Day 6

Partial Classes

- If the definition of a class is split into multiple source files(.cs) then each part is referred as a partial class
- All the parts available at compile time will form the final class.

Partial Classes

- All the parts must have the same accessibility, such as public, private, etc.
- If any part is declared abstract, then the whole type is considered abstract.
- If any part is declared sealed, then the whole type is considered sealed.
- If any part declares a base type, then the whole type inherits that class.
- Parts can specify different base interfaces, and the final type implements all the interfaces listed by all the partial declarations.

Partial Methods

- A partial class or struct may contain a partial method.
- One part of the class contains the signature of the method.
- Implementation can be defined in the same part or another part.

shared assembly

- By default, every assembly is a private assembly.
- If we add a reference to a private assembly to any project, a copy of the assembly is given to the project.
- So each project maintains a private copy of the assembly
- A shared assembly is an assembly available for use by multiple applications on the computer.

Reflection

- Reflection is a programming feature by which a program can understand and manipulate itself..

File I/O

- System.IO namespace contains the set of classes and interface for doing fie operations.

DirectoryInfo class && Directory class

- DirectoryInfo class allows you to
  – create a Directorory
  – Create sub directories under current directory
  – Find list of sub directories under current directory
  – Find list of files under current directory
  – Delete directory
- Directory class provides similar functionality as DirectoryInfo.
- But methods under Directory class are static methods

DriveInfo

- The DriveInfo class under System.IO namespace can be used to find properties about a drive such as drive type, total size of drive, available free space etc;

FileInfo class & File class

- FileInfo class contains methods for
  - creating a file
  - Opening a file for reading/writing
  - Deleting a file etc
- File class also provides similar functionality but as static methods

Writing to a file/reading from a file

- Read/Write to a file can be done using FileStreams
- But FileStreams can work with byte data only.
- So you can do read/write a bytes or byte array.

Writing to a file/reading from a file

- Read/Write to a file can be done using StreamWriter and StreamReader classes
- StreamReader and StreamWriter can do reading/writing of character-based data

Thread

Thread

- A thread is a light-weight process, which can execute in parallel with other threads.
- To create a thread do the folloing steps
- Create a method to be the entry point for the new thread.
- Create a new ParameterizedThreadStart (or ThreadStart) delegate, passing the address of the method defined in step 1 to the constructor.
- Create a Thread object, passing the ParameterizedThreadStart/ThreadStart delegate as a constructor argument.

Thread

- The Thread constructors can take a function name or a delegate to the method
  - If the method has no arguments, you can pass methodname or object of ThreadStart delegate to the constructor.
  - If the method has an argument, you pass object of ParameterizedThreadStart delegate to the constructor.

Thread

- ThreadStart has the signature:

```
public delegate void ThreadStart()
```

- ParameterizedThreadStart has the signature:

```
public delegate void ParameterizedThreadStart(object obj)
```

Thread pool

- Thread pool is a collection of threads.
- It is used to perform tasks in the background.
- When a thread completes a task, it is sent to the queue wherein all the waiting threads are present.
- Threads from thread pool are reusable.

Managing Concurrency

- When multiple threads are running then a mechanism to ensure that shared data if changed is done by only 1 thread at a time.

lock Keyword

- lock keyword allows you to define a scope of statements that must be synchronized between threads.
- The lock keyword requires a token (an object reference) that must be acquired by a thread to enter within the lock scope.
- Normally the current object reference will be used as the token

Monitor

- We have 2 use 2 statements Monitor.Enter and Monitor.Exit to show the critical section.
- Both Enter and Exit uses an object key/token to lock the statements

System.Threading.Interlocked Type

Is providing a set of atomic operations for variables that are shared by multiple threads.

Interlocked Proves methods for example Incrememt, Decrement, Add, Substract, Exchange etc as atomic operations.

It ensures that only 1 thread will be able to do these operations at a time.