Programming Languages Java and C++

Katheryn Knutson

*Programming Languages, Dr. Obuomar*

*Lewis University*

Romeoville, United States

katherynmknutson@lewisu.edu

*Abstract*—This is an analysis of the similarities and differences between the programming languages known as Java and C++. This paper will individually discuss each language and do a comparison between the two languages. It will also use real world examples.

Keywords—computer science, programming languages, Java, C++

# **I.** **Introduction**

I have been working on computers since I was about the age of six. My father has several certifications in the field although it is not his primary area of study. Because of his immense knowledge and skill, my father often fixed computers, built new computers, and owned many computers. Each member of my household had at least two computers and sometimes owned many more. This began my lifelong interest in the field of computer science. During high school I took computer hardware, CISCO networking, and computer graphic classes. I also took a year-long college course in coding within the Java programming language. Throughout my time in two or three years of programming I have often heard that Java and C++ were similar in syntax. I had never used C++ and was always curious about the language due to it being mentioned so often in the field. I know that many games and virtual reality worlds are coded in C# and C++ and during my recent acquisition of the role of vice president of the virtual reality club I thought it may be beneficial to me to learn more about some of the C languages. This paper was the perfect opportunity to learn more about C++ with the background knowledge I have of years of experience with Java.

# **II.** **Project Description**

## **A.** **Discussion of Java**

Java, a programming language, was developed by Sun Microsystems in the 1990s. It was primarily used for internet applications. But Java is also a “simple, efficient, general-purpose language” [1]. This language is usable across platforms, object oriented and interpreted. Java is extremely portable; “The same Java application will run identically on any computer, regardless of hardware features or operating system, as long as it has a Java interpreter” [1]. Java has several key security features such as protection from erroneous code, malicious code, and viruses. A user can safely run Java programs on their computer because the language prevents the code from accessing a hard drive or network connections. Java also has a feature called Java Applets. This feature is renowned in the computer industry as it allows for small programs to be embedded on web pages. “Java can be considered both a compiled and an interpreted language because its source code is first compiled into a binary bytecode. This byte-code runs on the [Java Virtual Machine](https://www.sciencedirect.com/topics/computer-science/java-virtual-machines) (JVM), which is usually a software-based interpreter” [1] . Due to this feature, the interpreter can run the code nearly as fast as the CPU running naive. This also allows the language to be multi-platform as it can run on any computer with the interpreter. Java is also very popular as most web browsers use it to run the applets explained earlier.

## **B.** **Discussion of C++**

C++ was developed by Bjarne Stroustrup in 1979 at Bells Lab [4]. C++ is a compiled language that is for general purpose coding [4]. It is statically typed and free form [4]. The language is also an object-oriented language, just like Java [4]. C++ is generally known as a middle-level difficulty language [4]. Both high-level and low-level programmers can use this language because techniques from all difficulties are supported. Any C program will also work in C++ because C++ is a superset of C. C++ supports encapsulation, data hiding, inheritance and polymorphism [4]. The language supports the basic building blocks of code such as data types and literals. The Standard Library of C++ helps with file manipulation and strings. The Standard Template Library is used to manipulate data structures. The language is used in almost every application domain across many programming spheres. It is often used to write drives and software for computer hardware. The difficulty and usage of basic concepts makes the language a popular one for low level programming classes. Primary user interfaces in Mac and Windows are written with C++.

# **III.** **Similarities and Differences**

## **A.** **Similarities**

C++ and Java have been known throughout the computer science world to have similar syntax. One could say that learning either language would help you learn the other language and vice versa. Commenting is an important way of letting others know the functionality of code when developing. Both languages have identical comment methods [6]. C++ and Java support inheritance and pointers, although used in different ways [5]. The primitive types in C++ and Java are nearly identical [6]. The fundamental mathematical operators in both languages do not have any significant differences [6]. The control constructs, such as if, for, while etc. are almost identical [6]. Both languages have a ‘main’ within the code in which the entry point is executed. Both languages support polymorphism, abstraction, and encapsulation [7]. Many of the keywords the languages use such as break, continue, and char are the same [7]. Both languages support multiple threading [7]. Both languages have a mass of resources and libraries [8]. C++ and Java are backwards compatible with C++ [8]. Both languages are used in Android, Mac, and Windows applications. They are used co-dependently with other languages and environments and are often used in the field to develop highly complex systems. Even with different aspects, both languages can build similarly functioning applications.

## **B.** **Differences**

There are some key differences between Java and C++, which do not make one language better than the other, but instead better for different uses than the other. For example, “compared to C++ (another object-oriented language), Java code runs a little slower (because of the JVM) but it is more portable and has much better security features” [1]. While C++ permits the user to overload the computer, Java does not. Java is also a dynamic language, meaning that the user can modify the code while it is running as C++ does not support this. This is a key feature that makes Java better for networks as they may not be able to afford downtime. C++ contains predefined data types which may change from platform to platform while Java is the same on any platform. Java is of a higher structure than C++ equivalents, “all functions (or Java methods) and executable statements in Java must reside within a class while C++ allows function definitions and lines of code to exist outside of classes (as in C-style programs). Global data and methods cannot reside outside of a class in Java, whereas C++ allows this” [1]. This allows the security and forces object orientedness of Java while it also lets C++ be a more dynamic and possibly user-friendly language. C++ is mainly used for system programming while Java is mainly used for application programming [5]. C++ supports the go to statement and operator overloading while Java does not [5]. C++ is a compiled language while Java can be interpreted and compiled [5]. C++ supports structures and unions while Java does not [5]. C++ supports call and value while Java only supports value [5]. The inheritance trees between the two languages are different: java only uses a single inheritance tree because all classes are the children of the Object class [5]. While the languages differ on some key aspects, both languages contain many similarities. However, these key differences are the reasons why the languages are best suited for different applications.

## **C.** **Problem**

The problem originates from a developer position. When given a goal, a project or an assignment from work or school, a student or developer most usually has the task of choosing the language or languages that would be best suited in achieving the goal. In some situations, companies and professors may have required language that the developer is forced to use. But in many situations in the field, the developer is going to have to make an educated decision about the language that will be used to complete a task. Some developers may choose the language that they have the most experience with and are most comfortable with; this method is immature. To best further computer science and most efficiently complete a goal, the language used needs to be the best one for the task. When choosing a language, these things need to be considered [2]:

* Type of Application
* Complexity of the Application
* Company Culture
* Time to Market
* Maintainability
* Scalability and Performance
* Security

When these requirements are considered, the choice of language is almost mathematical. But if the developer does not have the knowledge of more than one language, the choice does not matter. A well-rounded programmer may only have experience with less than five languages but would be educated on more than ten languages. This education is a key aspect in computer science. In a situation where a language needs to be chosen, and the developer looks at these considerations they may choose a language that they do not have experience with. But the project would be done in the most efficient and correct way. A skilled programmer would most likely have a less than difficult time in learning a new language in which they are educated on the way the language is written and performed.

## **D. Project Methodology**

The experiment consists of phenomena common in a coding or programming sphere. It will contain basic processing of mathematical concepts. It will include things such as if-statements and loops. The programs will take in user input and contain print statements.

We will test how the languages perform differently with different types of applications and different complexities of applications. We will test the differences between the culture surrounding the developer. The developer (author) is familiar with Java and entirely unfamiliar with C++. This will simulate a real-world application in which programmers have to adapt to different environments and learn new tools and technologies. We will test how quickly the code can be developed.

The list of experiment programs used to compare the programming languages Java and C++:

* Rock, Paper, Scissors Game [3]
* Weight Conversion Tool [3]
* Choose Your Own Adventure Text Game

All three experiments will test the ease of user input. The first experiment will test loops, comparison statements and print capability. The second experiment will test simple mathematical functionality within the languages and print statements. The third experiment will be a higher complexity test in which comparison statements and print statements will be used and stored with continuous user input.

These experiments are meant to test the necessities of a programming language. These experiments are common in introductory programming courses. If the tools and methods used in the experiments are easy to understand and use, they will most definitely be scalable for higher level projects. With each experiment I will use these concepts to compare:

* Ease of Development
* Speed of Development
* Language Performance
* Language Readability
* Compatibility with Basic Programming Fundamentals (loops, input, print statements and mathematical functions)

Since the languages are similar at their core, I assume that all of these tests will have the same outcomes in both languages.

# **IV.** **Experiment**

[yellow highlighted text indicates user input]

## **A.** **Experiments in Java**

#### **Figure 1.J** Rock, Paper, Scissors Game [3]

import java.lang.Math;

import java.util.\*;

public class RPS{

public static void rps(int i){

int rand = (int)(Math.random() \* 3);

if (rand == 0){

//rock

if (i == 1){

System.out.println("The computer picked rock!");

System.out.println("You're a Winner!");

}

else{

System.out.println("The computer picked rock!");

System.out.println("You lose!");

}

}

else if (rand == 1){

//paper

if (i == 2){

System.out.println("The computer picked paper!");

System.out.println("You're a Winner!");

}

else{

System.out.println("The computer picked paper!");

System.out.println("You lose!");

}

}

else if (rand == 2){

//scissors

if (i == 0){

System.out.println("The computer picked scissors!");

System.out.println("You're a Winner!");

}

else{

System.out.println("The computer picked scissors!");

System.out.println("You lose!");

}

}

}

public static void main(String[] args){

int result;

Scanner sc = new Scanner(System.in);

System.out.println("Welcome to the Rock, Paper Scissors Game! You will play the computer!");

System.out.print("Type 0 to choose rock, 1 to choose paper and 2 to choose scissors ");

result = sc.nextInt();

rps(result);

}

}

**Output:**

**Welcome to the Rock, Paper Scissors Game! You will play the computer!**

**Type 0 to choose rock, 1 to choose paper and 2 to choose scissors 1**

**The computer picked paper!**

**You lose!**

* **Ease of Development**

The math.random concept can be difficult. The function had to have a specific return value. The user input had to be taken in at main. The if and else statements were simple and clear. The = and == concept in Java is very difficult to understand in instances of strings and integers. This ranks as an easy development.

* **Speed of Development**

Development took around 15 minutes. Some concepts and imports were somewhat difficult to get correct. Scanners and user input can be difficult in Java. This ranks as an easy development.

* **Language Performance**

The language performed well and had a clear output statement. It compiled quickly. The error messages were clear. There was no indication of an area in which the user was supposed to enter input, only coded text can be used to indicate that. This ranks as a mid-level language performance.

* **Language Readability**

The function and series of if statements are clear and understandable. The fact that the main scanner took in the user input as an integer is something that may be confusing to developers. If the scanner took in the input at face, it would need to be type casted. The comments make the code clear. The code is concise and simple. This ranks as easy language readability.

* **Compatibility with Basic Programming Fundamentals (loops, input, print statements and mathematical functions)**

All of the if statements, print statements and mathematical functions worked with no difficulty. The java.lang math library was necessary. This ranks as highly compatible.

#### **Figure 2.J** Weight Conversion Tool [3]

import java.util.\*;

public class wct {

public static void main(String[] args){

int result;

Scanner sc = new Scanner(System.in);

System.out.println("Enter a weight in lbs and we will convert it! ");

result = sc.nextInt();

//grams

System.out.println("The weight in grams is: " + (result\*453.592));

//kilograms

System.out.println("The weight in kilograms is: " + (result\*0.000453592));

//ounces

System.out.println("The weight in ounces is: " + (result\*16));

}

}

**Output:**

**Enter a weight in lbs and we will convert it!**

**4**

**The weight in grams is: 1814.368**

**The weight in kilograms is :0.001814368**

**The weight in ounces is :64**

* **Ease of Development**

Simple mathematical concepts and print statements were the only thing necessary to complete this task. The code did need a scanner to get user input. Everything could be done in main. This ranks as an easy development.

* **Speed of Development**

This took 5 minutes to develop. The simplicity of the program made it easy to develop. It did not take many lines of code to do the mathematical calculations, they could be done inside the print statements. This ranks as a quick development.

* **Language Performance**

The language took in the user input, did calculations with it, and printed those calculations with ease. Only one library was necessary. All of the calculations worked when done inside the print statements. This language has high performance.

* **Language Readability**

The code is simple, readable, and commented. This ranks as high language readability.

* **Compatibility with Basic Programming Fundamentals (loops, input, print statements and mathematical functions)**

The calculations working with user input, inside of a print statement without flaw shows high compatibility.

#### **Figure 3.J** Choose Your Own Adventure Text Game

import java.util.\*;

public class game{

public static void main(String[] args){

Stack<String> stack = new Stack<String>();

int score = 0;

int total = 6;

Scanner sc = new Scanner(System.in);

//System.out.println("");

System.out.println("You are in a forest, you do not know how you got there...");

System.out.println("You hear a loud scream behind you");

System.out.println("Do you run, or hide? ");

String decide = sc.next();

if (decide.contains("run")){

stack.push("run");

score = score +1;

System.out.println("You run and run as fast as you can hearing footsteps behind you");

System.out.println("After a while of running the footsteps fade, you are alone");

System.out.println("When you catch your breath you see a small hut in the distance");

System.out.println("Do you go towards the hut or keep going straight? ");

decide = sc.next();

if (decide.contains("hut")){

stack.push("hut");

score = score + 1;

System.out.println("you walk up to the hut, knock on the door....");

System.out.println("no answer");

System.out.println("You push the door and by your luck it is unlocked");

System.out.println("Do you explore the main floor or the basement? ");

decide = sc.next();

if (decide.contains("basement")){

stack.push("basement");

score = score + 2;

System.out.println("You walk to the basement doors no where near the kitchen");

System.out.println("You open the door and try to turn on the light but it does not work");

System.out.println("You slowly go down the large staircase");

System.out.println("You feel the walls trying to get a grasp of where you are");

System.out.println("Suddenly a small blue light emerges...");

System.out.println("You go towards the light and when you reach into it you find...");

System.out.println("A magic 8 ball!");

stack.push("8-ball winner");

System.out.println("Congratulations");

}

if (decide.contains("main")){

stack.push("main floor");

score = score + 1;

System.out.println("You're walking around carefully making sure no one is in the hut");

System.out.println("You explore the living room");

System.out.println("Nothing but dusty photos of weird looking children in there");

System.out.println("You start heading to the kitchen when...");

System.out.println("BOOM. Floor falls out from beneath you");

stack.push("dead");

}

}

if (decide.contains("straight")){

stack.push("straight");

score = score + 1;

System.out.println("You keep running past the hut hoping to find some end to this madness");

System.out.println("You look behind you to see if anyone is following");

System.out.println("While your head is turned you run into a tree");

stack.push("dead");

}

else{

stack.push("invalid");

}

}

else if (decide.contains("hide")){

stack.push("hide");

score = score + 1;

System.out.println("You hide for a while...");

System.out.println("You haven't heard anything for a long time now...");

System.out.println("You come out of hiding to see if the coast is clear");

System.out.println("You get shot in the head with an arrow");

stack.push("dead");

}

else{

stack.push("invalid");

}

System.out.println("Here are the choices you made:");

System.out.println(stack);

System.out.println("Your score is " + score + "/" + total );

sc.close();

}

}

**Outputs depend on user choice, all outputs work as developed and equally in both languages.**

* **Ease of Development**

This was an easy development with simple concepts. Although this code has the most lines out of all the codes, it is the one with the simplest concepts.

* **Speed of Development**

This was a quick development. The concepts were easy to understand and apply. There were not many errors that needed to be fixed.

* **Language Performance**

This language has a high performance. The stack structure worked flawlessly throughout the entire application and printed easily. The user input and regular expression testing worked without flaw as well.

* **Language Readability**

The language is extremely readable and simple. A beginner would have an easy time just looking at the code and being able to understand it from that.

* **Compatibility with Basic Programming Fundamentals (loops, input, print statements and mathematical functions)**

All of the concepts used in this code were extremely compatible.

## **B.** **Experiments in C++**

[yellow highlighted text indicates user input]

#### **Figure 1.C** Rock, Paper, Scissors Game [3]

#include <iostream>

#include <string>

#include <vector>

#include <math.h>

using namespace std;

string rps(int i);

int main()

{

int result;

cout << "Welcome to the Rock, Paper Scissors Game! You will play the computer!";

cout << "Type 0 to choose rock, 1 to choose paper and 2 to choose scissors ";

cin >> result;

cout << rps(result);

return 0;

}

string rps(int i){

int random = rand() % 3;

if (random == 0)

{

// rock

if (i == 1)

{

std::cout << "The computer picked rock!" << std::endl;

std::cout << "You\'re a Winner!" << std::endl;

}

else

{

std::cout << "The computer picked rock!" << std::endl;

std::cout << "You lose!" << std::endl;

}

}

else

{

if (random == 1)

{

// paper

if (i == 2)

{

std::cout << "The computer picked paper!" << std::endl;

std::cout << "You\'re a Winner!" << std::endl;

}

else

{

std::cout << "The computer picked paper!" << std::endl;

std::cout << "You lose!" << std::endl;

}

}

else

{

if (random == 2)

{

// scissors

if (i == 0)

{

std::cout << "The computer picked scissors!" << std::endl;

std::cout << "You\'re a Winner!" << std::endl;

}

else

{

std::cout << "The computer picked scissors!" << std::endl;

std::cout << "You lose!" << std::endl;

}

}

}

}

}

**Output:**

**Welcome to the Rock, Paper Scissors Game! You will play the computer!Type 0 to choose rock, 1 to choose paper and 2 to choose scissors 1**

**The computer picked paper!**

**You lose!**

* **Ease of Development**

This code was somewhat difficult to create because of the intricacies of C++. The functions must be declared before main but created after main. That order can be confusing to beginners and difficult because the organization and planning happens in many different areas. Taking in user input and creating random numbers is easier in C++ than Java. This ranks as an easy development.

* **Speed of Development**

This code took the same time to develop as the Java code did. It is simple, with easy concepts. This ranks as a quickly developable program.

* **Language Performance**

The language performed well with the user input and random number creation. It completed the task without any clear issues. This ranks as high-level performance.

* **Language Readability**

The functions being initialized before main but created after main is confusing. To the beginner’s eye, the taking in of user input is not clearly defined. This ranks as a mid-level readable program.

* **Compatibility with Basic Programming Fundamentals (loops, input, print statements and mathematical functions)**

Getting random numbers in C++ is easy and the math library is all encompassing. The language is compatible with user input, if statements and print statements and completed the goal of the task without any issue. This ranks as highly compatible.

#### **Figure 2.C** Weight Conversion Tool [3]

#include <iostream>

#include <string>

#include <vector>

#include <math.h>

using namespace std;

int main()

{

int result;

cout << "Enter a weight in lbs and we will convert it! ";

cin >> result;

cout << "The weight in grams is: " << (result\*453.592);

cout << "The weight in kilograms is: " << (result\*0.000453592);

cout << "The weight in ounces is: " << (result\*16);

return 0;

}

**Output:**

**Enter a weight in lbs and we will convert it! 4**

**The weight in grams is: 1814.37The weight in kilograms is: 0.00181437The weight in ounces is: 64**

* **Ease of Development**

Simple mathematical concepts and print statements were the only thing necessary to complete this task. The code did need a scanner to get user input. Everything could be done in main. This ranks as an easy development.

* **Speed of Development**

This took 5 minutes to develop. The simplicity of the program made it easy to develop. It did not take many lines of code to do the mathematical calculations, they could be done inside the print statements. This ranks as a quick development.

* **Language Performance**

The language took in the user input, did calculations with it, and printed those calculations with ease. Only one library was necessary. All the calculations worked when done inside the print statements. This language has high performance.

* **Language Readability**

The code is simple, readable, and commented. This ranks as high language readability.

* **Compatibility with Basic Programming Fundamentals (loops, input, print statements and mathematical functions)**

The calculations working with user input, inside of a print statement without flaw shows high compatibility.

#### **Figure 3.C** Choose Your Own Adventure Text Game

#include <iostream>

#include <stack>

using namespace std;

int main() {

int score = 0;

int total = 4;

string decide = "";

stack<string> stack;

cout<<"You are in a forest, you do not know how you got there...\n";

cout<<"You hear a loud scream behind you\n";

cout<<"Do you run, or hide? \n";

cin >> decide;

if (decide.find("run") != std::string::npos){

stack.push("run");

score = score +1;

cout<<"You run and run as fast as you can hearing footsteps behind you\n";

cout<<"After a while of running the footsteps fade, you are alone\n";

cout<<"When you catch your breath you see a small hut in the distance\n";

cout<<"Do you go towards the hut or keep going straight? \n";

cin >> decide;

if (decide.find("hut") != std::string::npos){

stack.push("hut");

score = score + 1;

cout<<"you walk up to the hut, knock on the door....\n";

cout<<"no answer\n";

cout<<"You push the door and by your luck it is unlocked\n";

cout<<"Do you explore the main floor or the basement? \n";

cin >> decide;

if (decide.find("basement") != std::string::npos){

stack.push("basement");

score = score + 2;

cout<<"You walk to the basement doors no where near the kitchen\n";

cout<<"You open the door and try to turn on the light but it does not work\n";

cout<<"You slowly go down the large staircase\n";

cout<<"You feel the walls trying to get a grasp of where you are\n";

cout<<"Suddenly a small blue light emerges...\n";

cout<<"You go towards the light and when you reach into it you find...\n";

cout<<"A magic 8 ball!\n";

stack.push("8-ball winner");

cout<<"Congratulations\n";

}

if (decide.find("main") != std::string::npos){

stack.push("main floor");

score = score + 1;

cout<<"You're walking around carefully making sure no one is in the hut\n";

cout<<"You explore the living room\n";

cout<<"Nothing but dusty photos of weird looking children in there\n";

cout<<"You start heading to the kitchen when...\n";

cout<<"BOOM. Floor falls out from beneath you\n";

stack.push("dead");

}

}

if (decide.find("straight") != std::string::npos){

stack.push("straight");

score = score + 1;

cout<<"You keep running past the hut hoping to find some end to this madness\n";

cout<<"You look behind you to see if anyone is following\n";

cout<<"While your head is turned you run into a tree\n";

stack.push("dead");

}

else{

stack.push("invalid");

}

}

else if (decide.find("hide") != std::string::npos){

stack.push("hide");

score = score + 1;

cout<<"You hide for a while...\n";

cout<<"You haven't heard anything for a long time now...\n";

cout<<"You come out of hiding to see if the coast is clear\n";

cout<<"You get shot in the head with an arrow\n";

stack.push("dead");

}

else{

stack.push("invalid");

}

cout<<"Here are the choices you made:\n";

while(!stack.empty()) {

cout << stack.top() << " \n";

stack.pop();

}

cout<<"Your score is "<< score << "/" << total ;

}

**Outputs depend on user choice, all outputs work as developed and equally in both languages.**

* **Ease of Development**

This development had a high level of difficulty. The difficulty is extreme for beginners but is at the ‘frustrating’ level for developers who are familiar with other languages.

* **Speed of Development**

The development for this code took the longest out of all the codes researched. Checking the strings against other strings, also known as regular expression testing, is a concept that is not too well done in C++. It is complicated and took much research. However, the other elements were very simple. This is a somewhat quick level of development.

* **Language Performance**

The language did well with taking in user input, it is arguably easier to do that in C++ than it is in Java on a small scale. The usage of the stack concept was similar if not the same. The if-statements worked with no issue. The language performance is high.

* **Language Readability**

When all the issues were worked out, the language is arguably more readable than the Java language. The final print statement for the stack is entirely confusing and took a lot of research. Checking a string against another string is almost completely unreadable in C++. Without intricate knowledge of the details of this language, some concepts in this code do not make sense. This is a mid-level readability problem.

* **Compatibility with Basic Programming Fundamentals (loops, input, print statements and mathematical functions)**

The user input, print statements and if-statements worked flawlessly. These concepts are compatible. Concepts like regular expression checking and printing complicated techniques such as stacks are not compatible.

# **VI.** **Experiment Results**

## **A.** **Discussion of Results**

My prediction was that both languages would perform similarly in almost all implementations. C++ and Java performed equally well with:

* Print statements
* If statements
* Loops
* Mathematical Operations

Java performed better with its handling of high-level concepts such as stacks and queues. Java had the ability to directly access the information within the stack. This makes Java better than C++ to deal with stacks sand queues because the information is mutable and accessible.

C++ performed better with its handling of user-input. Java needs to import a scanner library to access user input. Scanners are not dependable, and how they take in information can be difficult. Scanners can take in entire lines, the next input and more. With a scanner, it must be specified as to what it is taking in such as a string or integer or the information must be parsed later. With C++ a variable is initialized, then the program takes in user input and assigns it to the variable. That variable is already declared as a string or an integer, no parsing is needed.

Although each language had its minor successes, the differences are negligible.

## **B.** **Conclusion**

I have concluded that it would take the same amount of time to solve a problem in each language. Each of the codes are equally readable. They both handle operations and low to high level concepts similarly.

Java was easier to develop because of my experience with it. I used the things I knew about Java and applied them to C++. Although this strategy did not work every time, it helped develop each of the codes in a similar fashion and time period.

Throughout my research I learned that the applications for each language in the real world are entirely different. While each language would perform similarly in a small-scale academic pretense, they would perform differently in a high-level environment in which larger problems needed to be solved.

I propose that high level projects be conducted to reach a more accurate conclusion.

##### **References**

[1] Austerlitz, Howard. *Data Acquisition Techniques Using PCs*. 2nd ed., 2003, Accessed 24 Jan. 2022.

[2] Nguyen, Phuong. *How to choose the best programming language for a software development project*, 27Global, 17 Nov. 2017. Accessed 20 Mar. 2022.

[3] Grant, Anthony. "Want to learn programming but don't know where to start? These beginner programming projects and tutorials will start you off." *The 10 Best Beginner Projects for New Programmers*, Make Use Of, 20 Sep. 2021. Accessed 20 Mar. 2022.

[4] *C++ Quick Guide*, sceweb.sce.uhcl.edu/helm/WEBPAGE-Cpp/my\_files/Quick%20Guide/cpp\_overview.html. Accessed 21 Mar. 2022.

[5] *C++ vs Java* , JavaPoint, www.javatpoint.com/cpp-vs-java. Accessed 21 Mar. 2022.

[6] Nguyen. *Similarities between Java and C++*, CSULB, home.csulb.edu/~pnguyen/cecs282/lecnotes/similarities.pdf. Accessed 21 Mar. 2022.

[7] *Similarities between Java and C++*, tutorialspoint, tutorialspoint.dev/language/cpp/similarities-between-java-and-c. Accessed 21 Mar. 2022.

[8] *Java Vs C++: Differences, Similarities and Importance*, Coding Ninjas, 19 Aug. 2021, www.codingninjas.com/blog/2021/08/19/java-vs-c-differences-similarities-and-importance/. Accessed 21 Mar. 2022.