Design Patterns 1

2-29-2002

Opening Discussion

- What did we talk about last class? Do you have any questions about assignments?
- Did you do the reading? What are design patterns? What do they help us with? What are the patterns that were in the reading for today?
- Templates and separate compilation in C++.

Design Patterns

- Certain things that you want to do in programs come up repeatedly. General solutions to these are often called design patterns.
- Understanding and being able to use design patterns can make your life programming easier, but they are not without their pitfalls as the code to do them can be complex at times.

Functors

- Classes that have no data are sometimes called functors. Especially if they have only one method in them and they are used as arguments to templated functions.
- A functor class allows the flexability of easily passing a function as an argument to another function. This can be helpful for things like sorting on different keys.

Writing Functors

- You can have a functor where the method is a normal method with some appropriate name. This is how you would do it in most OOPLs.
- In C++ you have the ability to overload the function call operator, operator(), to take different numbers of different types of arguments. When you invoke these you use the functor like a function.

Wrappers

- Sometimes you want to add an extra layer of abstraction on top of a given data type. This is most common when the data type is a primitive and the default behavior isn't exactly what you want it to do.
- With a wrapper class you create a class that stores the primitive type, but has a more appropriate interface for doing what you want it to do.

Adapters

- At times you might have a class that does something you want, but the interface on it isn't quite right. This could be especially true if you are going to use it with templates and the template functions or classes require certain methods of the class.
- An adapter uses private inheritance to put a new interface over the old one.

Details

- All of these patterns we discussed today gain significance from templates. In some cases they are things that you wouldn't want to even do without templates (adapters). In others, the templates make them fast and efficient (functors).

Minute Essay

- We might have a use for functors in the project so that you can have a sorted list that gets sorted on different aspects of SubStr. It might even be helpful for assignment #3. Write a functor class where the method takes two SubStr objects and compares them in some way.
- Remember that the design for assignment #3 is due today.