Tanner Kleintop

Professor Richard Martin

Computer Science 122

2 May 2023

SRTE Exercise

1043 Words

"C++ vs Java: Comparing Object-Oriented Programming Languages"

C++ and Java are two of the more popular object-oriented programming (OOP) languages used in the world today. Both languages were developed to make it easier and more efficient to make code look nicer and have less clutter to it. While they can achieve similar goals in terms of OOP, some key differences should be considered when looking at the two.

To give some background, C++ was developed in the 1980s by Bjarne Stroustrup. Stroustrup at the time worked at Bell Labs looking for a way to extend the C programming language to allow for support of object-oriented programming. The first version of C++ would release in 1985, it would quickly gain popularity among programmers due to its ability to be efficient and flexible.

One key design of C++ was efficiency. This is why C++ would is a compiling language, this means that the code translates from code into machine-readable instructions before being executed by the computer. This allows C++ to run its programs faster than other languages like Python, or Javascript. C++ also includes several advanced features such as templates, operator overloading, and multiple inheritance, that make it well-suited for building complex software systems.

In terms of object-oriented programming for C++, many different basic parts were available like encapsulation, inheritance, and polymorphism. C++ allowed for the creation of objects to encapsulate data and behavior into a single part of the code. It also allowed for the creation of subclasses that can hold properties and behaviors to their parent functions. C++ also had support for dynamic binding which allowed programs to determine a runtime and method to call based on the objects involved with a specific code.

Java, on the other hand, was developed by a team at Sun Microsystems in the early 1990s. The team was led by James Gosling, who was looking for a way to create a portable programming language that could run on a variety of hardware platforms. The first version of Java was released in 1995, and it quickly became popular among developers due to its simplicity and portability.

One of the key features developed with Java was the ability to be as portable as possible. This is why Java would be an interpreted language, this means that the code is run by a virtual machine rather than being directly complied by the machine itself. This approach allows Java to be a highly portable program, as it can run with any device that supports and has Java Virtual Machine (JVM) installed. Java also includes several advanced features, such as garbage collection and automatic memory management, that make it well-suited for building large-scale enterprise applications.

In terms of object-oriented programming, Java shares many similarities with C++. Both languages support the basic principles of OOP such as encapsulation, inheritance, and polymorphism. They both also allow for the creation of objects that can encapsulate data into a single unit. Both also allow for the creation of subclasses that inherit properties and behavior to that of the parent class. They also support dynamic binding, which allows a program to determine at runtime which method to call based on the type of object involved.

On the other hand, there are also many significant differences between the two languages when it comes to Object-oriented programming. One major difference is that Java does not support multiple inheritance, while C++ does. Multiple inheritance allows a class to inherit from more than one parent class, which can be useful in certain situations but can also lead to complications and ambiguity in the code.

Another difference is that C++ allows for more low-level memory manipulation than Java does. This can be useful in certain situations where performance is critical, but it can also lead to errors and security vulnerabilities if not used carefully.

Additionally, another significant difference between C++ and Java in terms of OOP is the approach to garbage collection. In C++, the programmer is responsible for managing memory allocation and deallocation, which can be a complex and error-prone task. C++ provides features such as destructors and smart pointers to help with memory management, but ultimately, it is up to the programmer to ensure that memory is used efficiently and correctly. Java uses automatic memory management through a process known as garbage collection. The JVM automatically manages the allocation and deallocation of memory for objects, freeing up memory when it is no longer needed. This approach to memory management makes Java more convenient and less error-prone than C++.

Another significant difference between the two languages is the way they handle exceptions. In C++, exceptions are typically thrown as objects and caught using try-catch blocks. This can be useful in certain situations, but it can also be error-prone if not used correctly. In Java, exceptions are handled using the try-catch-finally block, which provides a more structured and safer way of handling exceptions.

Finally, Both C++ and Java have different levels of platform independence. While Java is designed to be heavily portable-friendly with JVM, C++ code needs to be recompiled for each platform of intended use. This can make C++ more difficult to work with as you add multiple machines into a situation, but it can also lead to a more efficient and flexible workspace.

In conclusion, both C++ and Java are two excellent and powerful choices when it comes to the object-oriented programming portion of coding. They both have great approaches to OOP with many good factors to look at. However, they also have some key differences that put them apart, these key differences should be taken into consideration before starting a project with object-oriented programming. C++ is a compiled language that is well known for its efficiency and its list of advanced features such as templates and multi-inheritances while Java is an interpreted language that prioritizes portability over features. Java also gives the user ease of use through features such as automatic memory management and garbage collection, feature C++ is lacking. Other differences between the two languages could be the memory management approach, exception handling, and platform independence. Ultimately, the choice between C++ and Java is entirely up to the user, but the specific needs for each project should be looked at to make use of the powerful features that Java has over C++ or vice versa!

Works Cited

Coursera. “Java Vs. C++ Comparison: What Are the Differences in These Programming Languages?” *Coursera*, Aug. 2022, [www.coursera.org/articles/java-vs-c](http://www.coursera.org/articles/java-vs-c).

Stroustrup, Bjarne. “What Is Object-oriented Programming?” *IEEE Software*, vol. 5, no. 3, IEEE Computer Society, May 1988, pp. 10–20, doi:10.1109/52.2020.

GeeksforGeeks. “Similarities and Difference Between Java and C.” *GeeksforGeeks*, Apr. 2023, [www.geeksforgeeks.org/similarities-and-difference-between-java-and-c](http://www.geeksforgeeks.org/similarities-and-difference-between-java-and-c).

Admin. “Java Vs C++: Comparison, Key Differences, and Similarities.” *CodeAvail*, 13 May 2022, www.codeavail.com/blog/java-vs-c-plus-plus.

*What Is Java? | IBM*. [www.ibm.com/topics/java](http://www.ibm.com/topics/java).

GeeksforGeeks. “Object Oriented Programming in C.” *GeeksforGeeks*, Apr. 2023, www.geeksforgeeks.org/object-oriented-programming-in-cpp.