01 - HARDWARE ABSTRACTIONS

COMP256 – COMPUTING ABSTRACTIONS

DICKINSON COLLEGE

COMP256 – COMPUTING ABSTRACTIONS

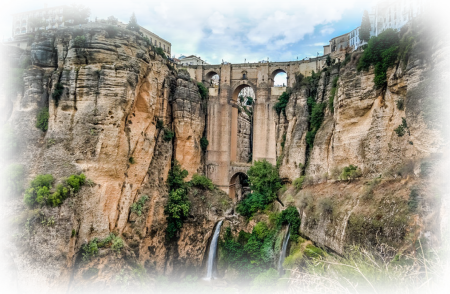
- Prof. Grant Braught
  - http://users.dickinson.edu/~braught/
    - Office hours, Meetings, Course Link
- Course Info:
  - Meetings:
    - MWF 10:30-11:20 Tome 118
    - T Lab 3:00-5:00 Tome 213 / Tome 118
  - Schedule
  - Syllabus
  - Moodle

ACTIVITY

- Abstractions and Computing Abstractions
  - Pairs / Threes
  - Shared GoogleDoc on Moodle
    - 1. ~10 min
    - 2. ~2 min
    - 3. ~2 min

ABSTRACTIONS

- *Abstractions* and *abstraction hierarchies* build bridges across chasms of complexity.



ABSTRACTIONS IN THIS COURSE

- Hardware Abstractions
- Language Abstractions
- Operating System Abstractions
- Networking Abstractions
- Web Architecture Abstractions

HARDWARE ABSTRACTIONS

- *Hardware abstractions* bridge the gap from transistors to programmable machines that perform useful computation.
  - *Transistors* make logic gates
  - *Logic gates* make circuits
  - *Circuits* make bigger circuits and eventually *machines*

Image from: https://ploda.com.ua/g27814829-pererabotka-loma-othodov

LANGUAGE ABSTRACTIONS

- *Language Abstractions* bridge the gap from the 1's and 0's (bits) of "computer language" to textual or graphic languages that are easier for people.
  - Compiled Languages
    - *High level language* (HLL) is translated (compiled) to an intermediate language (e.g. Assembly Language)
    - *Intermediate language* is translated (assembled) into machine language
    - *Machine language* is executed directly by the hardware
  - Interpreted Languages
    - HLL is read and processed by a compiled program (interpreter) running on the hardware.

Image from: https://www.archer.eu/blog/what-makes-functional-programming-a-viable-choice-for-artificial-intelligence-projects/

OPERATING SYSTEMS ABSTRACTIONS

- **Operating systems abstractions** bridge the gap between the physical hardware and the user experience.
  - E.g. Bits and pixels vs files, folders, text and graphics.
  - *User interfaces* are presented by HLL programs,
  - *HLL programs* invoke library routines
  - *Library routines* make system calls
  - *System calls* invoke operating system code
  - *Operating system code* uses that uses device drivers
  - *Device drivers* interact with the hardware.

Image from: https://www.supinfo.com/articles/single/4344-introduction-to-operating-systems

NETWORKING ABSTRACTIONS

- *Networking abstractions* bridge the gap between wanting to transfer data between two computers and the process of actually sending electrical signals over wires or through the air, though tens or hundreds of intermediate machines to get it where it needs to go.
  - *Application layer protocols allow* HLL programs to exchange information using the transport layer.
  - *Transport layer* (end-to-end) protocols allow client and server machines to exchange information via the network layer.
  - *Network layer (routing)* protocols direct information from the source to the destination using the link layer.
  - *Link layer* protocols move data from one machine to the next machine on the path using the physical network.
  - *Physical layer* hardware devices transmit the data from one machine to the next over wires or through the air.

Image from: https://dealna.com/Article/Post/6564/Networking-Accessories

WEB ARCHITECTURE ABSTRACTIONS

- *Web architecture abstractions* bridge the gap between the front-end user experience of visiting a web site in a browser and the server and databases where the information for that page is actually stored/generated.
  - *Presentation layer technologies* (HTML/CSS) present a user interface
  - *Front-end code* (JavaScript/Ajax and wrappers – React/Angular) exchange information with servers through application programming interfaces (APIs).
  - *API calls* invoke back-end code on the server
  - *Back-end code* interacts with databases and returns information to the front-end code which modifies the presentation.

Image from: https://vistapointe.net/web-world-map.html

SO WHY THIS COURSE…

- Breadth of preparation for a wide range of possible futures.
  - Hardware, languages, operating systems, networks, web development
- Understanding how things work builds intuition for:
  - Learning new abstractions
  - Creating new abstractions
  - Debugging across levels of abstraction
    - Sometimes, what looks like a bug at one level can only be understood by looking at a lower level of abstraction.

ABSTRACTION FAILURE #1

- Is this an infinite loop?

```
int i=1;

while (i > 0) {
    i++;
}

System.out.println("Done!");
```

ABSTRACTION FAILURE #2

- What is the output?

```
System.out.println(Integer.MAX_VALUE); // 2147483647
System.out.println(-Integer.MAX_VALUE);

System.out.println(Integer.MIN_VALUE); // -2147483648
System.out.println(-Integer.MIN_VALUE);

System.out.println(Math.abs(Integer.MIN_VALUE));
```

ABSTRACTION FAILURE #3

- What is the output?

```
double f=4.35;

int n = (int)(100*f);

System.out.println(n);
```

ABSTRACTION FAILURE #4

- Which takes longer to run?

```
int[][] x = new int[100000][10000];
for (int i = 0; i < x.length; i++) {
  for (int j = 0; j < x[0].length; j++) {
       x[i][j] = 1;
  }
}
```

```
int[][] x = new int[100000][10000];
for (int i = 0; i < x[0].length; i++) {
  for (int j = 0; j < x.length; j++) {
    x[j][i] = 1;
  }
}
```