Slide 1

Administrivia

- If you aren't subscribed to CSMajors, it might be a good idea — we circulate announcements of CS-related events, job opportunities, etc. Not just for majors. Instructions for subscribing on department home page. (I mention this because there was a recent announcement about summer internships.)

Slide 2

Java Basics — Recap

- Java programs consist of classes. Each class can contain
  - Variables — instance and static.
  - Methods — instance and static.
  - Classes (more about this later).

  Notice that each source-code file can contain at most one public class.
- Variables and methods can be `public` or `private`.
- Variables and methods can be `final`. (Use `static final` for constants.)

Slide 3

Java Syntax

- Basic syntax based on C — variable declarations, method definitions, expressions — with some additions (as discussed in class and in the text).
- (This was by design.)

Slide 4

Variable Types

- Primitive types provided for efficiency (not purely object-oriented):
  - `boolean`, `short`, `int`, `long`, `float`, `double` are pretty much as in C.
  - `char` is 16-bit Unicode.
  - `byte` is 8-bit byte.
- All other variables are *references to objects*, similar to pointers:
  - `MyClass x` creates a *reference*, not an object — use `new` to create objects.

    Type of `x` is `MyClass` (just as type of an `int` variable is `int`).
  - Value of `null` means it doesn't point to anything.

Slide 5

Variable Scope

- As in C, variables have "scope" (region of the program in which they're valid), but possibilities are somewhat different:
- Instance variables — data for object, can be used in any method.
- Class variables — data for class (one copy for all objects), can be used in any method.
- Local variables — declared within a method *or block*, only valid within that method or block. Notice also that you can declare variables anywhere, not just at start of method.
- Advice: Use narrowest scope that will work.

Slide 6

Creating Objects

- Create object of class `MyClass` using `new` operator, e.g.,

  ```
  MyClass x = new MyClass();
  ```

  This object contains its own copy of all instance variables defined in `MyClass`.
- `new` above invokes a *constructor* for `MyClass` — method with no return type. Can have any number of these, with zero or more parameters. If none is supplied, compiler generates one with zero parameters. Useful for setting initial values for variables.

Slide 7

Deleting Objects

- No need to explicitly free/delete objects — Java has “garbage collection”.
- (Contrast with C, where you must free dynamically-allocated memory yourself.)

Slide 8

Referencing Objects, Variables, and Methods

- Within `MyClass`, reference members of class (variables and methods) using just their names. If you have multiple objects of this class, which one is meant? “current object”.
- In code using `MyClass`, reference as, e.g., `x.foo(parameters)` for instance methods, and `MyClass.staticFoo(parameters)` for static methods.

  Similar syntax for variables, but likely to be used less, since variables are normally private. (Exception is constants.)

Slide 9

Passing Parameters

- Syntax is like C.
- As in C, *everything is passed by value*. (Some languages provide other options, e.g., passing "by reference".)
- C has pointers, which can point to any data type, and this allows you fake passing parameters by reference. Not possible in Java — Java has references, which can only point to objects.
- *However*, when you pass an object reference by value, both caller and callee have references to the same object, so in some ways you appear to be passing the object by reference.

Slide 10

Comments

- Can use C-style comments, C++-style comments.
- One type of C-style comments are special — "documentation comments" or "Javadoc comments". These start with `/**` and end with `*/`, and the command-line tool `javadoc` turns them into HTML documentation similar to what Sun provides for the library functions. (IDEs, Eclipse among them, also have a way to do this.)
- Use documentation comments to describe what people using your class need to know. Use other types of comments to document code itself — something that would be useful to humans reading it.

Slide 11

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), called "for-each". 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 12

Example

- Example — `Account` class.

Slide 13

Minute Essay

- Make your best try at writing a method for our `Account` class that computes and returns one month's interest. Have it take one `int` parameter representing the monthly interest rate as a percentage (e.g., 5 for five percent per month interest).

Slide 14

Minute Essay

- Here is one way, somewhat simplified in that there's no attempt to round.

```
/**
 * Compute and return one month's interest.
 */
public long computeInterest(int interestRate) {
        // unrounded
        return (dollars * interestRate) / 100;
}
```