CS1004: Intro to CS in Java, Spring 2005

Lecture #1: Introduction

Janak J Parekh
janak@cs.columbia.edu

Yes, I know this room is crowded

- Unfortunately…
  - The University doesn't have a bigger room during this timeslot
  - We can't change the class time without me being lynched and mobbed
  - The CS department didn't anticipate the demand to make a second section
- So…
  - We're stuck here
  - Fortunately, the class is capped
  - For most classes, we should be okay space-wise
  - No notes today – just sit back (stand up?) and relax

What is this class?

- An introduction to Computer Science
- Required for all CS majors with no Java experience, and for some non-majors
- Two main components:
  - The *theory* behind Computer Science
  - Basics of Java, the programming language
  - As we'll see, a programming language is not equivalent to CS

CS1004 vs. CS1007

- CS1004 is an Introduction for those with no formal CS/Java training
  - Assumes only basic computer skills (email, web, mouse, brain)
  - Focuses on basic theoretical knowledge as well as basic Java fluency
- CS1007 assumes basic Java knowledge
  - If you know Java and/or took the AP CS exam, you're in the wrong room
  - Emphasis on more advanced Java and algorithmic skills
- If you have questions, ask me after class

Basic information

- Instructor: Janak J Parekh (janak@cs.columbia.edu)
  - Call me Janak, please
  - 10th year at Columbia (in various capacities)
  - OH: to be finalized once we get all our TAs; for now, ask questions right after class
- Class website: http://www.cs.columbia.edu/~janak/cs1004
  - Make sure to check it regularly
  - Only thing you need to write down
- No lab, like last semester
  - Two lectures a week, both by me

TAs and other resources

- Teaching assistants:
  - William Beaver (wmb2013@columbia.edu)
  - William Beaver (wmb2013@columbia.edu)
  - The rest are TBD (hopefully sooner than later)
- Class webboard
  - Since this class is large, the webboard will be an invaluable resource

Textbooks

- Multiple textbooks
  - Everyone must buy two textbooks (sorry!)

Schneider & Gersting,
Invitation to Comp. Sci., Java Version

Lewis & Loftus,
Java Software Solutions, 4th Ed., Java 5 Version

- Books can be obtained from Morningside Bookshop, SW 114th & Broadway; full details (including ISBN and online links) on website

Course structure

- 6 homeworks, 25 points each = 150 points
  - Roughly every 2 weeks
- 50 point midterm, 100-point final (open-book)
- Class participation (see next slide)
- In other words, homeworks are most important component of class
  - Learning programming is useless unless you actually do it hands-on

Class participation and attendance

- Attendance is expected; participation is beneficial
  - I'm *going* (ha!) to try and learn everyone's name
  - Useful for your grade at the end of the semester…
- If you miss class, you're expected to catch up
  - I'll post slides to the schedule page 12 hours in advance, but…
  - I *will* have examples and material that are class-only

Homeworks

- Will consist of written and programming parts
  - Programming part will be submitted online
  - Programming to be done on CUNIX (or at least tested there)
- Late policy: you are given 3 grace days during the semester
  - A late day is exactly 24 hours
  - Can use up to two on any individual homework
  - After late days used up, late submissions will *not be accepted*

Homework "0"

- Make sure you have a CUNIX account, preferably "extended"
  - Most ugrads get this automatically for "free"
- Make sure you can log into it
- Thursday tutorial will cover most of the topics you need to know in how to work with it
- Free AcIS hands-on lessons as well – check the class's homepage

Cheating

- Plagiarism and cheating: unacceptable
  - You're expected to do homeworks *by yourself*
  - http://www.cs.columbia.edu/academics/honesty -- especially the end of the page
  - Automated tools to catch plagiarizers
    - http://www.cs.berkeley.edu/~aiken/moss.html
    - I caught five students last semester
    - Moving stuff around, renaming, etc. doesn't help
- Results: instant zero on assignment, referral to academic committee
  - Columbia takes dishonesty *very seriously*
  - I'd much rather you come to me or the TAs for help

Feedback

- The course isn't strictly set in stone, and I'm open to suggestions
- I can't promise I'll make your dreams come true, but I will take any constructive feedback seriously
  - Not just template-speak: ask my students from last semesters
- I'm here to help you succeed

Poll time!

- School
  - CC
  - SEAS
  - GS
  - Other
- Year
  - Freshman
  - Sophomore
  - Junior
  - Senior
  - Masters or later

Poll (II)

- Have you programmed before?
  - No
  - Yes (BASIC, VB)
  - Yes (C, C++, C#, Java)
- Have you used…
  - UNIX
  - Windows command prompt
- Planning to be a…
  - CS major?
  - CS minor?
  - Nothing much – just taking course out of interest?

What is Computer Science?

- Well, what *isn't* it?
  - Computer Science isn't (just) the study of computers: much of it was developed before computers were actually invented (logic and applied math).
  - Fellows, Parberry (p. 3): "Computer science is no more about computers than astronomy is about telescopes, biology is about microscopes, or chemistry is about beakers and test tubes. Science is not about tools. It is about how we use them and what we find out when we do."

Other misconceptions

- Computer Science is the study of how to write computer programs.
  - Programming is important, but it's ultimately a tool *once you figure out what you want to do*
  - Figuring out what you want to do is what's the key (design vs. implementation)
  - There's no "one" right programming languages, besides
- Computer Science is the study of the uses and applications of computers and software
  - Why would you be in this class, then? You should all know how to use Microsoft Word at this point
  - Rather, how do we *design a word processor?*

So what is it?

- We can ask Google: http://www.google.com/search?q=define:Computer+Science
- I like this one best: "The systematic study of algorithmic processes that describe and transform information: their theory, analysis, design, efficiency, implementation, and application."
  - "Information age": we're presented with tons of information, and need tools to help organize it and manipulate it.

Book's definition

- Gibbs, Tucker in '86:
- "Computer Science is the study of algorithms, including:
  - Their formal and mathematical properties
  - Their hardware realizations
  - Their linguistic realizations
  - Their applications"
- That sounds immensely boring, but hopefully you won't find it so

What's an algorithm?

- Dictionary: "n. *A procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation*, broadly: *a step-by-step method for accomplishing some task*."
- In other words, we'll take problems we want to solve, break them down into steps, and realize them with a computer
- This course will help you to build those problem-solving skills

Who cares?

- "I'm taking this class because I have to know how to write code."
- "I'm taking this class because my advisor said I have to and I need an A."
- Several reasons:
  - Rising importance of computers in the world (and for your job)
  - A good coder does *not* necessarily make a good programmer or good computer scientist
  - Theory does have some practical implications (e.g., network security, what's a blue screen, etc.)
  - Brainteasers…

So what are we going to do?

- Study *algorithms*
  - Not necessarily as intuitive as you may think
- Study (the basics of) *hardware:* how do we represent these algorithms, and on what do they run?
- Study *programs/software*
  - A program is machine-compatible representation of an algorithm, written in a *programming language*

Abstraction

- While we're studying all this, maintain the fundamental principle of abstraction
- What is abstraction?
  - http://www.google.com/search?q=define:abstraction
  - "Abstraction means ignoring many details in order to focus on the most important elements of a problem."
  - At any given time, we focus on one aspect of a problem, and abstract away the details of others
  - Lets us build a "big picture" of Computer Science, brick by brick

Topics we'll cover

- Programming language basics
- Bottom-up approach to the computer
  - How is information stored and manipulated in hardware?
  - How do you tell the hardware to manipulate information?
  - How do you run this software in a reasonable fashion on a hardware?
- We'll build algorithm and problem-solving skills
- Finally, we'll look at some interesting directions for Computer Science
  - AI: the "future"?
  - Computation theory: what makes a computer a computer from a theoretical perspective?

Meshing theory and practice

- Let's look briefly at the syllabus
- I've put lots of effort into synchronizing the content of the two books
- "Survey" of topics that CS offers
  - Eventually, you'll want to take specialized classes in *each topic*
  - You're *not* going to write an operating system right now

Let's start thinking…

- You've got a five quart jug, a three quart jug, and a lake. How do you come up with exactly a gallon of water?
  - This is (was?) a brainteaser asked at Microsoft interviews
- Let's model this as (x,y) where x == # of quarts in five-quart jug, y == # of quarts in three-gallon jug

Something more pragmatic, perhaps?

- Given a map of the NYC subway system, design an algorithm that finds the "optimal route" between two stations
  - This is not quite so easy, and you're not going to know enough to do this in this class
  - But we can think about it conceptually: got any ideas?
  - http://www.mta.info/nyct/maps/submap.htm

Something simpler?

- Given 10 numbers, sort them
  - Easy, you say?
  - Sort 100 numbers
  - Sort 1,000 numbers
  - Do it fast

Being a good programmer…

- Takes more than knowing how to write code
- It takes the ability to take a problem and break it down into small enough steps to write code that solves it
- It takes the ability of knowing enough of the field (and the language) to know what a "step" is
- Hopefully, that's what you'll learn this Spring

Before we go any further…

- Let me prove that I, unlike most professors, know how to program
  - Both myself and the TAs know Java well, so don't hesitate to ask for help
- First program: always "Hello, world!"
- We'll go through the details next week

Next class

- UNIX tutorial
- Check out the homepage for AcIS class information