Namespaces

Namespaces allow you to subdivide global scope into "sub-scopes". You can do this with the namespace keyword:

```
namespace yourspace
{
      // classes, variables, methods, etc.  inside yourspace
}
```

Each namespace is considered its own scope, so you could have two variables of the same name in different namespaces. This is where namespaces are useful, as they avoid redefinition errors if you have two identifiers of the same name in separate namespaces.

Here is a simple example:

```
namespace first
{
  int x = 5;
  int y = 10;
}

namespace second
{
  double x = 3.1416;
  double y = 2.7183;
}
```

To specify which variable we are interested in, use the scope resolution operator, ::

```
cout << first::x << endl;              // 5
cout << second::x << endl;     // 3.1416
```

If we want access to all the entities in a namespace, we use:

```
using namespace first;
using namespace std;
```

A better technique is to only use the specified identifiers or classes that we need. For example, if we are only using cin, cout, and endl, then we would use:

```
using std::cin;
using std::cout;
using std::endl;
```

This is considered a better technique, as it specifies just those entities we need to use from the std namespace. It is akin to Java and using import java.util.Scanner instead of import java.util.*. This helps the programmer know what entities we are using from the other namespace in the top of the code.