New Namespace Features

What C++ now adds is the ability to create named namespaces by defining a new kind of declarative region, one whose main purpose is to provide an area in which to declare names. The names in one namespace don't conflict with the same names declared in other namespaces, and there are mechanisms for letting other parts of a program use items declared in a namespace. The following code, for example, uses the new keyword namespace to create two namespaces, Jack and Jill.

```cpp
namespace Jack {
    double pail;
    void fetch();
    int pal;
    struct Well { ... };
}
namespace Jill {
    double bucket(double n) { ... }
    double fetch;
    int pal;
    struct Hill { ... };
}
```

Namespaces can be located at the global level or inside other namespaces, but they cannot be placed in a block. Thus, a name declared in a namespace has external linkage by default (unless it refers to a constant).

In addition to user-defined namespaces, there is one more namespace, the global namespace. This corresponds to the file-level declarative region, so what used to be termed global variables are now described as being part of the global namespace.

The names in any one namespace can't conflict with names in another namespace. Thus, the fetch in Jack can coexist with the fetch in Jill, and the Hill in Jill can coexist with an external Hill. The rules governing declarations and definitions in a namespace are the same as the rules for global declarations and definitions.

Namespaces are open, meaning you can add names to existing namespaces. For example, the statement