CS W3134: Data Structures in Java

Lecture #1: Introduction

9/7/04

Janak J Parekh

What?

- This is the second Computer Science class in a two-course sequence for non-majors
  - As opposed to CS3137, this one more focused on "practical"
  - Minors: special curriculum info session 4-6pm, Carleton
- The first class introduced fundamental Computer Science concepts; this class will build on them and continue to develop your programming and algorithmic skills
- Prerequisite: COMS W1004 or equivalent, i.e., basic fluency in Java and CS
- If unsure about anything, find me today right after class!

Who?

- Instructor: Janak J Parekh (janak@cs.columbia.edu)
  - Call me Janak, please
  - 10th year student at Columbia (sort of)
- TAs
  - William Beaver (wmb2013@columbia.edu)
  - Rachel Goldman (rg2020@barnard.edu)
  - Matthew Waymost (mw708@columbia.edu)

Where? When?

- Class will be held here, in 833 Mudd, TR 11-12:15
  - "Tell me if you're not here"
- Class website: http://www.cs.columbia.edu/~janak/cs3134
  - Lecture slides will be posted there
- Office hours
  - Mine will be held right after class or by appointment
  - TAs' office hours TBD

How?

- Textbook
  - Lafore, Robert. *Data Structures & Algorithms in Java, Second Edition.* SAMS, 2003.
  - I really like this book – very practical, lots of code examples, applets to demonstrate concepts
  - Available from Morningside Bookshop, SW 114th and Broadway

How? (II)

- Course structure: 300 points
  - 6 homeworks * 25 points ea. = 150 points
  - 50 point midterm
  - 100 point final
  - Occasional extra credit
- Class participation is important
  - I hate standing up here and talking nonstop
  - Board material is fair game
  - "Reasonable person principle"
  - Feedback!

How? (III)

- Homeworks: theory and programming parts
  - Important – they're worth 50% of the class
  - Submission, late policy
  - Computing environment: CUNIX
  - Please, don't try to plagiarize or cheat – you **will** get caught
    - http://www.cs.berkeley.edu/~aiken/moss.html
- Exams are open-book, open-notes
  - Midterm tentatively on 10/21

Poll

- Why are you here?
- School (GS, SEAS, CC)
- Level of Java knowledge
  - CS1004: With Prof. Aho? With me? No 1004?
  - Basic applications
  - Basic applets, AWT, Swing
  - OO: Subclassing, interfaces, polymorphism, inheritance, visibility modifiers
  - Java Collections: Vector/ArrayList, Hashtable/HashMap, etc.
- C/C++ knowledge
- **Don't worry if you don't know most of these**
- Java recitation/basic hands-on?

Why?

- What are the two primary things computers do?
  - Store information
  - Manipulate information
- Why do we need to know how? Doesn't Java have built-in data structures?
  - There's no one way of doing it
  - Each approach has advantages and disadvantages
  - Raw CPU power can't overcome inefficiency
  - Java "Collections" don't handle everything

Why? (II)

- Don't we need to know the problem beforehand?
  - Not necessarily
  - We want to develop a "toolkit" to be useable in the future
  - One fundamental concept makes it feasible: abstraction
- Abstraction
  - Fundamental concept in Computer Science, especially applies here
  - Lafore defines it as "considered apart from detailed specifications or implementation"
  - Car analogy

Abstraction

- We create a layered system
- *Abstract data types* as fundamental building blocks of information
  - What data types does Java support?
  - Primitive vs. reference data types
- *Abstract algorithms* as fundamentally useful to a broad range of applications
  - Data manipulation, sorts, searches
- You won't always have to design them, but you'll always have to use them
  - Understanding how they work, even under the scenes, is key in making your code work better

Example

- Music database
  - How can we represent this information?
  - What kinds of operations would we do on such a application?
  - What problems do we encounter with a naïve implementation?
  - Can we do better?
- Can an abstract knowledge of data structures and algorithms help?

What's out there?

- Data structures?
  - Arrays (sorted or unsorted), stacks, queues, linked lists, trees, hashtables, heaps, graphs
- Algorithms?
  - Insert
  - Search
  - Delete
  - Iterate
  - Sort
  - Recurse

Object-Oriented Programming and Java

- What is OO?
- How does OO help?
  - Improves abstraction
  - Allows code reuse
  - Access control to data: makes it more reliable – *encapsulation*
- Why do we use Java in a class like this?
  - OO is nice, but…
  - Java has no pointers
  - Strongly-typed
  - Garbage collection

What we'll be doing the rest of the semester…

- Learning about these data structures
- Learning about some of the algorithms for them
- Learning which is best when
  - Elementary analysis of algorithms
  - Take the real class if you want to know the details
- Becoming better programmers!

Homework & Next Time

- No "official" homework until next week
- HW0 posted on webpage – no submission
  - Intended to get you up to speed
- Get the book
- Next time: start looking at ADTs and OO design more closely, "refresher" on Java OO constructs