Contents

[**Java Overview** 1](#_Toc509938926)

[**Java Introduction** 1](#_Toc509938927)

|  |
| --- |
|  |
| **Java Introduction** |
|  |
| What do you know about Java? |
| Java is a popular high-level programming language and computing platform.  Java runs on 3 billion devices worldwide.  Main features of Java are:   |  |  |  | | --- | --- | --- | | Simple  -   * Syntax is based on C++ * Removed many confusing and/or rarely-used features e.g., explicit pointers, operator overloading etc. * Automatic Garbage Collection | Object Oriented     * Everything is object | Distributed     * We can create distributed applications in java. * We may access files by calling the methods from any machine on the internet. | | Platform Independent     * Bytecode | Architecture-neutral     * Int - 4 byte for both (32/64 bit architecture) | Portable     * Java program gets converted into Java Byte Codes * Byte codes can be executed on any platform without any dependency. | | Robust     * Strong memory management. * There is lack of pointers that avoids security problem. * Automatic garbage collection * Exception handling and type checking mechanism | Secure     * Run inside virtual machine sandbox * Security Manager: determines what resources a class can access such as reading and writing to the local disk. |  | | Multithreaded     * A thread is like a separate program, executing concurrently. * We can write Java programs that deal with many tasks at once by defining multiple threads. | Interpreted     * Java uses compiler and interpreter both. | High Performance     * Java is faster than traditional interpretation since byte code is "close" to native code | |  |  | Dynamic     * Dynamic Polymorphism * Dynamic memory allocation | |
| How Java is platform independent  What gives Java its 'write once and run anywhere' nature? |
| * Platform Independence is also called build/write once, run anywhere. * When Java is compiled, it is not compiled into platform specific machine. The output is a class file, which contains an internal java representation called bytecode. * JVM converts bytecode to executable instructions (different in different operating systems). * There are different JVM's for different operating systems.   Example:  A JVM for windows is different from a JVM for mac. However, both the JVM's understand the bytecode and convert it to the executable code for the respective operating system. |
| How Java is Architecture-neutral? |
| * There is no implementation dependent feature e.g. size of primitive types is fixed. * In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. * But in java, it occupies 4 bytes of memory for both 32 and 64 bit architectures. |
| How Java is Portable? |
| Java program gets converted into Java Byte Codes that can be executed on any platform without any dependency. |
| Why Java is Simple? |
| Java language is simple because:   1. Syntax is based on C++ (so easier for programmers to learn it after C++). 2. Removed many confusing and/or rarely-used features e.g. explicit pointers, operator overloading etc. 3. No need to remove unreferenced objects because there is Automatic Garbage Collection in java. |
| Why Java is Secure? |
| Java is secured because:   * Java Programs run inside virtual machine sandbox   + - Classloader: adds security by separating the package for the classes of the local file system from those that are imported from network sources.     - Bytecode Verifier: checks the code fragments for illegal code that can violate access right to objects. * Security Manager: determines what resources a class can access such as reading and writing to the local disk. |
| Why Java is Robust/Strong? |
| All these points make java robust.   * Strong memory management. * There is lack of pointers that avoids security problem. * Automatic garbage collection * Exception handling and type checking mechanism |
| How Java is Multithreaded? |
| * A thread is like a separate 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. These resources are called Not-Thread safe. * Threads are important for multi-media, Web applications etc. |
| How Java is Interpreted language? |
| Java uses compiler and interpreter both. |
| How Java is High-performance? |
| Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language (e.g., C++). |
| How Java is Distributed language? |
| We can create distributed applications in java.  We may access files by calling the methods from any machine on the internet. |
| How Java is Dynamic (always active or changing)? |
| In terms of programming language the dynamic refer as the things which are executed as and when required rather than in advance.  Below are some feature related to dynamic in java  1. Dynamic Polymorphism: Compiler doesn’t understand which method to call in advance. JVM decide which method to call at run time.  2. Dynamic memory allocation: All Java objects are dynamically allocated. |