CSC207.01 2013F, Class 22: OOD in Practice: Designing a List Interface

Overview

- Preliminaries.
  - Admin.
  - Amazon code ninja challenge.
- The design of ADTs, revisited.
- Exercise: Designing a list ADT.
- Quick notes on implementation.

Admin

- I've brought some swag back from GHC. We'll try to find an equitable way for each of you who did not attend GHC to get an item.
- I finished the Amazon Code Ninja challenge that I tried faster than anyone around me. I've tried to replicate it and I'll give you a few minutes to try it.
  - No, you may not compile and run the code.
- Just so you know, most of this week will be dedicated to in-class design problems, problems that we will do as a group. (Although maybe sometimes in small groups.)
- I'm booked solid doing a SHACS review all day today and part of tomorrow.
- HW6 is now reduced to the Dutch National Flag problem.
- Exam 1 should be distributed in rough form on Tuesday.
- Upcoming extra credit opportunities
  - Road to Rio, Tuesday 7:00 p.m., Natatorium.
  - CS Extras, Thursday: Graduate School in CS
  - Learning from Alumni, Thursday: Tony Stubblebine '00 - CEO at Lift
  - Codebreaker Friday night @ 7pm.
  - ???

Amazon Code Ninja Challenge

- There were a few. This is the one I attempted (and solved faster than anyone around me).
- Come up with an answer (without using the compiler) and show it to me.

The design of ADTs, revisited

- Think about the what, not the how
- Three or so steps in designing an ADT
  - Overall goal or philosophy

        - Arrays: Collections of data, indexed by sequential integers
            - Fixed size vs. Dynamic
            - Starting at 0 or startnig where you want
    - Applications/client code (use case)
    - Think about what procedures the ADT needs
- Once you've designed the ADT, you can think about implementation
    - Layout in memory
        - Big chunk of data
        - Small chunks of data with interlinked pointers
    - Fields
    - Implement methods
    - Find running time

Exercise: Designing a list ADT

- You've seen lists multiple times
    - Scheme lists in 151 and 208.
    - Linked lists in 161.
    - UshahidiLists in 207.
    - Everyday written lists outside of CS classes
    - ArrayLists in Java
- What's the big picture philosophy a list?
    - Collections of data
    - Ordered - one comes after another comes after another
        - All but one element has a successor
        - All but one element has a predecessor
    - Resizable / modifiable
    - Insert in arbitrary places
    - Intended for sequential access
- Things not necessarily in lists
    - Efficient access to "middle" elements
    - Support a fast "find" method
- Lists are resizable ordered collections of data that support sequential accesss (iteration)
- Categories of operations
    - insert/add
    - get / access sequentially
    - remove elements
    - big picture mutation - reverse
    - medium picture mutation - swap positions of two values; sort, find
        - vs. replace
    - toString
    - sublist

Quick notes on implementation