Comparison Between the C++ and Java Programming Languages

It is easy to see the similarities between the C++ and Java programming languages, especially with the following: syntax or manner of writing the programs, the comments used, and method of definition.

Both programming languages also use similar constructors and method overloading. The two also utilize primitive types for accessing efficiency. Java uses boolean, char, byte, short, int, long, float, and double. However, type-checking and type requirements are tighter in Java, such that conditional expressions can be only boolean and not integral.

There are some notable things that make Java easier to work with than C++. Java has no use for a scope resolution operator. On the other hand, Java uses class definitions instead of class declarations. It also calls up static methods differently. Static quoted strings are automatically converted into String objects. In C++, strings are represented by arrays of characters.

Arrays in C++ and Java appear similar, but these are actually different in structure and behavior. Forward declarations are unnecessary in Java. It is not necessary to define a class or method before using it. The compiler makes certain that the appropriate definition exists. This eliminates the typical forward referencing issue present in C++ programming.

Java does away with the preprocessor, which C++ uses in order to use classes in another library. It also has no use for namespaces; it uses packages instead. The designation of names is managed by grouping everything into a class and by using a "packages" facility that functions to break up the equivalent name space for class names. This tool also collects library components under a single library name. A package is simply imported and the compiler does the rest.

Other things that are different in Java from C++ that may be noted here include the guarantee of initialization of primitive data class members even if they are not initialized explicitly or get a default value or zero, the nonexistence of goto, the use of singly-rooted hierarchy, the doing away with templates and parameterized types, and the lesser incidence of memory leaks by using a set of collections: Vector, Stack, and Hashtable that hold Object references and meet the needs for collection.

The garbage-collecting capacity is a big improvement over C++ because it makes many problems in programming disappear. There is more to Java than what has already been mentioned. It has built-in support for comment documentation, standard libraries for solving specific tasks, and Java Beans standard, which allows the creation of components for use in visual programming environments.

Information technology author Bruce Eckel of *Thinking in* Java and *Thinking in* C++ generally accepts that Java is richer in capabilities than C++ because of the following:

*object handles initialized to null
*handles are always checked and exceptions are thrown for failures
*array accesses are checked and exceptions are thrown for failures
*array accesses are checked for bounds violations
*automatic garbage protection keeps memory leaks minimal
*clean exception handling
*basic language support for multithreading
*bytecode verification of network applets

Java is not a perfect programming language, but programmers say that the improvements in Java outweigh the shortcomings as well as the limitations of its parent, C++.

For more reference, click on the link below, if you have Internet access: http://www.javacoffeebreak.com/articles/thinkinginjava/comparingc++andjava.html