Slide 1

Administrivia

- First quiz will be Thursday in class. 10 minutes at the end of class, 10 points, should be low-pressure. Open "book" and notes — meaning that you can use your own notes, the online textbook, the Java API, and anything on the course Web site.
- Due dates for Homework 1 posted. First phase ("design") due a week from today, code two days later.

Slide 2

Tools, Revisited

- I've been using BlueJ for examples in class, since it's a better tool (IMO) for people new to Java than a more-professional(?) IDE.
- Starting today, though, I'll use Eclipse. (Short demo.)

Slide 3

A Little About Homework 1

- What you will be turning in for the "design" phase is mostly a description of your game.

  You're not committing yourself to anything at this point, but try to be as detailed as you can — so I can try to spot potential trouble. Also good to think in terms of a basic design (not too ambitious) plus extras. Keep in mind that what you do has to fit into an existing framework. (That's actually one of the pedagogical goals.)

- What you will actually turn in is HTML documentation of your planned game's main class — put it in your `Local/HTML-Documentation` and send me mail saying "ready to be graded". (Complete instructions in homework writeup.)

Slide 4

A Little About Inner Classes

- As mentioned earlier in passing: Java classes can contain classes ("inner classes") as well as variables and methods.
- Inner classes can be named, local, or "anonymous". We'll see good examples of the first and last later. For now, just realize that this is possible.

Slide 5

"Generics" in Java

- Java library includes classes for collections of things (`ArrayList`, e.g. — like an expandable array). Originally, could put any kind of `Object` in one of these. Nice, except that then there's no way to know anything about types of objects inside except by using reflection (*much* later, if at all) or `instanceof` operator. Must also use explicit casts to do much with objects retrieved from collection.
- So Java 1.5 (a.k.a 5.0) introduced "generics" — Java's answer to C++ template classes, though not exactly the same. Idea is to allow you to specialize a collection — so, an `ArrayList` of `Integer` objects only, or an `ArrayList` of `Account` objects only, etc., etc. Syntax uses angle brackets, e.g., an `ArrayList` that can hold only `Account`s:

```
ArrayList<Account> list = new
ArrayList<Account>();
```

Slide 6

"Generics" in Java, Continued

- Used extensively in the game framework (see API for examples).
- (Example(s) as time permits.)

Slide 7

Other Features of Interest

- Enumerations — useful when you want to represent something that has to be one of a fixed number of choices. C allows something similar, but not as nicely packaged.
- "For each" loops (no explicit loop counter).
- `Scanner` class that makes reading from standard input easy.
- (Examples as time permits.)

Slide 8

Minute Essay

- Is the reading and what we do in class making sense, or do you have questions?