Slide 1

Administrivia

- Reminder: Homework 1 design (description of game) due at 11:59pm today.
- If you didn't take CSCI 1320 here, see "Useful links" page (here) for suggested reading about Linux and the command line (at the bottom of the page).

Slide 2

More Administrivia

- A little more about rebooting (or not) the machines in HAS 340!

  If a previous user has left a machine in the "locked by screensaver" state, you can bail out by pressing control-alt-backspace to restart X (the graphical subsystem) without disturbing background processes.

  If you log out from the "System" menu, it might be easy to shut down by mistake. Can put an icon on the task bar for logout to avoid this.
- Prox card access should be enabled now, so you should be able to get into the labs after hours. (Details in mail to CSMajors mailing list — ask me if you didn't get it.)

Slide 3

Java Basics, Continued — Control Structures

- Most control structures are the same as C — `if` , `while`, `do`, `switch`, `for`, etc. Also a simplified `for`, as of Java 5.0 (a.k.a. 1.5). More about it later.
- Also have "exceptions" — a way to deal with unusual or error conditions, break out of current flow of control. Can be "thrown" and "caught" (or not caught, in which case the program crashes). More about them later.

Slide 4

Arrays, Briefly

- Syntax is like C, except for explicit `new`:
  - `int[] x = new int[10];` creates 10 integers.
  - `String[] args = new String[20];` creates 20 *references* to strings.
- Arrays are "first-class" objects, with `length` variable.
- Java checks for out-of-bounds array references.

Slide 5

Miscellaneous Other Stuff

- No operator overloading (except "+" for `String` class).
- On reference variables, = and == operate on references, not objects. (So, you may instead want copy constructors or `equals()`.)
- No C-style strings, but a `String` class.
- A little about packages, and Java "generics" (new with 5.0), later.

Slide 6

Example

- Example — `Account` class.
- (This example, and most other code from class, will be on the Web, linked from the "Sample programs" page (here).)

Slide 7

UML Class Diagrams

- "Unified Modeling Language" — formal graphic representation of software analysis and design.

  Many types of diagrams, some of which you'll probably encounter in other courses. Tools exist for drawing them, but worth noting that they were designed to be whiteboard-friendly.
- We will mainly use class diagrams:
  - Box representing a class has name, attributes, operations.
  - Subclass points to its superclass (represents the path to follow to figure out inheritance).

Slide 8

Inheritance (Short Version)

- Given a class, it can be useful to define specialized versions — "subclasses".
- A subclass inherits attributes and operations from its superclass (which can in turn have a superclass . . .).
- Subclasses also form "subtypes" — e.g., if `CheckingAccount` is a subclass of `Account`, can use a `CheckingAccount` anywhere we need a `Account`.

Slide 9

Polymorphism (Short Version)

- "Many shapes" — something that works with many types.
- E.g., a function that works on `Account`s should work on `CheckingAccount`s, `SavingsAccount`s, ...

Slide 10

Minute Essay

- Make your best try at writing a method for our `Account` class that computes one month's interest and adds it to the balance. Assume `interestRate` is the monthly interest.

  (For minute essays where there's a "right answer", it will be in the final version of the notes online.)

Slide 11

Minute Essay Answer

- See the version of `Account` on the "Sample programs" page (here).)