Introduction

COMP 401, Fall 2018

Lecture 1

Why Program

- Think about how you would complete this sentence:
  - People program in order to _____.
- Share your answer with the people sitting on either side of you.
  - Introduce yourself if you don't know them.
  - Find something you have in common.
    - Example: "We both prefer sativa over indica"
  - Find something you don't have in common.
    - Example: "I'm totally Slytherin, she's a Hufflepuff"

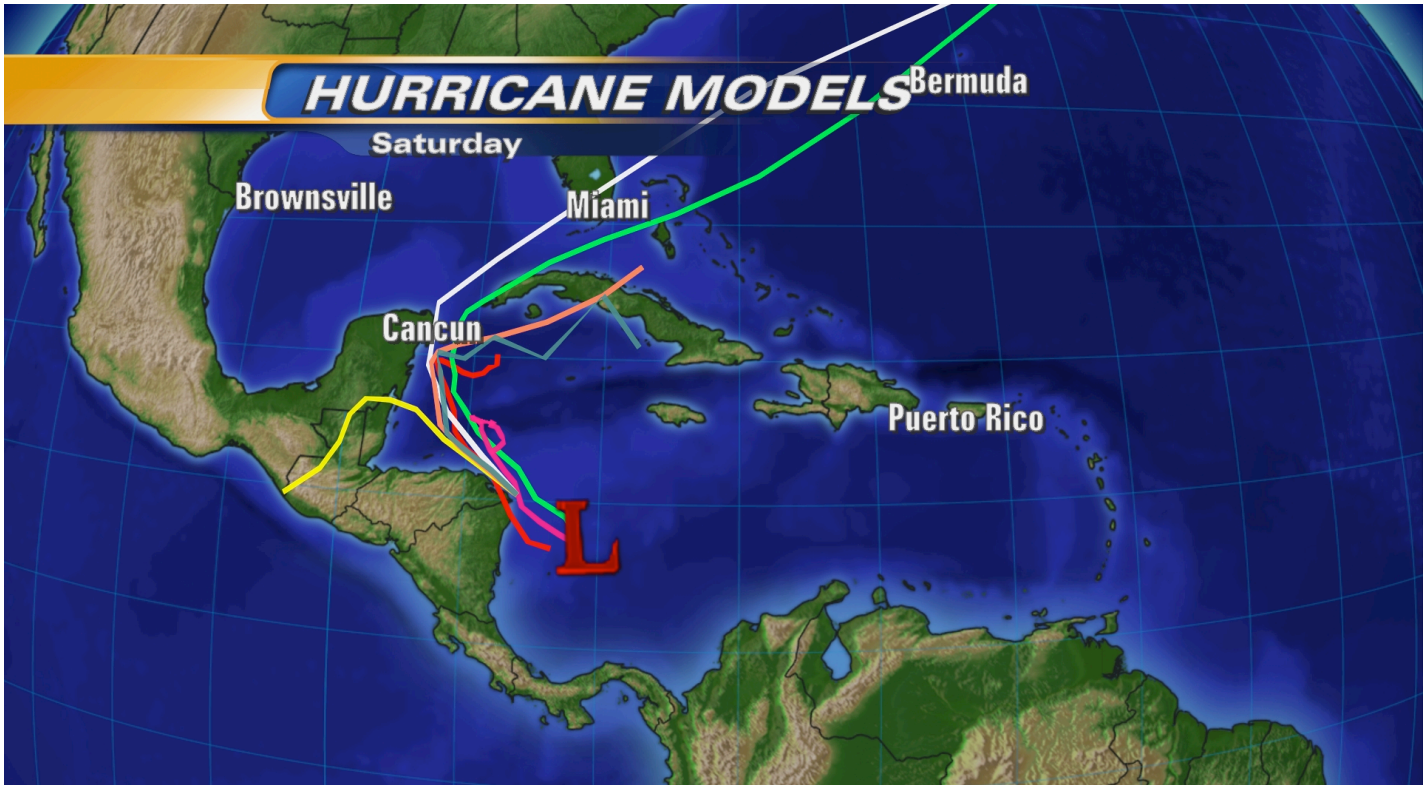
Why Do We Program?

- To answers questions
  - What is the average height of the people in this class?

```
double[] heights = readHeightData();
double sum_of_heights = 0.0;
for (int i=0; i<heights.length; i++) {
    sum_of_heights += heights[i];
}
double avg_height = sum_of_heights / heights.length;
```

Why Do We Program?

- To understand our world



Why Do We Program?

- To discover

Why Do We Program

- To make money

Why Do We Program?

- To create new worlds

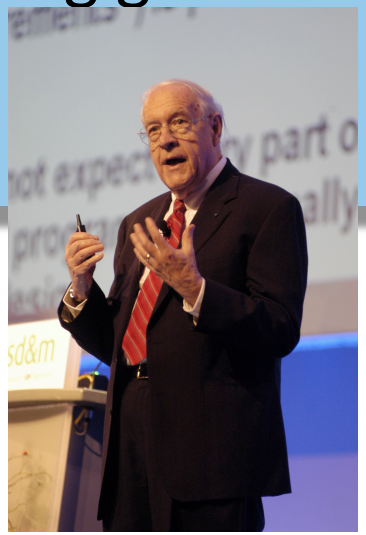
Why Do We Program?

- To evolve

What Is Programming?

The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures.

-- Fredrick P. Brooks Jr.



Programming is Magic

How do we understand the world?

The Evolution of Programming

- Simple programs are programmed simply.
  - A program as a sequence of instructions.
    - Each instruction causes a change in overall state.
      - Translation: shit happens
    - Named variables used to store values.
    - May employ loops and conditional execution
      - while loop
      - if / then

The Evolution of Programming

- As a program becomes more complex…
  - Refactor common operations into parameterized functions or procedures.
    - Can be generalized and put into libraries to be used across different programs.

The Evolution of Programming

- As complexity continues to grow…
  - … developing a program as recipe-like sequence of instructions becomes increasingly difficult.
- Need to be able to develop and understand our programming more like the way we understand the world.
  - As a set of interacting abstractions.
    - With different parts of the program operating at different levels of abstraction.
  - This is object-oriented programming.

Object-Oriented Programming

- Programs construct and manipulate software “objects”.
  - Each object is associated with data and a set of functions or procedures that operate with that data.
- Objects are defined by their “class”.
  - Each class represents an abstraction.
  - Abstractions often layered on top of each other and/or composited into more complex abstractions.
  - A key challenge is developing the appropriate abstractions.
- The operation of the program is the result of creating objects and having them interact with each other.

Major Themes Of This Course

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism
- Delegation
- Design Patterns
- Asynchronous Programming

A little bit about myself

- Background
- What's with the name?
  - And why can't I remember yours

Course Management

- Sakai
  - As a gradebook only
- Piazza
  - Course notes, resources, Q&A, and all of other course related communication.
  - http://piazza.com/unc/fall2018/comp401001/home
    - Not connected to official university rolls, so you need to go here and sign up.
  - Can post anonymously to classmates…
    - ... but instructional staff sees all.
- Textbook / Resources
  - No required text.
  - Will be posting links to on-line materials.

Prerequisites

- This should not be your first introduction to programming.
  - COMP 110 or 116
  - High school
  - Hobbyist / Professional experience
- If you have never programmed in Java before
  - Links posted on Piazza:
    - Oracle Java Tutorials
    - Java for Python Programmers
  - Many others.
    - Your goal should be to get up to speed on basic syntax and control structures.
- What programming language are you now *most* familiar with?
  - PollEverywhere at PollEv.com/comp401
  - By phone, text COMP401 to 22333

Etiquette

- Empathy and The Law Of Large Numbers
- Your screens are part of the environment.
- No conversations please

Grading

- 40% assignments
- 15% in-class participation / recitation exercises
- 20% midterm
- 25% final
- Computing your grade
  - Assignment points on a straight percentage.
  - Recitation exercises, in-class participation
    - Total number of points not known for sure until end of course.
      - Recitation exercises will be at least 80% of possible points.
    - Most will be simple participation (i.e., not “graded” per se)
    - Total points for the semester will be discounted by 20%
      - This is to account for absences.
  - Exams will be curved.
    - Exact curve determined by statistics of class distribution.

Letter Grade Mapping

Letter	Percentage
A	94% <= wa
A-	90% <= wa < 94%
B+	86% <= wa < 90%
B	83% <= wa < 86%
B-	80% <= wa < 83%
C+	76% <= wa < 80%
C	73% <= wa < 76%
C-	70% <= wa < 73%
D+	65% <= wa < 70%
D	60% <= wa < 65%
F	wa < 60%

Assignments

- Assignment schedule on Piazza
- Initial Class Survey
  - 5 points
  - Mandatory
    - You may be dropped from the course if you don't fill it out.
  - Do this ASAP, no later than Thursday night.
- 10 Programming Assignments
  - 10-30 points each, 135 points total
  - Some of the assignments will build on top of each other.
    - You should keep working on an assignment until you do get it working.
    - Late assignments will only count for 50%
      - If you get some of it working on time and then continue to work on it and get more of it working later, then you get full credit for what you did get working and 50% for any additional points earned.
- Coding Style
  - 10 points
- Total: 175 points

Autograder

- Not up yet, but will be soon.
- Must use campus VPN to access from off-campus.
- Grades every hour or so.
- Feedback from grading script may be hidden for some assignments.
- Counts best score and deals with late points appropriately.
- If you think your program is right but autograder gives you a zero, the problem is usually in your program.
- Detailed instructions and tips/tricks will be posted on Piazza.

In-Class Participation

- Google Forms and/or PollEverywhere
  - UNC's PollEverywhere now uses Single-Sign On with your onyen.
  - Go to poll.unc.edu to login, instructions, FAQs

Exams

- Midterm
  - Tuesday, 10/16, 7:00 – 9:00 PM
  - Location TBD
    - NOTE: This is not during class
- Final Exam
  - Saturday, 12/8, 12:00- 3:00 PM
  - Location TBA (likely in classroom)

Honor Code

- Exams and participation exercises.
  - No outside help given or accepted is allowed.
- Assignments
  - Allowed to collaborate in developing, debugging, and/or discussing solutions.
    - Must write your own code.
    - Report any significant collaborations in a comment.
      - Provides immunity to being flagged by code comparison tools.
    - May be asked to explain and / or recreate your solution.

Our LA Team and Office Hours

- Head LAs
  - Katherine Griffin
  - Nikhil Komirisetti
- Other LAs
  - COMP 227: Effective Peer Teaching In Computer Science

Recitations

- Hands-on exercise led by head LAs
  - Will generally either include a quiz or some sort of deliverable to be turned in.
- 9:05, 10:10
  - FB 007
- 11:15, 12:20
  - Twice as big, spread out between FB 007 and SN 011
- 1:25, 2:30
  - SN 011
- Try to go to the recitation you are scheduled for.
  - But if necessary, you can go to any of the recitations on a one off basis.
  - If you need an official recitation switch, fill out form posted on Piazza.

Recitation This Week

- Optional
  - No material, quizzes, or exercises
  - Drop in troubleshooting for installing Java and Eclipse.
  - Opportunity to meet the head LAs

Absences

- We will not keep track of absences.
  - Excused or otherwise. Just be absent.
  - 20% discount on lecture participation/recitation points intended to account for this.

Adding The Course

- Attendance form.
  - http://www.cs.unc.edu/~kmp/comp401fall18/fdoc
  - Form asks about registration status.
  - We will figure out how many more we can accommodate by sometime Friday
- If you are going to drop, please do so soon.
- If you are able to attend the other section, you should sign up for that one.
  - It is very unlikely that students who are able to take the other section will be let in to this one.

OO Programming Languages

- Almost all modern programming languages support some sort of object-oriented programming.
  - Not all of them will do so in the same way.
    - There isn't just one way of being object-oriented.
- We'll be using Java
  - Some things will be conceptual and will apply broadly to many other OO programming languages.
  - Some things will be specific to the object-oriented mechanisms provided by Java

Some Characteristics of Java

- Strongly typed
  - Variables must be declared with a type specified.
- Compiled into "byte code"
  - Write once, run everywhere
- Primitive value types vs. reference types.
  - In some OO languages, everything is an object.
  - Not quite true in Java.
    - Some basic data types are defined entirely by their value.
      - Numbers (integers, floating point), booleans, characters
- Garbage collected memory
  - Memory is automatically allocated when objects are created.
  - Memory is automatically reclaimed when no possible reference to an object can exist.

Installing the Java SDK

- Java SE
  - Latest released version is 8
  - You want to install the Software Development Kit (SDK)
    - Note this is more than just the Java Runtime Environment (JRE)
  - Available from Oracle at:
    - http://www.oracle.com/technetwork/java/javase/

Eclipse

- Java-based Integrated development environment (IDE)
  - Not necessarily the best, but reasonable, free, and fairly straightforward to use.
  - Available at http://eclipse.org

GitHub Classroom

- Will need a GitHub account.
  - http://www.github.com

Preconfigured Virtual Machine

- VirtualBox
  - Free virtual machine hosting platform.

- Preconfigured virtual windows box will Eclipse, Java 10, and git installed available.

A Java Program

```
1 public class AverageHeightApp {
2
3     public static void main(String[] args) {
4       double[] heights = readHeightData();
5       double sum_of_heights = 0.0;
6
7       for (int i=0; i<heights.length; i++) {
8         sum_of_heights += heights[i];
9       }
10
11      double avg_height = sum_of_heights / heights.length;
12
13      System.out.println("The average height is: " +
14                         avg_height + " inches");
15    }
16
17
18    static double[] readHeightData() {
19      double[] height_data = {66.0, 72.0, 69.5, 68.2, 75.0, 64.5, 63.0};
20      return height_data;
21    }
22
23 }
```

You might want to drop the class if…

- … you didn't recognize the idea of an array of values and/or the notion of retrieving a value from the array by its index.
- … you can't recognize what in the program is acting as a variable.
- … you are unfamiliar with what a *for* loop is.
- ... you don't grock the notion of defining a function (or procedure) and/or calling a function (or procedure)
- … you can't write a program of similar complexity in some other language if not Java.
- The above are necessary conditions for success in this class, but may not be sufficient conditions for success.
  - Trust your own judgment, don't be afraid of a challenge.
  - Programming takes practice.