Administrivia

- Linux accounts for new students should have been created, with passwords mailed to Trinity e-mail addresses.

  Problems? Talk to me. To change your password, `yppasswd` from the command line.

Slide 1

- Link added to “Useful links” page to Linux command-line information.

More Administrivia

- *Please* do not reboot the machines in this room (HAS 340); people rely on their being available for remote access.

Slide 2

Slide 3

About the Readings

- The textbook (*Java Software Structures*) is intended for a second-semester programming course, for people who already know Java.
- Since we teach our first-semester course in C — Dr. Lewis has been writing a "From C to Java" document.
- You should probably skim all assigned reading, but those with Java background will find some parts review.

Slide 4

The Project

- Did you start reading the project description? Do you have (quick) questions?

Slide 5

"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 6

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 7

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 8

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 9

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 10

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` or `private`. Good practice to define as private, except for constants that need to be used outside the class.

Slide 11

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 12

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 usually do.)
- However, many professional programmers use an IDE (Integrated Development Environment), so we will too. It's Eclipse, and open-source, so you should be able to install a copy on your home machine if you like.

Slide 13

Example(s)

- Let's write a "hello world" program.
- We'll use Eclipse to
  - Define a project, a package, and a class with a `main` method.
  - Compile and run.
  - Generate HTML documentation.
- Other example(s) as time permits.

  (All example code will be available linked from the "Sample programs" page.)

Slide 14

Minute Essay

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