Slide 1

Administrivia

- "Open lab" hours 2pm to 4pm Tuesdays and Thursdays in HAS 329.
- Reminder: Homework 2 design due today (11:59pm), code Tuesday.
- Reminder: Quiz 2 Tuesday.
- Career networking event ("Making Connections") today, starting at 5:45pm in the Great Hall. Sounds like it might be worthwhile.

Slide 2

Homework 2 — How to Approach Defining a Class

- What methods do I need? If implementing an interface, you at least need the methods in the interface. May want additional methods. If making a subclass, remember you automatically inherit all methods from superclass. Can override them and/or provide additional methods.
- What variables do I need to implement the needed methods? e.g., if defining a `Rectangle` class that has a `getArea` method, probably need either area or width and height.
- The class where this advice will be most relevant is the one implementing the `Screen` interface. You will need to represent your 2D grid of blocks and a list of entities. What kinds of variables would be good? (Look at the game framework API for hints about the list.)

Slide 3

Homework 2 Code — Some Tips

- 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.)

Slide 4

Sorting and Searching Arrays in Java

- As mentioned last time, 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.
- Examples . . . .

Slide 5

Concurrency Basics

- Textbooks on operating systems talk about "processes" — "threads of control" executing "concurrently", i.e., at the same time (in fact or in effect).

  Each is a sequence of steps, like the (sequential) programs you've written.
- How does it work? Conceptually, all processes not waiting for something (such as I/O) run at the same time. Operating system basically simulates one CPU per thread, with real CPU(s) switching back and forth among them.
- This turns out to be a good mental model for managing applications, and activities of the O/S itself. It also means you could get better performance with more than one CPU/core — can potentially have more than one thing actually running at the same time.
- But there are some potential pitfalls, involving interaction among processes.

Slide 6

Processes Versus Threads

- Two basic ways to implement this idea of concurrent execution — "processes" and "threads".
- "Processes" don't (usually) share memory, and must communicate in some fairly restricted way.
- "Threads" do share memory, which is convenient but has potential pitfalls ("race conditions").

Slide 7

Minute Essay

- Homework 2 – what was difficult? interesting? educational?