Namespace Essentials C++

In C++ the namespace gives us a convenient way to refer to a variable that will not conflict with another global variable. The namespace can be declared by the following syntax “*using namespace* [name]”. Adding this syntax prior to any code will reduce the hassle of having to call the namespace to use a variable of that namespace within local scope. Namespaces are equivalent to Java packages. In java one can refer to a method or variable that’s declared in another package by using the syntax: “*import java...”* Despite their similarities namespaces and packages do have their differences.

For example in Java packages are used primarily to organize files or public types which will reduce type conflicts. Package constructs can also be mapped to a file system. In C++ however namespaces ae used to organize programs both internally and externally. One good site to reference that contrasts the differences between the two is <http://www.javacamp.org/javavscsharp/namespace.html>. Even though namespaces can be used to organize programs, using them isn’t recommended as namespaces were initially designed as a means of preventing conflicts within large libraries not for expressing concepts. Instead of using namespaces a good program would be designed around the principles of object oriented programming in terms of using classes to organize a program. The benefits of using classes to organize a program are that classes have constructors, destructors, private members, and they cannot be reopened. In Java it’s quite difficult to escape needing packages because many functions included with the native Java libraries require programmers to import the necessary packages.

Check out stackoverflow for a more general discussion on this topic <http://stackoverflow.com/questions/2108172/c-namespaces-comparison-to-java-packages>