Introduction to Java

JAVA was developed by James Gosling at **Sun Microsystems** Inc in the year **1991**, later acquired by Oracle Corporation. It is a simple programming language. Java makes writing, compiling, and debugging programming easy. It helps to create reusable code and modular programs.

Java is a class-based, object-oriented programming language and is designed to have as few implementation dependencies as possible. A general-purpose programming language made for developers to *write once run anywhere* that is compiled Java code can run on all platforms that support Java. Java applications are compiled to byte code that can run on any Java Virtual Machine. The syntax of Java is similar to c/c++.

History

Java’s history is very interesting. It is a programming language created in 1991. James Gosling, Mike Sheridan, and Patrick Naughton, a team of Sun engineers known as the **Green team** initiated the Java language in 1991. **Sun Microsystems** released its first public implementation in 1996 as **Java 1.0**. It provides no-cost -run-times on popular platforms. Java1.0 compiler was re-written in Java by Arthur Van Hoff to strictly comply with its specifications. With the arrival of Java 2, new versions had multiple configurations built for different types of platforms.

In 1997, Sun Microsystems approached the ISO standards body and later formalized Java, but it soon withdrew from the process. At one time, Sun made most of its Java implementations available without charge, despite their proprietary software status. Sun generated revenue from Java through the selling of licenses for specialized products such as the Java Enterprise System.

On November 13, 2006, Sun released much of its Java virtual machine as free, open-source software. On May 8, 2007, Sun finished the process, making all of its JVM's core code available under open-source distribution terms.

The principles for creating java were simple, robust, secured, high performance, portable, multi-threaded, interpreted, dynamic, etc. **James** Gosling in 1995 developed Java, who is known as the Father of Java. Currently, Java is used in mobile devices, internet programming, games, e-business, etc.

Java programming language is named JAVA. Why?

After the name OAK, the team decided to give a new name to it and the suggested words were Silk, Jolt, revolutionary, DNA, dynamic, etc. These all names were easy to spell and fun to say, but they all wanted the name to reflect the essence of technology. In accordance with James Gosling, **Java** the among the top names along with **Silk**, and since java was a unique name so most of them preferred it. Java is the name of an **island** in Indonesia where the first coffee(named java coffee) was produced. And this name was chosen by James Gosling while having coffee near his office. Note that Java is just a name, not an acronym.

Java Terminology

Before learning Java, one must be familiar with these common terms of Java.

1. Java Virtual Machine(JVM): This is generally referred to as JVM. There are three execution phases of a program. They are written, compile and run the program.

- Writing a program is done by a java programmer like you and me.

- The compilation is done by the **JAVAC** compiler which is a primary Java compiler included in the Java development kit (JDK). It takes Java program as input and generates bytecode as output.
- In the Running phase of a program, **JVM** executes the bytecode generated by the compiler.

Now, we understood that the function of Java Virtual Machine is to execute the bytecode produced by the compiler. Every Operating System has a different JVM but the output they produce after the execution of bytecode is the same across all the operating systems. This is why Java is known as a **platform-independent language.**

2. Bytecode in the **Development process:** As discussed, the Javac compiler of JDK compiles the java source code into bytecode so that it can be executed by

JVM. It is saved as **.class** file by the compiler. To view the bytecode, a disassembler like javap can be used.

3. Java Development Kit(JDK): While we were using the term JDK, when we learn about bytecode and JVM . So, as the name suggests, it is a complete Java development kit that includes everything including compiler, Java Runtime Environment (JRE), java debuggers, java docs, etc. For the program to execute in java, we need to install JDK on our computer in order to create, compile and run the java program.

4. Java Runtime Environment (JRE): JDK includes JRE. JRE installation on our computers allows the java program to run, however, we cannot compile it. JRE includes a browser, JVM, applet supports, and plugins. For running the java program, a computer needs JRE.

5. Garbage Collector: In Java, programmers can’t delete the objects. To delete or recollect that memory JVM has a program called Garbage Collector. Garbage

Collectors can recollect the of objects that are not referenced. So Java makes the life of a programmer easy by handling memory management. However, programmers should be careful about their code whether they are using objects that have been used for a long time. Because Garbage cannot recover the memory of objects being referenced.

6. ClassPath: The classpath is the file path where the java runtime and Java compiler look for **.class** files to load. By default, JDK provides many libraries. If you want to include external libraries they should be added to the classpath.

Primary/Main Features of Java

1. Platform Independent: Compiler converts source code to bytecode and then the JVM executes the bytecode generated by the compiler. This bytecode can run on any platform be it Windows, Linux, macOS which means if we compile a program on Windows, then we can run it on Linux and vice versa. Each operating system has a different JVM, but the output produced by all the OS is the same after the execution of bytecode. That is why we call java a platform-independent language.

2. Object-Oriented Programming Language: Organizing the program in the terms of collection of objects is a way of object-oriented programming, each of which represents an instance of the class.

The four main concepts of Object-Oriented programming are:

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

3. Simple: Java is one of the simple languages as it does not have complex features like pointers, operator overloading, multiple inheritances, Explicit memory allocation.

4. Robust: Java language is robust that means reliable. It is developed in such a way that it puts a lot of effort into checking errors as early as possible, that is why the java compiler is able to detect even those errors that are not easy to detect by another programming language. The main features of java that make it robust are garbage collection, Exception Handling, and memory allocation.

5. Secure: In java, we don’t have pointers, and so we cannot access out-of-bound arrays i.e it shows **ArrayIndexOutOfBound Exception** if we try to do so. That’s why several security flaws like stack corruption or buffer overflow is impossible to exploit in Java.

6. Distributed: We can create distributed applications using the java programming language. Remote Method Invocation and Enterprise Java Beans are used for creating distributed applications in java. The java programs can be easily distributed on one or more systems that are connected to each other through an internet connection.

7. Multithreading: Java supports multithreading. It is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU.

8. Portable: As we know, java code written on one machine can be run on another machine. The platform-independent feature of java in which its platform-independent bytecode can be taken to any platform for execution makes java portable.

9. High Performance: Java architecture is defined in such a way that it reduces overhead during the runtime and at some time java uses Just In Time (JIT) compiler where the compiler compiles code on-demand basics where it only compiles those methods that are called making applications to execute faster.

10. Dynamic flexibility: Java being completely object-oriented gives us the flexibility to add classes, new methods to existing classes and even creating new classes through sub-classes. Java even supports functions written in other languages such as C, C++ which are referred to as native methods.

11. Sandbox Execution: Java programs run in a separate space that allows user to execute their applications without affecting the underlying system with help of a bytecode verifier. Bytecode verifier also provides additional security as it’s role is to check the code for any violation access.

12. Write Once Run Anywhere: As discussed above java application generates ‘.class’ file which corresponds to our applications(program) but contains code in binary format. It provides ease t architecture-neutral ease as bytecode is not dependent on any machine architecture. It is the primary reason java is used in the enterprising IT industry globally worldwide.

13. Power of compilation and interpretation: Most languages are designed with purpose either they are compiled language or they are interpreted language. But java integrates arising enormous power as Java compiler compiles the source code to bytecode and JVM executes this bytecode to machine OS-dependent executable code.

Example

- Java

```
// Demo Java program

// Importing classes from packages
import java.io.*;

// Main class
public class GFG {

    // Main driver method
    public static void main(String[] args)
    {

        // Print statement
        System.out.println("Welcone to
GeeksforGeeks");
```

```
    }
}
```

Output

```
Welcome to GeeksforGeeks
```

Explanation:

1. **Comments:** Comments are used for explaining code and are used in a similar manner in Java or C or C++. Compilers ignore the comment entries and do not execute them. Comments can be of a single line or multiple lines.

Single line Comments:

Syntax:

```
// Single line comment
```

Multi-line **comments:**

Syntax:

```
/* Multi line comments*/
```

2. import java.io.*: This means all the classes of io package can be imported. Java io package provides a set of input and output streams for reading and writing data to files or other input or output sources.

3. class: The class contains the data and methods to be used in the program. Methods define the behavior of the class. Class **GFG** has only one method Main in JAVA.

4. **static void Main(): static** keyword tells us that this method is accessible without instantiating the class.

5. **void:** keywords tell that this method will not return anything. The main**()** method is the entry point of our application.

6. **System.in:** This is the **standard input stream** that is used to read characters from the keyboard or any other standard input device.

7. **System.out:** This is the **standard output stream** that is used to produce the result of a program on an output device like the computer screen.

8. **println():** This method in Java is also used to display text on the console. It prints the text on the console and the cursor moves to the start of the next line at the console. The next printing takes place from the next line.

Similarities and Difference between Java and C++

Nowadays Java and C++ programming languages are vastly using in competitive coding. Due to some awesome features, these two programming languages are mostly used in Industry also. C++ is a widely popular language among coders for its efficiency, high speed, and dynamic memory utilization. Java is widely in the IT industry, It is incomparable with any other programming language in terms of software development.
Let's go through the various points to compare these popular coding languages:

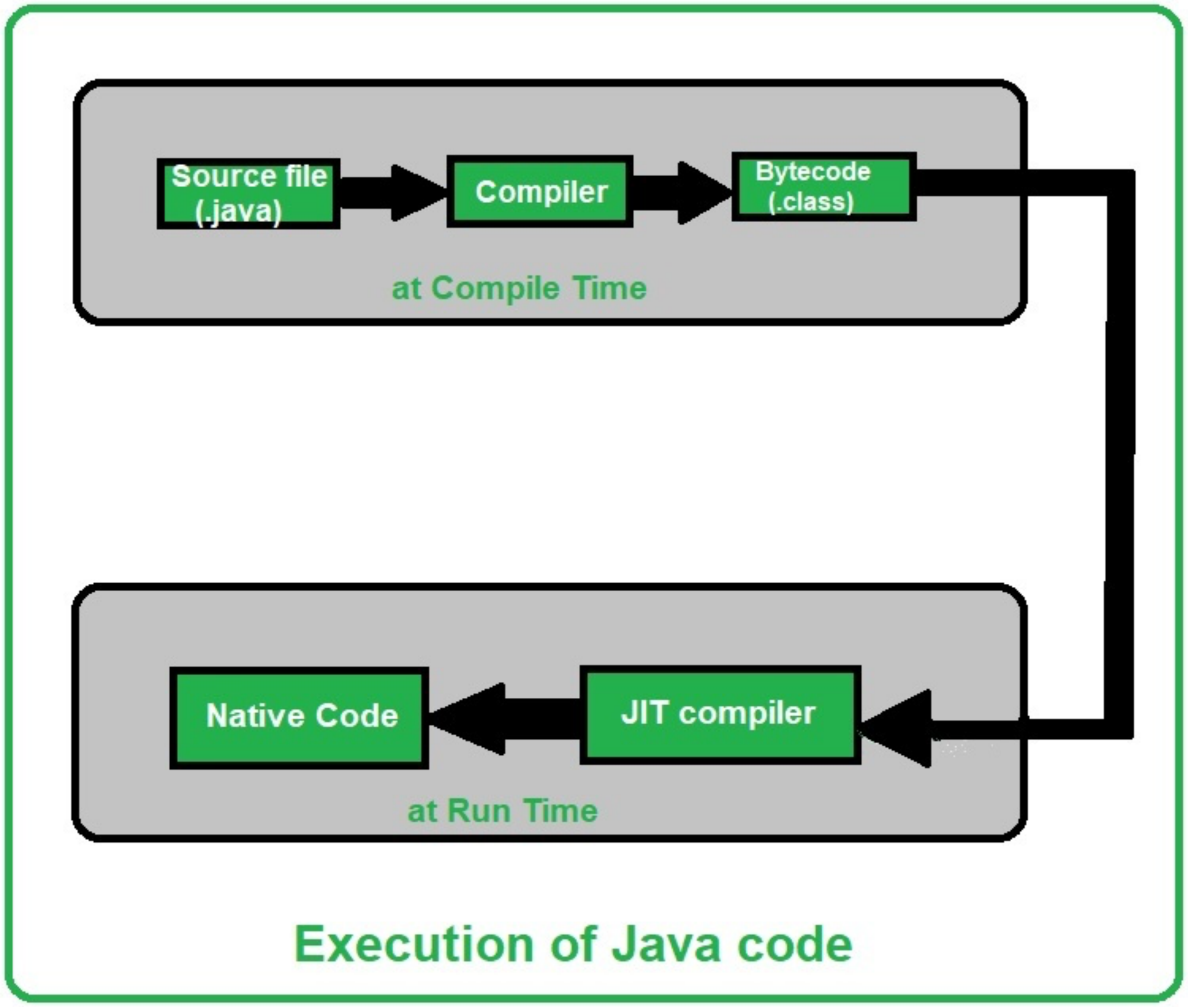
Similarities between Java and C++

1. **Execution:** At compile-time, Java source code or **.java** file is converted into a bytecode or **.class** file. At runtime, JVM(Java Virtual Machine) will

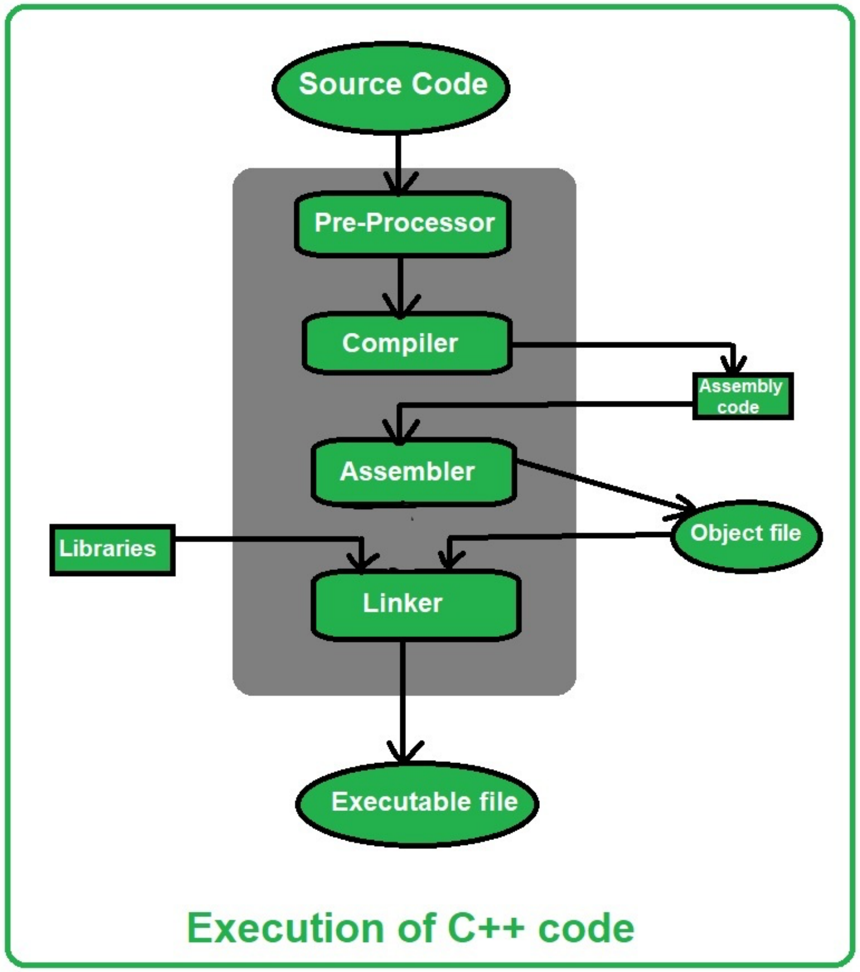
load the **.class** file and will convert it to machine code with the help of an interpreter. After compilation of method calls(using Just-In-Time (JIT) compiler), JVM will execute the optimized code. So Java is both compiled as well as an interpreted language.
On the other hand, C++ executes the code by using only a compiler. The C++ compiler compiles and converts the source code into the machine code. That’s why c++ is faster than Java but not platform-independent.

1. Below is the illustration of how Java and C++ code are executed:

Execution of a Java Code


Source file (.java)
Compiler
Bytecode (.class)
at Compile Time
Native Code
JIT compiler
at Run Time
Execution of Java code

1. **Execution of C++ code**



2. Features: C++ and Java both have several Object Oriented programming features which provide many useful programming functionalities. Some feature is supported by one and some are not. Below is the table which shows the features supported and not supported by both the programming languages:

Features	C++	Java
Abstraction	Yes	Yes
Encapsulation	Yes	Yes
Single Inheritance	Yes	Yes
Multiple Inheritance	Yes	No
Polymorphism	Yes	Yes

Features	C++	Java
Static Binding	Yes	Yes
Dynamic Binding	Yes	Yes
Operator Overloading	Yes	No
Header Files	Yes	No
Pointers	Yes	No
Global Variables	Yes	No
Template Class	Yes	No
Interference and Packages	No	Yes
API	No	Yes

Application: Both C++ and Java have vast areas of application. Below is the application of both languages:

- **Application of C++ Programming language:**
-
    1. Suitable for Developing large software(like passenger reservation system).
    2. MySQL is written by C++.
    3. For fast execution, C++ is majorly used in Game Development.
    4. Google Chromium browser, file system, cluster data processing are all written in C++.
    5. Adobe Premiere, Photoshop, and Illustrator these popular applications that are scripted in C++.
    6. Advance Computations and Graphics- real-time physical simulations, high-performance image processing.

7. C++ is also used in many advanced medical equipments like MRI machines, etc.

-

1. **Application of Java Programming language:**
   - Desktop GUI Applications development.
   - Android and Mobile application development.
   - Applications of Java are in embedded technologies like SIM cards, disk players, TV, etc.
   - Java EE(Enterprise Edition) provides an API and runtime environment for running large enterprise software.
   - Network Applications and Web services like Internet connection, Web App Development.

Environment: C++ is Platform dependent while Java is the platform-independent programming language. We have to write and run C++ code on the same platform. Java has the **WORA(Write Once and Run Everywhere)** feature by which we can write our code in one platform once and we can run the code anywhere.

Differences between Java and C++:

Parameters	Java	C++
Founder	Java was developed by James Gosling at Sun Microsystems.	C++ was developed by Bjarne Stroustrup at Bell Labs since 1979 as an extension of the C language.
First Release	On May 23, 1995	In October 1985

Parameters	Java	C++
Stable Release	Java SE 14 or JDK 14 was released on March 17, 2020.	C++17 was released in December 2017.
Official Website	oracle.com/java	isocpp.org
Influenced By:	Java was Influenced by Ada 83, Pascal, C++, C#, etc languages.	C++ was Influenced by Influenced by Ada, ALGOL 68, C, ML, Simula, Smalltalk etc language.
Influenced to:	Java was influenced to develop BeanShell, C#, Clojure, Groovy, Hack, J#, Kotlin, PHP, Python, Scala, etc languages.	C++ was influenced to develop C99, Java, JS++, Lua, Perl, PHP, Python, Rust, Seed7, etc languages.
Platform Dependency	Platform independent, Java bytecode works on any operating system.	Platform dependent should be compiled for different platforms.

Parameters	Java	C++
Portability	It can run in any OS hence it is portable.	C++ is platform-dependent hence it is not portable.
Compilation	Java is both Compiled and Interpreted Language.	C++ has only Compiled Language.
Memory Management	Memory Management is System Controlled.	Memory Management in C++ is Manual.
Virtual Keyword	It doesn't have Virtual Keyword.	It has Virtual Keyword.
Multiple Inheritance	It supports only single inheritance. Multiple inheritances are achieved partially using interfaces.	It supports both single and multiple Inheritance.
Overloading	It supports only method overloading and doesn't allow operator overloading.	It supports both methods and operators overloading.

Parameters	Java	C++
Pointers	It supports limited support for pointers.	It strongly supports Pointer.
Libraries	It doesn’t support direct native library calls but only Java Native Interfaces.	It supports direct system library calls, making it suitable for system-level programming.
Libraries	libraries have a wide range of classes for various high-level services	C++ libraries are comparatively low-level functionalities.
Documentation Comment	It supports documentation comment (e.g, /**.. */) for source code.	It doesn’t support documentation comments for source code.
Thread Support	Java supports documentation comment (e.g, /**.. */) for source code	C++ doesn’t have built-in support for threads, depends

Parameters	Java	C++
		on third-party threading libraries.
Type	Java is the only object-oriented programming language.	C++ is both a procedural and object-oriented programming language.
Input-Output mechanism	Java uses the (System class) System.in for input and System.out for output.	C++ uses cin for input and cout for an Output operation.
goto Keyword	Java doesn’t support goto Keyword	C++ supports goto keyword.
Structures and Unions	Java doesn’t support Structures and Unions.	C++ supports Structures and Unions.
Parameter Passing	Java supports only the Pass by Value technique.	C++ supports both Pass by Value and pass by reference.

Parameters	Java	C++
Global Scope	It supports no global scope.	It supports both global scope and namespace scope.
Object Management	Automatic object management with garbage collection.	It supports manual object management using new and deletes.

Applet Programming

1. An applet is a Java program that runs in a Web browser. Important Points regarding Applet are the following:

1. An applet is a Java class that extends the java.applet.Applet class

2. A main() method is not invoked on an applet, and an applet class will not define main().

3. Applets are designed to be embedded within an HTML page

4. When a user views an HTML page that contains an applet, the code for the applet is downloaded to the user's machine.