# Exercise 2: Implementing the Factory Method Pattern (.NET)

## What I Learned

In this task, I implemented the Factory Method Pattern using C# in a .NET environment. This design pattern helped me create different types of documents like Word, PDF, and Excel without tightly coupling the object creation logic. It allowed me to centralize and manage the creation of various document types through a common interface.

## Concepts Covered

- Factory Method Design Pattern

- Abstract classes and interfaces

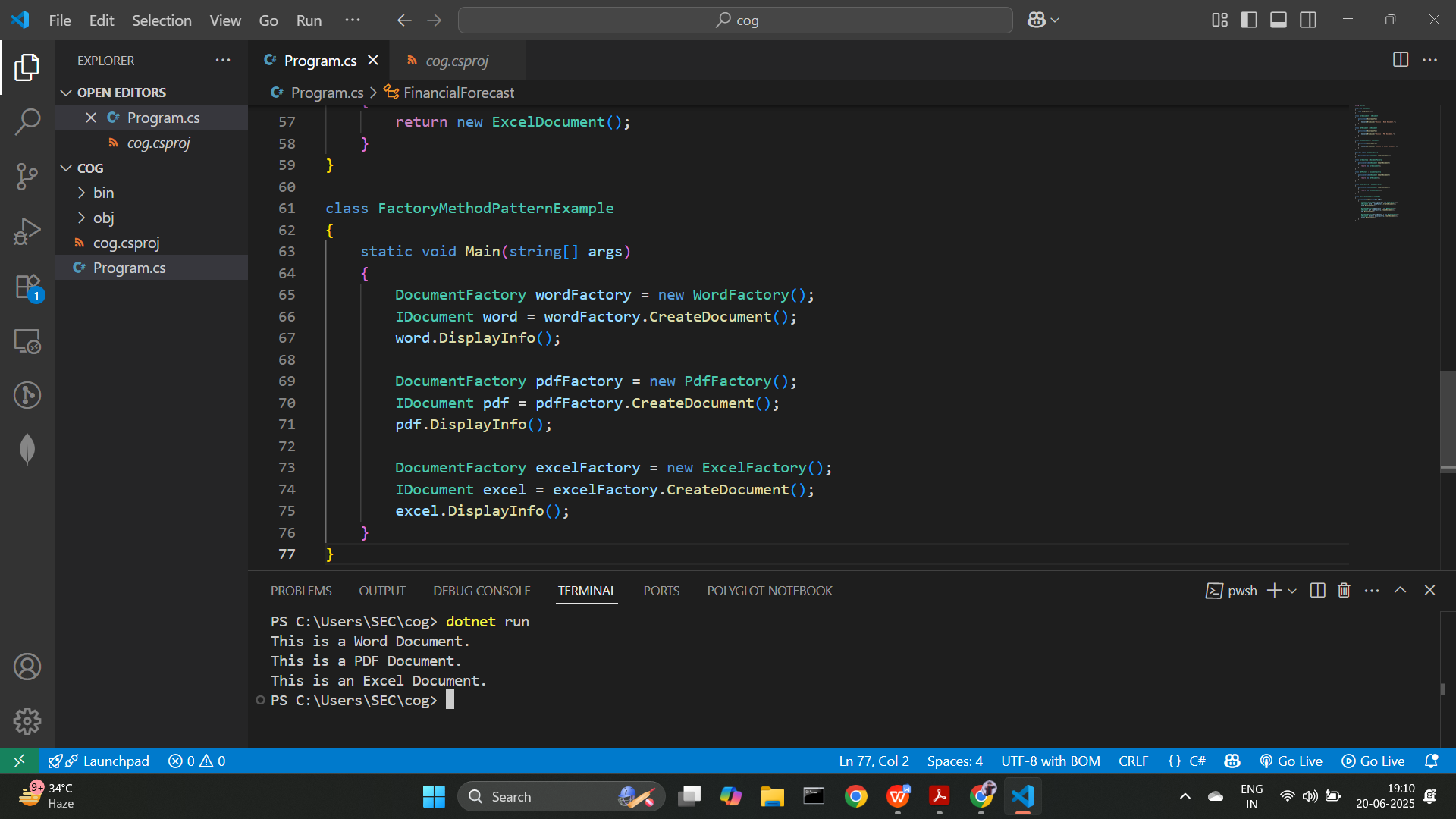
- Concrete implementation and object creation

- Loose coupling and flexibility

## C# Code

using System;  
  
interface IDocument  
{  
 void DisplayInfo();  
}  
  
class WordDocument : IDocument  
{  
 public void DisplayInfo()  
 {  
 Console.WriteLine("This is a Word Document.");  
 }  
}  
  
class PdfDocument : IDocument  
{  
 public void DisplayInfo()  
 {  
 Console.WriteLine("This is a PDF Document.");  
 }  
}  
  
class ExcelDocument : IDocument  
{  
 public void DisplayInfo()  
 {  
 Console.WriteLine("This is an Excel Document.");  
 }  
}  
  
abstract class DocumentFactory  
{  
 public abstract IDocument CreateDocument();  
}  
  
class WordFactory : DocumentFactory  
{  
 public override IDocument CreateDocument()  
 {  
 return new WordDocument();  
 }  
}  
  
class PdfFactory : DocumentFactory  
{  
 public override IDocument CreateDocument()  
 {  
 return new PdfDocument();  
 }  
}  
  
class ExcelFactory : DocumentFactory  
{  
 public override IDocument CreateDocument()  
 {  
 return new ExcelDocument();  
 }  
}  
  
class FactoryMethodPatternExample  
{  
 static void Main(string[] args)  
 {  
 DocumentFactory wordFactory = new WordFactory();  
 IDocument word = wordFactory.CreateDocument();  
 word.DisplayInfo();  
  
 DocumentFactory pdfFactory = new PdfFactory();  
 IDocument pdf = pdfFactory.CreateDocument();  
 pdf.DisplayInfo();  
  
 DocumentFactory excelFactory = new ExcelFactory();  
 IDocument excel = excelFactory.CreateDocument();  
 excel.DisplayInfo();  
 }  
}

## output



## Conclusion

Using the Factory Method Pattern, I was able to create objects without specifying their concrete classes directly. This approach made my code more maintainable and scalable. The different factories handled the creation logic independently, ensuring flexibility.