Administrivia

- Why it’s *not* \( n(n+1)/2 \) – we do one less comparison on the first step
  - If you want to see the slowness, run the applet with 100 bars…
- Who hasn’t started HW1?
  - Uh-oh….;)
  - Are you using the webboard?

Agenda

- Implement sort examples
  - Look at complexity metrics based on code
- Start looking at linked lists

Sorts

- By the way, look at applets
- Bubble (p. 85)
  - Sort pairwise repeatedly
  - Biggest placed each time
    - Outer loop starts at \( n-1 \) and goes down
- Selection (p. 89)
  - Search for smallest, swap with first
  - Search for smallest, swap with second
  - Outer loop starts with 1 and goes up
- Insertion (p. 95)
  - Take the next one, and put it into the existing sorted subset
  - Outer loop similar; difference is “slide” instead of swap
- By the way, this isn’t the only way to do these…

Sorts II

- Lexicographical comparisons?
- Stability of existing items?
  - Does your homework need a stable unordered array?
- Sidebar: Comparable interface
  - All you have to do is implement boolean compareTo(Object o)
  - Generally a good thing to program to, I prefer to book’s example
  - Arrays.sort()

Stacks and Queues

- Useful programmer’s tools, will encounter it in many places
- “Restricted access”: no index – only manipulate one item at a time
- More abstract – the underlying implementation is unimportant or not similar to the structure

Stacks

- Basic operations
– Push
– Pop
– Peek
– “LIFO”
• Analogy: mail basket
  – Not as rigorous as a real stack, of course
• Another analogy: life
  – Conversations
  – Workday
• Extraordinarily simple!

8 Array-based stacks
• Limited size; ways to get around this
• Decoupled from array index!
• Very simple to implement
  – Keep top variable, initialized to -1
• Boundary conditions?
• Complexity bounds?
  – Apart from simplicity, biggest reason to use

9 Next time…
• Reasons to use stacks
• Queues
• Arithmetic expression parsing