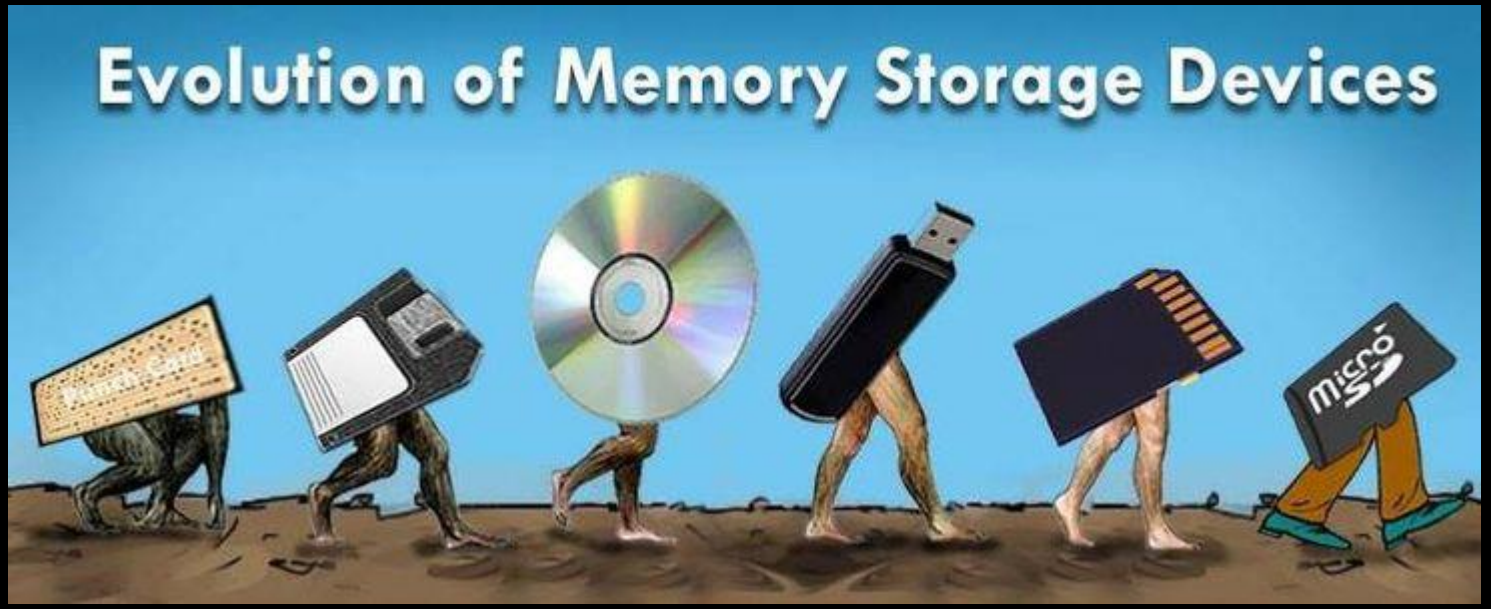

TUTORIAL: FILE I/O AND STRING

BME 121, Fall 2016

Rasoul Nasiri

Agenda:

- Reading File
- Checking the existence of file
- File write
- File modes and access levels
- String processing
  - Trim()
  - IndexOf()
  - Substring()
  - Split()

File I/O review the commands

1
```
using System.IO;
```

2
```
FileStream inFile = new FileStream(@"myfile.txt", FileMode.Open,
FileAccess.Read);
```

3
```
StreamReader inStream = new StreamReader(inFile);
```

4
```
string line = inStream.ReadLine();
```

5
```
inStream.Dispose();
inFile.Dispose();
```

Change file to count the number of chars in each line

```
using System;
using System.IO; // Don't forget to import System.IO

// in Main():
// Create an object of FileStream and bind it to a specific file
FileStream inFile = new FileStream(@"myfile.txt", FileMode.Open, FileAccess.Read);
StreamReader inStream = new StreamReader(inFile);

Console.WriteLine("Content of the file:");
int counter = 0;
while(!inStream.EndOfStream)
{
    // read a line of text from the stream, then display that line of text
    Console.WriteLine( inStream.ReadLine() );
    counter++;
}
Console.WriteLine("Number of lines = {0}", counter);

inStream.Dispose(); // close the stream
inFile.Dispose(); // then close the file
```

`EndOfStream` is a bool property which has the value of `true` when the hidden "read cursor" reaches the end of the file (after the last characte Otherwise it has the value of `false`.

The "read cursor" is moved forward one line at a time when we call ReadLine().

We use EndOfStream to loop until the end of the file.

Practice: Create a class for file operations

- Create a class named “myFileOperations”
- Constructor with input string (file path)
- Method countChars
- Write the first 10 lines in another file
  - Run this program two times and see what will happen
  - Try “CreateNew” instead of “Create” in file mode

Different file mode:

Members

	Member name	Description
	Append	Opens the file if it exists and seeks to the end of the file, or creates a new file. This requires FileIOPermissionAccess.Append permission. **FileMode.Append** can be used only in conjunction with **FileAccess.Write**. Trying to seek to a position before the end of the file throws an IOException exception, and any attempt to read fails and throws a NotSupportedException exception.
	Create	Specifies that the operating system should create a new file. If the file already exists, it will be overwritten. This requires FileIOPermissionAccess.Write permission. **FileMode.Create** is equivalent to requesting that if the file does not exist, use CreateNew; otherwise, use Truncate. If the file already exists but is a hidden file, an UnauthorizedAccessException exception is thrown.
	CreateNew	Specifies that the operating system should create a new file. This requires FileIOPermissionAccess.Write permission. If the file already exists, an IOException exception is thrown.
	Open	Specifies that the operating system should open an existing file. The ability to open the file is dependent on the value specified by the FileAccess enumeration. A System.IO.FileNotFoundException exception is thrown if the file does not exist.
	OpenOrCreate	Specifies that the operating system should open a file if it exists; otherwise, a new file should be created. If the file is opened with **FileAccess.Read**, FileIOPermissionAccess.Read permission is required. If the file access is **FileAccess.Write**, FileIOPermissionAccess.Write permission is required. If the file is opened with **FileAccess.ReadWrite**, both FileIOPermissionAccess.Read and FileIOPermissionAccess.Write permissions are required.
	Truncate	Specifies that the operating system should open an existing file. When the file is opened, it should be truncated so that its size is zero bytes. This requires FileIOPermissionAccess.Write permission. Attempts to read from a file opened with **FileMode.Truncate** cause an ArgumentException exception.

Different file accesses:

	Member name	Description
	Read	Read access to the file. Data can be read from the file. Combine with **Write** for read/write access.
	ReadWrite	Read and write access to the file. Data can be written to and read from the file.
	Write	Write access to the file. Data can be written to the file. Combine with **Read** for read/write access.

String processing

- Main operations on string
  - Main methods in class string

method	What it does
Trim	Removes all leading and trailing white-space characters from the current String object.
IndexOf	Reports the zero-based index of the first occurrence of the specified string in this instance.
LastIndexOf	Reports the zero-based index position of the last occurrence of a specified character within this instance.
Substring	Retrieves a substring from this instance.
Split	Splits a string into substrings that are based on the characters in an array.

```
public string Trim()

public int IndexOf(
        string value
)

public int LastIndexOf(
        char value
)

public string Substring(
        int startIndex
)

public string[] Split(
        params char[] separator
)
```

Work with strings:

- We have an information file for each student.
- File includes: name, family, #std, email, and midterm score.
- Read a file for specific student line by line
- First: trim lines to remove spaces and print each line
- Second: find the index of “:” in each line and print it
- Third: find the info after “:” in each line using Substring method
- Do the same thing with split

THE END …