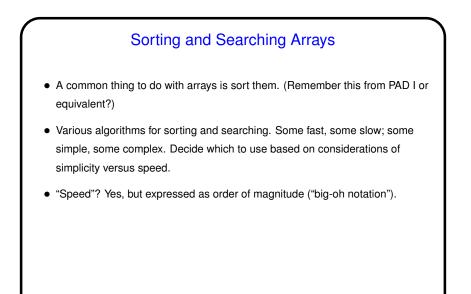


Slide 1

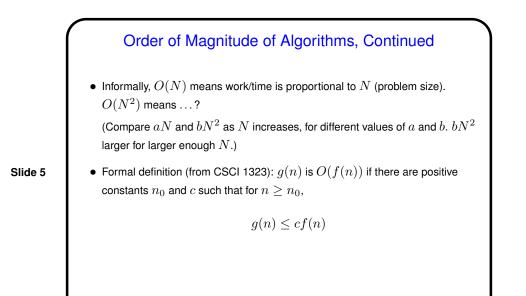
Homework 2 Code Eclipse will suggest adding a variable called serialVersionUID to some of your classes. Do that. (Notice there's one of these in some of the provided code.) Value can be anything. We will talk later about what this means and how to make use of it. Notice that x/y coordinates of framework are opposite of row/column. getSize() in screen class should return width by height. To confirm that your code works: Start the game, and verify that the playing field is what you defined (dimensions, plus appearance of blocks — for now, solid colors are okay). Try running the screen editor (directions in "project description" document). If it comes up, and shows all the kinds of blocks you defined, all is well. (Actually it doesn't have to do that if you don't plan to use it — it just has to not crash.)

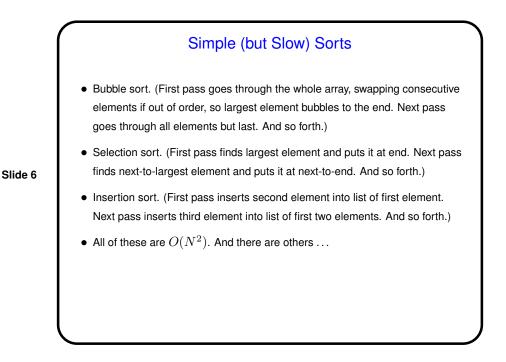


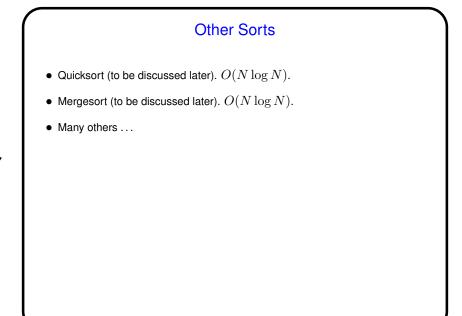
Order of Magnitude of Algorithms
Idea is to estimate how work (execution time) for algorithm varies as a function of "problem size" (e.g., for sorting, size of array). (Similar idea can be applied to how much memory is required.)
Usually do this by counting something that represents most of the "work" in the algorithm and varies with problem size (e.g., for sorting, how many comparisons).

Slide 3

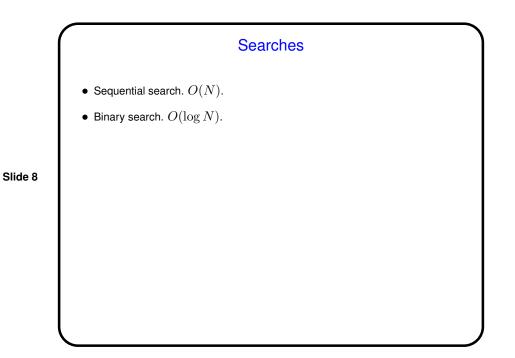
Slide 4

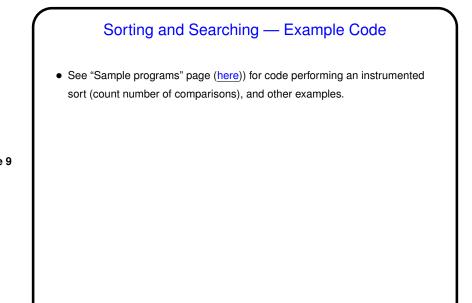






Slide 7





Slide 9

Slide 10

Sorting and Searching Arrays in Java Writing your own sorting routines is pedagogically useful, but in practice you would probably use something from Java library. Arrays class has some useful methods. One thing that's nice about Java is "polymorphic sorting"; can sort objects of any class that implements Comparable. Can also provide, when you call Arrays.sort, a Comparator that defines the ordering you want. Example: case-insensitive sorting of strings.

