A using-declaration, then, makes a single name available. In contrast, the using-directive makes all the names available. A using-directive consists of preceding a namespace name with the keywords `using namespace`, and it makes all the names in the namespace available without using the scope resolution operator:

```cpp
using namespace Jack;  // make all the names in Jack available
```

Placing a using-directive at the global level makes the namespace names available globally. You've seen this in action many a time:

```cpp
#include <iostream>    // places names in namespace std
using namespace std;   // make names available globally
```

Placing a using-directive in a particular function makes the names available just in that function.

```cpp
int vorn(int m)
{
    using namespace jack; // make names available in vorn()
...
}
```

**Using-Directive Versus Using-Declaration**

Using a using-directive to import all the names wholesale is *not* the same as using multiple using-declarations. It's more like the mass application of a scope resolution operator. When you use a using-declaration, it is as if the name is declared at the location of the using-declaration. If a particular name already is declared in a function, you can't import the same...