Abstraction

Object-Oriented Concepts
• There are many software design approaches
  o We focus in this course on Object-oriented analysis, design, and programming
  o Software solutions are created by arranging for multiple agents to cooperate
• Object-oriented analysis (OOA)
  o What needs to be done?
  o What are the objects (agents, players, actors, etc.)?
  o How do they need to communicate?
• Object-oriented design (OOD)
  o Forming a concrete design from the requirements uncovered during OOA
• Object-oriented programming (OOP)
  o Implementing the design in an OO language (e.g., C++ or Java)

ADTs vs. Objects
• Abstract Data Type (ADT)
  o An encapsulation of data and operations on the data.
• “Object-oriented”
  o Encapsulation of data and operations (i.e., ADT)
  o Inheritance of properties from “base classes” or “super classes”
  o Runtime polymorphism
• What is polymorphism? Consider two examples:
  o Static polymorphism
    • int solve(double a, double b, double c, double roots[]);
  o Runtime polymorphism
    void apply(Widget& w)
    {
      w.doSomething(); // potentially goes to different code on each call
    }
ADTs vs. Objects (cont’d)

- Sometimes ADTs are sufficient; sometimes OO leads to a superior and more extensible solution.
- In languages like Java and C++, the basic tool for ADTs (the “class”) is also the tool for inheritance and runtime polymorphism as employed in object oriented programs.

Design

- Design is hard, but can be learned.
  - Seek out designs (and code) that can be reused.
  - Make your own designs and code reusable.
- Making designs reusable (Design Patterns)
  - State the problem being solved, assumptions, restrictions, etc. in as generic terms as possible.
  - Describe a general solution design that satisfies the problem.
- Making code reusable
  - Write single-purpose, highly cohesive methods (and classes)
  - When combining methods and creating cooperating agents, minimize their interdependencies and knowledge of each other Þ loose coupling.

Abstraction?

- Abstraction is simply focusing on what you need to know to solve the problem at hand
  - Know enough, but not too much
  - In terms of software design, know what input you are given and what you are to do with it.
  - Do NOT worry about
    - How your clients built the data they gave you or how they will use what you produce.
    - How methods and objects you use go about fulfilling the requests you make of them.
- http://people.eecs.ksu.edu/~miller/Courses/268/ Materials/Abstraction.html