Christopher Koepke

CS 4080.03

February 10, 2022

Language Evaluation of C++

INTRODUCTION

My first introduction into programming was in Fall 2015 at Cerritos College. I had taken a Java programming class that was required for my Associate Degree in Computer Science for Transfer. I do not remember much about how the class material was presented but I do remember how to write code in that language and if needed, look up the Java documentation for any needed resources. I do know that I have completed many projects from that class and for other classes taken at a later date, with all of my projects saved on my personal computer. The next class that I had taken related to computer programming was in Spring 2017 at the same community college. It was Computer Organization and Assembly Language Programming. I honestly do not remember much from the class but do remember that the programming language was lower-level and dealt with registers. My next programming language was C/C++ in Fall 2017 at the same community college. I later took an Object-Oriented C++ programming class in Spring 2019, again at the same community college. I transferred into Cal Poly Pomona in Fall 2020. In Spring 2021, I had a class in Systems Programming which dealt with C programming and Unix environment. In Summer 2021, I took a class in Probability and Statistics for Computer Science, in which I briefly learned about R programming. I have also dabbled in SQL for my Database System class in Fall 2021. I have a wide time frame of my programming experience, starting from Fall 2015 till the present. I have written a few Java and C++ programs for various classes during this time outside of my main programming classes. I do not write programs in my spare time for personal experience, only what was required for my classes. I do not consider myself an expert of any programming language, nor do I even consider myself as somewhat proficient in these two languages; But I am experienced enough to know how to start, get the basic design and logic of the project, and intelligent enough to research my chosen programming language in order to complete the project. I do not really have a preference for either Java or C++. As of this moment, I feel more comfortable writing code in C++ but could easily transition to Java if needed but will need to refresh myself in that language.

READABILITY OF C++ AND JAVA

What was initially hard for me during my first introduction into programming was not the readability of a program but the concepts of it. Starting my introduction in a higher-level programming language was easier since the programming language can be written using simple English statements. Some examples of this are: if, else, for, while, do…while, print, return, and many other such statements. Higher-level programming language to me is easier to read than say Assembly Language programming. An example of what I mean is shown from the response to a question posted on stackoverflow.com. The response was authored by Maximilian Schier [1].

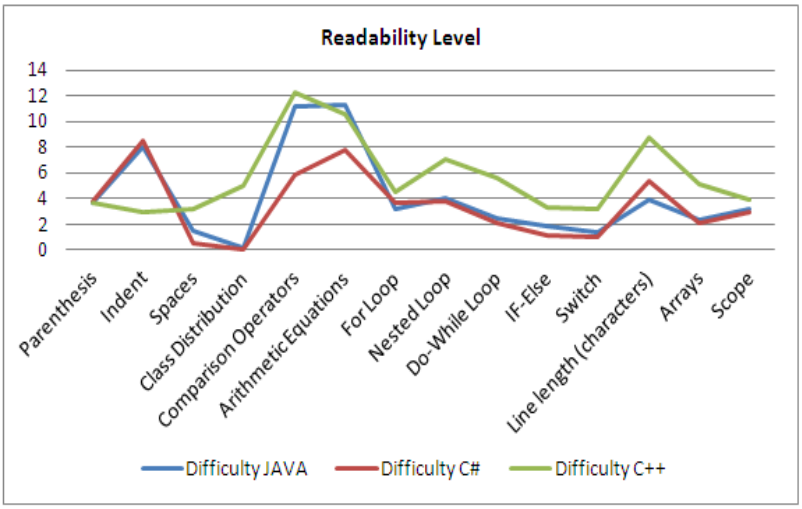
While-Loop in C:

```
While(x==1){
    //DO SOMETHING
}
```

The same loop in assembler:

```
        jmp loop1   ;  Jump to condition first
cloop1  nop         ;  Execute the content of the loop
loop1   cmp ax,1    ;  Check the condition
        je cloop1   ;  Jump to content of the loop if met
```

From this one example, it is easy to see that readability in a higher-level language is better than in a lower-level language. Between programming languages of the same level, it is a littler harder to justify one being easier to read than the other, especially once you become more proficient in a specific language and understand the context of its syntaxes. According to a very brief read of an article in the International Journal of Advanced Computer Science and Applications from March 2020, Java and C# were easier to read than C++ in many of the criteria that they had established. The article, Code Readability Management of High-level Programming Languages: A Comparative Study, showed that only parenthesis and indents were easier to read in C++ and all other criteria, C++ was harder to read [2]. The figure to the right shows their criteria on the readability level between the three programming languages, found on page 600 of their article. I personally do agree with them on some level that C++ is more difficult to read than Java, but in my limited experience, not enough to justify one being superior to the other in readability. As we become more comfortable in a language, we become more instinctive on the syntax and semantics of that product, making readability easier on the user.



WRITABILITY OF C++ AND JAVA

Writability is easier for me to justify than readability. Reading is the ability to understand the words in a sentence and the meaning of the context of that sentence. Knowing the syntax and semantics of a given programming language allows that person to read the code that makes up that program, to understand its intended function; unless that program was deliberately written to obfuscate its designed function. Even if a program was designed to conceal its logic, a person experienced in that given language will still be able to extract some understanding of its function. Writability on the other hand is almost strictly governed by a language's semantics. Writing code on a given language out of its intended order could cause its logic to become unreliable or will not even be allowed to be compile and/or interpreted, but that would not change its writability. An easy example I can give of writability is on the output between C++ and Java. In C++, to output some text you would use the "std::cout," and for Java it would be "System.out.print()." Both of these examples are considered basic output statements in their respective languages. In C++, we could shorten the output statement by including a declarative in the beginning of our program, written as "using namespace std." Another example is how the main code block of a program should be written. In C++, you write your main program as shown in the text box on the left and for Java as shown in the text box on the right.

```
int main(){
   //SOME CODE
   return 0; //OR SOME VARIATION
}
```

```
public class something{
    static void main (String[] args){
    //SOME CODE
    }
}
```

I am sure that there are other parameters that you could include in the main code block declarations of each programming language but from what I was taught, this is what I should use. There is visibly less writing in C++ than in Java from the above examples. These two simple examples I have presented are not indicative of the complexities of each programming language. A slightly more advanced example of the writability differences is in extracting a character in a string. In Java we would use the charAt() method and in C++ we would use the [] operator. An example of each is as follows, with C++ on the left and Java on the right.

```
String  example = "something";
cout << example[2];
```

```
String example = "something";
System.out.print(example.charAt(2));
```

From the two examples, only six extra characters are needed in Java in comparison to C++. These extra characters might be trivial but does show how the same operation is implemented differently between the two, with C++ using an operator versus Java using a method. From my limited perspective, both programming languages are relatively easy to write code for and are quite similar with only minor differences. If I spent more time in writing code in both languages, I am sure I would find more definitive examples of the differences in their writability.

RELIABILITY OF C++ AND JAVA

Reliability is something that I feel less confident about than I did with my explanation of writability and readability. Something that I remember specifically from this class is on our discussion of a programming language being weakly or strongly type checking. In C++, the language was deliberately designed to be more strongly typed than C but not as strongly as Java. What this means is that the language will check during compile time to make sure that the data associated with the variable is of the same type that it was declared as, meaning an integer variable is associated with an integer value, it also checks the other primitive data types, such as: char, float, double, etc. In the case of C++, it might produce some errors of mismatched types but will normally just convert the associated value to the type that was declared, such as typecasting a float value into an integer value if that variable was declared an integer type. For Java, mismatched values associated with declared variables will always produce a type mismatch error. Being strongly typed will increase the reliability of a language but not enough to justify one being better than the other since you should normally test you program before distribution and should be able to see incorrect output if you have a type mismatch. Another difference I can recall between these two languages is out-of-bounds checking. In Java, an out-of-bound index will produce an error of the same name, but in C++, it will continue to run without error but will use the data at that index, even if the data accessed is not associated with the bounds of the array. This type of issue in C++ will usually display so kind of jumbled output; I have also personally seen my program cause my computer to freeze, or outright crash the program but not the operating system. I can guess at how much of an issue this would be for a large commercial program when attempting to test it before release. So in this case, I would give the reliability crown to Java, as this issue could potentially take a long time to find and fix. Memory management in C++ is almost entirely manual, where as Java is almost entirely automatic. A good programmer should be aware of how to manage memory while writing code in C++. The larger capacities of memory available in today's machines should be less of a concern than say a few decades ago, but badly managed memory could cause memory leaks, leading to the program using all of the available memory in poorly designed programs. I have personally seen these issues in some video games where the video ram would increase the longer, I played until all available memory was used, requiring me to restart the game. To me, this is another win for Java when it comes to reliability. I understand that C++ is a completely complied language whereas Java is an interpreted language. I have limited knowledge of the reliability of interpretation and compilation, but I could see how an interpreted language could have unforeseen issues since it is running the program in a virtual environment, in the case of Java. I could see the portability benefits of interpretation but still believe it is better just to compile a program for each system you plan to release your product for. Overall, in my limited knowledge, I believe that the reliability of Java in its active approach to error checking lends it to be slightly better than C++.

CONCLUSION

Although C++ and Java are both higher-level languages with many similarities, they are different enough to justify using one over the other, depending on your preferences, or the preferences of the business who owns the resulting product. As stated in my introduction and within each section of this essay, my knowledge of both C++ and Java is limited to what I have been taught in my academic career. For readability, my familiarity with both makes it difficult to

determine whether one language is better than the other. For me personally, I feel that I can read code in C++ better than in Java, but only because I have more recent experience in that language but not enough to definitively say one is superior between the two. The readability of C++ also lends itself into writability. Because I have an easier time writing code in C++, I also have an easier time reading the code. I cannot say with certainty, but I feel that C++ uses shorter syntax for its language in comparison to Java, with the only concrete example I can give would be the setup of the main method of each, and the other examples explained in the writability section of this essay. Reliability is clearer for me to articulate. Java has more error checking and automatic memory management that C++ lacks. The common argument that I remember is that Java was designed more for portability while C++ was more for speed. I cannot attest to that statement since I do not run a large company with multiple users, servers, and other such infrastructure. The speed difference for me is negligible, but the reliability is more evident. I will give the win of reliability to Java but am not certain that Java is truly more reliable than C++. My enthusiasm for programming is minimal. I do not wish to be a professional program, I am more into networking and cybersecurity but my lack of sufficient knowledge on the topics of this class, and the process of writing this essay has made me more involved in the retention of the knowledge that you will present to us in this course.

REFERENCES

[1] M. Schier, "Stack Overflow," Stack Exchange Inc, 23 Feburary 2015. [Online]. Available: https://stackoverflow.com/questions/28665528/while-do-while-for-loops-in-assembly-language-emu8086. [Accessed 09 Feburary 2022].

[2] M. U. Tariq, M. B. Bashir, M. Babar and A. Sohail, "Code Readability Management of High-level Programming Languages: A Comparative Study," *International Journal of Advanced Computer Science and Applications,* vol. 11, no. 3, pp. 595-602, 2020.