Slide 1

Administrivia

- Reminder: Reading assignments will be on the "lecture topics and assignments" Web page. Ideally you will read before class!

Slide 2

"Object Orientation"?

- A "programming paradigm" — contrast with procedural programming, functional programming, etc.
- No accepted-by-all definition, but most definitions mention encapsulation:
  - Data and functionality grouped together into "objects".
  - Some data/functionality is hidden.
- Origins in simulation/modeling, where the goal is to model complex systems consisting of many (real-world) objects.

Slide 3

What's An Object?

- Object — set of data (attributes) and associated functions (methods, behaviors, operations) that can act on data.
- Objects interact by calling each other's methods, or by sending each other messages.
- Often makes sense to have many similar objects — hence "classes".

Slide 4

What's a Class?

- Can be thought of as a blueprint for objects of a given type; individual objects are "instances" of the class.
- Defines attributes and methods each object will have (instance variables/methods), attributes and methods shared by all objects of a class (class variables/methods).
- "Public interface" — attributes and methods visible from outside the class.

Slide 5

Java and Object Orientation

- Java is not purely object-oriented — also includes “primitive types” for efficiency — but it’s much more strongly object-oriented than a hybrid language such as C++.
- Java programs consist of definitions of classes. (No free-standing functions like the ones in C.)
- Java variables (except primitives) are references to objects, classes define types.
- Classes, attributes, methods have varying “visibilities” (from public to private).

Slide 6

Program Structure

- In Java, everything (variables and code) is part of a class. Typically have only one class per source code file (exception is inner/nested classes — more about them later).
- Any class can have a `main` method that can be launched by the runtime system (more about that later).

Slide 7

Defining a Class

- Each class is like a blueprint for objects of a particular kind, and can include:
  - Variables — instance (one copy per object) or static (one copy shared by all objects).
  - Methods — similar to C functions, but can be static or non-static ("instance methods"). Instance methods are "invoked on an object'.
  - Classes (more later).
- Variables and methods can be `public`, `private`, etc. Good practice to expose only what needs to be exposed (so variables are usually private).

Slide 8

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 9

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 . . .

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

Slide 10

Tools

- Java programs are text, so you can write them with a text editor and compile and run them from the command line. (In fact I often do.)
- However, many professional programmers use an IDE (Integrated Development Environment), so we will too. For most of the semester we will use Eclipse, which is a free open-source tool written in Java, so you should be able to install a copy on your home machine if you like. (Versions seem to be available for Windows, Linux, and Mac OS X.) But in some ways it's *too* helpful, so we will start using BlueJ.

Slide 11

Example(s)

- Let's first write the traditional "hello world" program using PAD-I-style tools (`javac` and `java` to compile and execute programs).
- Then let's do it again in BlueJ (`bluej` from the command line), and also start writing another simple class.

  (Notice that in BlueJ you have to start by creating a "project" — very typical of IDEs.)
- (Now you should know enough to start trying examples as you do the reading — and you should!)

Slide 12

Minute Essay

- Was there anything today that was particularly unclear?
- If you have programmed in Java before, what tool(s) did you use?