Slide 1

Administrivia

- Deadlines for Homework 2 have been moved (one class period later).
- Comments on project design mailed earlier today. Comments on code coming soon, plus grade.

  Please follow the conventions for class and variable names: Class names start with a capital letter, variable names don't.
- "Shapes" example from lecture is now available via "Sample programs" page. To note about comments:

  At least a brief comment for every class and every method.

  Comments describing parameters and return values, unless it's totally obvious.
- Together is giving some students problems. My experience is that it generally is okay *if* you don't try to do anything else while it's starting up.

Slide 2

Recap — Classes and Objects

- Objects are a "nice" way of packaging together related data and code — a little like C `struct` but with code too.
- A class is a template for making objects — defines variables (one copy per object, unless static) and related functions ("methods").
- Non-static methods operate on objects — must have an object to apply them to, which acts like a hidden parameter to the method.
- Static methods don't have this hidden parameter.
- Java variables are either "primitives" (like C variables) or *references* to objects. Objects are created only with `new`.

Slide 3

String Class

- Recall — no C-style strings (arrays of characters) in Java. Instead, `String` class, similar to C++ STL `string` class or similar.
- To see what's available, look at the API . . .

Slide 4

String Class, Continued

- In general, no operator overloading in Java, with one exception — "+" for strings.
- To compare two strings, "==" is rarely what you want. Instead, use `equals`.
- Strings are "immutable" — once created, can't be changed. (Why? allows them to be safely shared.) Methods you would think change the value return a new string.
- Use `StringBuffer` if you need something you can change, or for efficiency.

Slide 5

Minute Essay

- None — quiz!