CSCI 2270

Data Structures & Algorithms

Gabe Johnson

Lecture 25 Mar 15, 2013

Subclassing, Operator Overloading Graphs and Graph Algorithms

Lecture Goals

1. Huffman HW Delay (Weather)
2. About Next Week's Test
3. Finish Classes/Objects
4. Graphs

Upcoming Homework Assignment

HW #7 **Due: Mon, Mar 18**

Huffman Encoding

It is too nice to be inside programming. Huffman assignment is now due on Monday, 6pm.

Questions?

Test on Wednesday

Test will cover everything we've done so far. Exam 1 topics will definitely be revisited:

- Pointers
- Recursion
- Complexity

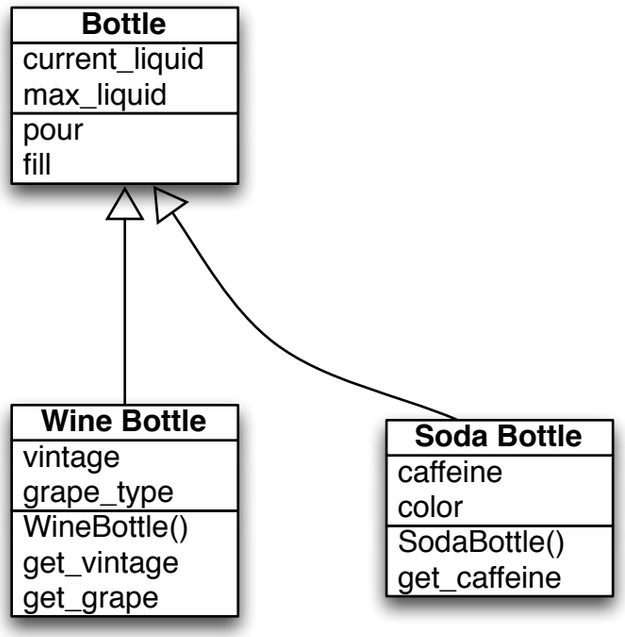
HW and Lecture topics since the exam: B-Trees, Priority Queues, Huffman Encoding, Standard Container Types, Treaps.

Specific Test Things

The next test will not have nearly as much coding (either pseudocode or actual C++), but there will still be a question or two about design, and you might write some pseudocode there.

You will **not** need to know how to implement crazy B-Tree algorithms. But you will need to understand from a high level how and why B-Trees work. E.g., "*what's the complexity of an order **m** B-Tree 'contains_key' operation?*"

Why Subclass?



Some software engineers *love* these diagrams. They help you visualize the relationship among classes. Here you can see there are three classes, and Wine Bottle and Soda Bottle are subclasses of plain old Bottle. A subclass can inherit the (non-private) variables and functions of the superclass. We can also treat the subclass as its superclass.

Classes Are Types

Just like structs can contain other structs, objects can contain other objects.

```
class Rectangle {
  Point top_left;
  Dimension size;
  float angle;
};
```

```
class Point {
  float x;
  float y;
};

class Dimension {
  int w;
  int h;
};
```

Operator Overloading

Since objects are types, just like every other variable, we can apply operations to them.

Operators can be member functions of a class, or they can be defined separately. Sadly, we can't do this for operators like << that take the stream as the first argument, since the object would be the first (implied) argument.

Op. Overload Example

```
// LinkedList.hpp
class LinkedList {
  friend std::ostream &operator <<
         (std::ostream& out, LinkedList list);
};

// LinkedList.cpp
ostream &operator << (ostream& out, LinkedList list) {
  if (list.root != NULL) {
    out << "[ ";
    LinkedListNode* cursor = list.root;
    while (cursor != NULL) {
      out << cursor->value << ", ";
      cursor = cursor->next;
    }
    out << "]";
  }
  return out;
}
```

Then we can use it with cout:
LinkedList george;
cout << “list is: “ << george << endl;

Graphs!

There is little chance I'll get done with the Graphs lecture today. The first homework after break will be to implement a breadth-first search and a depth-first search. I will give you a half-implemented C++ class, you will just need to fill in the DFS and BFS member functions.

*Please see **Graphs.pdf** on GitHub for the rest of the slides. It is in the HW 8 directory.*