Handouts/cse687/Code/Hiding

How to Avoid Hiding
and a few other Evils

Jim Fawcett
CSE687 – Object Oriented Design
Spring 2006

Avoid:

1. Redefining, in derived classes, non-virtual base class functions
    a. Non-virtual member functions do not have vtable entries and so the function called is the type of the pointer or reference, not the type of object attached to the pointer or reference.
    b. So it is possible for a base class function to be called on a derived class object, with possibly disasterous results.
2. Overloading non-virtual base class functions in derived classes
    a. Overloads work only within a single scope, not across both base and derived class scopes.
    b. The result may be hiding of base class member functions that are inherited by the derived class.
3. Overloading virtual functions
    a. If a derived class redefines a base class virtual function, which is a correct procedure, that will hide the base class overloads that are inherited.
4. Using default parameters in virtual functions
    a. Parameters don't have vtable entries, so they are bound based on the type of pointer or reference to an object, not of the object type.
    b. This results in a derived class using base class defaults even though the derived class defined different values for the defaulted parameters.

Handouts/cse687/Code/Hiding

Always:

1. Provide a virtual destructor if your class may be used as the base class for a derivation.
    a. If you don't do that, and a client creates an instance of a class derived from your base class on the heap, bound to a base pointer, then when the client calls delete on that pointer, the destructor called is based on the type of pointer not the type of object, so the base destructor only will be called.

Definitions:

1. Overriding:
   Providing, in a derived class, a declaration and definition of a virtual base class function, using exactly the same function signature and the same or covariant return type.
    a. A covariant return type is a pointer or reference of the derived type, when the base virtual function returns a pointer or reference of the base type.
2. Overloading:
   Providing, in the same class, or in the same global scope, a function definition that uses the function identifier of another existing function with a different sequence of formal parameter types.
    a. Note that you cannot overload on return type, because a client is not compelled to use the return type, so the compiler cannot figure out which function to bind to.