Slide 1

Administrivia

- Web page said Homework 1 design was due today. I intended a due date of Thursday. Fixed now!
- Slides from class will be on Web — preliminary version shortly before class, final version later that day.

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).
- Variables and methods can be `public` or `private`.
- Variables and methods can be `final`. (Use `static final` for constants.)

Slide 3

Variables

- 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 4

Java Syntax

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

Slide 5

Creating and Deleting 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 no-parameters constructor for `MyClass`. Can have additional constructor(s) with parameters as desired.

- No need to explicitly free/delete objects — Java has "garbage collection". (This may not seem remarkable unless you've used a language without it — e.g., C or C++.)

Slide 6

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 7

Passing Parameters

- Syntax is like C.
- *Everything is passed by value* — but for reference variables, copying just creates two pointers to the same object, and the called method can change the object.

  (More about this later.)

Slide 8

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.
- 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 9

Naming Conventions

- Java library classes and methods follow these conventions:
  - If it's mixed-case and starts with uppercase, it's a class.
  - If it's mixed-case and starts with lowercase, it's a variable or method.
  - If it's all uppercase, it's a constant.
- You should follow them too, so your code will be easier for experienced Java programmers to read.

Slide 10

Compiling and Running Programs — Java Versus C/C++

- With C/C++, your program ("source code") is transformed by a compiler into . . .

  "object code" (different for different processors), which is combined with library object code to produce . . .

  an "executable" (different for different operating systems) that can be run like other applications.
- With Java, your program (source code) is transformed by a compiler into . . .

  "byte code" (same on any processor), which is executed by . . .

  "Java virtual machine" (which has access to library byte code).

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

Slide 13

Minute Essay

- None — sign in.