Module 9

What to learn

Learn File Handling
File Class
File Stream
Binary Reader
Binary Writer
Stream Reader
Stream Writer
FileInfo
Directory Info
DriveInfo
Path
Converting object to json

Practice Exercise

Practice 1
https://learn.microsoft.com/en-us/dotnet/api/system.io.file?view=net-7.0

Practice 2
https://learn.microsoft.com/en-us/dotnet/api/system.io.fileinfo?view=net-7.0

Practice 3
https://learn.microsoft.com/en-us/dotnet/api/system.io.directoryinfo?view=net-7.0

Practice 4
https://www.tutorialsteacher.com/articles/convert-object-to-json-in-csharp

Practice 5

File Class

Create a text file: Write a C# program to create a text file named output.txt and write some sample content into it using the File class.

Read from a file: Given a text file sample.txt, write a C# program to read the content of the file using the File class and display it on the console.

Check if a file exists: Write a C# program to check whether a file named data.txt exists in the current directory and output a message accordingly.

Delete a file: Write a C# program to delete a file named deleteMe.txt from the current directory using the File class.

Copy a file: Create a program that copies the file source.txt to destination.txt using the File.Copy method.

Practice 6

File Stream

Write to a file using FileStream: Write a C# program to create a FileStream that writes the string "Hello World!" to a file named output.txt.

Read from a file using FileStream: Write a C# program that reads from the file input.txt and displays its content on the console using a FileStream.

Append to a file using FileStream: Write a C# program that appends the string "Appended Text" to the file example.txt using the FileStream class.

FileStream for binary data: Write a C# program that writes an array of bytes to a file named binaryData.dat using a FileStream.

Using FileStream to check file length: Write a C# program to open a file data.txt using a FileStream and print its length in bytes.

Practice 7

Binary Reader

Read a number using BinaryReader: Write a C# program to read an integer stored in a binary file data.dat using a BinaryReader.

Read a string using BinaryReader: Write a C# program that reads a string from a binary file sample.dat using the BinaryReader class and prints it.

Read multiple types from a binary file: Create a program that uses BinaryReader to read an integer, a double, and a string from a binary file and prints each one.

Handling end of file with BinaryReader: Write a C# program that reads from a binary file data.dat using BinaryReader and stops reading when the end of the file is reached.

BinaryReader for reading a float: Write a program to read a floating-point number from a binary file floatData.dat using BinaryReader and display it.

Practice 8

Binary Writer

Write an integer using BinaryWriter: Write a C# program that writes an integer 1234 to a binary file data.dat using BinaryWriter.

Write a string to a binary file: Write a C# program that writes the string "Hello Binary" to a binary file message.dat using BinaryWriter.

Write multiple data types using BinaryWriter: Write a program that uses BinaryWriter to write an integer, a double, and a string to a binary file.

Write a float to a binary file: Create a C# program that writes a floating-point number 3.14 to a binary file floatData.dat using BinaryWriter.

Append data with BinaryWriter: Write a C# program that appends the string "Appended text" to a binary file appendData.dat using BinaryWriter.

Practice 9

Stream Reader

Read from a text file using StreamReader: Write a C# program to read the content of a text file input.txt using StreamReader and display it on the console.

Read a line using StreamReader: Write a C# program that reads a single line from the file data.txt using StreamReader.

Read all lines using StreamReader: Write a C# program that reads all the lines from the file lines.txt using StreamReader and displays them.

StreamReader to check for end of file: Write a program that reads from file.txt using StreamReader and stops when the end of the file is reached.

Read a file with StreamReader and count words: Write a program that uses StreamReader to read a text file textfile.txt and counts the number of words in the file.

Practice 10

Stream Writer

Write to a file using StreamWriter: Write a C# program that writes "Welcome to C#" to a text file greeting.txt using StreamWriter.

Append text to a file using StreamWriter: Write a program that appends the text "Appended Text" to an existing file file.txt using StreamWriter.

Write multiple lines using StreamWriter: Write a C# program that writes multiple lines (e.g., "Line 1", "Line 2") to a text file lines.txt using StreamWriter.

Write formatted data using StreamWriter: Write a program that uses StreamWriter to write a formatted string, such as "Name: John, Age: 30", to a file.

Write and flush with StreamWriter: Write a program that writes some text to a file and explicitly calls StreamWriter.Flush() to ensure the content is written to the file.

Practice 11

FileInfo

Get file size using FileInfo: Write a C# program that uses FileInfo to get the size of the file data.txt and displays it.

Check if a file exists using FileInfo: Write a program that uses FileInfo to check whether the file config.txt exists and displays a message.

Get the creation time of a file: Write a C# program that uses FileInfo to get and display the creation time of a file file.txt.

Rename a file using FileInfo: Write a program that uses FileInfo to rename a file oldname.txt to newname.txt.

Delete a file using FileInfo: Write a program that uses FileInfo to delete a file unwantedfile.txt.

Practice 12

DirectoryInfo

Get the list of files in a directory: Write a program that uses DirectoryInfo to get all the file names in a directory C:\files and display them.

Create a directory using DirectoryInfo: Write a C# program that creates a new directory C:\newFolder using DirectoryInfo.

Get directory creation time using DirectoryInfo: Write a program that uses DirectoryInfo to get and display the creation time of a directory.

Check if a directory exists using DirectoryInfo: Write a program that checks whether the directory C:\test exists using DirectoryInfo and displays a message.

Delete a directory using DirectoryInfo: Write a program that deletes the directory C:\tempDir using DirectoryInfo.

Practice 13

DriveInfo

Get available free space on a drive: Write a program that uses DriveInfo to get the available free space on the C: drive.

Get drive type using DriveInfo: Write a program that uses DriveInfo to determine if the C: drive is a fixed drive or a removable drive.

Get total size of a drive using DriveInfo: Write a C# program that uses DriveInfo to get and display the total size of the C: drive.

Get drive name using DriveInfo: Write a program that uses DriveInfo to get and display the name of the C: drive.

Check if a drive is ready using DriveInfo: Write a program that checks if the D: drive is ready and displays a message accordingly.

Practice 14

Path

Get the file name from a path: Write a C# program that extracts and displays the file name from the path C:\files\example.txt using the Path class.

Get the directory name from a path: Write a program that extracts and displays the directory name from the path C:\files\example.txt using the Path class.

Combine two paths using Path.Combine: Write a program that combines the directory path C:\files and the file name data.txt using Path.Combine.

Get the file extension using Path: Write a program that extracts and displays the file extension from the path C:\files\document.pdf using Path.GetExtension.

Check if a path is rooted using Path: Write a program that checks if the path C:\files\example.txt is rooted using Path.IsPathRooted.

Practice 15

Converting Object to JSON

Convert an object to JSON: Write a C# program that converts a Person object with properties Name and Age to a JSON string using JsonConvert.SerializeObject().

Serialize a list of objects to JSON: Write a program that serializes a list of Product objects to a JSON array using JsonConvert.SerializeObject().

Deserialize JSON to an object: Write a C# program that deserializes a JSON string representing a Person object back to a Person class.

Handle JSON array deserialization: Write a program that deserializes a JSON array string into a list of Product objects.

JSON conversion with nested objects: Write a program that converts an object containing nested objects (e.g., an Order object with a list of Items) into a JSON string.

Assignment 1


```
[
  {
    "MovieID": 1,
    "MovieName": "Pathan",
    "Details": {
      "DirectorName": "Sidharth Anand",
      "ActorsNames": [
        "Shahrukh Khan",
        "Deepika Padukone",
        "John Abraham"
      ],
      "VideoLink": "https://tutorial.radixind.in/videos/Pathaan.mp4"
    }
  },
  {
    "MovieID": 2,
    "MovieName": "Dabang 3",
    "Details": {
      "DirectorName": "Prabhu Deva",
      "ActorsNames": [
        "Sonakshi Sinha",
        "Salman Khan"
      ],
      "VideoLink": "https://tutorial.radixind.in/videos/KKBKKJ.mp4"
    }
  },
```

```
{
  "MovieID": 3,
  "MovieName": "Kisi Ka bhai Kisi ki Jaan",
  "Details": {
    "DirectorName": "Farhad Samji",
    "ActorsNames": [
      "Pooja Hegde",
      "Salman Khan"
    ],
    "VideoLink": "https://tutorial.radixind.in/videos/Dabaang3.mp4"
  }
},
{
  "MovieID": 4,
  "MovieName": "Any Body can Dance",
  "Details": {
    "DirectorName": "Prabhu Deva",
    "ActorsNames": [
      "Prabhu Deva",
      "Punit Pathak",
      "Ganesh Acharya",
      "Noorin sha"
    ],
    "VideoLink": "https://tutorial.radixind.in/videos/abcd.mp4"
  }
}
```

]

store above data in movie.json file in your project folder.


```
{
  "Directors": [
    "Prabhu Deva",
    "Sidharth Anand",
    "Farhad Samji"
  ],
  "Movies": [
    {
      "MovieID": 1,
      "MovieName": "Pathan"
    },
    {
      "MovieID": 2,
      "MovieName": "Dabang 3"
    },
    {
      "MovieID": 3,
      "MovieName": "Kisi Ka bhai Kisi ki Jaan"
    },
    {
      "MovieID": 4,
      "MovieName": "Any Body can Dance"
    }
  ],
```

```
  "Actors": [
    "Shahrukh Khan",
    "Deepika Padukone",
    "John Abraham",
    "Sonakshi Sinha",
    "Salman Khan",
    "Pooja Hegde",
    "Salman Khan",
    "Prabhu Deva",
    "Punit Pathak",
    "Ganesh Acharya",
    "Noorin sha"
  ]
}
```

store this info.json file in your project folder.

Task:

List all the director to the user.

Create classes and interfaces accordingly.

- Input a director name from the user.
- based on the director name its movies should be displayed
- Show all the movies name and take input movie name from the user and displays its actors name.
- Print that movie name where director is working as a actor in the movie.

Assignment 2

Display folder hierarchy of any of folder. Recursion should be used

example output

```
A
        B
```

C
D
E
F
info.txt
hell.txt
p.txt

Assignment 3

There is one image gallary folder in your system. keep only that images which is modified in the current month other files should be deleted

Assignment 4

Scenario: File Management System for a Company

You are tasked with creating a simple file management system for a company that keeps track of employee records and the documents they work with. The system should be able to read, write, and modify files, handle directories, and store employee information in a structured format. You will also implement serialization and deserialization of data to store employee records in JSON format for future use.

Requirements:

File and Directory Operations:

Create a directory structure for storing employee files. The main directory will be C:\CompanyFiles\Employees\, where each employee will have a folder named after their Employee ID. For example: C:\CompanyFiles\Employees\12345\.
In each employee folder, create a text file called info.txt containing the employee's details (Name, Age, Department, Position).
Use the File and DirectoryInfo classes to check if the employee folder exists. If it does not, create the folder and the info.txt file with sample data.

Stream Operations (Reading and Writing Files):

Implement a function that reads the employee's details from the info.txt file in their respective folder using StreamReader. If the file does not exist, handle the error by displaying an appropriate message.
Implement a function to update the info.txt file with new information using StreamWriter. The program should ask for the new data (e.g., new position) and update the text file.

File Info and Drive Info:

Using FileInfo, get and display the size of the info.txt file for a specific employee.

Use DriveInfo to check if there is enough free space in the drive (C:) for creating new employee folders (set a threshold of 100MB).

JSON Serialization and Deserialization:

Create a class Employee with properties EmployeeId, Name, Age, Department, and Position. Serialize an instance of this class to a JSON string using JsonConvert.SerializeObject() and store it in a file named employee.json inside the employee folder.
Implement a function to deserialize the employee.json file and load the employee data into the Employee object.

Additional Operations:

Use Path class methods to determine the file extension of info.txt and print it to the console.
Implement a function that appends a document name (e.g., resume.pdf) to a documents.txt file inside the employee folder using StreamWriter (if documents.txt does not exist, create it).

Bonus: File Management System Operations

Implement the functionality to display all employee folders in the C:\CompanyFiles\Employees\ directory.
Write a method that takes an employee's ID and retrieves the document list from documents.txt, displaying the names of all files associated with that employee.

Online Reference

No online Reference

.NET Core Web API

WEB API (old)

Authentication And Authorization (WEBAPI)(old)

FullStackDevelopment_WIth_Dotnet_AND_Angular