Some simple differences between Java and C++

- Java eliminated manual memory allocation and deallocation and added automatic GarbageCollection
- Java introduced true arrays and eliminated PointerArithmetic
- Java eliminated MultipleInheritance, replacing it with a notion of *interface* derived from ObjectiveC
- Java has no pre-processor, no include files, no macros (although it is possible to find preprocessor extensions for Java, for example http://www.google.com/search?q=java+preprocessor), This is a bit of a bummer especially for debug builds where you want more error info.
- Java has mapping of fully-qualified ClassNames to a directory and file structure. This means when the Java compiler needs to read in a specified class file, it always knows where to find it. Not so C++, have to hunt up header files and libraries manually and adjust build settings to account for their locations.
- Java has universal use of UniCode, so no need to worry if Japanese or Chinese char sets will fit in 8 bits. You have at least 4 bytes.
- Java strictly defines the size and signedness of its types. An int is always 32 bits. Unlike C++ where an int can be 2 bytes or 4 bytes depending on the processor architecture
- Java - An object must be dynamically allocated on TheHeap
- Java has built in language support for thread protection with the *synchronized* keyword plus other concurrency constructs. C++ has it as part of the C++ 11 standard but many compilers only partially support it if at all.
- Java has built-in network awareness. Network coding is MUCH, MUCH, MUCH easier to work with in Java than C++.
- Java forces you to deal with exceptions (see JavaExceptionHandling). Its optional in C++, you often have to read the docs to see if C++ will throw an exception at all.
- Java handles resource management automatically for memory only; other resources must be released explicitly each time they are used. Can use a try finally, but must know what to wrap in the {} of the try finally. In C++ you can use an objects destructor to release resources automatically, for instance handles to files. Sort of a primitive Garbage collector for resources.

- With C++, you can control what version of build tools will run your compiler. This gives you a lot more control than Java, where you may not know which version of whose JavaVirtualMachine/JIT your code is going to have to work with.

- If you want a program to run on multiple platforms (windows, Mac, and linux say) Java is a bit more convenient. Java- compile to bytecodes ONCE and submit to any of the VMs for the target platforms. C++ compile for each platform, so 3 compiles, 3 executables that only run on target platform. Portable but a pain.
- Despite all that JIT stuff, C++ code is still much faster on average (this point is not so clear cut; discussion at IsJavaSlow).