Lecture notes/Miscellaneous

SDK -> .Net Core

- Assembly vs. Namespace
  - Namespace: logically arranges types (classes) to avoid naming conflict
  - Assembly: physical separation of types (classes); ends with .exe or .dll; Deployable units
- Properties like getters and setters in Java called smart fields by Microsoft
  - encapsulate value of private variables

C# is typesafe language and case sensitive

Access Modifiers

- change behaviors and scope of types
- public
  - type or member can be accessed by any other code in same assembly
  - or another assembly referencing it
- private
  - type or member can be accessed only by code in same class or struct
- protected
  - type or member that can be accesed only by code in same class
  - or class derived from that class
- internal
  - type or member can be accessed only by code in same class
  - or in class derived from that class
- protected internal
  - type or member can be accessed by any code in assembly where it's declared
  - or within derived class in another assembly
- private protected
  - type or member can only be accessed within declaring assembly
  - can be accessed by code in same class or in type derived from that class

---

Data Types (reference, value)

- reference and value types in C#
- reference: store reference to data (objects)
  - value stored in heap memory
  - expensive retrieval process
  - examples include:
    - classes, interface, and delegates
    - also predefined types: string, arrays, collections, etc.
- value: directly contain data
  - stored in memory Stack

- faster access
- structs and enums
- predefined types: int, long, short, byte, DateTime, char

Extended Modifiers

Class, Struct, Interface, Enum

Semantic code (DRY, Comments-inline, Comments-XML, KISS)

Common Language Runtime (BCL, CIL, CLI, CLR, CTS, JIT, VES)

CLI -> Common Language Infrustructure

C# f# VB

CSC fsc VBC -- language specific; all can be combined using dotnet build using CIL

CIL (makes code platform independent) -> Runtime (CLR) also platform independent

use dotnet new console -l VB -o directory name (to create console with different language than c#)

see .NetCLI on Day 1 slides notes