Unit 1: Java's Lineage

What is Java?

- Java is a **programming language and a platform**.
- Java is a high level, robust, secured and object-oriented **programming language**.
- Any hardware or software environment in which a program runs, is known as a **platform**. Since Java has its own runtime environment (JRE) and API, it is called platform.

Java's Lineage

Java is related to C++, which is a direct descendant of C. Much of the character of Java is inherited from these two languages. From C, Java derives its syntax. Many of Java's object oriented features were influenced by C++. In fact, several of Java's defining characteristics come from—or are responses to—its predecessors. Moreover, the creation of Java was deeply rooted in the process of refinement and adaptation that has been occurring in computer programming languages for the past several decades.

By the end of the 1980s and the early 1990s, object-oriented programming using C++ took hold. Indeed, for a brief moment it seemed as if programmers had finally found the perfect language. Because C++ blended the high efficiency and stylistic elements of C with the object-oriented paradigm, it was a language that could be used to create a wide range of programs. However, just as in the past, forces were brewing that would, once again, drive computer language evolution forward. Within a few years, the World Wide Web and the Internet would reach critical mass. This event would precipitate another revolution in programming.

The Creation of Java

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. in 1991. It took 18 months to develop the first working version. This language was initially called "Oak," but was renamed "Java" in 1995.

- ✓ The primary motivation for creation of Java was the need for a platform-independent (that is, architecture-neutral) language that could be used to create software to be embedded in various consumer electronic devices, such as microwave ovens and remote controls.
- ✓ As you can probably guess, many different types of CPUs are used as controllers. The trouble with C and C++ (and most other languages) is that they are designed to be compiled for a specific target. Although it is possible to compile a C++ program for just about any type of CPU, to do so requires a full C++ compiler targeted for that CPU. The problem is that compilers are expensive and time consuming to create.
- ✓ An easier—and more cost-efficient—solution was needed. In an attempt to find such a solution, Gosling and others began work on a portable, platform independent language that could be used to produce code that would run on a variety of CPUs under differing environments. **This effort ultimately led to the creation of Java.**

- ✓ About the time that the details of Java were being worked out, a second, and ultimately more important, factor was emerging that would play a crucial role in the future of Java. This second force was, of course, the World Wide Web. Had the Web not taken shape at about the same time that Java was being implemented, Java might have remained a useful but obscure language for programming consumer electronics. However, with the emergence of the World Wide Web, Java was propelled to the forefront of computer language design, because the Web, too, demanded portable programs.

How Java Changed the Internet?

The Internet helped catapult Java to the forefront of programming, and Java, in turn, had a profound effect on the Internet. In addition to simplifying web programming in general, Java innovated a new type of networked program called the applet that changed the way the online world thought about content. Java also addressed some of the thorniest issues associated with the Internet: **portability and security**

- JAVA innovated a new type of networked program called the applet that changed the way the online world thought about content.
- Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.
- **Security :** JAVA achieved this protection by confining(restrict) an applet to the JAVA execution environment and not allowing its access to other parts of the computer
- **Portability**: In case of an applet, the same applet must be able to be downloaded and executed by the wide variety of CPU's , operating systems and browsers connected to the internet.

The Bytecode

The key that allows Java to solve both the security and the portability problems just described is that the output of a Java compiler is not executable code. Rather, it is bytecode. Bytecode is a highly optimized set of instructions designed to be executed by the Java runtime system, which is called the Java Virtual Machine (JVM).

When we write a program in Java, firstly, the compiler compiles that program and a bytecode is generated for that piece of code. When we wish to run this .class file on any other platform, we can do so. After the first compilation, the bytecode generated is now run by the Java Virtual Machine and not the processor in consideration. This essentially means that we only need to have basic java installation on any platforms that we want to run our code on. Resources required to run the bytecode are made available by theJava Virtual Machine, which calls the processor to allocate the required resources.

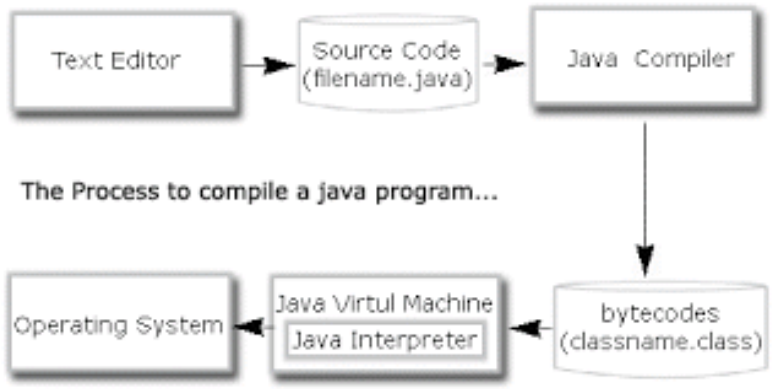


Figure 1: Process to compile a java program

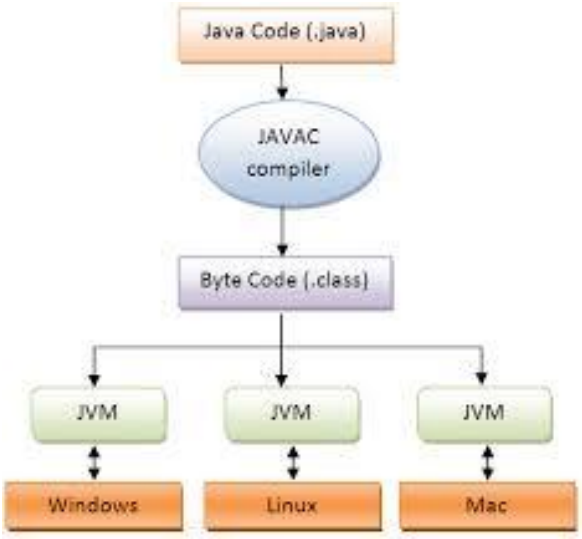


Figure 2:Java is platform independent language

Translating a Java program into bytecode makes it much easier to run a program in a wide variety of environments because only the JVM needs to be implemented for each platform. Once the run-time package exists for a given system, any Java program can run on it. Remember, although the details of the JVM will differ from platform to platform, all understand the same Java bytecode.

Advantage of Java Bytecode

Platform independence is one of the soul reasons for which James Gosling started the formation of java and it is this implementation of bytecode which helps us to achieve this. Hence bytecode is a very important component of any java program.The set of instructions for the JVM may differ from system to system but all can interpret the bytecode. A point to keep in mind is that bytecodes

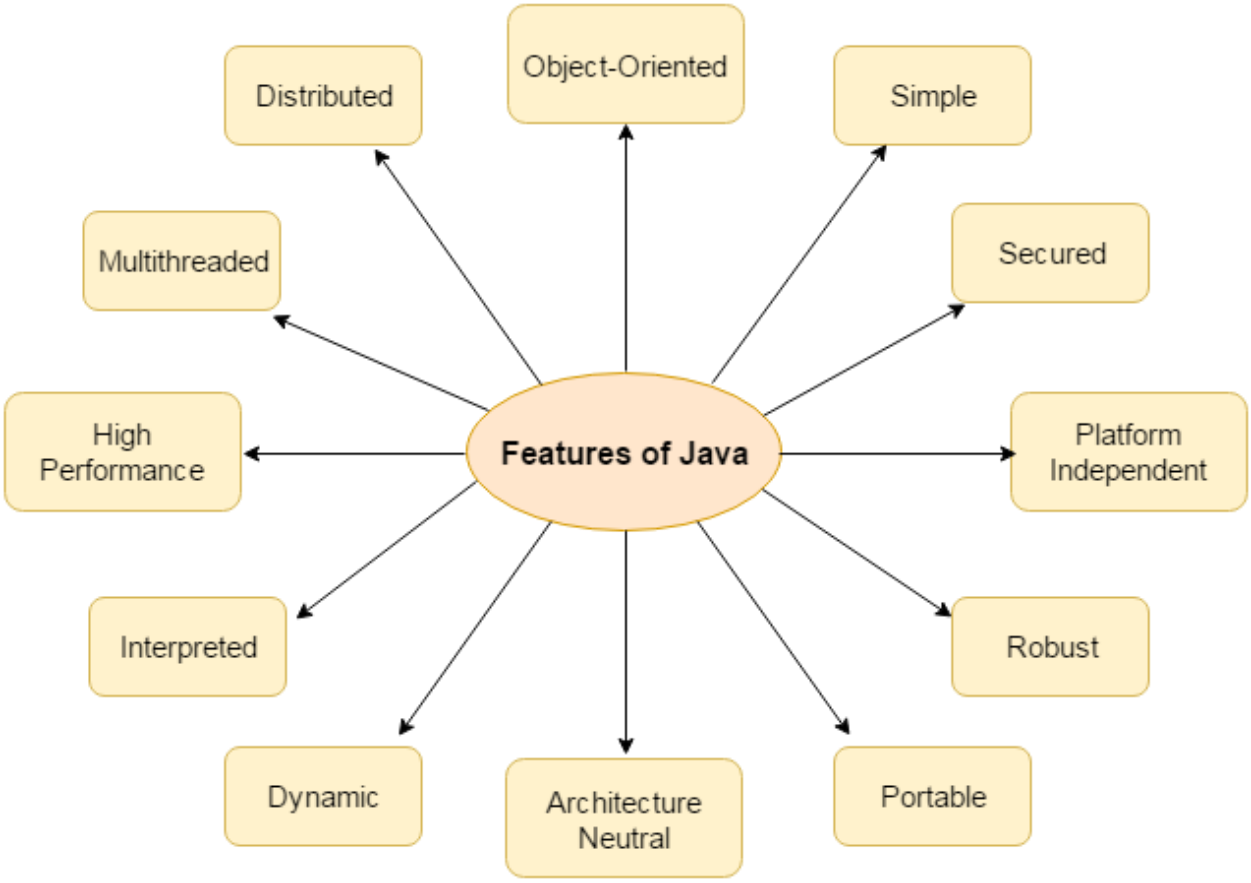
are non-runnable codes and rely on the availability of an interpreter to execute and thus the JVM comes into play.

Bytecode is essentially the machine level language which runs on the Java Virtual Machine. Whenever a class is loaded, it gets a stream of bytecode per method of the class. Whenever that method is called during the execution of a program, the bytecode for that method gets invoked. Javac not only compiles the program but also generates the bytecode for the program. Thus, we have realized that the bytecode implementation makes Java a platform-independent language. This helps to add portability to Java which is lacking in languages like C or C++. Portability ensures that Java can be implemented on a wide array of platforms like desktops, mobile devices, severs and many more. Supporting this, Sun Microsystems captioned JAVA as "write once, read anywhere" or "WORA" in resonance to the bytecode interpretation.

Servlets: Java on the Server Side

- A servlet is a small program that executes on the server. Just as applets dynamically extend the functionality of a web browser, servlets dynamically extend the functionality of a web server. Thus, with the advent of the servlet, Java spanned both sides of the client/server connection.
- Servlets are used to create dynamically generated content that is then served to the client.
- Because servlets (like all Java programs) are compiled into bytecode and executed by the JVM, they are highly portable. Thus, the same servlet can be used in a variety of different server environments. The only requirements are that the server support the JVM and a servlet container.

The Java Buzzwords (features of java)



Simple

Java was designed to be easy for the professional programmer to learn and use effectively. Assuming that you have some programming experience, you will not find Java hard to master. If you already understand the basic concepts of object-oriented programming, learning Java will be even easier. Best of all, if you are an experienced C++ programmer, moving to Java will require very little effort. Because Java inherits the C/C++ syntax and many of the object-oriented features of C++, most programmers have little trouble learning Java.

Object-Oriented

Although influenced by its predecessors, Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean, usable, pragmatic approach to objects. Borrowing liberally from many seminal object-software environments of the last few decades, Java manages to strike a balance between the purist's "everything is an object" paradigm and the pragmatist's "stay out of my way" model. The object model in Java is simple and easy to extend, while primitive types, such as integers, are kept as high performance non objects.

Java is an object oriented language. It supports various object oriented features like class, object, encapsulation, polymorphism, inheritance, abstraction, etc.

Robust

Robust simply means strong. The ability to create robust programs was given a high priority in the design of Java. To gain reliability, Java restricts you in a few key areas to force you to find your mistakes early in program development. At the same time, Java frees you from having to worry about many of the most common causes of programming errors. Because Java is a strictly typed language, it checks your code at compile time. However, it also checks your code at run time.

- → It provides many features that make the program execute reliably in variety of environments.
- → Java is a strictly typed language. It checks code both at compile time and runtime.
- → Java takes care of all memory management problems with garbage-collection.
- → Java, with the help of exception handling captures all types of serious errors and eliminates any risk of crashing the system.

Multithreaded

Java was designed to meet the real-world requirement of creating interactive, networked programs. To accomplish this, Java supports multithreaded programming, which allows you to write programs that do many things simultaneously. The Java run-time system comes with an elegant yet sophisticated solution for multiprocess synchronization that enables you to construct smoothly running interactive systems.

A thread is like a sub program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

Architecture-Neutral

A central issue for the Java designers was that of code longevity and portability. At the time of Java's creation, one of the main problems facing programmers was that no guarantee existed that if you wrote a program today, it would run tomorrow—even on the same machine. Operating system upgrades, processor upgrades, and changes in core system resources can all combine to make a program malfunction. The Java designers made several hard decisions in the Java language and the Java Virtual Machine in an attempt to alter this situation. Their goal was "write once; run anywhere, any time, forever." To a great extent, this goal was accomplished.

Distributed

Java is designed for the distributed environment of the Internet because it handles TCP/IP protocols. In fact, accessing a resource using a URL is not much different from accessing a file. Java also supports Remote Method Invocation (RMI). This feature enables a program to invoke methods across a network.

Dynamic
Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time. This makes it possible to dynamically link code in a safe and expedient manner. This is crucial to the robustness of the Java environment, in which small fragments of bytecode may be dynamically updated on a running system.

- ✓ Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand.
- ✓ Java supports dynamic compilation and automatic memory management (garbage collection).
- ✓ Java is capable of linking in new class libraries, methods, and objects.
- ✓ It can also link native methods (the functions written in other languages such as C and C++).

Compiled and Interpreted

Usually a computer language is either compiled or Interpreted. Java combines both this approach and makes it a two-stage system.

Compiled : Java enables creation of a cross platform programs by compiling into an intermediate representation called Java Bytecode.

Interpreted : Bytecode is then interpreted, which generates machine code that can be directly executed by the machine that provides a Java Virtual machine.

High Performance
As described earlier, Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java bytecode. This code can be executed on any system that implements the Java Virtual Machine. Most previous attempts at cross-platform solutions have done so at the expense of performance. As explained earlier, the Java bytecode was carefully designed so that it would be easy to translate directly into native machine code for very high performance by using a just-in-time compiler.

- ✓ Java performance is high because of the use of bytecode.
- ✓ The bytecode was used, so that it would be easily translated into native machine code.

Portable
Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

Secure
- ✓ Java provides a “firewall” between a networked application and your computer.
- ✓ When a Java Compatible Web browser is used, downloading can be done safely without fear of viral infection or malicious intent.
- ✓ Java achieves this protection by confining a Java program to the java execution environment and not allowing it to access other parts of the computer.