Object-Oriented Design

Motivation ?

- Manage program complexity
- Partition solution into a collection of smaller, well-defined components
- Partition around the "data", rather than the program-flow
  Rather than "top down" or "bottom up" design,
  identify, design and implement "objects" that interact with each other.

What is a **Data Type**?

A "set of values" and a "set of operations" that can manipulate those values

Abstract Data Type: the implementation of the operations is unknown or unspecified. Only the operation "interfaces" are firmly specified.

What is an **OBJECT**?

An "Intelligent" entity

• Data	"What it Knows"
• Methods	"What it Can Do"

C++: Object → **CLASS**

- Similar to a *struct*
  - Data members
  - Methods (functions defined inside the Class object definition)

  - Provides an Interface (or definition) for the object

  - Allows Access/Visibility control
    - public
    - private

  - Constructors/Destructor:
    - guarantee initialization, etc.
    - guarantee proper deallocation

OO-terminology

Class:	An object design and implementation
Object:	An instance of a class (e.g., a variable)
attribute:	A data member of a class object
method:	A member function of a class
message-send:	A call to a member function of a class
message-reply:	The value returned by the function called in a message-send.
Interface:	The messages to which an object can respond
Encapsulation:	Design and implementation of an object can be self-contained. Internal implementation details can be hidden. Only the “official” interface of the object is visible to other objects.
Inheritance:	Design of a new object class, based on existing object
Polymorphism:	Each object class responds to a message in its own way. Related objects (inherit from same base class) can have “virtual” methods that do not “bind” in the application until runtime, when the message is sent.

- **Object Encapsulation**
    - Establish public/private "visibility" of class members
    - Ensures that users of the class object can only use the defined interface
    - Allows object implementation to be "hidden" from other objects
    - Constructors enable an object implementation to be self-contained
    - C++ fully supports "using" an object without knowledge of the private implementation details
        - pass as parameter
        - return as function value
        - assignment

- **Object Inheritance**
    - Design a new object, based on an existing object
    - Extend the functionality of an object
    - Restrict access to existing features of an existing object
    - Define a new "interface" for an existing object

Encourage Re-use, adapt to new situations

Example: Windows programming

CWindow

CScrollableWindow

CDialogBox

- **Object Polymorphism**
    - Allow related, but different, object types to define different implementations of a common operation or method

      Example: a graphical "drawing" package

      a list of "shape" objects (square, triangle, circle, …)

      each "shape" object has it's own "draw yourself" method
    - Container classes and run-time binding

      The main drawing package can simply traverse a list of shape objects and tell each object to draw itself at a specified location.