How Java code gets executed

.class file

(byte code)

.java

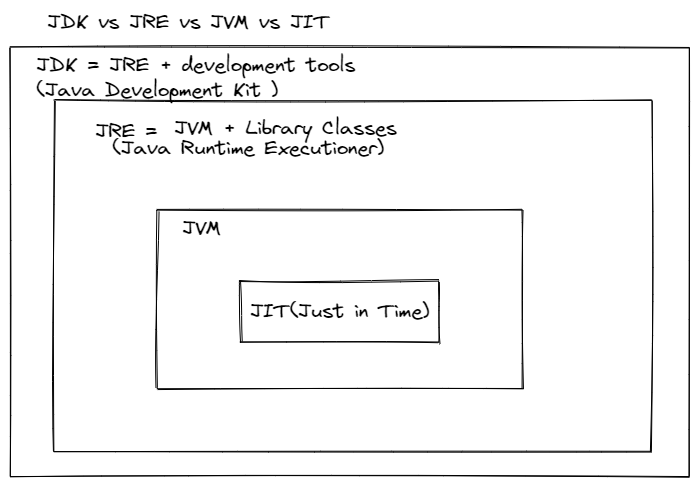
Jvm(Comp.) interpreter

Machine code

* Java bytecode is platform independent
* It won’t run directly in the system

Why Platform Independent?

* The byte code (.class file ) can run on any OS
* C/C++ is platform dependent, meaning the compiler needs separate configuration based on the system architecture
* In java we get the byte code which is converted to machine code
* However JVM is platform **dependent,** meaning it needs separate configuration as well to execute/interpret the byte code.



Java Development Kit

* It provides environment to run a java program
* It’s a package that includes

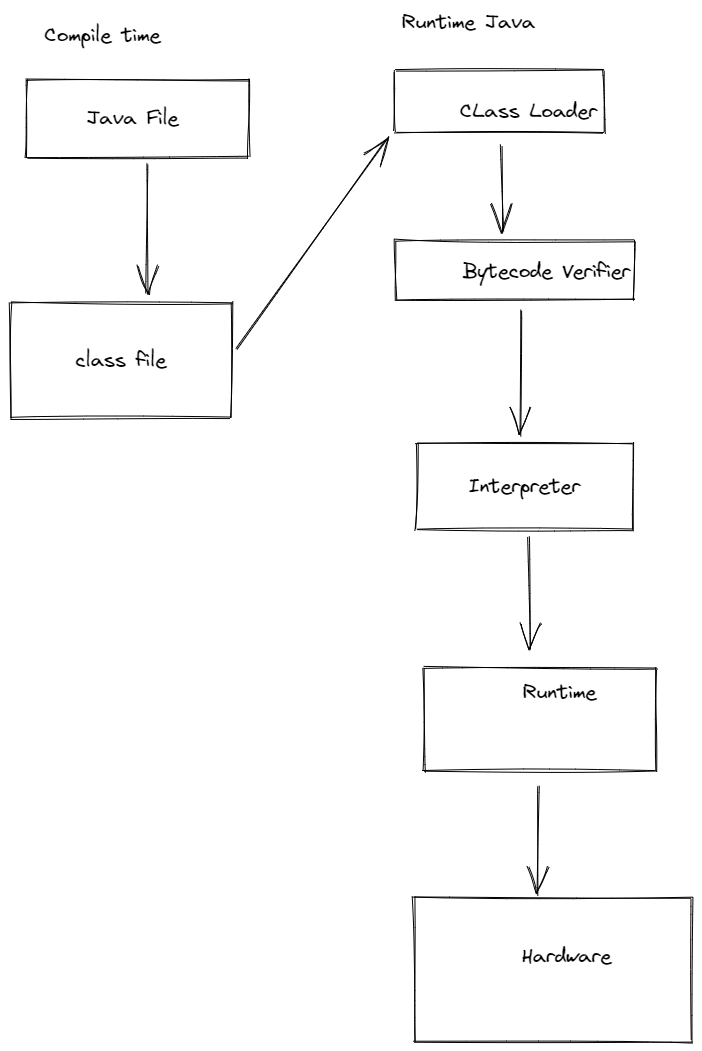
1. Development tools – to provide a environment to develop the program
2. JRE – runtime executioner
3. Javac – compiler
4. Javadoc – for docs writing
5. Java interpreter /loader
6. Archiver – jar

Java Runtime Environment ( inside Java Development Kit )

* It an installation package that only runs the program
* It consists of

1. UI tools
2. Deployment Technologies
3. Integration Libraries
4. Base Libraries
5. JVM

* After getting the .class file (source code to byte code)
* Class loader loads all the classes
* JVM sends code to byte verifier to check the byte code

****

(How JVM works) Class Loader

Loading

1. Reading the .class file and getting the binary data
2. Object of this class is created in the heap memory

Linking

1. JVM verifies the class file ( Looks for illegal practices)
2. Allocates memory for the class variables
3. Removing symbolic references with direct references

Initialization   
1. All static variables are initialized with values defined in its block

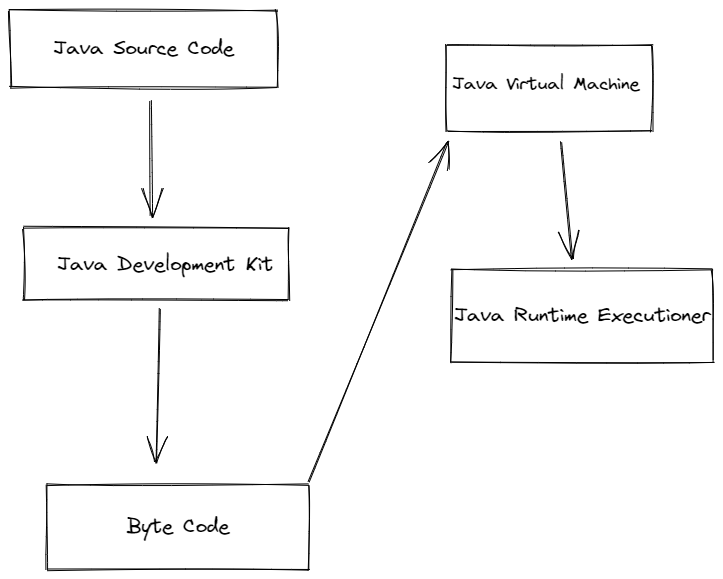
JVM contains the stack and heap memory

JVM Execution   
-Interpreted line by line

-same method are re-executed

JIT (Just in Time Compiler

-It stores the information of same method so that there won’t be any duplicity in interpretation



Hardware