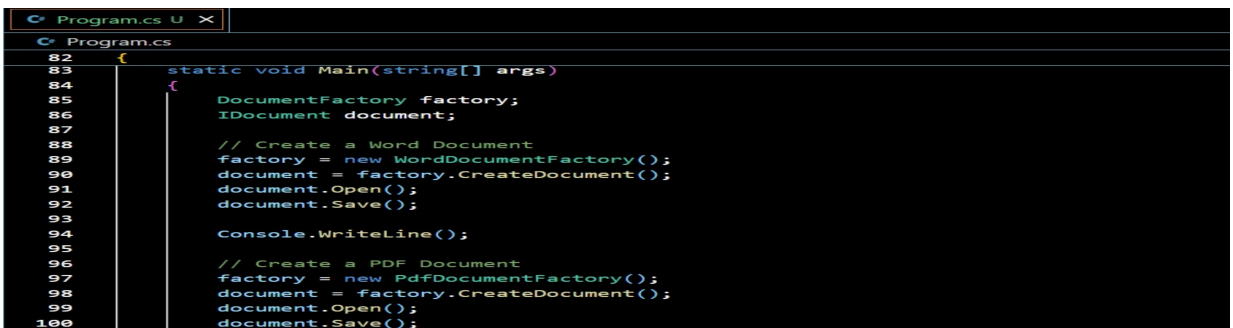
Code:

```
Program.cs U
Program.cs
using System;

// Document interface
public interface IDocument
{
    void Open();
    void Save();
}

// Concrete Documents
public class WordDocument : IDocument
{
    public void Open()
    {
        Console.WriteLine("Opening Word document...");
    }

    public void Save()
    {
        Console.WriteLine("Saving Word document...");
    }
}

public class PdfDocument : IDocument
{
    public void Open()
    {
        Console.WriteLine("Opening PDF document...");
    }
```

```
Program.cs U
Program.cs
    {

    public void Save()
    {
        Console.WriteLine("Saving PDF document...");
    }
}

public class ExcelDocument : IDocument
{
    public void Open()
    {
        Console.WriteLine("Opening Excel document...");
    }

    public void Save()
    {
        Console.WriteLine("Saving Excel document...");
    }
}

// Abstract Factory
public abstract class DocumentFactory
{
    public abstract IDocument CreateDocument();
}

// Concrete Factories
public class WordDocumentFactory : DocumentFactory
```



```
Program.cs U
Program.cs
// 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();
    }
}

class Program
{
    static void Main(string[] args)
    {
        DocumentFactory factory;
```



```
Program.cs U
Program.cs
{
    static void Main(string[] args)
    {
        DocumentFactory factory;
        IDocument document;

        // Create a Word Document
        factory = new WordDocumentFactory();
        document = factory.CreateDocument();
        document.Open();
        document.Save();

        Console.WriteLine();

        // Create a PDF Document
        factory = new PdfDocumentFactory();
        document = factory.CreateDocument();
        document.Open();
        document.Save();
```

```
100             document.Save();
101
102             Console.WriteLine();
103
104             // Create an Excel Document
105             factory = new ExcelDocumentFactory();
106             document = factory.CreateDocument();
107             document.Open();
108             document.Save();
109         }
110     }
```

OUTPUT :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                   powershell

>> nExample>
Opening Word document...
Saving Word document...

Opening PDF document...
Saving PDF document...

Opening Excel document...
Saving Excel document...
PS C:\Users\KIIT\Desktop\DotNet FSE\Engineering concepts\Design patterns and principles\FactoryMethodPatternExample>
```