**Java language**

Direct from the creators of the Java™ programming language, the completely revised fourth edition of The Java™ Programming Language is an indispensable resource for novice and advanced programmers alike. Developers around the world have used previous editions to quickly gain a deep understanding of the Java programming language, its design goals, and how to use it most effectively in real-world development. Now, Ken Arnold, James Gosling, and David Holmes have updated this classic to reflect the major enhancements in Java™ 2 Standard Edition 5.0 (J2SE™ 5.0). The authors systematically cover most classes in Java's main packages, java.lang. java.util, and java.io, presenting in-depth explanations of why these classes work as they do, with informative examples. Several new chapters and major sections have been added, and every chapter has been updated to reflect today's best practices for building robust, efficient, and maintainable Java software. Key changes in this edition include New chapters on generics, enums, and annotations, the most powerful new language features introduced in J2SE 5.0 Changes to classes and methods throughout to reflect the addition of generics Major new sections on assertions and regular expressions Coverage of all the new language features, from autoboxing and variable argument methods to the enhanced for-loop and covariant return types Coverage of key new classes, such as Formatter and Scanner The Java™ Programming Language, Fourth Edition, is the definitive tutorial introduction to the Java language and essential libraries and an indispensable reference for all programmers, including those with extensive experience. It brings together insights you can only get from the creators of Java: insights that will help you write software of exceptional quality.

**Java language development**

There is an alternative to compiling a high-level language program. Instead of using a compiler, which translates the program all at once, you can use an interpreter, which translates it instruction-by-instruction, as necessary. An interpreter is a program that acts much like a CPU, with a kind of fetch-and-execute cycle. In order to execute a program, the interpreter runs in a loop in which it repeatedly reads one instruction from the program, decides what is necessary to carry out that instruction, and then performs the appropriate machine-language commands to do so.

**Objects and Object-oriented Programming**

Programs must be designed. No one can just sit down at the computer and compose a program of any complexity. The discipline called software engineering is concerned with the construction of correct, working, well-written programs. The software engineer tends to use accepted and proven methods for analyzing the problem to be solved and for designing a program to solve that problem.

During the 1970s and into the 80s, the primary software engineering methodology was structured programming. The structured programming approach to program design was based on the following advice: To solve a large problem, break the problem into several pieces and work on each piece separately; to solve each piece, treat it as a new problem which can itself be broken down into smaller problems; eventually, you will work your way down to problems that can be solved directly, without further decomposition. This approach is called top-down programming.

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

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

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

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

**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.

**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.

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

**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.

**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.

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

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

**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.

**Java Basic Syntax**

It can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what do class, object, methods, and instance variables mean.

**Object** - Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behavior such as wagging their tail, barking, eating. An object is an instance of a class.

**Class** - A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.

**Methods** - A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.

**Instance Variables** - Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

**Objects in Java**

Let us now look deep into what are objects. If we consider the real-world, we can find many objects around us, cars, dogs, humans, etc. All these objects have a state and a behavior. If we consider a dog, then its state is - name, breed, color, and the behavior is - barking, wagging the tail, running. If you compare the software object with a real-world object, they have very similar characteristics. Software objects also have a state and a behavior. A software object's state is stored in fields and behavior is shown via methods. So in software development, methods operate on the internal state of an object and the object-to-object communication is done via methods.