Chapter 2

What is Java?

Java is a portable,interpreted,high-performance,simple,object-oriented programming language

Why Java

- Java is *object oriented*, yet it's still dead *simple*.
- The development cycle is much *faster* because Java technology is *interpreted*. The compile-link-load-test-crash-debug cycle is obsolete-now just compile and run.
- Applications developed using java are *portable* across multiple platforms. Develop the applications once, and they are never needed to be ported --they will run without modification on multiple operating systems and hardware architectures.
- Java applications are *robust* because the Java runtime environment manages memory for the programmer.
- Interactive graphical applications have *high performance* because multiple concurrent threads of activity in applications are supported by the *multithreading* built into the Java programming language and runtime platform.
- Applications are *adaptable* to changing environments because one can dynamically download code modules from anywhere on the network.
- End users can trust that applications developed in java are *secure*, even though they're downloading code from all over the Internet; the Java runtime environment has built-in protection against viruses and tampering.

History Of Java

In the early 90s, extending the power of network computing to the activities of everyday life was a radical vision. In 1991, a small group of Sun engineers called the "Green Team" believed that the next wave in computing was the union of digital consumer devices and computers. Led by James Gosling, the team worked around the clock and created the programming language that would revolutionize our world – 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 Green Team demonstrated their new language with an interactive, handheld home-entertainment controller that was originally targeted at the digital cable television industry. Unfortunately, the concept was much too advanced for the team at the time. But it was just right for the Internet, which was just starting to take off. In 1995, the team announced that the Netscape Navigator Internet browser would incorporate Java technology.

Today, Java not only permeates the Internet, but also is the invisible force behind many of the applications and devices that power our day-to-day lives. From mobile phones to handheld devices, games and navigation systems to e-business solutions, Java is everywhere!

Different Versions of Java

As of March 2019, Java 8 is supported; and both Java 8 and 11 as Long Term Support (LTS) versions. Major release versions of Java, along with their release dates:

- JDK 1.0 (January 23, 1996)
- JDK 1.1 (February 19, 1997)
- J2SE 1.2 (December 8, 1998)
- J2SE 1.3 (May 8, 2000)
- J2SE 1.4 (February 6, 2002)
- J2SE 5.0 (September 30, 2004)
- Java SE 6 (December 11, 2006)
- Java SE 7 (July 28, 2011)
- Java SE 8 (March 18, 2014)
- Java SE 9 (September 21, 2017)
- Java SE 10 (March 20, 2018)
- Java SE 11 (September 25, 2018)
- Java SE 12 (March 19, 2019)

Difference Between C,C++ and Java

while C++ is a superset of C, Java is neither a superset nor a subset of C or C++.

1. JAVA is Object-Oriented while C is procedural.

> Most differences between the features of the two languages arise due to the use of different programming paradigms. C breaks down to functions while JAVA breaks down to Objects. C is more procedure-oriented while JAVA is data-oriented.

2. Java is an Interpreted language while C is a compiled language.

> We all know what a compiler does. It takes your code & translates it into something the machine can understand-that is to say-0's & 1's-the machine-level code. That's exactly what happens with our C code-it gets 'compiled'. While with JAVA, the code is first transformed to what is called the bytecode. This bytecode is then executed by the JVM(Java Virtual Machine). For the same reason, JAVA code is more portable.

3. C is a low-level language while JAVA is a high-level language.

> C is a low-level language(difficult interpretation for the user, closer significance to the machine-level code) while JAVA is a high-level lagunage(abstracted from the machine-level details, closer significance to the program itself).

4. C uses the top-down **{sharp & smooth}** approach while JAVA uses the bottom-up **{on the rocks}** approach.

> In C, formulating the program begins by defining the whole and then splitting them into smaller elements. JAVA(and C++ and other OOP languages) follows the bottom-up approach where the smaller elements combine together to form the whole.

5. Pointer go backstage in JAVA while C requires explicit handling of pointers.

> When it comes to JAVA, we don't need the *'s & &'s to deal with pointers & their addressing. More formally, there is no pointer syntax required in JAVA. It does what it needs to do. While in JAVA, we do create references for objects.

6. The Behind-the-scenes Memory Management with JAVA & The User-Based Memory Management in C.

> Remember 'malloc' & 'free'? Those are the library calls used in C to allocate & free chunks of memory for specific data(specified using the keyword 'sizeof'). Hence in C, the memory is managed by the user while JAVA uses a garbage collector that deletes the objects that no longer have any references to them.

7. JAVA supports Method Overloading while C does not support overloading at all.

> JAVA supports function or method overloading-that is we can have two or more functions with the same name(with certain varying parameters like return types to allow the machine to differentiate between them). That it to say, we can overload methods

> with the same name having different method signatures. JAVA(unlike C++), does not support Operator Overloading while C does not allow overloading at all.

8. Unlike C, JAVA does not support Preprocessors, & does not really them.

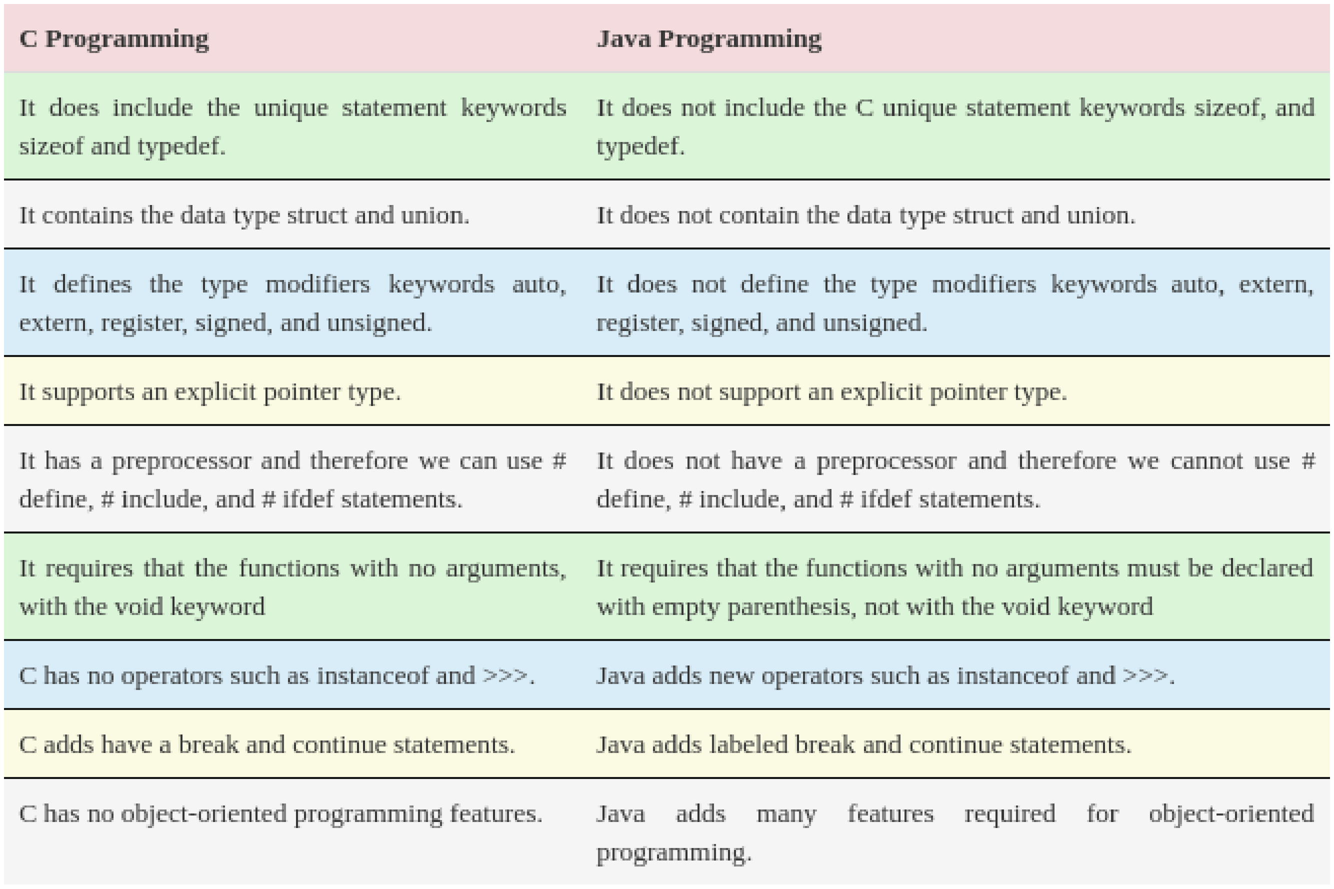
> The preprocessor directives like #include & #define, etc are considered one of the most essential elements of C programming. However, there are no preprocessors in JAVA. JAVA uses other alternatives for the preprocessors. For instance, public static final is used instead of the #define preprocessor. Java maps class names to a directory and file structure instead of the #include used to include files in C.

9. The standard Input & Output Functions.

> Although this difference might not hold any conceptual(intuitive) significance, but it’s maybe just the tradition. C uses the printf & scanf functions as its standard input & output while JAVA uses the System.out.print & System Resources and Information..read functions.

10. Exception Handling in JAVA And the errors & crashes in C.

> When an error occurs in a Java program it results in an exception being thrown. It can then be handled using various exception handling techniques. While in C, if there’s an error, there IS an error.



C Programming	Java Programming
It does include the unique statement keywords sizeof and typedef.	It does not include the C unique statement keywords sizeof, and typedef.
It contains the data type struct and union.	It does not contain the data type struct and union.
It defines the type modifiers keywords auto, extern, register, signed, and unsigned.	It does not define the type modifiers keywords auto, extern, register, signed, and unsigned.
It supports an explicit pointer type.	It does not support an explicit pointer type.
It has a preprocessor and therefore we can use # define, # include, and # ifdef statements.	It does not have a preprocessor and therefore we cannot use # define, # include, and # ifdef statements.
It requires that the functions with no arguments, with the void keyword	It requires that the functions with no arguments must be declared with empty parenthesis, not with the void keyword
C has no operators such as instanceof and >>>.	Java adds new operators such as instanceof and >>>.
C adds have a break and continue statements.	Java adds labeled break and continue statements.
C has no object-oriented programming features.	Java adds many features required for object-oriented programming.

C++ VS Java

C++ is platform-dependent.	Java is platform-independent.
C++ is mainly used for system programming and standalone desktop application programming	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
C++ was designed for systems and applications programming. It was an extension of C programming language.	Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience
C++ supports the goto statement.	Java doesn't support the goto statement.
C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java.
C++ supports operator overloading.	Java doesn't support operator overloading.
C++ supports pointers. You can write pointer program in C++.	Java supports pointer internally. However, you can't write the pointer program in java. It means

	java has restricted pointer support in java.
C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent.	Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent.
C++ supports both call by value and call by reference.	Java supports call by value only. There is no call by reference in java.
C++ supports structures and unions.	Java doesn't support structures and unions.
C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.	Java has built-in thread support.
C++ doesn't support documentation comment.	Java supports documentation comment (/** ... */) to create documentation for java source code.
C++ supports virtual keyword so that we can decide whether or not override a function.	Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default.
C++ doesn't support >>> operator.	Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator.
C++ creates a new inheritance tree always.	Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the inheritance tree in java.
C++ is close to hardware.	Java is not at all close to hardware.
C++ is an object-oriented language but not a strictly object oriented language	Java is a strictly object oriented language. Nothing can reside outside class.

There may be a lot of differences between C,C++ and Java but the major differences are as follows

- C,C++ are compiled languages whereas Java is a both a compiled and Interpreted Language.
- C,C++ produce code for target machine whereas Java produces code which can be used on any machine where jvm is installed.

Features of Java

The Java programming language is a high-level language that can be characterized by all of the following buzzwords

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

Simple

Primary characteristics of the Java programming language include a *simple* language that can be programmed without extensive programmer training while being attuned to current software practices. The fundamental concepts of Java technology are grasped quickly; programmers can be productive from the very beginning. Java uses the simple syntax of C and borrows object orientation from C++ and also removed almost all the drawbacks of C and C++; means that C and C++ programmers can migrate easily to the Java platform and be productive quickly.

Object-Oriented

The Java programming language is designed to be *object oriented* from the ground up. The needs of distributed, client-server based systems coincide with the encapsulated, message-passing paradigms of object-based software. To function within increasingly complex, network-based environments, programming systems must adopt object-oriented concepts. Java technology provides a clean and efficient object-based development platform.

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.

Multithreaded

Multi Threading means more than one sequence of execution in a single process.Multiple Tasks can be performed at the same time simultaneously. It can exploit potential of a Multi Processor system and can achieve true Parallesim.

Modern network-based applications, typically need to do several things at the same time. A user working with HotJava Browser can run several animations concurrently while downloading an image and scrolling the page. Java technology's *multithreading* capability provides the means to build applications with many concurrent threads of activity. Multithreading thus results in a high degree of interactivity for the end user.

Dynamic

While the Java Compiler is strict in its compile-time static checking, the language and run-time system are dynamic in their linking stages. Classes are linked only as needed. New code modules can be linked in on demand from a variety of sources, even from sources across a network.

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.

Architecture-Neutral

Java technology is designed to support applications that will be deployed into heterogeneous network environments. In such environments, applications must be capable of executing on a variety of hardware architectures. Within this variety of hardware platforms, applications must execute atop a variety of operating systems and interoperate with multiple programming language interfaces. To accommodate the diversity of operating environments, the Java Compiler TM product generates *bytecodes*--an *architecture neutral* intermediate format designed to transport code efficiently to multiple hardware and software platforms. The interpreted nature of Java technology solves both the binary distribution problem and the version problem; the same Java programming language byte codes will run on any platform.

Portable

Architecture neutrality is just one part of a truly *portable* system. Java technology takes portability a stage further by being strict in its definition of the basic language. Java technology puts a stake in the ground and specifies the sizes of its basic data types and the behavior of its arithmetic operators. Your programs are the same on every platform--there are no data type incompatibilities across hardware and software architectures.

High Performance

The Java platform achieves superior performance by adopting a scheme known as JIT by which the interpreter can run at full speed without needing to check the run-time environment. The *automatic garbage collector* runs as a low-priority background thread, ensuring a high probability that memory is available when required, leading to better performance. Applications requiring large amounts of compute power can be designed such that compute-intensive sections can be rewritten in native machine code as required and interfaced with the Java platform.

Robust

The Java programming language is designed for creating highly *reliable* software. It provides extensive compile-time checking, followed by a second level of run-time checking. Language features guide programmers towards reliable programming habits.

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.

Secure

Java technology is designed to operate in distributed environments, which means that *security* is of paramount importance. With security features designed into the language and run-time system, Java technology lets you construct applications that can't be invaded from outside. In the network environment, applications written in the Java programming language are secure from intrusion by unauthorized code attempting to get behind the scenes and create viruses or invade file systems.