**1. Platform Independent**

**Compiler** converts source code to **byte code**and then the JVM executes the bytecode generated by the compiler. This byte code can run on any platform be it Windows, Linux, or 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 the **byte code**. That is [**why we call java a platform-independent language.**](https://www.geeksforgeeks.org/java-platform-independent/)

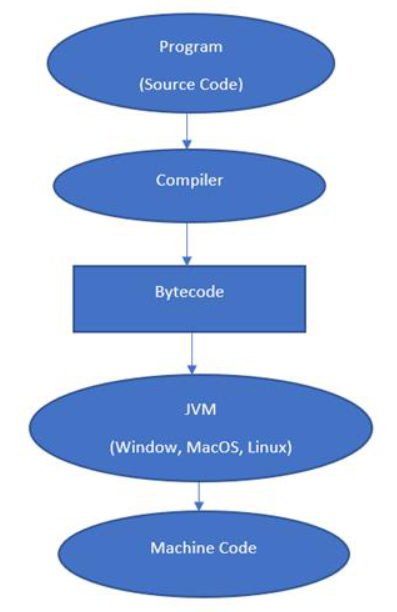
# Byte Code in Java

### ****Byte Code****

Byte Code can be defined as an intermediate code generated by the compiler after the compilation of source code(JAVA Program). This intermediate code makes Java a platform-independent language.

### ****How****is Byte Code****generated?****

Compiler converts the source code or the Java program into the Byte Code(or machine code), and secondly, the Interpreter executes the byte code on the system. The Interpreter can also be called JVM(Java Virtual Machine). The byte code is the common piece between the compiler(which creates it) and the Interpreter (which runs it).



/\*package whatever //do not write package name here \*/

import java.io.\*;

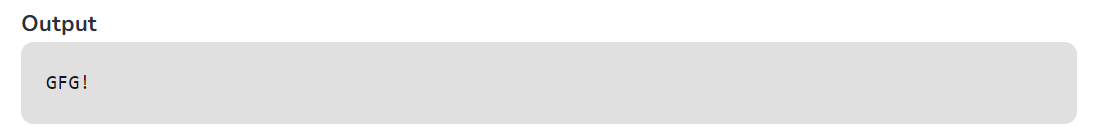
class GFG {

    public static void main (String[] args) {

        System.out.println("GFG!");

    }

}



* The above-written code is called JAVA source code.
* The compiler compiles the source code.
* Finally, Interpreter executes the compiled source code.

Whenever we write any program, it is not written in machine code. We write it in a high-level language like JAVA, C++, Python, etc. But the computer understands only the machine code. So when we execute our program, it is first converted into machine code or Byte code by the compiler and then executed by the Interpreter.

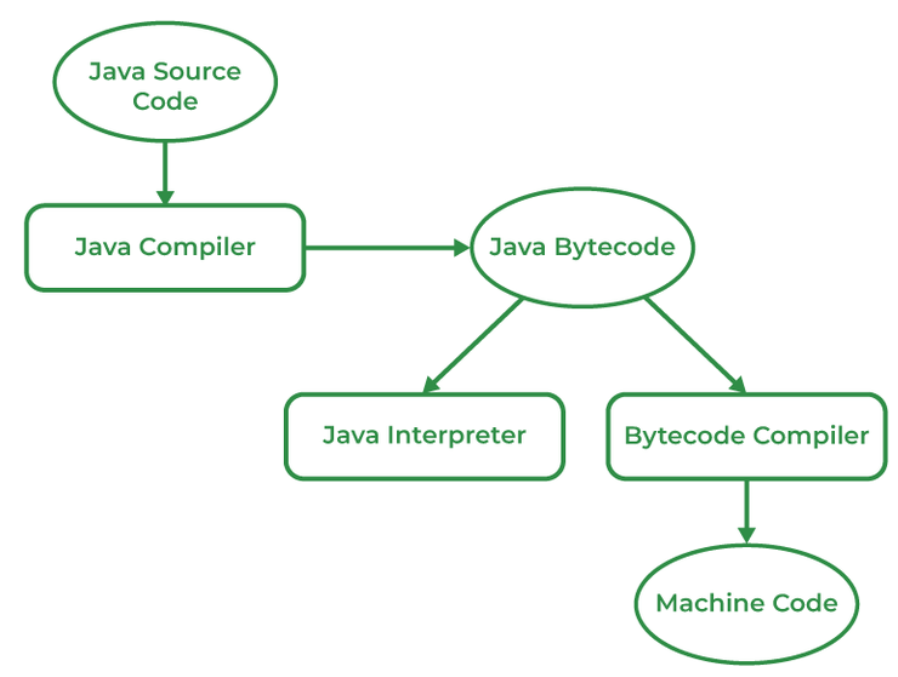
# How is Java platform independent?

The meaning of Java platform-independent is that the Java compiled code(byte code) can run on all operating systems. A program is written in a language that is human-readable. It may contain words, phrases, etc. which the machine does not understand. For the source code to be understood by the machine, it needs to be in a language understood by machines, typically a machine-level language. So, here comes the role of a compiler. The compiler converts the high-level language (human language) into a format understood by the machines.

Therefore, a compiler is a program that translates the source code for another program from a programming language into executable code. This executable code may be a sequence of machine instructions that can be executed by the CPU directly, or it may be an intermediate representation that is interpreted by a virtual machine. This intermediate representation in Java is the **Java Byte Code.**

## **Step-by-Step Execution of Java Program**

* Whenever a program is written in JAVA, the java compiles it.
* The result of the JAVA compiler is the **.class file or the bytecode** and not the machine’s native code (unlike the C compiler).
* The bytecode generated is a non-executable code and needs an interpreter to execute on a machine. This interpreter is the JVM and thus the Bytecode is executed by the JVM.
* And finally, the program runs to give the desired output.



In the case of C or C++ (languages that are not platform independent), the compiler generates an executable file (such as a .exe file) which is both OS-dependent and CPU-dependent. When we try to run this executable file on another OS or CPU architecture, it does not run, since it is specifically compiled for the target operating system and hardware architecture, making it incompatible with others.

## **Why Java is platform-independent but JVM is platform dependent?**

In Java, the main point here is that the JVM depends on the operating system – so if you are running Mac OS X you will have a different JVM than if you are running Windows or some other operating system. This fact can be verified by trying to download the JVM for your particular machine – when trying to download it, you will be given a list of JVM corresponding to different operating systems, and you will obviously pick whichever JVM is targeted for the operating system that you are running. So we can conclude that JVM is platform-dependent and it is the reason why Java is able to become “Platform Independent”.

**Important Points:**

* In the case of Java, **it is the magic of Bytecode that makes it platform-independent**.
* This adds to an important feature in the JAVA language termed **portability**. Every system has its own JVM which gets installed automatically when the JDK software is installed. For every operating system separate JVM is available which is capable to read the .class file or byte code.
* An important point to be noted is that while **JAVA is a platform-independent language, the JVM is platform-dependent.** Different JVM is designed for different OS and byte code is able to run on different OS.