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*
    - or *design philosophy* or *design paradigm*
  - Procedure and structure by which a design is created
- Some popular paradigms
  - Top down design
    - Typical for Pascal, C
    - Can be used with C++
  - Object-oriented design
    - Typical for Java, Ada, Smalltalk
    - Can be used with C++

Top-Down Design

- Also called:
  - Structured Design
  - Functional Decomposition
- Focus on overall control flow
  - Think of problem in terms of functions and algorithms and how they act on the data
    - Input-> Process -> Output
  - Often have a layered or hierarchical approach: make successively more detailed refinements to design
  - Call graph depicts the overall design

Object-Oriented Design

- 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

Often, objects are nouns and operations are verbs in the English description of the problem

Step 2: Determine organization of objects and operations

How do the objects relate to one another? Are there similarities, differences? Is-a and has-a relationships? Are there containers holding multiple 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

Astrachan's Maxim

"Ask not what you can do to an object, but what can the object do to itself"

Three Cornerstones of OO Programming

- Encapsulation
  - Packaging data and functions together as classes
  - Hiding implementation details from clients
- Inheritance
- Overloading (and related concepts)
  - polymorphic (overloaded) functions
  - virtual functions and dynamic dispatch
  - operator overloading

Historical Notes

- The object model was first thoroughly developed in **Smalltalk**
  - Smalltalk still looks modern!
- **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
- Many modern programming and scripting languages use aspects of O.O
  - Javascript, Perl, Visual Basic, Python, etc.