C# File Handling

INSTRUCTOR: ADEEN BIN WAHEED

LECTURE 14

Some Important Points

1) There is an alternative way of doing conversion or casting in C#.

int.parse(“123”);

float.parse(“12.5”);

2) We can use the System.Collection header file to import all the collections else than

System.Collection.* because .* keyword is not applicable in C#.

3) We can use urdu language syntax by concatenation in C# (GUI) but not on console.

File Handling

- Become familiar with the concept of an I/O stream.
- Learn how to save data in a file.
- Learn how to read data from a file.

File Handling

A file is a collection of data stored in a disk with a specific name and a directory path. When a file is opened for reading or writing, it becomes a stream**.**

For File Handling. First of all we have to add following header in our C# program.

```
using System.IO;
```

StreamWriter

This class writes strings or append strings to a text file. We can write numbers or the textual representation of anything. It also uses a "using" block.

Note:

Syntax forms like the using-statement are useful. They allow automatic cleanup of resources.

StreamWriter

```
// Write a line in the file.
using (StreamWriter s = new StreamWriter("File.txt"))
{
        s.Write ("Hello World");          //Writing a word in a file.
        s.WriteLine("Hello World");    //Writing a line in a file.
 }
```

Overwriting a File

- Opening an output file creates an empty file.
- Opening an output file creates a new file if it does not already exist.
- Opening an output file that already exists eliminates the old file and creates a new, empty one data in the original file is lost.

AppendText Method (File Class)

We can use this method, if we don’t want our file to overwritten again.

Syntax:

```
 using (StreamWriter s = File.AppendText("File.txt"))
 {
        s.WriteLine("Hello World");
 }
```

AppendAllText Method (File Class)

We can also use this method, if we don’t want our file to overwritten again.

Syntax:

```
File.AppendAllText(“File.txt", newContent);
```

WriteAllLines Method (File Class)

We can write our all data in an array and than can pass this array to our file.

Syntax:

```
string[] stringArray = {"cat", "dog", "arrow"};

File.WriteAllLines("file.txt", stringArray);
```

Writing Data in a File

Running Demos

StreamReader

For text files, StreamReader is the most useful types. We use StreamReader in a using block, a special syntax form to read data from a File.

ReadLine:

This is a method on StreamReader. It returns null if no further data is available in the file.

StreamReader

```
// Read every line in the file.
using (StreamReader reader = new StreamReader("File.txt"))
{
    string line;
    while ((line = reader.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }
}
```

ReadAllText Method (File Class)

We can use ReadAllText method to load our data from a file. Then it prints the contents of the file. The data is now stored in a string object.

ReadAllText is the easiest way to put a file into a string. It is part of the System.IO namespace.

Syntax:

```
string file = File.ReadAllText("C:\\file.txt");

Console.WriteLine(file);
```

ReadAllLines Method (File Class)

We can read all the lines from a file and place them in an array. The code reads lines from "file.txt" and uses a foreach-loop on them.

Syntax:

```
string[] lines = File.ReadAllLines("file.txt");

foreach (string line in lines)

{

   Console.WriteLine(line);

}
```

Count Lines in a File (File Class)

We count the number of lines in a file with few lines of code.

Syntax:

```
int lineCount = File.ReadAllLines("file.txt").Length;
```

Reading Data from a File

Running Demos