Module 1 Day 10

Classes & Encapsulation

Encapsulation

- The action of enclosing something in or as if in a capsule
- From Wikipedia
  - A language construct that facilitates the bundling of data with the methods (or other functions) operating on that data.[
  - A language mechanism for restricting direct access to some of the object's components.
- Mike's words
  - Bundling stuff together which goes together (as in classes)
    - Models real-world as closely as possible - **maintainable**
  - Not showing outsiders any more than they need to know (access modifiers)
    - **Loosely couples** your system; makes system more **flexible**

Encapsulation

- Access modifiers help us encapsulate

```
public int Property { get; set; }          //public set
public int Property { get; }               //readonly set w/in constructor
public int Property { get; private set; } //private set
```

- How can we better encapsulate the Card class?
  - Which properties should be set only when the card is created?
  - Which properties should be set only by the Card class itself?
  - Which properties should be freely available to be set by the public?

Lecture Code Goals

- Properly encapsulate the Card class
- Create lookup tables for
  - Converting from a Rank to a Face Name (“Ace”, “Two”…)
  - (Optionally) Finding a “Suit Emoji” for a highly enjoyable user experience!
    - https://unicode-table.com/en/#control-character
    - Console.OutputEncoding = System.Text.Encoding.UTF8;
  - Does each instance need its own table?
    - Then let’s make it *static*
- Create a CardDeck class to represent a standard deck of cards
  - How should we encapsulate the members of the CardDeck?
- In Program.cs, deal a hand to each of 2 players
  - Print out the two hands