

Slide 1

Administrivia

- Reminder (or notice, for those not subscribed to the CSMajors mailing list): “Research Opportunities Fair” today at 5pm in the Great Hall.
- (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.)

Slide 2

More Administrivia

- *Please* do not reboot the machines in this room (HAS 340); people rely on their being available for background work and remote access.
Also be careful not to inadvertently shut them down when trying to log off.
If a previous user has left the machine’s screen locked, you can use control-alt-backspace to restart the graphical subsystem.
If you think a reboot is needed, find a faculty member to decide and take responsibility.

“Object Orientation”?

Slide 3

- 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.

What’s An Object?

Slide 4

- 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”.

What's a Class?

Slide 5

- 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.

Java and Object Orientation

Slide 6

- 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).

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

Slide 8

Slide 9

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. 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.)

Slide 10

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.

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

Slide 11

- 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).

Example(s)

Slide 12

- 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.
- (Now you should know enough to start trying examples as you do the reading — and you probably should.)

Minute Essay

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

Slide 13