|  |  |
| --- | --- |
| **C++** | **JAVA** |
| functions | methods |
| const | “final” |
| using | “import” |
| shared\_ptr | reference |
| data members | fields / instance variables |
| static data members | class variables |
| range based for | enhanced for |
| auto (type deduction) | var |
| -> | . |
| destructor | “finalize” |
| namespaces / libraries | packages |
| derive | extend |
| base class | superclass |
| derived class | subclass |
| void\* | Object reference |
| -  (class that have at least 1 pure virtual func) | abstract |
| virtual | -  (everything virtual by default other than static and final methods) |
| pure virtual | abstract method |
| copy constructor | Object.clone() |
| templates | generics |

~ ALWAYS WRITE YOUR CODE CLOSE FOR MODIFICATIONS, OPEN FOR EXTENSIONS ~

“final” is same as in C++.

If name is same, meaning is same. If name is not same (like above table), meanings are similar.

If there is “ “, then similarity is not too much.

JAVA is designed to look like C++ or C bc they knew that C++/C programmers will move to JAVA.

In JAVA we don’t have namespaces, we have packages.

There is new keyword in JAVA same with in C++.

C++ reference and JAVA reference are different. You don’t delete them in JAVA.

There is no delete keyword in JAVA. You never delete anything. Everything is deleted automatically. It is like how the shared\_ptr supposed to be. With shared\_ptr you don’t delete anything, you just make a new object, when the object is not referenced anymore, it is deleted automatically, its destructor is called automatically.

In JAVA you make all the objects with new, there is no other way. Everything is on the heap.

Fundamental types are created on the stack. There is no “new int”.

In JAVA, there is no include and header files. So you don’t separate your class definition from the implementation. Once you start writing a class, you have to put all of your methods including their implementations inside the class.

There is no linking in JAVA. We don’t link any programs with any libraries. Also you don’t see any include statements in JAVA.

With an Object reference, you can point to anything you like in JAVA.

(You can make method calls with object reference)

With a void pointer, you can point to anything you like in C++.

(You can’t increment the void pointer, add values to it, make function calls with it)

Global namespace in C++ is kinda unnamed package in JAVA.

Static methods are like global functions.

In JAVA, we don’t have ++, -- operators. We have next and pre methods. Interable interface has them.