Chapter 1

INTRODUCTION TO JAVA:

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.

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. Between the initial implementation of Oak in the fall of 1992 and the public announcement of Java in the spring of 1995, many more people contributed to the design and evolution of the language. Bill Joy, Arthur van Hoff, Jonathan Payne, Frank Yellin, and Tim Lindholm were key contributors to the maturing of the original prototype.

The original intention for Java was not the Internet! Instead, 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.

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 Applets

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.

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.

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. 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 Bytecode(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. Rather, it is bytecode. Bytecode is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). In essence, the original JVM was designed as an interpreter for bytecode. This may come as a bit of a

surprise since many modern languages are designed to be compiled into executable code because of performance concerns. However, the fact that a Java program is executed by the JVM helps solve the major problems associated with web-based programs.

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. In general, when 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. However, with Java, the differential between the two is not so great. Because bytecode has been highly optimized, the use of bytecode enables the JVM to execute programs much faster than you might expect.

JVM,JRE and JDK:

All three JDK, JRE and JVM are interdependent and there is no common point so it is not effective to state their difference in the tabular form however we will discuss these in following paragraphs.

JVM

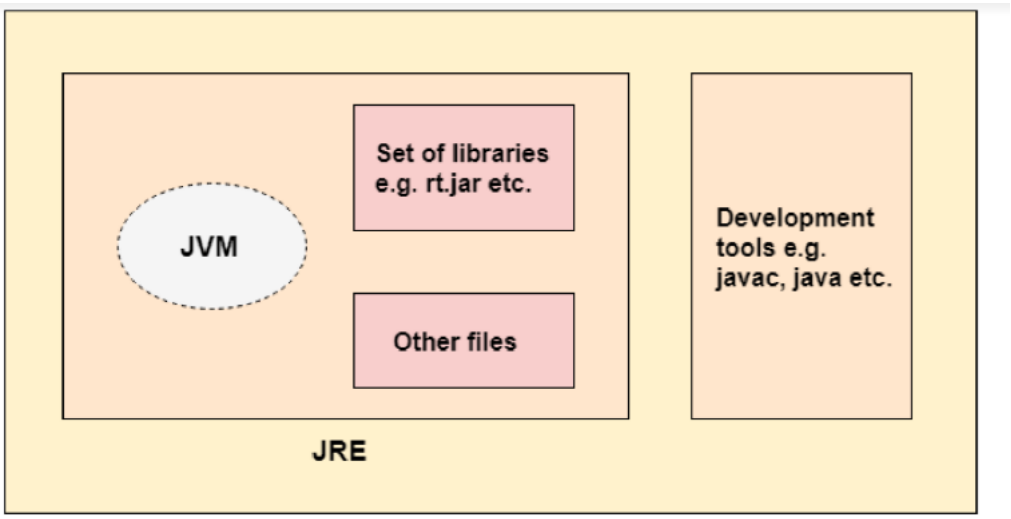
JVM is the abbreviation for Java virtual machine which is basically specification that provides a runtime environment in which Java byte code can be executed. It is JVM which is responsible for converting Byte code to the machine specific code.

JRE

JRE is Java runtime environment which is the implementation of JVM i.e the specifications which are defined in JVM are implemented and creates corresponding environment for the execution of code.JRE comprises mainly java binaries and other classes to execute the program alike of JVM it physically exists. Along with Java binaries JRE also consist of various technologies of deployment, user interfaces to interact with code executed, some base libraries for different functionalities and language and util based libraries.

JDK

JDK is abbreviation for Java Development Kit which includes all the tools, executable and binaries required to compile, debug and execute a Java Program.JDK is platform dependent i.e there is separate installers for Windows, Mac, and Unix systems.JDK includes both JVM and JRE and is entirely responsible for code execution. It is the version of JDK which represent version of Java.



The Java Buzzwords

Although the fundamental forces that necessitated the invention of Java are portability and security, other factors also played an important role in molding the final form of the language. The key considerations were summed up by the Java team in the following list of buzzwords:

- Simple
- Secure
- Portable
- Object-oriented Robust
- Multithreaded
- Architecture-neutral Interpreted
- High performance
- Distributed
- Dynamic

Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun, Java language is a simple programming language because:

- Java syntax is based on C++ (so easier for programmers to learn it after C++).
- Java has removed many complicated and rarely used features, for example: explicit pointers, operator overloading etc.
- There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

Object Oriented

Java is an object-oriented programming language. Everything in Java is an object. Object- oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.

Object-oriented programming (OOPS) is a methodology that simplifies software development and maintenance by providing some rules.

Robust

The multiplatform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. Thus, 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. Many hard-to-track- down bugs that often turn up in hard-to-reproduce run-time situations are simply impossible to create in Java. Knowing that what you have written will behave in a predictable way under diverse conditions is a key feature of Java.

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.

Architecture-Neutral

A central issue for the Java designers was that of code longevity and portability. One of the main problems facing programmers is that no guarantee exists that if you write a program today, it will 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.

Interpreted and High Performance

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. 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 run-time systems that provide this feature lose none of the benefits of the platform-independent code.

Java can be considered both a compiled and an interpreted language because its source code is first compiled into a binary byte-code. This byte-code runs on the JVM, which is usually a software-based interpreter.

Secured

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

• No explicit pointer

• Classloader: Classloader in Java is a part of the Java Runtime Environment(JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.

• Bytecode Verifier: It checks the code fragments for illegal code that can violate access right to objects.

• Security Manager: It determines what resources a class can access such as reading and writing to the local disk.

Some security can also be provided by an application developer explicitly using API's and libraries.

Portable

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

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.