# WEEK 1

8/22/22

* Use common sense when writing the code, good documentation and whatnot.
* Instead of using “import” to import libraries, use “Using” in C#.
* One source file can have multiple public classes, Java will yell at you.
* Classes function similarly to Java: public, private, protected, internal(this is the stock class, the same as using “public”).
* For all practical purposes, internal and public are basically the same thing; methods are only accessible to other classes in that assembly.
* Using system.out.println()? Don’t; C# uses Console.Write() and Console.WriteLine()
* Gotta use .NET sdk 6.0, 7.0 is out, but in the middle of semester. (Also has very nice documentation)

8/24/22

* Lots of excitement for day 2 of class.
* Having a quiz on Friday mainly covering the syllabus and getting to know C#
* Declaring public variables is a nono.
* Class foo {

private int a;  
public int A {  
 get return Int(a);

set a = *some value*;

}

}

* Class foo {  
  public int A {get; private set;}  
  }  
  This method creates “under the hood” variables to store A  
  Also allows you to have a public get and a private set
* Databases read the properties of the class, not the public variables.
* Properties are used to declutter the code “a lot”
* Implicit variables are exactly what they sound like, it makes refactoring “a whole lot easier”  
  int i = 3; (explicitly defined)  
  var j = 4; (implicitly defined)
* Using implicit variables still means that they are strongly typed.
* Using a try/catch/finally: you can have try/catch OR try/finally
* NEVER throw: Exception, NullReferenceException, SystemError, IndexOutOfBoundsException.
* Throwing any of these is “a giant jerk move”, you are passing your shit to someone else to deal with   
  “I am a lazy programmer who doesn’t want to correct this, so I'm gonna throw it onto someone else.”
* When submitting things, select the two files you need to zip, this is the .csproj and .cs project.
* Using DateTime? Don’t.

Use StopWatch instead

* For practice, remove the “./” in front of file names
* Always use forward slash for file separation “it’s the one next to the shift key!”

8/26/22

* Took quiz and did in class activity

# WEEK 2

8/29/22

* Reviewed the quiz’s solutions
* The parts of a computer are
  + Memory - temporary storage for a program and its data; it gets erased when the computer resets and is very fast.
  + CPU - It is the core and it executes instructions.
  + I/O - The interfaces between the computer and the outside world; this can be a freebie on the quiz.
* “interrupt and a trap difference is that interrupt is hardware and a trap is software; they pretty much do the same thing”
* Interrupts are asynchronous, it gets placed in the PIC(Programmable interrupt controllers) then the CPU reads it
* A trap is used to change a program via a debugger’s breakpoint or exceptions.
* Main memory is the RAM; most common is 16 GB in modern applications.
* Swapping - Non-sed memory copied to disk
* Virtual Memory - Contiguous block of logical memory is provided for a program
* Flash memory (M.2 SSD) is much faster because there is no rotations
* Rotational drives are typically larger but a bit slower; while maintaining more reliability.
* Multiprogramming - allows multiple programs to be scheduled to run and was originally created for batch systems.
* Cooperative Multitasking - Often referred to as time sharing; this leads to more than one process being able to be scheduled at the same time
* Preemptive multitasking - used in all modern operating systems, however the OS can interrupt a process if need be; makes the computer look like it is multitasking

# WEEK 3

9/5/22

* Processes vs Threads
  + Both use resources and are scheduled
  + Threads get the CPU registers, Code, Data, Stack/heap info, and OS resources
  + Process is a group of threads that share a set of resources.
    - This shared set of resources includes code, global data, heap, and OS resources
  + They do NOT share CPU registers, thread local data and the stack
  + Main differences include
    - Threads are designed for one program to do multiple things at once
    - Process is meant to group those items together
    - Threads the belong to a process share the same address space
    - Different processes have different address spaces
    - Threads from a different address space that are loaded need more work done by the CPU
  + Threads can be managed by the kernel via 1:1 mapping. this is commonly used by modern OS
  + Categories of threads
    - CPU Bound - does CPU work (sequencing DNA)
    - IO Bound - waits for the IO (keyboard input)
  + Deadlocks
    - Caused when threads need the same resource
    - Avoid them
* Interprocess Communication (IPC)
  + Shared Memory
    - We have to explicitly share memory and mapping physical memory
    - Was used before multiprocessor computers existed
  + Message Passing
    - Processes in the same system can send messages to another process
  + Network protocols
    - TCP (reliable and uses streams)
    - UDP (unreliable and uses Datagrams)
  + Remote procedure call
    - Ability to run function on remote computer as if it were local
  + Remote invocation
    - Lets you call methods on remote objects
  + IPC is used in distributed systems, while threads are used in parallel systems
* Concurrency - Doing more than one thing at the same time
* Multithreading - one form of concurrency that uses threads
* Parallel programming - Concurrent programming that relies of Futures to return data instead of manually creating threads
* Event driven programming - Having an application to respond to events on other threads.
* Always start your thread
* Parallel for methods are different from the usual.  
  Parallel.For(<start>, <end>, <item> => {anonymous method})

# WEEK 4

9/12/22

* Critical section - code that must execute without interruption, does not always, but it *should* run without interruption
* Atomic - code that runs without interruption
* All atomic sections are critical sections, but not all critical sections are atomic
* Interlocks are pretty useful; use interlock.add(ref a, 5);
  + Locks variable a and adds 5 to it in an atomic fashion
  + use it for ByteCount, FileCount, and FolderCount in project
* Semaphore - protects a resource (like variable a in above example)
* Monitor - protects a block of code (uses lock() {//code})
* Race conditions are not reproducible
* NAT - Network Address Translation
  + private ip addresses look like
    - 192.168.x.x
    - 172.16.x.x - 172.31.x.x
    - 10.x.x.x

# WEEK 6

9/26/22

* Basic networking week, Prof Brown’s favorite week
* A network is when two machines talk to each other using a predefined protocol
  + Network can be like a phone
  + Protocol can be like answering the phone with ‘hello’ and ending with ‘bye’
* Sizes of networks
  + LAN - Local Area Network
    - Very fast
    - Privately owned
    - Small area
  + WAN - Wide Area Network
    - Not as fast
    - High capacity
    - Operated by an ISP
* Layering
  + All network layers are layered and each layer is responsible for one key aspect of the network
  + lets you plug and play in different layers when necessary