Java v/s C++ Today 2024

By, Tyler Zenisek

1. Introduction
   1. Both Java and C++ may be old and, in some areas, considered outdated languages. However, they both have played such critical roles in software development that by todays standards companies still seek out developers for both languages and both languages are still amongst the most popular in colleges to be used as languages to teach no developers the concepts of coding. But which language should take precedence in today’s technological industry?
2. Abstraction
   1. In this paper I will resolve this question by providing a brief history of the languages and many of their strengths and weaknesses. Along with some project examples that both languages will excel at. Something you will discover just as I have is that it isn’t so black and white on which language is better. Both have their uses and today play critical roles in different areas of development.
3. Overview/History
   1. C++
      1. Before C++’s creation in 1979, there existed the coding language C. Now C has been around for a long while as one of the premiere coding languages designed specifically to easily work with system hardware. C is also a compiled language which made it a very efficient programming languages heavily suited to the functionality surrounding specific hardware based devices or the programming of operating systems. These features transfer over to C++ because C++ was built upon the C library. This was important for the creation of C++ because the C compiler was already a widely distributed library that most every system featured and was compatible with. This made the portability of C++ very easy and widespread upon its creation.
      2. In 1979 C++ was created by a Danish computer scientist at AT&T Bell Labs, Bjarne Stroustrup as a language that is well known syntactically by developers of the time in a more general-purpose design. The primary way this was done was by adapting the C language to support object orientated programming through classes, structured very similarly to that of the language Simula. Originally C++ was called C with Classes but later was adapted to be called C++.
   2. Java
      1. Java was created much later than C++ in 1995 by a development team known as Team Green. This team was led by James Gosling who is widely known as the father of Java. The language was originally designed for television programming but was adapted for computers given that TV hardware at the time couldn’t handle Java’s specs. It was designed as a language since it’s adaption to computers to be object orientated, extremely robust, as simplistic as possible to read and learn, and to be highly portable like C++. C++ was one of the most popular languages at the time for wide range object orientated development that was highly portable. But, C++ was difficult to learn, syntactically difficult to read and understand and often lead to encumber some bugs.
4. Comparisons
   1. Similarities
      1. Both Java and C++ are compiled languages and are Object Orientated. Supporting the four main aspects of OOP (Inheritance, Abstraction, Polymorphism, and Encapsulation). For the most part anything you can do with classes in Java you can do in C++ and vice versa.
      2. Both are highly portable between systems with their compilers. C++ because it is based off of the C compiler which most all systems support due to the popularity of the language and the necessity of it 50 years ago. Java on the other hand created the JVM (Java Virtual Machine) to compile java code and execute as necessary on any system.
      3. Both have a large array of supporting libraries and frameworks due to their age and popularity. The primary example of this being the STL (Standard Template Library) in C++ or all of the java.util libraries in Java designed to support all aspects of development. Extending from this both languages have detailed documentation, tons of Github source code, and IDE implementation.
      4. Both feature many high level language features and are flexible in their development options. Ranging from operating system development, GUI, servers, hardware programs, databases and more
   2. Java
      1. Java uses the JVM (Java Virtual Machine) to compile and execute java code, or any language based from Java such as Kotlin. This virtual machine acts as an all interpreter, loader, compiler and executer for Java which is what allows Java to be so portable over different systems. This does cause Java to be considered a clunky and slow language despite its flexible and robust coding style. Furthermore, java has a unique way of linking memory to different aspects of its code:
         1. The Method Area which is where shared resources go. Such as variables, object identifiers, and functions.
         2. The Heap Area where all data for classes is stored.
         3. The Stack Area where threads are reserved.
         4. And PC Registers reserved for addressing.
      2. In the JVM the Execution Engine is the compiler that first maps the source code out of the interpreter to the JIT (Just-In-Time) Compiler which outputs a single bytecode file to execute. This is all followed by a garbage collection program that cleans ups the program during it’s execution to maintain efficiency and security.
      3. Moving past the JVM, Java offers a few key features over C++. One of the big ones, which many other languages also offer over C++ is the automation of addressing and memory allocation. This is both a pro and a con over C++ that for most general-purpose use cases is a nice feature to have that simplifies the programming. Java is also easier to read and structure making it very robust as a language. Java also offers code injection over C++ and a plethora of libraries/frameworks suited to GUI and Android development that C++ doesn’t. Finally, Java has its own unique feature known as Java Reflection which essentially allows the modification of classes from within other classes that can be dynamic and isn’t done via abstraction or inheritance.
   3. C++
      1. Now to start off, C++ in comparison is a more agile language that offers a steeper learning curve. The implementation of namespaces and manual addressing with pointers provides C++ with a lot of functionality but some difficult concepts to understand along with some confusing looking syntax. This does, however, segway to C++’s best features which is its ability to manual control addressing and memory allocation almost like an assembly language might do but at a higher level. This is what makes C++ so versatile and efficient by today’s standards and its own leg up over Java for many use cases. It makes it much easier to develop when the stuff C++ makes a developer do manually is done automatically but it forces slower execution and more memory usage as a result.
      2. The C++ compiler is known as being one of the best when it comes to its efficiency, accuracy in detecting syntax errors and bug defense. It isn’t as encumber some as the JVM but does mostly everything the JVM does just slightly differently and with a few more steps. The main difference being the garbage collection that the JVM does happening before program execution which is what helps C++ with its efficiency with memory and execution speed.
         1. Firstly, the compiler takes all of the source code files in txt form and transforms them into object files. This is the Processing Stage.
         2. Next the object files are tokenized and parsed into concrete trees for the Syntax Analyzer to run through the object files.
            1. During this stage is the garbage collection where any unnecessary code or memory usage is glanced over in the build process.
         3. The C++ Linker will then use these object files to create a single binary file to execute.
5. Examples
   1. C++
      1. The main take away about C++ is that it is designed for efficiency and works best with applications that work directly with any hardware system. This is why C++ is great for coding databases, servers, operating systems, or back-end code. This can entail code for audio, graphics, networking, computational geography and video. One of C++’s main areas of use is game design due to its ease of access to manipulating data over the hardware level.
      2. Some Popular C++ libraries/Frameworks:
         1. Cinder-
            1. Great for servers and graphics.
         2. Boost-
            1. Designed for easy manipulation of the file system and enhancing the standard library data structures.
         3. Wt-
            1. Wt is a library that allows programmers to create web-based GUI applications without the need of JavaScript.
         4. Yocto/GL-
            1. Designed specifically for graphics.
      3. C++ does offer specific libraries for GUI design and web development, or android development but other languages are suited better for this. C# being the main of which, which is very similarly to C++ but designed specifically for these use cases.
   2. Java
      1. Both Java and C++ are general purpose languages, but where C++ is designed for efficiency, Java has been designed for simplicity and ease of use. From my research and experience, Java is best suited for servers, Android app development, business analytics, GUI and simplistic game design.
      2. Some popular libraries and frameworks:
         1. Spring Boot-
            1. This is a very powerful and large framework designed for creating servers of all types. Netflix and Facebook were built off of this.
         2. Android Studio Java Edition-
            1. Android Studio is an IDE that specializes in android app development and does offer support in C++ and Kotlin. But, Java is the main language used for this and offers a wide range of code templates and libraries for this
         3. GWT (Google Web Kit)-
            1. This library is often paired with Spring Boot for full stack web development. GWT helps with complex web development like where dynamic use cases are necessary.
6. Personal Work
   1. C++
      1. C++ was my first language I ever coded in and it was a steep learning curve that I’m glad I committed to because it’s made every other language much easier to learn and understand, along with many of the basic concepts of coding. I find the C++ manual addressing and memory allocation very useful in designing my own C++ library for dynamic datastrucutres and my own database. It is a powerful language that I haven’t had the full opportunity to work in but it is powerful and versatile
      2. When I needed to program hardware coding on a Bluetooth device for Galileo Innovations, I would have never been able to figure it out without using C++. It is a great language for hardware coding and transporting data over a network.
   2. Java
      1. Java is much easier to code in with simpler syntax however I find Java code to be disorganized and difficult to follow at times when thousands of line of code exists across dozens of files. There are better languages that are easier to work with if your goal is GUI development. I would recommend C#.
      2. For servers however, Spring Boot is great, with a decently steep learning curve that once over offers a great range of power and versatility for API calls, data manipulation, web development and more. I personally worked in Spring Boot to create a server for a cryptocurrency application that on the back end handled everything needed to buy, sell, trade, and acquire several different types of cryptocurrencies.
      3. Android development is also heavily supported in Java and if your goal is to only code an app for android then Java would be the way to go. The supporting libraries make it easy to set up and design over other phone app frameworks like React Native or Swift Programming.
      4. The biggest thing I have found Java to be great for and why I would recommend the language for anything server based is Java Reflection. The ability to change class parameter names on the fly and check various member names makes Java great for API calls and JSON based data. This is something that isn’t offered in any other language I’ve worked in and getting around this normally requires several extra functions to be created with different parameters. Java Reflection makes things much more robust which is Java’s key feature.
   3. To find credibility to my personal experience in these languages look at my githubt
      1. https://github.com/tzenisekj
7. Summary
   1. Both Java and C++ are great languages to learn and code in. They offer a wide variety of specialties, and you can’t really go wrong with either one in most cases. Java is more popular in the industry and is easier to use. Java also offers better development for more mid to front end development. C++ is better for back-end development, or any coding required to manipulate hardware. C++ is designed for efficiency whereas Java is designed for versatility and robustness.
8. Works Cited
   1. <https://www.perforce.com/blog/qac/misra-cpp-history>
   2. <https://www.javatpoint.com/history-of-java>
   3. <https://www.infoworld.com/article/3516151/12-top-notch-libraries-for-c-plus-plus-programming.html>
   4. <https://github.com/tzenisekj>