a place of mind

THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 213
Introduction to Computer Systems

Unit 0 – Introduction

All slides adapted from materials by Mike Feeley, Jonatan Schroeder, Robert Xiao, and Jordon Johnson

Staff

- 14:00 section – Mike Feeley
- 17:00 section – Geoffrey Tien
  - ICCS 245
  - Office hours – see Canvas

- Course coordinator: Olivia Zhou (temporarily)
  - cpsc213-admin@cs.ubc.ca
  - For all administrative queries (e.g. missing activities, etc.)

- And our many wonderful TAs
  - Meet them in labs and on Piazza

Course logistics

- Canvas – https://canvas.ubc.ca/courses/101867/
  - Starting point for everything
- PrairieLearn – https://ca.prairielearn.com/pl/course_instance/2464/
  - For assignments and quizzes
  - Access using CWL
- iClicker Cloud
  - For in-class questions; participation marks (register now via Canvas)
- Piazza – piazza.com/ubc.ca/winterterm12022/cpsc213101102022w1/home
  - discussion, help, announcements etc.
  - Discord server for informal, socialization, but not for asking course questions

Grading

- Participation
  - 1% for actively participating in 10 of 12 labs (1hr section)
  - 1% for lecture iClicker questions for 80% of lectures
- Assignments
  - 25% for 10 assignments
  - best 8 of A1-A9 + A10
- Quizzes
  - 30% for 3 quizzes
  - in-person in CBTF (self-schedule 1hr slot during quiz weeks)
- Final exam
  - 43% for final exam
  - Improve your quiz by final; up to 80%; by specific learning goal/topic
  - Must achieve at least 50% on quiz/final average AND submit at least 80% of homeworks

What is a computer?

- A computer is a machine that…
  - Stores information and can perform a simple set of operations
  - Uses existing information plus work to produce new information
  - Performs a sequence of operations determined by stored instructions
    - stored instructions are also part of the machine's stored information
  - Can be modeled by a Turing machine

- Examples:
  - The human brain, analog water computer, your smartphone

What is a computer **system**?

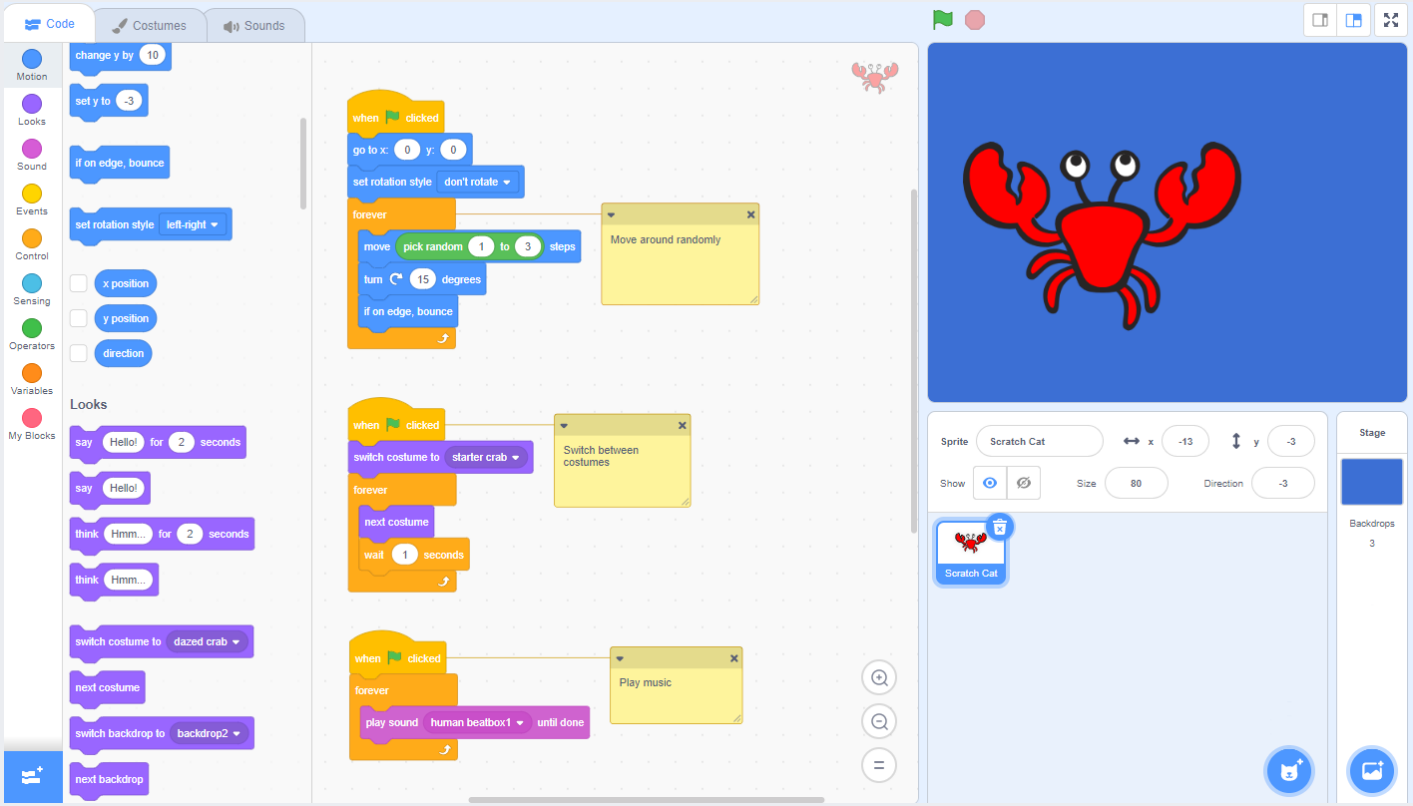
- An abstraction of a computing machine
- Used in various combinations by programmers

- Some abstractions:
  - Hardware instruction-set architectures (CPSC 213, 313)
  - Operating systems (CPSC 313, 415)
  - Compilers and high-level languages (CPSC 311, 411)
  - Middleware and libraries

- Programming is:
  - Building new abstractions in software
  - Using existing abstractions in libraries and the operating system

Why should you care?

- You can use abstractions you don't understand
- But your understanding is rooted in and restricted to the abstraction
- Example: https://scratch.mit.edu/projects/11483977



Why should you care?

- A computer scientist understands computation deeply
- Mental model of computation rooted in the machine
  - Informs understanding of abstraction
  - Enables generalization among abstractions (e.g. languages)
  - Empowers going deeper, beyond abstraction, when needed

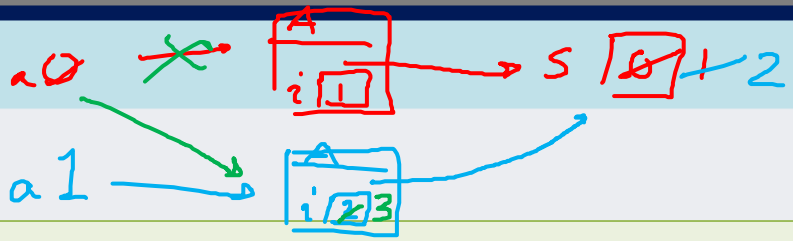
- Understanding abstractions will:
  - Let you write better, faster code by understanding and respecting limits
  - Make you a more powerful, flexible programmer able to switch between abstractions
  - Give you a holistic view of the computational stack and insight into what is actually happening

Let's practice – with iClickers!

- On Canvas, select iClicker Sync
  - Follow the instructions to login to CPSC 213 2022W1 5pm on iClicker Cloud
  - You can use your laptop, cell phone, tablet, etc.
  - Please make sure that your student number is correct in your account!

click 'B' when ready

iClicker 0.1




```
public class A {
  static int s = 0;
  int i;
  public A() {
    s = s + 1;
    i = s;
  }
  public static void main(String[] args) {
    A a0 = new A();
    A a1 = new A();
    a0 = a1;
    a1.i = a1.i + 1;
    System.out.printf("%i, %i", a0.i, a1.i);
  }
}
```

In the code on the left, what prints when `main()` executes?

A. `0, 2`

B. `1, 3`

C. `2, 3`

D. `3, 3`

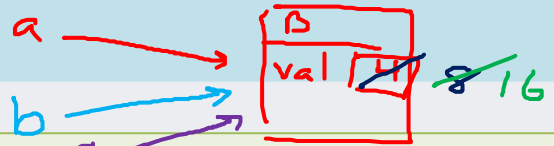
E. Something else

Draw a picture!

Inside the machine

- All state is stored in a single, main memory
- Each byte has an unique numeric address
- A variable is a name for a memory location that has an address, a size, and a value
  - Statically allocated variables: address is specified at compilation time
  - Dynamically allocated variables: address is determined at runtime
- A pointer is a variable whose value is a memory address

iClicker 0.2




```
public class B {
  int val;
  B(int val) {
    this.val = val;
  }
  int add(B x, B y) {
    this.val = this.val + x.val;
    this.val = this.val + y.val;
    return this.val;
  }
  public static void main(String[] args) {
    B a = new B(4);
    B b = a;
    B c = a;
    System.out.println( a.add(b,c) );
  }
}
```

In the code on the left, what is printed when `main()` executes?

A. 4

B. 8

C. 12

D. 16

E. 20

Draw a picture!

Inside the machine

- A new object is created only when the `new` keyword is called
- Object is identified ("named") by its memory address; a number
- Pointer variables are stored in memory too
  - Their value is the memory address of the/some object

- In the previous example, every pointer variable stores the same number (i.e. they all point to the same object)

Overall learning goals for 213

From the syllabus

- Be a better programmer
  - have a deeper understanding of the features of a programming language
  - understand in detail how your programs are executed
  - be able to more easily learn new programming languages
  - be able to evaluate design tradeoffs in considering languages most appropriate for solving a given problem

- Appreciate that system design is a complex set of tradeoffs
  - system may not have exactly one optimal answer
  - there are often many sub-optimal answers
  - tradeoffs exist in the hardware level, programming level, etc.
  - experience with tradeoffs prepares you to deal with tradeoffs in design in real world programming scenarios

Overall learning goals for 213

- Develop distinctions between static and dynamic components of programs and systems and be able to describe their implications
- Utilize synchronization primitives to control interaction in various situations including among processes and threads
- **Understand how computing systems work**

Approach via major topics

- Hardware context of a single executing program
  - CPU and main memory
  - Static and dynamic computation
  - CPU architecture for C / Java

- System context of multiple executing programs with I/O
  - I/O, concurrency, and synchronization
  - thread abstraction to hide asynchrony and to express parallelism
  - synchronization to maintain concurrency

How to succeed in 213

- Work hard
  - Current information plus work leads to new information
- Stay engaged
  - Be active in class, in labs, and on Piazza
  - You can learn by asking questions, and by helping to answer them
- Stay on schedule
  - Concepts we cover are very likely to be relevant for later concepts
- Get help as soon as you need it
- Learn from your mistakes