Data Structures in Java

Session 1

Instructor: Bert Huang

http://www.cs.columbia.edu/~bert/courses/3134

Session Plan

- Administrative overview
- Introduction to course content

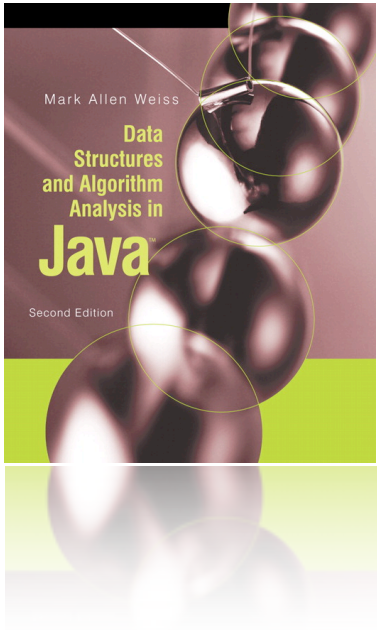
About the Course: Description

- Title COMS W3134; Data Structures in Java
- Lectures: Tuesday/Thursday 5:40-6:55 PM
- homepage: http://www1.cs.columbia.edu/~bert/courses/3134
- We'll study useful data structures, their applications and implementations. We'll gain intuition about designing our own

About the Course: Staff

- Bert Huang, 3rd year PhD candidate
  Office hours tentatively Wednesday 2-4 PM
  CEPSR/Schapiro Building 624
  bert@cs.columbia.edu
- TA: Nikhil Ramesh, UNI nf2241
  Office hours TBA

About the Course: Reading



- *Data Structures and Algorithm Analysis in Java, 2nd Edition* by Mark Allen Weiss.
  ISBN-10: 0321370139

About the Course: Resources

- Course homepage: http://www.cs.columbia.edu/~bert/courses/3134
- Courseworks: http://courseworks.columbia.edu
- Textbook Errata: http://users.cs.fiu.edu/~weiss/dsaajava2/errata.html
- Textbook Source Code: http://users.cs.fiu.edu/~weiss/dsaajava2/code/

About the Course: Prerequisites etc.

- COMS W1004, Introduction to Computer Science and Programming in Java (or equivalent)
- CompSci **majors** should be taking COMS W3137

About the Course: Grading

- 50% Homework Assignments (six)
- 20% Midterm Exam
- 30% Final Exam

About the Course: Academic Honesty

- You **must** read the Computer Science department's academic honesty policy listed at http://www.cs.columbia.edu/education/honesty/
- Additional Comments:
  - Plagiarism is easy to catch.
  - All homework and exams in *this class* are individual assignments. No collaboration.

About the Course: Expectations

- Attend class
  - Ask questions; slow me down
- Read assigned text
- Start homework early
- Write well and clearly
- Get help when you need it

Abstraction

- Stand on the shoulders of giants
- In practice: a well tested **class** should be treated as a black box with inputs and outputs, with no concern over implementation.
- In theory: a well tested **abstract data type** should be treated as a black box with inputs and outputs, with no concern over implementation.

Benefits of Abstraction

- Consider Java Strings
  - We use them all the time
  - How is the text in a String object stored?
  - When we call the length() method, how does it find the length?
  - How does it concatenate strings?

Abstract Data Types

- **Data structures** implement **Abstract Data Types**
  - ADTs are defined only as black box input and outputs
- ADTs vary in complexity.
  - E.g., bits*, ints*, arrays,
  - lists, stacks, queues, trees, heaps, hash tables, graphs

Array ADT

- You can:
  - insert elements into arrays by **index**
  - read elements by index
- You (typically) don't have to think about:
  - where is the data in memory?
  - how does the computer find the *i*th element?

Our dual role

- As programmers, it is good practice to shield our eyes and treat our black boxes as black boxes.
  - This yields easier design and cleaner programs.
- As computer scientists, we should understand the theory behind data structures
  - Helps us invent new structures, better understand when to use which ADT or implementation.

Homework 0

- http://spreadsheets.google.com/viewform?hl=en&formkey=dHE3c3V4X3E5SlFycFJDTWNybHN3bnc6MA..
- 1 percentage point "extra credit" survey
- Follow the link on homepage
- Due by next class

Homework 1

- Running time analysis theory
- Java refresher
- Collection data structure
- http://www1.cs.columbia.edu/~bert/courses/3134/hw1.pdf

Reading

- Course Website:
  http://www.cs.columbia.edu/~bert/courses/3134
- Academic Honesty policy

  http://www.cs.columbia.edu/education/honesty
- *Weiss* Chapters 1 and 2