Lecture 23

Namespaces
&
Mop up

Namespaces

- At the beginning of the semester, I asked you to take it on faith that if you wanted to use the functions defined in <iostream> you would need to add the following line of code to your program:
  - `using namespace std;`
- A *namespace* is a method by which you can encapsulate any combination of classes, constants, globals, types into a neat "package".
- Without using special directives, such classes, globals, etc., can only be accessed through the namespace's name.
- Namespaces look somewhat like a class declaration.
- You define a name for your namespace and use curly braces to define a scope for it

Namespaces (cont)

- Everything which occurs in that scope is "hidden" in your namespace.
- It might look something like this:

```
//
// Number.h
//

namespace CornellCS213        // Defines a namespace named CornellCS213
{
  class Number
  {
    public:
    …
  };
}
```

Namespaces (cont)

- Given the previous namespace declaration, you can access `Number` in one of three ways.
- First, you can use `Number`'s new fully qualified name anywhere in your code:
  - `CornellCS213::Number aNum;`
- Second you can specifically designate that, for the current file being compiled, you'd like to use the `Number` implementation in the `CornellCS213` namespace:
  - `using CornellCS213::Number;`
  - `Number aNum;`
- Third, you can simply state that you'd like to use all the contents of a given namespace without qualification. This is what we've been doing all year with the `std` namespace:
  - `using namespace CornellCS213;`
  - `Number aNum;`

Namespaces (cont)

- Now, a cool feature with namespaces is that a single namespace may span multiple files.
- Any time a namespace keyword is encountered, the contents that follow are "appended" to any previously existing entries in that namespace.
- So given our example which encapsulated Number into CornellCS213, if the following were encountered:

```
namespace CornellCS213
{
  class Person  { … };
  class Student : public Person { … };
  class Instructor : public Person { … };
}
```

- It would also be accessible in the CornellCS213 namespace, along with Number.

Namespaces (cont)

- You may also define "aliases" to existing namespaces to shorten their names.
- Consider the following namespace:
  - `namespace AReallyLongNameSpaceName { … }`
- You can use the following call to shorten it:
  - `namespace X = AReallyLongNameSpaceName;`


Mop-Up
Any Questions?