CSE 143

Object-Oriented Design

[Chapters 1, 8]

Design Methodology

- Changes may result in lots of wasted work! How to minimize their impact?
  - Use a good *design methodology*
  - Procedure and structure by which a design is created
- Top-Down Design, aka structured design
  - Focus on overall control flow rather than data.
  - Think of problem in terms of functions and algorithms and how they need to interact
  - Often have a layered or hierarchical approach: make successively more detailed refinements to design
  - Traditional design method for C and similar procedural languages

Object-Oriented Design

- An alternate design philosophy.
- Instead of control flow and functions, concentrate on different *kinds* of entities ("objects") in the problem (*data-driven* approach)
- Object = Collection of data and operations on that data
- All phases of design are in terms of objects
- Often easier to prototype a design or adapt to changing conditions

Designing in the OO Style

Step 1: Identify the objects in the problem, and the operations they should have

What are the objects in the problem?

Step 2: Determine organization of objects and operations

How do the objects relate to one another?

Are some contained inside another object, or need to organize other objects?

Drawing an *object hierarchy diagram* might help

What messages pass between objects?

Step 3: Implement objects (C++ classes, or off-the-shelf)

Tightly encapsulate data and operations

Step 4: Test and refine

Three Cornerstones of OO Programming

- Encapsulation
  - Packaging data and functions together as classes
  - Hiding implementation details from clients
- Inheritance
- Overloading
  - polymorphic functions, dynamic dispatch, operator overloading

Historical Notes

- The object model was first thoroughly developed in **Smalltalk**
  - Smalltalk still looks modern!
  - Roots go back to Simula in early 1970's
- **C** was as far from object-oriented as you get can get
- **C++** = C + O.O. features
  - Considered an ugly hybrid by many
- **Java** retains much C++ syntax
  - but simpler, "purer" - a lot like Smalltalk underneath the syntax