Slide 1

Administrivia

- Reminder: Homework 1 code due today (11:59pm). I have office hours this afternoon if you need help.
- Method `instance` in `BasicGameSetup` mentions "singleton". What's that about? Reference to "singleton design pattern" — idea that for some classes there should only ever be one instance.
- Homework 2 to be on Web later today. (I'll send you mail.) Please read through for class next time, and we can spend a little class time answering questions.
- When we have a quiz, I'll post a sample solution on the Web shortly after the quiz.

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 — more like C functions.
- Java variables are either "primitives" (like C variables) or *references* to objects. Objects are created only with `new`.

Slide 3

String Class — Example of Using a Class

- Recall — no C-style strings (arrays of characters ending in null character) in Java. Instead, `String` class. (C++ has a similar library class, `string`.)
- 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

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.

Slide 6

Arrays in Java

- Arrays are objects — unlike in C/C++, where they're basically pointers.
- Declaring (references to) arrays — denote by putting brackets after type.
- Creating arrays — use `new`, e.g.,

  ```
  new int[10]
  new String[n]
  ```

  (Remember that the second one only creates *references*.)
- All arrays have `length` variable.
- Otherwise, syntax is same as C/C++; indices start at 0.
- Java runtime does automatic bounds-checking — unlike in C/C++, get ArrayBoundsException rather than random problems.

Slide 7

Minute Essay

- None — quiz.