Module 1 Day 16

File I/O - Reading

What makes an application?

- Program Data
  - ✓ Variables & .NET Data Types
  - ✓ Arrays
  - ✓ More Collections (list, dictionary, stack, queue)
  - ✓ Classes and objects (OOP)
- Program Logic
  - ✓ Statements and expressions
  - ✓ Conditional logic (if)
  - ✓ Repeating logic (for, foreach, do, while)
  - ✓ Methods (functions / procedures)
  - ✓ Classes and objects (OOP)
  - ❑ Frameworks (MVC)

- Input / Output
  - User
    - ✓ Console read / write
    - ❑ HTML / CSS
    - ❑ Front-end frameworks (HTML / CSS / JavaScript)
  - Storage
    - ❖ File I/O
    - ❑ Relational database
    - ❑ APIs

Directory(Info), File(Info) and Path

- System.IO namespace
- Classes that allow you to navigate the file system
- Directory == folder
- *Directory* and *File*: static methods for navigating, creating and deleting folders and files
- *DirectoryInfo* and *FileInfo*: instance methods for detailed information on a single folder or file
- *Path* provides help parsing and combining paths together
- https://docs.microsoft.com/en-us/dotnet/standard/io/common-i-o-tasks

Directory and DirectoryInfo

Directory (static)	DirectoryInfo	Description
--	new DirectoryInfo(path)	Constructor
CreateDirectory(path)	Create()	Create a directory / folder
Delete(path)	Delete()	Delete a directory / folder
Exists(path)	Exists	This is a Property on DI class
Move(fromPath, toPath)	MoveTo(toPath)	Move a directory to another path
GetDirectories(path)	GetDirectories()	List subdirectories. Dir method returns string[]; DI method returns DI[]
GetFiles(path)	GetFiles()	List files. Dir method returns string[]; DI method returns FI[]
GetParent(path)	Parent	This is a Property on DI class, returns DI
GetDirectoryRoot(path)	Root	This is a Property on DI class, returns DI

- https://docs.microsoft.com/en-us/dotnet/api/system.io.directory?view=netcore-2.2
- https://docs.microsoft.com/en-us/dotnet/api/system.io.directoryinfo?view=netcore-2.2

File and FileInfo

File (static)	FileInfo	Description
--	new FileInfo(path)	Constructor
CreateFile(path)	Create()	Create a new file
Delete(path)	Delete()	Delete a file
Exists(path)	Exists	This is a Property on FI class
Move(fromPath, toPath)	MoveTo(toPath)	Move a file to another path / name
OpenRead(path)	OpenRead()	Open an existing file for reading
OpenWrite(path)	OpenWrite()	Open an existing file file for writing, or create a new file.

- https://docs.microsoft.com/en-us/dotnet/api/system.io.file?view=netcore-2.2
- https://docs.microsoft.com/en-us/dotnet/api/system.io.fileinfo?view=netcore-2.2

Path

- Provides help parsing and combining paths together
  - Cross-platform (separators, drive letters, roots, etc.)
  - Absolute vs. relative path
- These do not modify files and folders in the file system
- Combine, GetDirectoryName, GetExtension, GetFileName, GetFullPath, GetRelativePath, GetTempPath
- https://docs.microsoft.com/en-us/dotnet/api/system.io.path?view=netcore-2.2
- Common I/O Tasks:
  - https://docs.microsoft.com/en-us/dotnet/standard/io/common-i-o-tasks

Reading from a File

```
using (StreamReader stream = new StreamReader(path))
{
    // Read a line at a time.
    while (!stream.EndOfStream)
    {
        string line = stream.ReadLine();
        // Process the line however you want to here...
    }
}
```

- Use a *StreamReader*
  - Allows you to read and process chunks of a file sequentially
  - Think streaming a movie vs. downloading
- *EndOfStream* property
  - Tells when we have reached the end of the file
- *using* construct
  - Creates, uses and disposes a resource within the block
  - Important for cleaning up un-managed resources (*not* garbage-collected)
  - IDisposable

Exceptions

- Exceptions are how the .Net Framework reports runtime errors
- Exceptions are thrown when an error occurs
- Your code can catch an error and handle it
  - You can re-throw it with throw;
- Examples of runtime errors:
  - Attempting to int.Parse a non-numeric value
  - Attempting to read a File that does not exist
  - Divide by zero
  - NULL reference exception
- You can define and throw your own Exceptions
- Exceptions “unwind the stack” until someone catches it

Exceptions

```
try
{
    // Do some work here...
}
catch (ArgumentNullException e)
{
    // catch most specific Exceptions first
}
catch (Exception e)
{
    // (optional) catch more general exceptions later
    // (optional) re-throw the same exception so it can be caught further up the stack
    throw;
}
finally
{
    // (optional) Do work that shouldexecute whether the above succeeded or failed
}
```

Bonus: Intro to Console Menu Framework

- Nuget packages allow you to add functionality written by other developers
- Nuget is known as a "package manager"
- This package written by Josh & Mike
- Easily create menu-driven programs
- Func<MenuOptionResult> means "a pointer to a method which returns type MenuOptionResult"