1 The History and Evolution of Java

Java lineage

Computer language innovation and development occurs for two fundamental reasons:
• To adapt to changing environments and uses
• To implement refinements and improvements in the art of programming.

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.

James Gosling initiated Java language project in June 1991 at Sun Microsystems for use in one of his many set-top box projects. The language, initially called "Oak" after that "an oak tree" that stood outside Gosling's office, also went by the name "Green" and ended up later being renamed as Java, from a list of random words. The public announcement of Java in the spring of 1995.

Motivation

1. The primary motivation 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.
2. The emergence of the World Wide Web, Java was propelled to the forefront of computer language design, because the Web, too, demanded portable programs. the Internet consists of a diverse, distributed universe populated with various types of computers, operating systems, and CPUs. Even though many kinds of platforms are attached to the Internet, users would like them all to be able to run the same program.

This realization caused the focus of Java to switch from consumer electronics (a small scale) to Internet programming (large scale).

Java and Internet

An *applet* *:* is a special kind of Java program that is designed to be transmitted over the Internet and automatically executed by a Java-compatible web browser. Furthermore, an applet is downloaded on demand, without further interaction with the user. If the user clicks a link that contains an applet, the applet will be automatically downloaded and run in the browser. Applets are intended to be small programs. They are typically used to display data provided by the server, handle user input, or provide simple functions, such as a loan calculator, that execute locally, rather than on the server. In essence, the applet allows some functionality to be moved from the server to the client.

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.
For example, an online store might use a servlet to look up the price for an item in a database. The price information is then used to dynamically generate a web page that is sent to the browser. Although dynamically generated content is available through mechanisms such as CGI (Common Gateway Interface), the servlet offers several advantages, including increased performance. 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.

Security
Every time you download a "normal" program, you are taking a risk, because the code you are downloading might contain a virus, Trojan horse, or other harmful code. At the core of the problem is the fact that malicious code can cause its damage because it has gained unauthorized access to system resources. For example, a virus program might gather private information, such as credit card numbers, bank account balances, and passwords, by searching the contents of your computer's local file system.
In order for Java to enable applets to be downloaded and executed on the client computer safely, it was necessary to prevent an applet from launching such an attack. Java achieved this protection by confining an applet to the Java execution environment and not allowing it access to other parts of the computer.
Security: The ability to download applets with confidence that no harm will be done and that no security will be breached may have been the single most innovative aspect of Java.

Portability

Portability : is a major aspect of the Internet because there are many different types of computers and operating systems connected to it. If a Java program were to be run on virtually any computer connected to the Internet, there needed to be some way to enable that program to execute on different systems. For example, in the case of an applet, the same applet must be able to be downloaded and executed by the wide variety of CPUs, operating systems, and browsers connected to the Internet. It is not practical to have different versions of the applet for different computers. The same code must work on all computers. Therefore, some means of generating portable executable code was needed. The same mechanism that helps ensure security also helps create portability.

Java's Magic

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. : The 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)***. In essence, the original JVM was designed as **an *interpreter for bytecode***.

JVM : The run-time package exists for a given system, any Java program can run on it. So, 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.
Remember, although the details of the JVM will differ from platform to platform, all understand the same Java bytecode. The fact that a Java program is executed by the JVM also helps to make it secure. Because the JVM is in control, it can contain the program and prevent it from generating side effects outside of the system. As you will see, safety is also enhanced by certain restrictions that exist in the Java language

Problem : a program is compiled to an intermediate form and then interpreted by a virtual machine, it runs slower than it would run if compiled to executable code.

bytecode has been highly optimized, the use of bytecode enables the JVM to execute programs much faster than you might expect.
the HotSpot technology was introduced not long after Java's initial release. HotSpot provides a ***Just-In-Time (JIT) compiler*** for bytecode. When **a JIT compiler is part of the JVM, selected portions of bytecode are compiled into executable code in real time, on a piece-by-piece, demand basis**. It is important to understand that it is not practical to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. Instead, a JIT compiler compiles code as it is needed, during

execution. Furthermore, not all sequences of bytecode are compiled—only those that will benefit from compilation. The remaining code is simply interpreted. However, the just-in-time approach still yields a significant performance boost.

The latest release of the Java Standard Edition is Java SE 8. 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 (WORA**).

Java features:

1. Object Oriented: In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

2. 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 bytecode. This bytecode is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

3. Simple: Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

4. Secure: With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

5. 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. **Their goal was "write once; run anywhere, any time, forever."**

6. 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.

7. Robust: Java makes an effort to eliminate error prone situations by emphasizing mainly on **compile time error checking** and **runtime checking**.

8. 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.

9. 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.

10. High Performance: With the use of Just-In-Time compilers, Java enables high performance.

11. Distributed: Java is designed for the distributed environment of the internet.

12. 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.