* **About Java**

**-History:** Microprocessors have had a profound impact in intelligent consumer electronic devices including computers. Sun microsystems recognized this and wanted to develop a programming language that would work on all kinds of devices(like TVs). Java is created by James Gosling with Sun Microsystems in 1995. It has been bought by oracle. It is an improved version of C and C++.

When the web exploded in popularity in 1993, Sun saw the potential of using Java to add dynamic content to web pages. And java is still being used to developed large scale enterprise applications.

Java is also the key language for developing Android smartphone and tablet apps.

Sun Microsystems was acquired by Oracle in 2010.

**-Paradigms:** Prior to Java SE 8, Java supported three programming paradigms. Procedural programming, object-oriented programming and generic programming. Java SE 8 adds functional programming,

Imperative, Object oriented, Functional, Procedural, Generic, Reflective, Event Driven, Concurrent, Structured.

**-Typing System:** Static, strong, safe, nominative, manifest.

<https://en.wikipedia.org/wiki/Java_(software_platform)>

<https://en.wikipedia.org/wiki/Template:Java_(software_platform)>

https://en.wikipedia.org/wiki/Java\_(programming\_language)

<https://en.wikipedia.org/wiki/Java_version_history>

See “Programming Languages” folder for more on paradigms and type systems.

* **JDK/JRE:**

<https://en.wikipedia.org/wiki/Java_virtual_machine>

<https://en.wikipedia.org/wiki/Java_bytecode>

https://en.wikipedia.org/wiki/Java\_bytecode\_instruction\_listings

JDK is a subset of what is loosely defined as a Software Development Kit (SDK). JDK is responsible for writing and running of java programs. The remainder of SDK is composed of extra software, such as Application Server, Debuggers and Documentation (Glassfish, MySQL, Netbeans, etc.)

JDK (Java Development Kit) contains **JRE** (Java Runtime Environment) which contains JVM (Java Virtual Machine) and API. JDK is needed for compiling, running, debugging and other development needs. JRE is needed for running java programs. If you need the JRE on a server and do not want the abilty to run RIAs (Rich Internet Application), use server JRE. The classes are grouped into packages. They are collectively referred to as the Java class libraryor the Java API (Java Application Interface) is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces; these libraries are known as packages. Also the api are optimized for speed so you should use them instead of making your own.

-Inlining the code: <https://stackoverflow.com/questions/2096361/are-there-inline-functions-in-java>

**Java Platform Editions**

**-Java SE(Standart Edition)/ Java SE API:** Provides the core functionality of the java programming language. It defines everything from the basic types and objects of the java programming lancguage to high-level classes that are used for networking, security, database access, graphical user interface(GUI) development and XML parsing. Plus it has JVM, dev tools, redeployment techs and other class libraries and toolkits used in java technology apps.

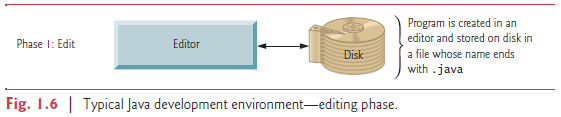
**-Java EE(Enterprise Edition:** It is built on top of Java SE platform. Provides API and runtime environment for developing and running large-scale, multitiered, scalable, reliable and secure network apps.

**-Java ME(Micro Edition:** Provides an API and a small-footprint virtual machine for running Java apps on small devices, like mobile phones. The API is a subset of Java SE API, along with special class libraries useful for small device app development. Java ME apps are often clients of Java EE platform services.

**-Java FX:** A platform for creating rich internet apps using lightweight user-interface API. JavaFX apps use hardware-accelerated graphics and media engines to take advantage of higher-performance clients and a modern look-and-feel as well as high-level APIs for connecting to networked data sources. JavaFX apps may be clients of Java EE platform services.

* **Java Development Steps:**

**1- Creating a Program:** You write java code (**source code**) on a file with an editor program (**editor**) like notepad. Then you save it on a secondary storage device such as your hard drive. Java source code files are given a name ending with the **.java** extension. **Integrated development environments (IDE)** provide tools that support the software development process such as editors, debuggers, compilers, interpreters etc. Some examples are Netbeans, Eclipse, IntelliJ Idea.



**2- Compiling a java program into bytecodes: javac java compiler**: After saving the file with the extension “.java”, change your directory to address of the file on command prompt. Compile using:

**javac** ClassName.java // javac stands for JAVA Compiler.

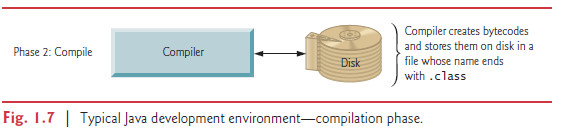
Java code -> Java compiler -> bytecode. Javac command makes the compiler turn **.java** files into **.class (bytecode)** files. Bytecode is platform independant.

-You can compile more than one file like so:

javac Account.java AccountTest.java

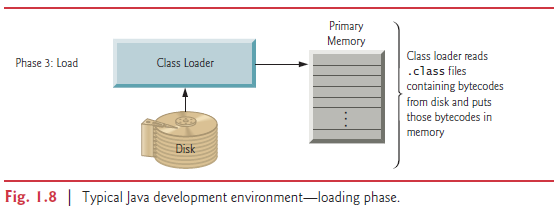
- \* means “all” in programming. It is being used a lot in database just like we are doing now. We are querying all files that has the extension .java). If you want to compile all .java files in a directory:

javac \*.java

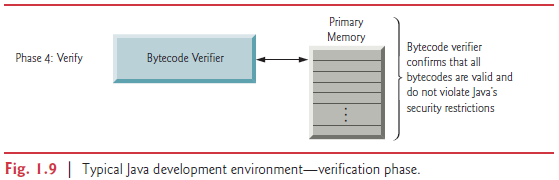


**3- Loading a program into memory:** JVM places the program in memory to execute it. This is known as **loading.** JVM **class loader** takes .class files containing the program’s bytecodes and transfers them to primary memory. It also loads any of the .class files provided by java that your program uses. The .class files can be loaded from disk on your system or over a network.

<https://stackoverflow.com/questions/2424604/what-is-a-java-classloader>



**4- Bytecode verification:** Bytecode verifiers examine their bytecodes to ensure that they are valid and do not violate java’s security restrictions. Java enforces strong security to make sure that java programs arriving over the network do not damage your files or your system.



**5- Execution:** Go to the directory on command prompt and use the command:

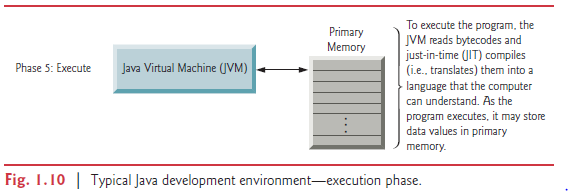
**java** program\_name

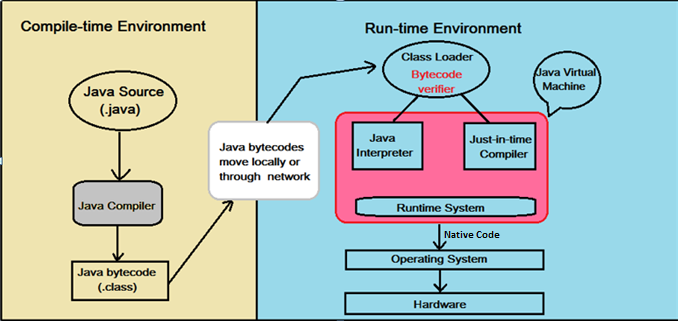
java command starts JVM, puts bytecode in JVM and runs it. **JVM** is located at ram. Class loader in JVM loads .class files to ram. Loaded bytecodes are verified for security reasons. Then execution engine bytecodes are transformed to **native machine code** by JVM that can be understood by operating system and hardware. That is why bytecode runs everywhere. Java is both **write once compile anywhere(WOCA)** and **write once and run everywhere(WORA**). Sometimes the word everywhere is used instead of anywhere. Languages like C and C++ are only WOCA and not WORA. WORA means native machine code will be created where the program is ran, not where the program got compiled. Java can be a portable language but compiler, JVM and computer differences might create problems.

JVM uses a combination of interpreting and just in time compiling.

The catch is that since there are multiple JVM implementations, on top of a wide variety of different operating systems, there could be subtle differences in how a program executes on each JVM/OS combination, possibly requiring an application to be tested on each target platform. This gave rise to a joke among Java developers: "Write Once, Debug Everywhere."

java [ <option> ... ] <class-name> [<argument> ...]





<https://stackoverflow.com/questions/8194713/difference-between-java-javaw-javaws>

**-Compiling and Interpreting:**

Java is compiled to an intermediate "bytecode" at compilation time. This is in contrast to a language like C that is compiled to machine language at compilation time. The Java bytecode cannot be directly executed on hardware the way that compiled C code can. Instead the bytecode must be interpreted by the JVM (Java Virtual Machine) at runtime in order to be executed. The primary drawback of a language like C is that when it is compiled, that binary file will only work on one particular architecture (e.g. x86). When you have an interpreted language, code can be run as is, without a compilation stage that creates a non portable executable program. You run the interpretable code in your virtual machine. This interpretable code is translated to machine code by virtual machine (Java, PHP, Perl).

Languages can be strictly compiled(C, C++), non-strictly interpreted(intermediate code), or strictly interpreted (Shells, or Tcl).

Interpreted languages like PHP are effectively system independent and rely on a system and architecture specific interpreter (You need to install a PHP interpreter just like you install a system specific JVM to your machine that runs all Java code). This leads to much greater portability (The same PHP scripts work on Windows machines and Linux machines, etc.). However, this interpretation leads to a significant performance decrease. High-level languages like PHP require more time to interpret than machine-specific instructions that can be executed by the hardware.

Java seeks to find a compromise between a purely compiled language (with no portability) and a purely interpreted language (that is significantly slower). It accomplishes this by compiling the code into a form that is closer to machine language (Actually, Java bytecode is a machine language, simply for the Java Virtual Machine), but can still be easily transported between architectures. Because Java still requires a software layer for execution (the JVM). It is an interpreted language. However, the interpreter (the JVM) operates on an intermediate form known as bytecode rather than on the raw source files. This byte code is generated at compile time by the Java compiler. Therefore, Java is also a compiled language. By operating this way, Java gets some of the benefits of compiled languages, while also getting some of the benefits of interpreted languages. However, it also inherits some limitations from both of these languages. You can think of JVM as a processor with the instruction set as bytecode.

-There are some strategies for increasing the performance of Java code (and other bytecode languages like .Net) through the use of Just in Time (JIT) compilation. [Just-in-time](https://realityworld.trade/wiki/Just-in-time_compilation?__cpo=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3Jn) compilers that compile bytecodes to machine code during runtime were introduced from an early stage. Java program’s execution speed improved significantly with the introduction of [just-in-time compilation](https://realityworld.trade/wiki/Just-in-time_compilation?__cpo=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3Jn) in 1997/1998 for [Java 1](https://realityworld.trade/wiki/Java_version_history?__cpo=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3Jn).

The actual process varies from implementation to implementation based on the requirements, but the end-result is that the original code is compiled into bytecode at compile time, but then it is run through a compiler at runtime before it is executed. By doing this, the code can be executed at near-native speeds. Some platforms (.Net) saves the result of the JIT compilation, replacing the bytecode. By doing this, all future executions of the program will execute as though the program was natively compiled from the beginning.

If there are more than one class in a .java file, the compiler produces a seperate .class file for each compiled class. Compiler places both class files in the same direction.

The compiler can optimize programs by removing calls to simple methods and replacing them with the expanded code of their declarations— a technique known as inlining the code, which improves program performance.

-Java Native Interface(JNI) is a [programming framework](http://phx.4everproxy.com/secure/kc6oT8rcKWQ2jdawuE40Tt0OoXakFb4lC3QpCFz8NXE0N2OA~dk5Kcmb0swmzCWLKLBFSiaSErdR_T6bPQiCGg--) that enables [Java](http://phx.4everproxy.com/secure/kc6oT8rcKWQ2jdawuE40Tpr6_Cdane3CGqEqy92BhhIUiwcw8QD4fOsarOH2qAy2hOheHYbR~hoTlxyd4d3hqA--) code running in a [Java Virtual Machine](http://phx.4everproxy.com/secure/kc6oT8rcKWQ2jdawuE40Tpr6_Cdane3CGqEqy92BhhLdKau6LT0S76ng5rmpVNDpC4AHCBCYdKNVo0jvqj4kmg--) (JVM) to call and be called by native applications (Programs specific to a hardware and [operating system](http://phx.4everproxy.com/secure/kc6oT8rcKWQ2jdawuE40TvPVzqrZ~Bh5_h4qyoxjtpKRj6RXLKo4jl4EsTHYF8ju) platform.) and libraries written in other languages such as C, C++ and assembly.

JNI enables programmers to write native methods to handle situations when an application cannot be written entirely in the Java programming language. One good example is Math.pow(double, double);

<https://en.wikipedia.org/wiki/Java_Native_Interface>

-System software that converts source code to object code is called language processor. There are three types of language interpreters.

**-Assembler:** Converts assembly level program into machine level program.

**-Interpreter:** Converts high level programs into machine level program line by line.

**-Compiler:** Converts high level programs into machine level programs at one go rather than line by line.

<https://en.wikipedia.org/wiki/Programming_language_processor>

<https://www.tutorialspoint.com/basics_of_computers/basics_of_computers_system_software.htm>

**-JAR(Java ARchiving tool)** is a compression format. To add a jar file to your project, you right click them on IDE and choose build path -> add to build path.

**-JBD(Java DeBuging tool):** They provide tools that support development process, including editors for writing and editing programs and debuggers for locating errors and warnings.

* **Java API (Application Programming Interface)/ Java Class Libraries:**

**-API is a collection of Classes and Interfaces grouped together mainly according to their functionality.**

-They are the classes which have the functionality written in them to do things like sorting, making various network connections like HTTP, FTP connections, File handling .etc..

-For example java.net will have classes for networking, java.io for various input-output functionality

-When using an API class or interface, you either need to write the whole path or use import statement.

**-Full path:**

java.util.ArrayList<String> arr = new java.util.ArrayList<String>();

**-Import:**

import java.util.ArrayList ;

ArrayList<String> arr = new ArrayList<String>();

* **­Object oriented analysis and design (OOAD)**

Object oriented analysis and design is needed to determine project’s requirements(what the system is supposed to do) and developing a design(how the system should do it). **UML** is a good way to design a software before starting to code it.

-See “OOAD” folder for more information.