Programming in the small, medium, large

- You must be able to write class correctly to be able to write project
- You must understand the difference between a pointer and a reference to write your project
- You must know about templates, copy constructors, arrays, pointers to functions to write masteries

- You must know about built-in arrays and c-style strings to be able to implement a string and vector class, or be able to cope without them
- You must know about factories and proxies to design and implement using object-oriented principles
- You must know about patterns to communicate
Writing C++ classes

- Write classes so that you are a good programming citizen
  - problems guaranteed to occur when you do not make effort
  - but must understand what C++ does for you

- What’s going on in the following code?

```cpp
Vector<Date> jan(31);
Vector<Date> feb = maybeLeapYearMonth();
Vector<Date> mar(jan);
Vector<Date> apr(30);
Vector<Date> may = jan;
// etc.
```
copy constructor

- **Used for “first-time” creation**
  
  ```cpp
  Date d(1, 1, 2000);
  Date copy(d);
  ```

- **Used for pass-by-value**
  
  ```cpp
  DoStuff(Date d);
  //…
  Date first(1, 1, 2000);
  DoStuff(first);
  ```

- **what about use of myLength in code as opposed to length()?**

```cpp
Template <class Item>
Vector(const Vector<Item> & vec)
// precondition: Item supports assignment
// postcondition: return copy of vec
{
    // allocate storage
    myLength = vec.myLength;
    myList = new Item[myLength];
    assert(myList != 0);
    // copy elements
    for (int k = 0; k < vec.myLength; k++)
    {
        myList[k] = vec.myList[k];
    }
}
```
public versus private member functions

- **public member functions**
  - can be called by any other class in the system
  - defines class *interface* --- what user can do with your object

- **private member functions**
  - cannot be called by any other class
  - used primarily to provide *helper* functions
  - also used to hide functions C++ silently writes for you

- **C++ automatically generates default implementations for**
  - assignment, copy constructor
  - default constructor, destructor
Has-a versus has-a-pointer to

- **Benefits of** `string * myName` **vs** `string myName`?
- **Downside of using pointers**?

- **Responsibilities for memory management, garbage collection**
  - not always possible for that which news data to delete it
  - deletes cause bugs, introduce destructors into the system iteratively, chasing one bug at a time
  - use “smart” pointer class, stands in for pointer

- **Pointers and references are the same size, both allow one variable to refer to memory created/stored elsewhere**
  - reference values bound on construction, no change
  - pointer values can be changed
C++ idioms

● What happens with the statement `myDay = d;`?
  ➤ assignment is memberwise unless operator `=` overloaded
  ➤ copy constructor used in passing parameters by value

● If you need one of: destructor, assignment operator, copy constructor, you need all of them
  ➤ heuristic only: managing resources other than memory
  ➤ preventing objects from being copied
  ➤ what about non-copyable state, e.g., stream

● In assignment operator, watch for self-assignment
  ➤ what problems might occur?
Inheritance guidelines in C++

- **Inherit from Abstract Base Classes (ABC)**
  - one pure virtual function needed (=0)
  - must have virtual destructor implemented
  - can have pure virtual destructor implemented, but not normally needed

- **Avoid protected data, but sometimes this isn’t possible**
  - data is private, subclasses have it, can’t access it
  - keep protected data to a minimum

- **Single inheritance, assume most functions are virtual**
  - multiple inheritance ok when using ABC, problem with data in super classes
  - virtual: some overhead, but open/closed principle intact
Templated classes

- If interface (.h) and implementation (.cc) are separate, client code must have access to implementation
  - can define all code inline, in class (vector.h)
  - can explicitly instantiate all uses in a separate file
  - can #include “tempclass.cc” in .h file, problems?
    - -frepo flag fixes this in g++
    - VC++ (and other PC compilers) use this approach

- STL classes use inline approach
  - see the <vector> or <map> header files

- Dangers of code-bloat
  - no literal code sharing between Vector<string> and Vector<int>