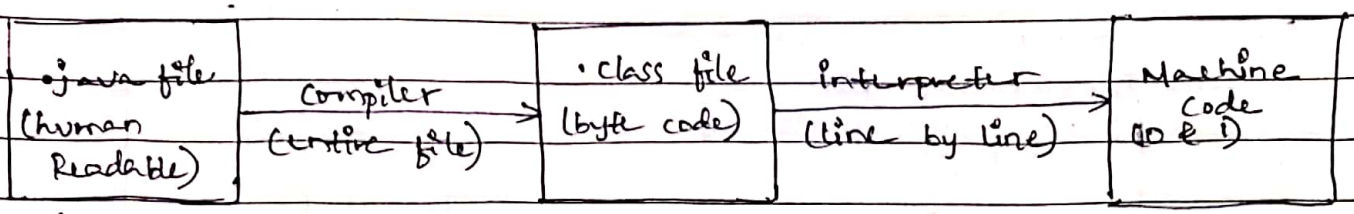
Introduction to Java - Architecture and Installation

How Java code executes?



This is the source code

- This code will not run directly on a system
- We need JVM to run this
- This is the reason why Java is platform independent

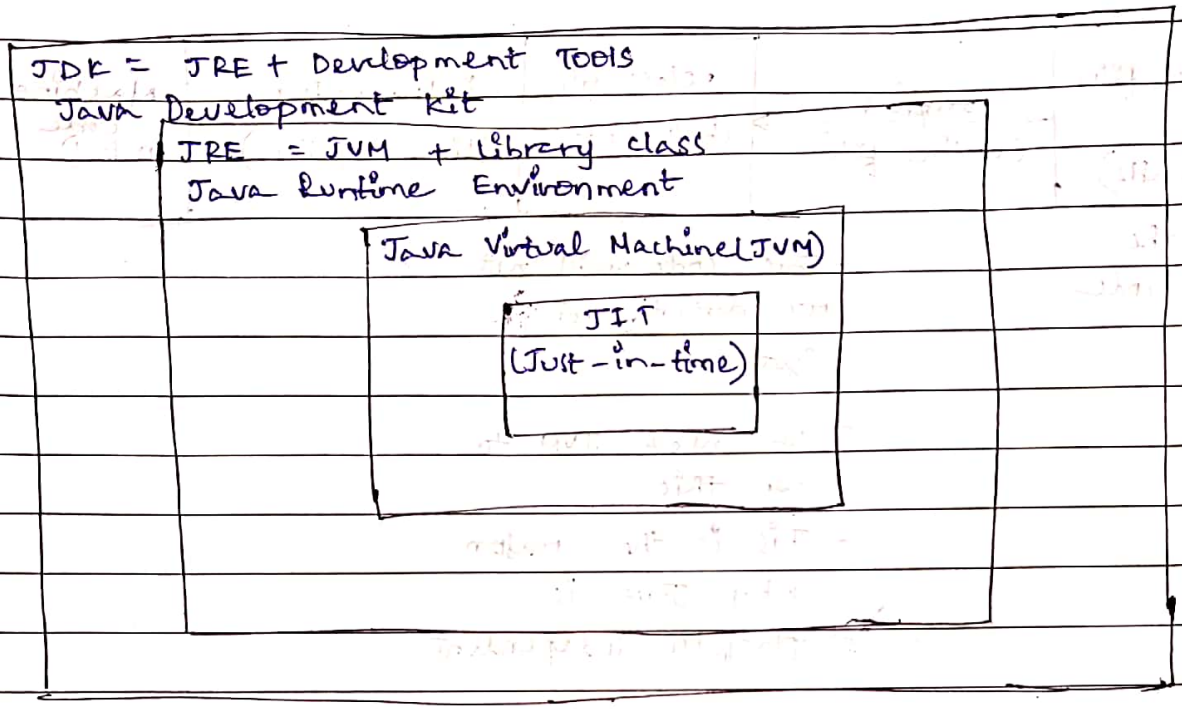
- In c & c++, the compiler converts source code to Machine code directly whereas in Java there is an intermediate step in which source code is converted to byte code (.class file) which is a sort of an another language by Java.
- When JVM converts byte code to Machine code, then it will be understood by the system.

Platform independence:

- Byte code can run on all operating systems.
- We need to convert source code to machine code so computer can understand.
- Compiler helps in doing this by turning it into an executable code, which is a set of instructions for the computer.
- After compiling c/c++ code we get .exe file which is platform dependent.
- In Java we get bytecode, which is converted to machine code by JVM.

- Java is platform independent but JVM is platform dependent.

JDK vs JRE vs JVM vs JIT



JDK:

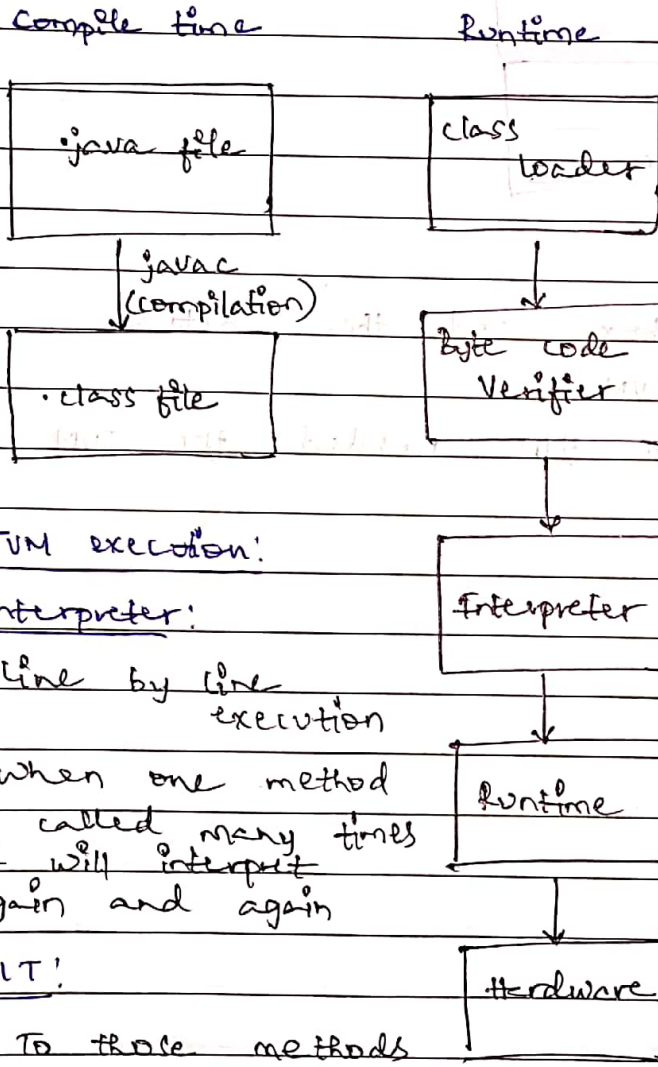
- Provides an environment to develop and run Java programs
- It is a package that includes
  - Dev tools: to provide an environment to develop program
  - JRE: to execute the program
  - a compiler - Javac
  - archiver - jar
  - docs generator: javadoc
  - interpreter/loader

JRE:

- It is an installation package that provides the environment only to run the program.
- It consists of
  1. Deployment technologies
  2. User interface toolkit
  3. Integration libraries
  4. Base libraries

5. JVM

- After we get the .class file, the things happening at run time are:
  1. class loader loads all the classes needed to execute the program.
  2. JVM sends code to byte code verifier to check the format of code.



(How JVM works) class loader:
- Loading
  - reads .class file and generates binary data
  - an object of this class is created in heap
- Linking:
  - JVM verifies the .class file
  - allocates mem for class variables and default values
  - replace symbolic references from the type to with direct references
- Initialization:
  - all static variables are assigned with their values defined in the code and static block

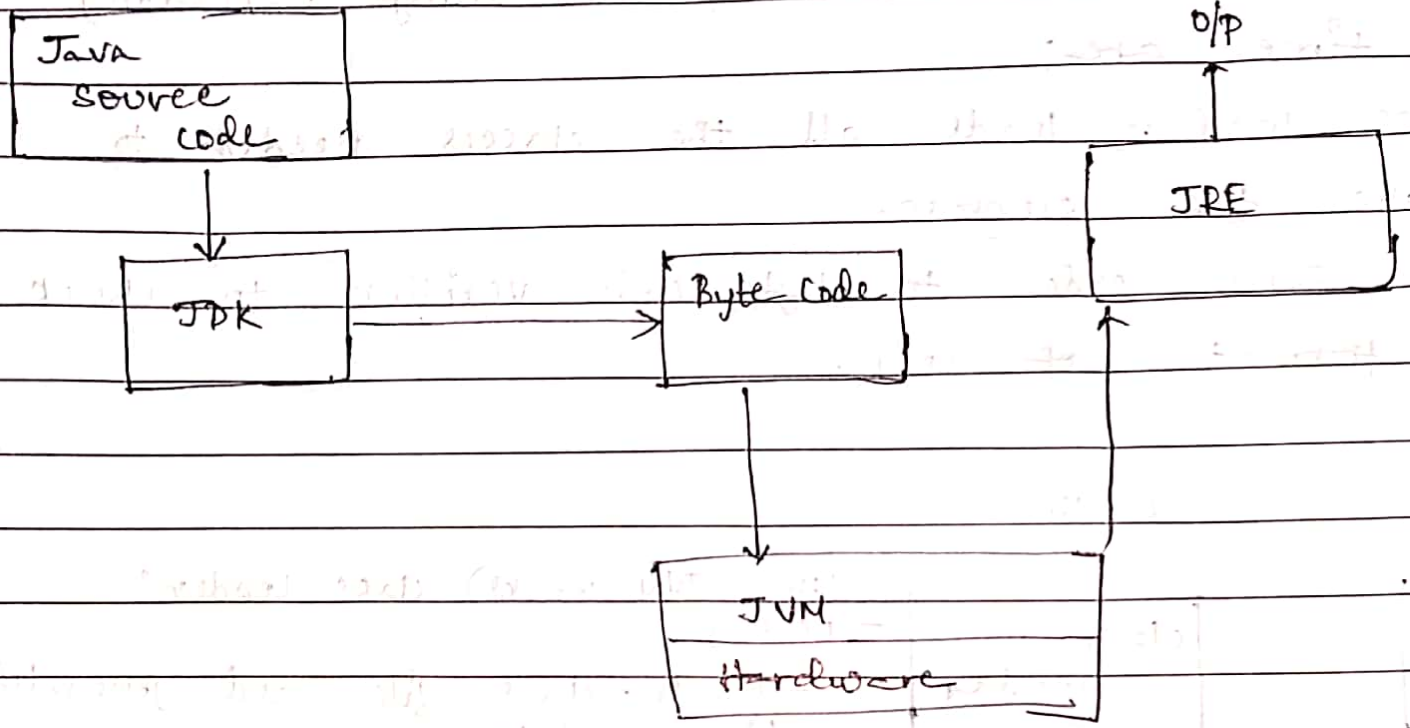
JVM contains stack and heap mem allocations.

JVM execution:

Interpreter:
- Line by line execution
- When one method is called many times it will interpret again and again

JIT:
- To those methods that are repeated, JIT provides direct Machine code so that re-interpretation is not repeated, which makes the execution of code faster
- Garbage collector

Working of Java Architecture:



JRE vs JVM:

- JRE is a sort of a box and the actual content inside in it is JVM.
- Whatever the libraries and files needed for JVM are provided by the JRE.