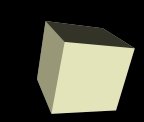
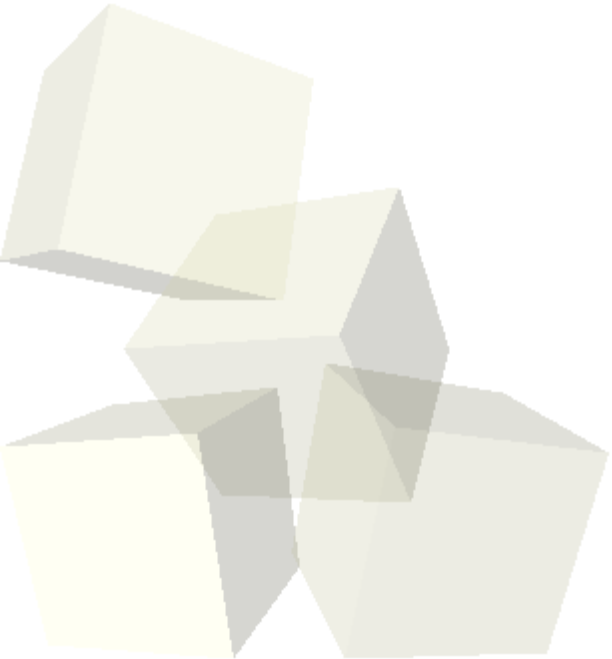


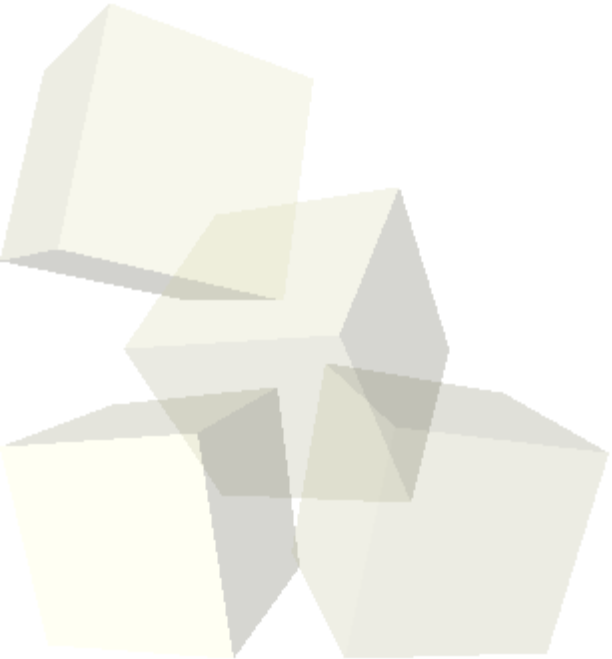
Searching and Sorting

2-7-2006



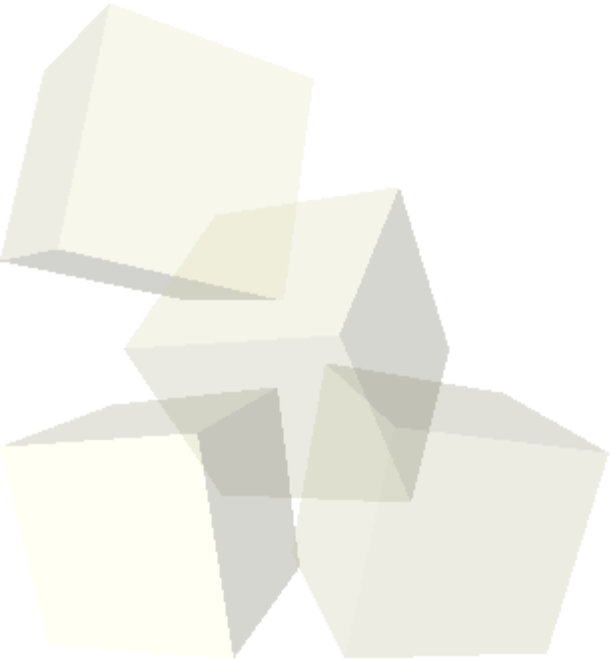
Opening Discussion

- Do you have any questions about the reading?
- Do you have any questions about the assignment?



Searching

- What are the two main methods of searching for objects that you could employ? When can each one be employed?
- How did they make it so their code could search through any type of array?

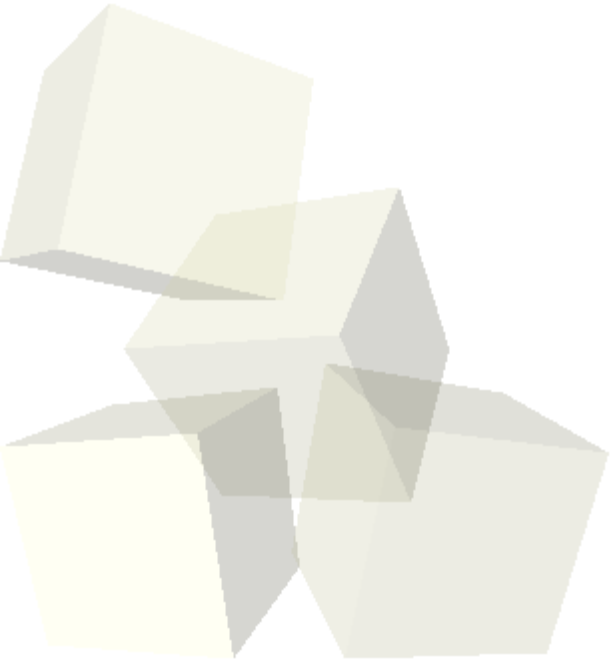


Comparators

- There is an alternate way to make searching and sorting code that will work with any type. The method in the book only works with data that is Comparable. Instead you can write the functions to take an extra argument that is a Comparator. Comparator<E> is an interface with one method: compare(E e1,E e2). Notice it is a generic interface.
- Not only does using a Comparator allow you to sort data that isn't Comparable, it also allows you to easily change the sort order. The book creates a type contact that sorts by last name. What if you wanted to sort by first name or phone number?

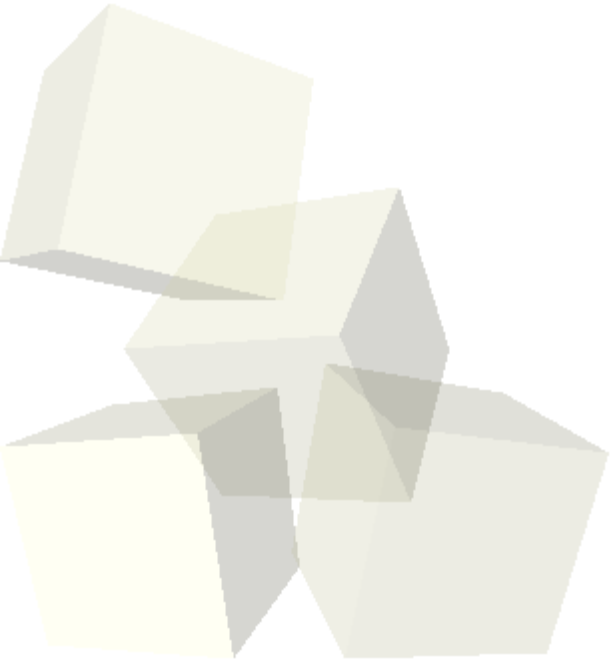
Code a Search

- I want you to write a linear search method that uses a comparator. Put it in your ArrayUtil class we created last time.
- In practice you could use Arrays.binary search.



A Non-Recursive Binary Search

- Your book uses a recursive version of binary search. This is a perfectly valid approach and you should all try to understand how it works. I want to show you a version that works with a loop just to show it can be done and some of the tricks I put into such code to make sure it works well.



Sorts

- What were the three “slow” sorts in your reading? I call them slow sorts because they are all $O(n^2)$.
- A formal definition of the O notation is as follows. A function g(n) is O(f(n)) iff

$$\exists n, c : \forall m > n, c * f(m) > g(m)$$

- I want you to write one of the sort algorithms using a comparator. You can pick which one. Bubble sort is the simplest one to write but it is also the least efficient.
- In practice you will likely use Arrays.sort.
- Let's look at code for those three sorts and instrument our comparator to see how many swaps they do.

Shell Sort

- Arrays.sort uses either quicksort or merge sort, depending on the data type. We will talk about those later when we do recursion. There are faster sorts you can write without using recursion.
- One example of that is Shell sort. Shell sort is also called the decreasing gap sort. It seems rather magical because do insertion sort multiple times with smaller gaps each time and you actually come out with a sort that is faster than a single insertion sort.

Minute Essay

- Remember that your design for assignment #2 is due today. It should include every class that you are going to write for this assignment and every public method. All of those need to be documented properly with a description of what they do.