Sorting and Searching

2/10/2009

Opening Discussion

- Let's look at solutions to the interclass problem.
- Do you have any questions about the reading?
- Do you have any questions about the assignment?
    - Let's talk about how to submit the assignment.
- How to write replaceAll.

Finishing ArrayMap

- Last class we started the code for a class called ArrayMap that we were going to use to make our CommandProcessor more flexible.
- Let's work on finishing that code today.

Sorting and Searching Arrays

- These are topics that you should have talked about a fair bit in PAD1 so I'm not going to lecture on them much now.
- Instead, we'll write some code that uses arrays and these concepts.

Order

- A function g(n) is O(f(n)) iff

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

- Let's look at what this means graphically.

Polymorphic Sorts

- One of my motivating examples for polymorphism was a sort. In C you have to write a separate sort for every type, or you have to do some very odd stuff. In Java we can write polymorphic sorts of object types in at least two ways.
- You can write a sort/search that only takes subtypes of Comparable.
- You can write a sort/search that works an any Object, but that also takes an object of type Comparator.
- I prefer the second method as it is far more flexible.
- The java.util.Arrays class contains some utility methods.

Write a Sort

- Let's write a method that uses one of the sorts you know to sort any object type. Try to make this a generic method so that it will be type safe. You can put it in a class called ArrayHelper.
- Let's make it so our comparator counts how many comparisons are made so we can see what sorts are best.
- If we have time, we can write a search as well.
- This is something we can integrate into our command processing class. We can write a sort command that takes two arguments: sort type and number of elements.

Minute Essay

- What sorts do you remember from PAD1? Explain how one of those sorts works.
- Remember that quiz #2 is on Thursday.
- Interclass Problem – Write the best sort you know how to write in a polymorphic way using a comparator. Compare its performance to Arrays.sort.