### Platform Independent

The design objective of javasoft people is to develop a language that must work on any platform. Here platform means a type of operating system and hardware technology. Java allows programmers to write their program on any machine with any configuration and to execute it on any other machine having different configurations.

In Java, Java source code is compiled to bytecode and this bytecode is not bound to any platform. In fact, this bytecode is only understandable by the Java Virtual Machine which is installed in our system. What I meant to say is that every operating system has its own version of JVM, which is capable of reading and converting bytecode to an equivalent machine’s native language. This reduces the overhead of programmers writing system-specific code. Now programmers write programs only once, compile them, generate the bytecode and then export it anywhere.

**Portable**

The WORA (Write Once Run Anywhere) concept and platform-independent feature make Java portable. Now using the Java programming language, developers can yield the same result on any machine, by writing code only once. The reason behind this is JVM and bytecode. Suppose you wrote any code in Java, then that code is first converted to equivalent bytecode which is only readable by JVM. We have different versions of JVM for different platforms. Windows machines have their own version of JVM, Linux has its own and macOS has its own version of JVM. So if you distribute your bytecode to any machine, the JVM of that machine would translate the bytecode into the respective machine code. In this way portability lets the programmers focus on development and productivity rather than writing different code for different platforms.

### Secure

In today’s era, security is a major concern of every application. As of now, every device is connected to each other using the internet and this opens up the possibility of hacking. And our application built using java also needs some sort of security. So Java also provides security features to the programmers. Security problems like virus threats, tampering, eavesdropping, and impersonation can be handled or minimized using Java. Encryption and Decryption feature to secure your data from eavesdropping and tampering over the internet. An Impersonation is an act of pretending to be another person on the internet. The solution to

The impersonation problem is a digital signature, a file that contains personal identification information in an unreadable format. Digital Signature can be generated using Java. Virus is a program that is capable of harming our system and this is generally spread with .exe files, image files, and video files but cannot be spread using a text file the good thing is java bytecode is also a text file (yes .class file also a text file with non-human-readable format). Even if somebody tries to add virus code in a bytecode file, then also we are safe, because our JVM is smart enough to distinguish viruses from normal programs. If a virus is found in a bytecode file, JVM will throw an exception and abort execution.

### Robust

The Java Programming language is robust, which means it is capable of handling unexpected termination of a program. There are 2 reasons behind this, first, it has a most important and helpful feature called Exception Handling. If an exception occurs in java code then no harm will happen whereas, in other low-level languages, the program will crash.

Another reason why Java is strong lies in its memory management features. Unlike other low-level languages, Java provides a runtime Garbage collector offered by JVM, which collects all the unused variables. The garbage collector is a special program under JVM that runs from time to time and detects any unused variables and objects and removes them from the memory to free up space. But in the case of other prior languages, there is no such program to handle memory management, programmers are solely responsible for allocating and deallocating memory spaces, otherwise, the program may crash due to insufficient memory.

### Multi-Threaded

Thread is a lightweight and independent subprocess of a running program (i.e, process) that shares resources. And when multiple threads run simultaneously is called multithreading. In many applications, you have seen multiple tasks running simultaneously, for example, Google Docs where while typing text, the spell check and autocorrect tasks are running.

The server also uses multithreading to provide its services to multiple client requests. In Java, you can create threads in two ways, either by implementing the Runnable interface or by extending the Thread class.