Why inheritance?

- **Add new shapes easily without changing code**
  - Shape * sp = new Circle;
  - Shape * sp2 = new Square;

- **abstract base class:**
  - interface or abstraction
  - pure virtual function

- **concrete subclass**
  - implementation
  - provide a version of all pure functions

- **remember “is-a” view of inheritance**

User’s eye view: think and program with *abstractions*, realize different, but conforming *implementations*
What is a map (aka dictionary)

- **Keys are mapped to values**
  - word --> # occurrences
  - student id --> grades
  - login-id --> working, home directory
- **Maps are templated classes**
  - `Map<string,int> map;`
  - `Map<int, Grades> gmap;`
- **operations on maps:**
  - insert
  - delete
  - find
  - element access (all?)
Iterators

- A pattern (realized as a class) for accessing all the elements in a map (or, more generally, a container)
  ```cpp
  for(it.First(); ! It.IsDone(); it.Next())
  {
    process(it.Current());
  }
  ```
- Two kinds of iterator, internal, external
  - Internal: pass a function or object to a container, container applies function to all its elements
  - External: ask container for sequential access to all elements in the container
- Implementations? How to do internal and external iterators
YAIH: yet another inheritance hierarchy

- Ask “give me an iterator”
  - do not ask, “give me an hmapiterator”

- Iterators must supply access to internals of a container, how does iterator get access to these internals?
  - Internals live in some private section, not accessible except to class, but iterator is a separate class!
  - Even if accessible, the private stuff lives somewhere else, not in the iterator

- Iterator must save state between invocations of current to allow Next() function to work properly
Iterators

- For vector:
  - state: current index
  - how to tell when done?
  - what about Next()?

- For hash table
  - state: current node, current index (both)
  - how to tell when done?
  - what about Next()?

- Binary search tree?
More iterators

- If iterating inorder, how much information is needed to implement `Next()`?
- Other iterators for BST?
- Internal iterators:
  - passing functions as parameters
    ```cpp
    void Print(string & s);
    void Pigify(string & s);
    void RemovePunct(string & s);
    ```
  - generalize: `map.Apply(Print)`
- Restrictions from functions: move to classes
  - average word length?