Slide 1

Administrivia

- Updated syllabus (minor wording changes).
- If problems with Linux accounts (old or new), talk to me and/or send mail to CSAdmin list.
- Look online for updated reading assignment later today.

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.

What's An Object?

Slide 3

- 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 4

- 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 5

- 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).

Inheritance (Short Version)

Slide 6

- Given a class, it can be useful to define specialized versions — “subclasses”.
- A subclass inherits attributes and operations from its superclass (which can in turn have a superclass . . .).
- Subclasses also form “subtypes” — e.g., if Triangle is a subclass of Shape, can use a Triangle anywhere we need a Shape.

Polymorphism (Short Version)

- "Many shapes" — something that works with many types.
- E.g., a function that works on Shapes should work on Triangles, Circles, . . .

Slide 7

UML Class Diagrams

- "Unified Modeling Language" — formal graphic representation of software analysis and design.
- We will mainly use class diagrams:
  - Box representing a class has name, attributes, operations.
  - Subclass points to its superclass (represents the path to follow to figure out inheritance).

Slide 8

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 . . .

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

Slide 10

Sample Programs

- Let’s write a “hello world” program . . .

Minute Essay

- Was there anything today that was particularly unclear?