

1. FILE HANDLING IN C#

1.1 Introduction

File handling is a mechanism in C# that allows developers to perform various operations on files such as creating, reading, writing, appending, copying, and deleting files. The System.IO namespace provides several classes that enable easy and efficient file manipulation.

C# supports both **binary** and **text-based** file operations. File handling is important when data must be stored persistently on disk instead of being held temporarily in memory.

1.2 Important Classes in System.IO Namespace

Class Name	Description
File	Provides static methods to create, copy, delete, move, and open files.
FileInfo	Provides instance methods for file operations. Similar to File but allows working with object-oriented features.
StreamReader	Reads characters from a text file sequentially.
StreamWriter	Writes text data to a file.
BinaryReader	Reads primitive data types as binary values.
BinaryWriter	Writes primitive data types in binary format.
Directory	Provides static methods for creating, moving, and deleting directories.
DirectoryInfo	Similar to Directory, but provides object-based access.
Path	Provides methods to manipulate file or directory path information.

1.3 Basic File Operations

1. Creating a File

To create a new file, use the File.Create() or File.WriteAllText() method.

```
File.WriteAllText("sample.txt", "Welcome to Dhruv Training");
```

2. Reading from a File

The File.ReadAllText() or File.ReadAllLines() methods are used to read data from a file.

```
string content = File.ReadAllText("sample.txt");
```

```
Console.WriteLine(content);
```

3. Appending to a File

To add content to an existing file without overwriting it, use the `File.AppendAllText()` method.

```
File.AppendAllText("sample.txt", "\nThis is an additional line.");
```

4. Deleting a File

To delete a file, use the `File.Delete()` method.

```
File.Delete("sample.txt");
```

1.4 Using StreamReader and StreamWriter

For reading or writing text files line by line, the `StreamReader` and `StreamWriter` classes are used.

Example:

```
using (StreamWriter sw = new StreamWriter("data.txt"))
```

```
{
```

```
    sw.WriteLine("Hello Udaya");
```

```
    sw.WriteLine("Welcome to .NET Training");
```

```
}
```

```
using (StreamReader sr = new StreamReader("data.txt"))
```

```
{
```

```
    string line;
```

```
    while ((line = sr.ReadLine()) != null)
```

```
    {
```

```
        Console.WriteLine(line);
```

```
    }
```

```
}
```

The `using` block ensures that resources are automatically released after the operation.

1.5 Exception Handling in File Operations

File operations often generate exceptions such as file not found, access denied, or path not valid.

Common exceptions include:

- FileNotFoundException
- IOException
- UnauthorizedAccessException
- DirectoryNotFoundException

It is good practice to use **try-catch-finally** blocks to handle such exceptions.

2. LINQ (LANGUAGE INTEGRATED QUERY)

2.1 Introduction

LINQ (Language Integrated Query) is a feature introduced in C# 3.0 that provides a uniform syntax for querying data from different sources like arrays, collections, XML, or databases. It enables developers to write SQL-like queries directly within C# code.

Namespace:

```
using System.Linq;
```

2.2 Advantages of LINQ

1. Unified approach to querying different data sources.
2. Strongly typed – errors are caught at compile time.
3. Readable and concise code.
4. Supports deferred execution.
5. Eliminates the need for complex loops and conditions.

2.3 LINQ Syntax Types

LINQ provides two syntaxes:

1. **Query Syntax**

Similar to SQL:

2. var result = from n in numbers
3. where n > 50
4. orderby n descending
5. select n;

6. **Method Syntax**

Uses lambda expressions and extension methods:

```
7. var result = numbers.Where(n => n > 50).OrderByDescending(n => n);
```

2.4 Common LINQ Methods

Method	Description
Where()	Filters a sequence based on a condition.
Select()	Projects elements into a new form.
OrderBy()	Sorts elements in ascending order.
OrderByDescending()	Sorts elements in descending order.
GroupBy()	Groups elements that share a common attribute.
Sum(), Count(), Average(), Max(), Min()	Performs aggregate calculations.
First(), Last(), Single()	Retrieves specific elements.

2.5 LINQ with Collections Example

Example using a list of integers:

```
List<int> nums = new List<int>() {10, 20, 30, 40, 50};
```

```
var result = from n in nums
```

```
    where n > 30
```

```
    select n;
```

```
foreach (var n in result)
```

```
    Console.WriteLine(n);
```

2.6 LINQ with Objects Example

```
List<Student> students = new List<Student>()
```

```
{
```

```
    new Student(){Name="Udaya", Marks=85},
```

```
    new Student(){Name="Arun", Marks=45},
```

```
    new Student(){Name="Shetty", Marks=75}
```

```
};
```

```
var passed = students.Where(s => s.Marks >= 50)
    .OrderByDescending(s => s.Marks);
```

```
foreach (var s in passed)
    Console.WriteLine($"{s.Name} - {s.Marks}");
```

2.7 LINQ Grouping and Aggregation Example

```
var group = students.GroupBy(s => s.Marks >= 50 ? "Pass" : "Fail");
```

```
foreach (var g in group)
{
    Console.WriteLine(g.Key + ":");
    foreach (var student in g)
        Console.WriteLine("  " + student.Name);
}
```

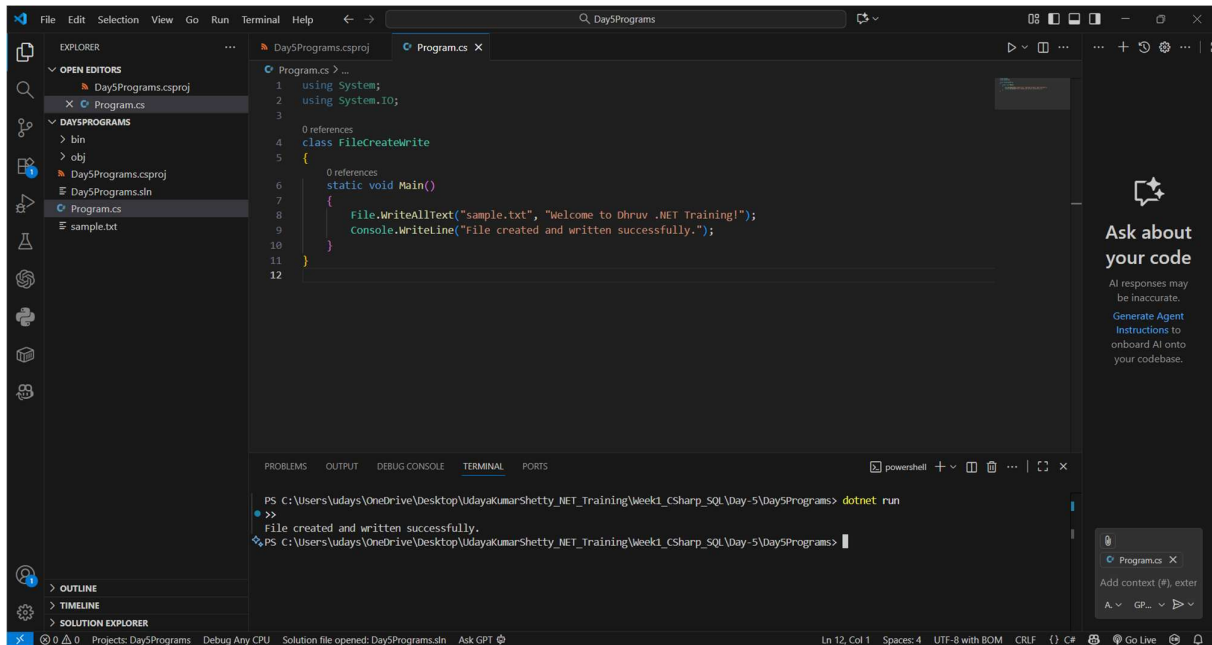
SUMMARY

Concept	Description
File Handling	Reading, writing, and managing files.
StreamReader/Writer	Sequential text file reading/writing.
LINQ	Query syntax integrated with C# for data filtering and sorting.

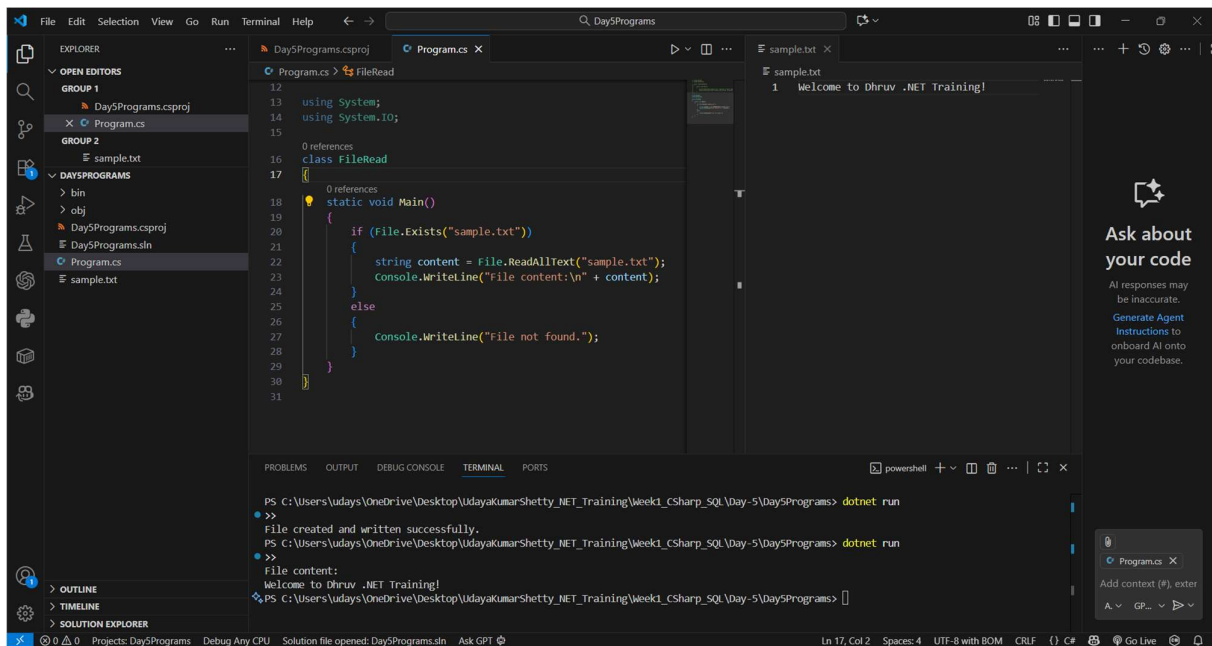
Snapshots:

1. Basic File Handling

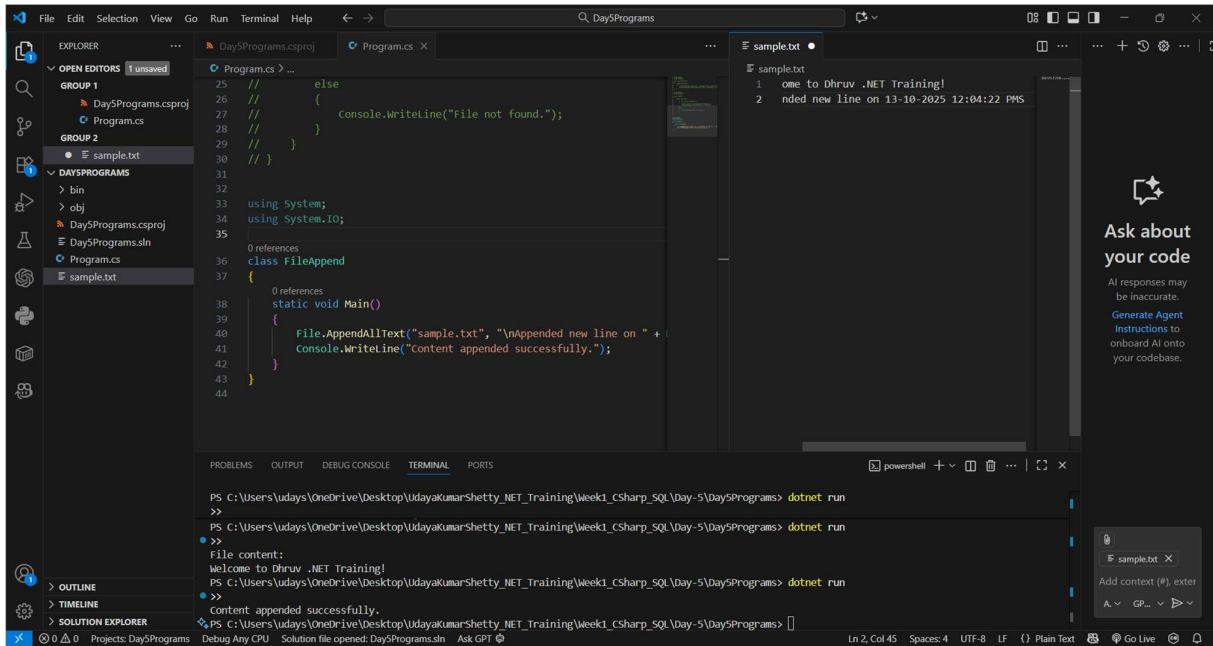
○ Create and Write



○ Read



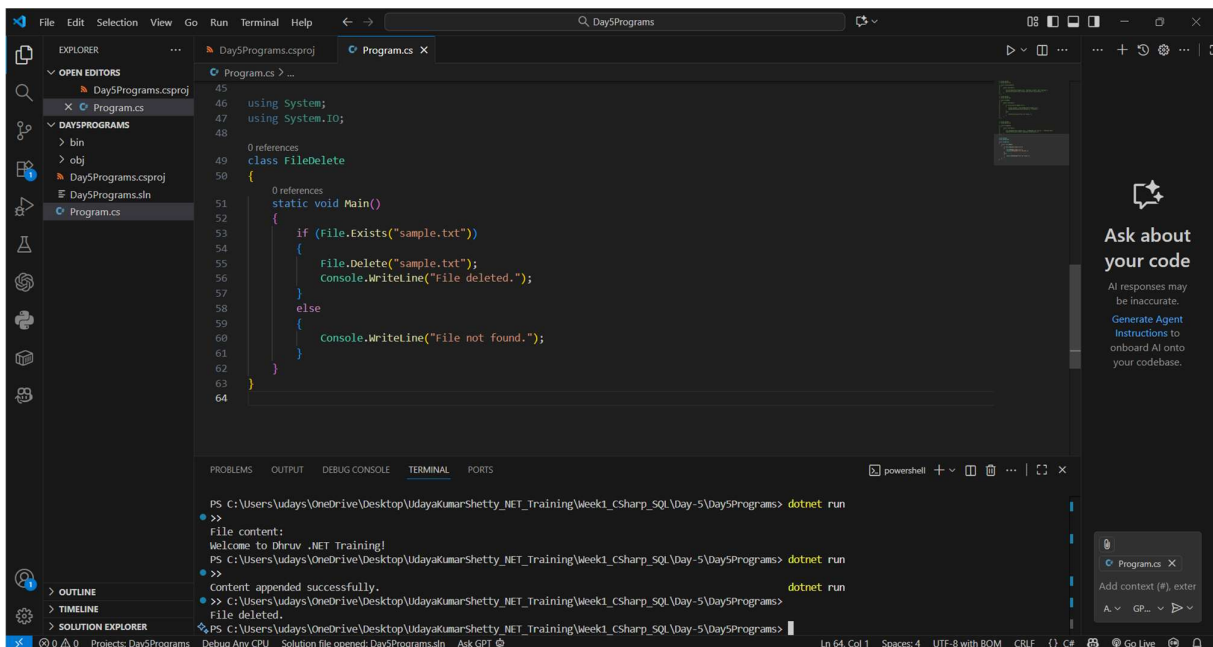
○ Append



```
25 // else
26 // {
27 //     Console.WriteLine("File not found.");
28 // }
29 // }
30 // }
31
32
33 using System;
34 using System.IO;
35
36 0 references
37 class FileAppend
38 {
39     0 references
40     static void Main()
41     {
42         File.AppendAllText("sample.txt", "\nAppended new line on " +
43             Console.WriteLine("content appended successfully.");
44     }
45 }
```

```
PS C:\Users\udays\OneDrive\Desktop\UdayakumarShetty_NET_Training\Week1_CSharp_SQL\Day-5\Day5Programs> dotnet run
>>
PS C:\Users\udays\OneDrive\Desktop\UdayakumarShetty_NET_Training\Week1_CSharp_SQL\Day-5\Day5Programs> dotnet run
File content:
Welcome to Dhruv .NET Training!
PS C:\Users\udays\OneDrive\Desktop\UdayakumarShetty_NET_Training\Week1_CSharp_SQL\Day-5\Day5Programs> dotnet run
>>
Content appended successfully.
PS C:\Users\udays\OneDrive\Desktop\UdayakumarShetty_NET_Training\Week1_CSharp_SQL\Day-5\Day5Programs>
```

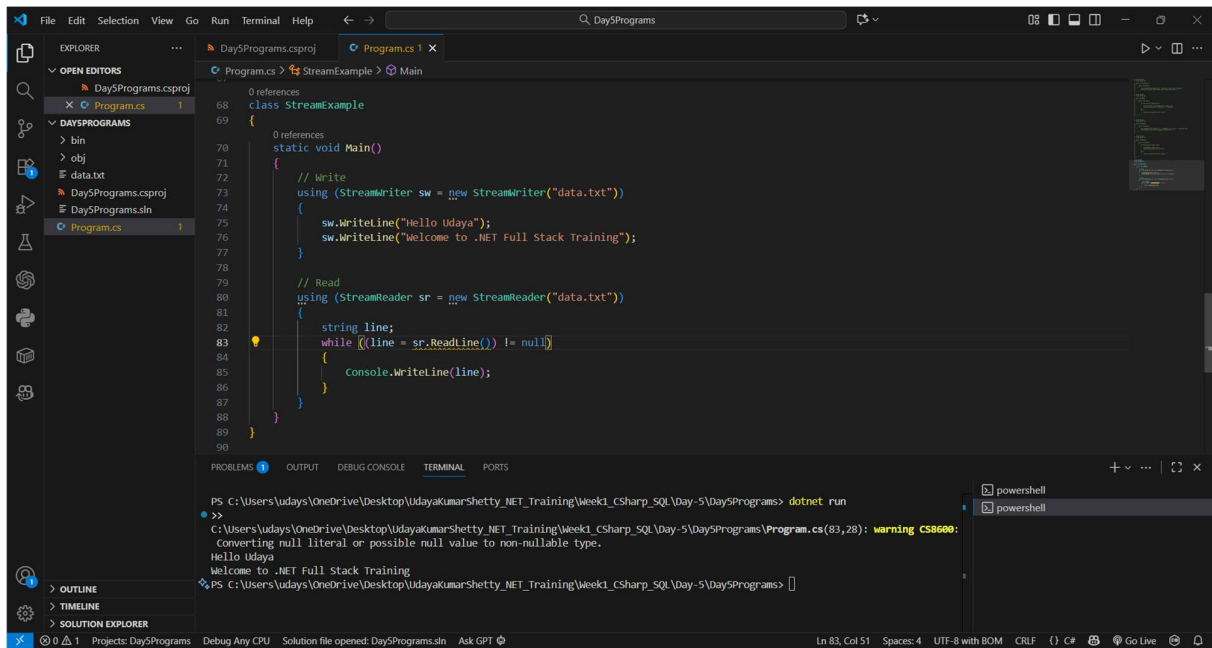
○ Delete



```
45
46 using System;
47 using System.IO;
48
49 0 references
50 class FileDelete
51 {
52     0 references
53     static void Main()
54     {
55         if (File.Exists("sample.txt"))
56         {
57             File.Delete("sample.txt");
58             Console.WriteLine("File deleted.");
59         }
60         else
61         {
62             Console.WriteLine("File not found.");
63         }
64     }
65 }
```

```
PS C:\Users\udays\OneDrive\Desktop\UdayakumarShetty_NET_Training\Week1_CSharp_SQL\Day-5\Day5Programs> dotnet run
File content:
Welcome to Dhruv .NET Training!
PS C:\Users\udays\OneDrive\Desktop\UdayakumarShetty_NET_Training\Week1_CSharp_SQL\Day-5\Day5Programs> dotnet run
>>
Content appended successfully.
PS C:\Users\udays\OneDrive\Desktop\UdayakumarShetty_NET_Training\Week1_CSharp_SQL\Day-5\Day5Programs> dotnet run
File deleted.
PS C:\Users\udays\OneDrive\Desktop\UdayakumarShetty_NET_Training\Week1_CSharp_SQL\Day-5\Day5Programs>
```

2. StreamReader and StreamWriter

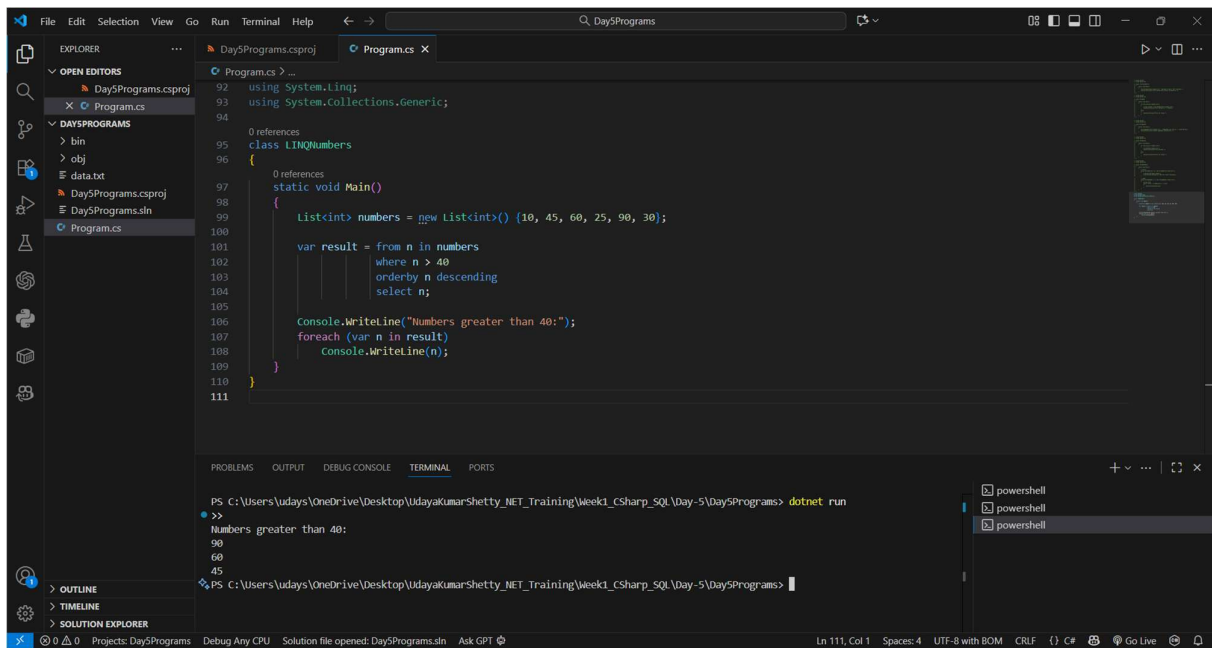


```
File Edit Selection View Go Run Terminal Help
Day5Programs.csproj Program.cs X
OPEN EDITORS
Day5Programs.csproj
Program.cs 1
DAYSPROGRAMS
bin
obj
data.txt
Day5Programs.csproj
Day5Programs.sln
Program.cs
class StreamExample
{
    static void Main()
    {
        // Write
        using (StreamWriter sw = new StreamWriter("data.txt"))
        {
            sw.WriteLine("Hello Udaya");
            sw.WriteLine("Welcome to .NET Full Stack Training");
        }

        // Read
        using (StreamReader sr = new StreamReader("data.txt"))
        {
            string line;
            while ((line = sr.ReadLine()) != null)
            {
                Console.WriteLine(line);
            }
        }
    }
}
```

```
PS C:\Users\udays\OneDrive\Desktop\UdayakumarShetty_NET_Training\Week1_CSharp_SQL\Day-5\Day5Programs> dotnet run
c:\Users\udays\OneDrive\Desktop\UdayakumarShetty_NET_Training\Week1_CSharp_SQL\Day-5\Day5Programs\Program.cs(83,28): warning CS8600:
Converting null literal or possible null value to non-nullable type.
Hello Udaya
Welcome to .NET Full Stack Training
PS C:\Users\udays\OneDrive\Desktop\UdayakumarShetty_NET_Training\Week1_CSharp_SQL\Day-5\Day5Programs>
```

3. LINQ Basics – List of Numbers



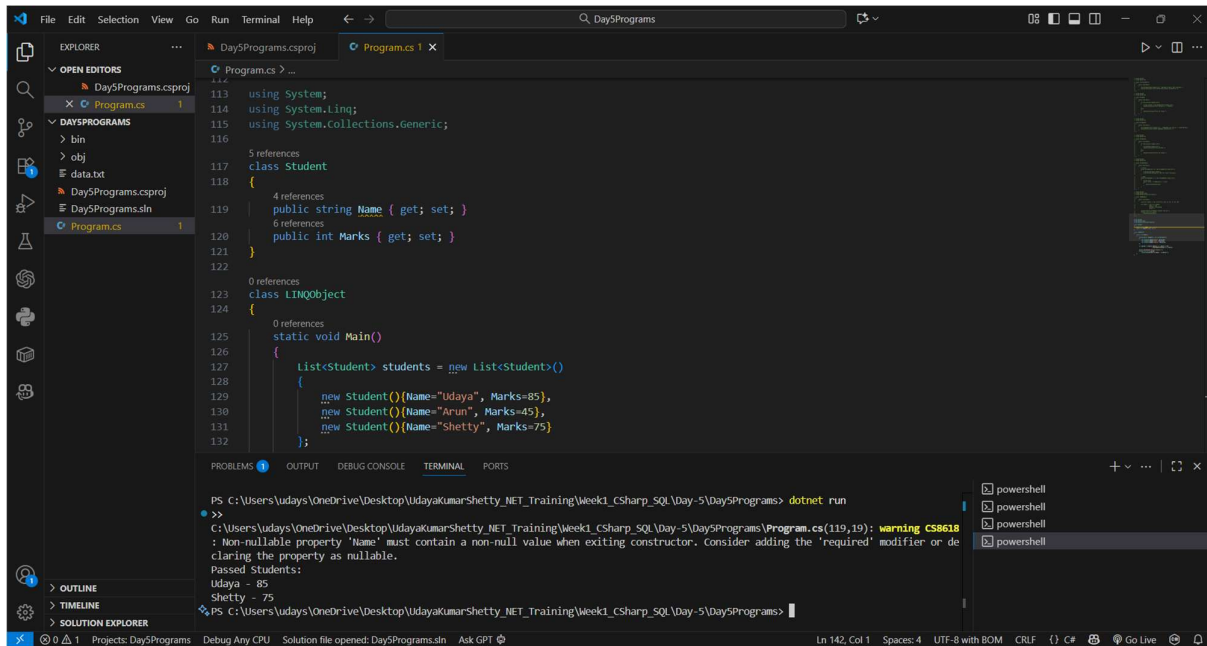
```
File Edit Selection View Go Run Terminal Help
Day5Programs.csproj Program.cs X
OPEN EDITORS
Day5Programs.csproj
Program.cs
DAYSPROGRAMS
bin
obj
data.txt
Day5Programs.csproj
Day5Programs.sln
Program.cs
using System.Linq;
using System.Collections.Generic;
class LINQNumbers
{
    static void Main()
    {
        List<int> numbers = new List<int>() {10, 45, 60, 25, 90, 30};

        var result = from n in numbers
                     where n > 40
                     orderby n descending
                     select n;

        Console.WriteLine("Numbers greater than 40:");
        foreach (var n in result)
        {
            Console.WriteLine(n);
        }
    }
}
```

```
PS C:\Users\udays\OneDrive\Desktop\UdayakumarShetty_NET_Training\Week1_CSharp_SQL\Day-5\Day5Programs> dotnet run
Numbers greater than 40:
90
60
45
PS C:\Users\udays\OneDrive\Desktop\UdayakumarShetty_NET_Training\Week1_CSharp_SQL\Day-5\Day5Programs>
```


4. LINQ with Objects



5. LINQ GroupBy and Aggregation

