Java vs C++

Java is a general-purpose, high-level programming language. Java is used for web development, Machine Learning, and other cutting-edge software development. Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE])

C++ is a middle-level, case-sensitive, object-oriented programming language. Bjarne Stroustrup created C++ at Bell Labs. C++ is a platform-independent programming language that works on Windows, Mac OS, and Linux. C++ is near to hardware, allowing low-level programming. This provides a developer control over memory, improved performance, and dependable software.

Read through this article to get an overview of C++ and Java and how these two programming languages are different from each other.

What is Java?

The latest release of the Java Standard Edition is Java SE 21. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

The new J2 versions were renamed as Java SE, Java EE, and Java ME respectively. Java is guaranteed to be **Write Once, Run Anywhere.**

Features of Java

Java is –

- **Object Oriented** – In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform Independent** – Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Java Virtual Machine (JVM) on whichever platform it is being run on.

- **Simple** – Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
- **Secure** – With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architecture-neutral** – Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
- **Portable** – Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
- **Robust** – Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multithreaded** – With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
- **Interpreted** – Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
- **High Performance** – With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed** – Java is designed for the distributed environment of the internet.
- **Dynamic** – Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

Java Example

Take a look at the following simple Java program –

```
package com.tutorialspoint;

import java.util.Scanner;

public class JavaTester {
```

```
   public static void main(String args[]) {
      String a, b;
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter The value for variable a");
      a = scanner.nextLine();
      System.out.println("Enter The value for variable b");
      b = scanner.nextLine();

      System.out.println("The value you have entered for a is " + a);
      System.out.println("The value you have entered for b is " + b);
      scanner.close();
   }
}
```

In our example, we have taken two variables "a" and "b" and assigning some value to those variables. Note that in Java, we need to declare datatype for variables explicitly, as Java is strictly typed language. As Java is an object oriented language, we uses objects to perform any action. In the example, we've used Scanner class object to read user input from console which is represented by System.in object. System.out object method println() is used to print the values received.

Output

On execution, this Java code will produce the following output –

```
Enter The value for variable a
10
Enter The value for variable b
20
The value you have entered for a is 10
The value you have entered for b is 20
```

What is C++?

C++ is a statically typed, compiled, multi-paradigm, general-purpose programming language with a steep learning curve. Video games, desktop apps, and embedded systems use it extensively. C++ is so compatible with C that it can build practically all C source code without any changes. Object-oriented programming makes C++ a better-structured and safer language than C.

Features of C++

Let's see some features of C++ and the reason of its popularity.

- **Middle-level language** – It's a middle-level language since it can be used for both systems development and large-scale consumer applications like Media Players, Photoshop, Game Engines, etc.
- **Execution Speed** – C++ code runs quickly. Because it's compiled and uses procedures extensively. Garbage collection, dynamic typing, and other modern features impede program execution.
- **Object-oriented language** – Object-oriented programming is flexible and manageable. Large apps are possible. Growing code makes procedural code harder to handle. C++'s key advantage over C.
- **Extensive Library Support** – C++ has a vast library. Third-party libraries are supported for fast development.

C++ Example

Let's understand the syntax of C++ through an example written below.

```
#include
using namespace std;

int main() {
   int a, b;
   cout << "Enter The value for variable a \n";
   cin >> a;
   cout << "Enter The value for variable b";
   cin >> b;
   cout << "The value of a is "<< a << "and" << b;
   return 0;
}
```

In our example, we are taking input for two variables "a" and "b" from the user through the keyboard and displaying the data on the console.

Output

On execution, it will produce the following **output** –

```
Enter The value for variable a
10
Enter The value for variable b
20
The value of a is 10 and 20
```

Difference Between Java and C++

Both Java and C++ are among the most popular programming languages. Both of them have their advantages and disadvantages. In this tutorial, we shall take a closure look at their characteristic features which differentiate one from another.

Sr.No.	Criteria	Java	C++
1	Developed by	Java was developed by James Gosling at Sun Microsystems. Initially it was designed for embedded systems, settop boxes, televisions etc. Later it become a preferred language for internet based application development	C++ was developed by Bjarne Stroustrup at Bell Labs, as an extension to C language. It supports both high level as well as low level machine code access. C++ is mainly used to develop system softwares, compilers etc.
2	Influenced by	It was influenced by Ada 83, Pascal, C++, C#.	It was influenced by Ada, ALGOL 68, C, ML, Simula, Smalltalk.
3	Dependency on Architecture	The Java bytecode works on any operating System. Bytecode is targeted for JVM. JVM or Java Virtual Machine then interpret the byte code and run the underlying machine specific code. Thus Java code needs not to be	It doesn't work on every operating system since libraries are different on different systems.

		changed to run on different machines.	
4	Platform independence	It can run on any OS. Java code is platform independent. No platform specific code is required. Size of int, long remains same on all platforms.	It is compiled differently on different platforms, can't be run on any OS.
5	Portablity	It is portable. Being platform independent, a java code can be transferred as it is on any machine without any plaform specific modification. A java code written in Windows machine can run in Unix machine in same fashion without any modification.	It isn't portable.
6	Interpreted/Compiled	It is an interpreted language.	It is a compiled language.
7	Memory management	Memory management is done automatically. Java provides Garbage Collector service which automatically deallocates memory once an object is not required.	Memory management is to be done manually.
8	virtual Keyword	It doesn't have ‘virtual' keyword.	It has the ‘virtual' keyword.
9	Multiple Inheritance support	It supports single inheritance only. Multiple inheritance can be achieved using interfaces (partial only).	It supports single and multiple Inheritance. Using virtual keyword, ambigous reference can be resolved.

		A class can extend only a single class. Although interface can extend multiple inheritance. Multiple inheritance can lead to ambigous results. As virtual keyword is not supported in Java, multiple inhertance is not supported.	
10	operator overloading support	It doesn't support operator overloading. Java supports only method overloading. Operator overloading is considered to add the complexity to base language so is not implemented to keep language simple.	It supports operator overloading. In C++, we can overload both methods and operators.
11	pointers	It provides limited support to pointers. Pointers being a complex functionality, Java refrains from using them. It provides concept of reference to point to objects or precisely their addresses.	It supports pointer operations. Developers can perform complex operations, can write optimized memory based code using pointers. But it is quite complex and requires strong programming skills to master them.
12	Low level machine code access	They have high level functionalities. Java is platform independent language and the compiled code of java as byte code is for JVM which further converts code to low level code. So using java,	They have low level functionalities. As C++ supports low level machine code code. It is mainly used to write system softwares, compilers etc.

		developer cannot write low level machine code. This is the reason, that Java is mainly used for application development.	
13	Native libraries access	It doesn't support direct native library call. Java is not designed to work with low level machine code and it is not supporting native call. But we can configure native call using third party libraries.	It supports direct system library calls.
14	documentation comment	It supports **documentation comment** (/**.. */) for source code. Javadoc tool can read the documentation comments from source code and generate html based java documentation based on the comments.	It doesn't support documentation comment for source code.
15	Multithreading	It supports thread operations. Java has by default support for multithreading. It allows concurrent programming to improve efficiency and to reduce time taken	It doesn't support threads by design. It can be done by using third party threading libraries.
16	Console Input	It uses the **'System' class**, i.e System.in for input. System.in class can be used to take	It uses **'cin'** for input operation. cin allows user to enter value in console.

		input from user on console.	
17	Console Output	It uses System.out for output. System.out.println() method prints the required value on system's console.	It uses **'cout'** for an output operation. cout prints the required value on system's console.
19	global support	It doesn't support global scope. Java is a strict object oriented language and global scope is not available. Using packages, it supports across package scope though.	It supports global scope as well as namespace scope.
20	struct/union support	It doesn't support structures and unions.	It supports structures and unions.
21	goto keyword	It doesn't have the 'goto' keyword. But same functionality is achivable using label. A break/continue statement can jump to a labelled statement location.	It supports the **'goto' keyword**. Using goto keyword, we can jump to any labelled location.
22	pass by value/reference	It supports Pass by Value method only. Even object reference is technically passed to a method as value.	It supports Pass by Value and pass by reference methods. In case of pass by reference, pointer or & notation is required.
23	Memory Management	It performs object management automatically using garbage collector. Developers are not required to do memory	It performs object management manually with the help of 'new' and 'delete'. Developers have to take measures to

		allocation for objects or deallocate them when objects are not in use. Garbage Collector service automatically detects and frees up the space. Due to GC service, memory leaks are very less possible in Java.	ensure memory is properly allocated/deallocated to prevent memory leaks.