Inheritance Heuristics

● A base class is an interface
  ➤ Subclasses implement the interface
    • Behavior changes in subclasses, but there’s commonality
  ➤ The base class can supply some default behavior
    • Derived classes can use, override, both
  ➤ The base class can have state
    • Protected: inherited and directly accessible
    • Private: inherited but not accessible directly
  ➤ Abstract base classes are a good thing

● Push common behavior as high up as possible in an inheritance hierarchy

● If the subclasses aren’t used polymorphically (e.g., through a pointer to the base class) then the inheritance hierarchy is probably flawed
Inheritance Heuristics in C++

- One pure virtual (aka abstract) function makes a class abstract
  - Cannot be instantiated, but can be constructed (why?)
  - Default in C++ is non-virtual or *monomorphic*
    - Unreasonable emphasis on efficiency, sacrifices generality
    - If you think subclassing will occur, all methods are virtual
  - Must have virtual destructor, the base class destructor (and constructor) will be called

- We use public inheritance, models *is-a* relationship
  - Private inheritance means is-implemented-in-terms-of
    - Implementation technique, not design technique
    - Derived class methods call base-class methods, but no “usable-as-a” via polymorphism
    - Access to protected methods, and can redefine virtual funcs
Inheritance and Layering/Aggregation

- Layering (or aggregation) means “uses via instance variable”
  - Use attributes if differences aren’t behavioral
  - Use inheritance when differences are behavioral

- Consider Student class: name, age, gender, sleeping habits
  - Which are attributes, which might be virtual methods

- Lots of classes can lead to lots of problems
  - It’s hard to manage lots of classes in your head
  - Tools help, use speedbar in emacs, other class browsers in IDEs or in comments (e.g., javadoc)

- Inheritance hierarchies cannot be too deep (understandable?)
Inheritance guidelines (see Riel)

- **Watch out for derived classes with only one instance/object**
  - For the CarMaker class is GeneralMotors a subclass or an object?

- **Watch out for derived classes that override behavior with a no-op**
  - Mammal class from which platypus derives, live-birth?

- **Too much subclassing? Base class House**
  - Derived: ElectricallyCooledHouse, SolarHeatedHouse?

- **What to do with a list of fruit that must support apple-coring?**
  - Fruit list is polymorphic (in theory), not everything corable