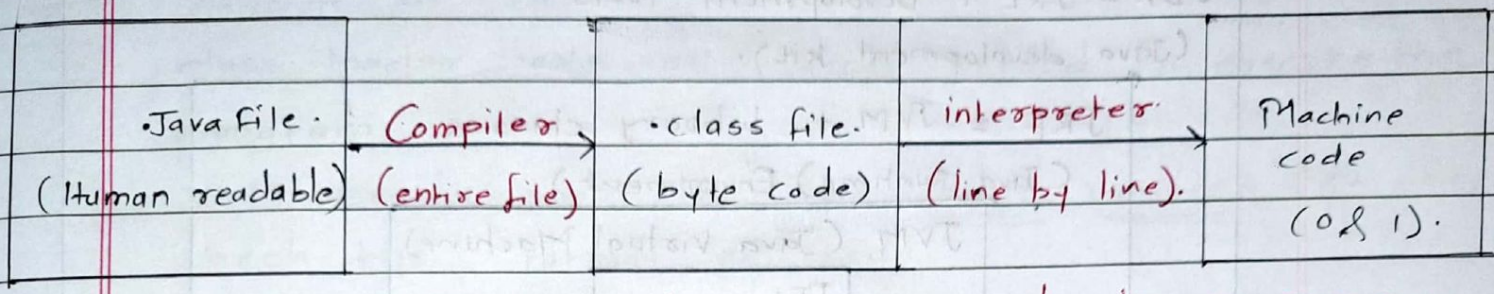
#8 Introduction to JAVA.

classmate
Date
Page

8 Aug 2021.

How Java code Executes

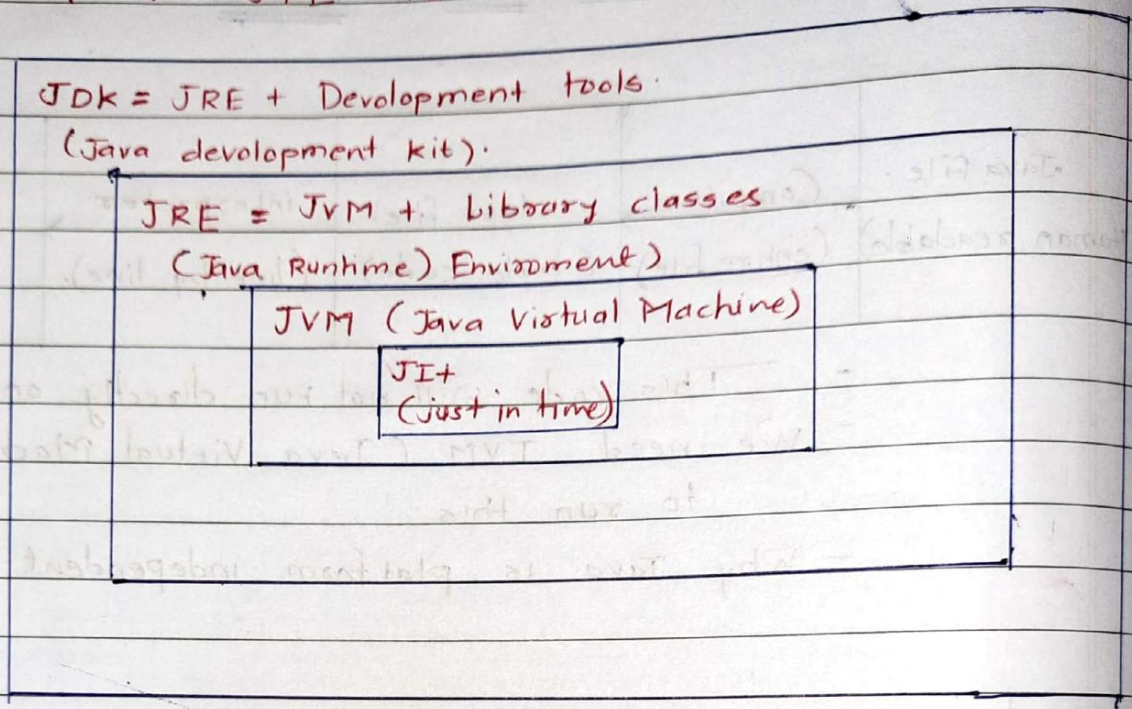


- This code will not run directly on ~~mac~~ sys.
- We need JVM (Java Virtual Machine) to run this.
- Why Java is platform independent.

★ More about platform independence :-

- It means that byte code can run on all operating sys.
- We need to convert source code to machine code so Computer can understand.
- Compiler helps us by doing this, turning it into executable code.
- this executable code is a set of instruction for Comput
- After Compiling c/c++ code we get .exe file which is platform dependent.
- In Java we get byte code, JVM Converts it into Machine code.
- Java is platform-independent but JVM is platform dependent.

★ JDK Vs JRE Vs JVM Vs JIT.



★ JDK =

- Provides enviroment to develop and run the Java Pro
- It is a Package that includes:

1) development tools - to provide an enviroment to develop your program.
2) JRE - To Execute your program.
3) A compiler - Javac.
4) archiver - Jar.
5) docs generator - Java docs.
6) interpreter / loader.

★ JRE :-

- It is an installation package that provides enviroment to only run the program.
- It consist of:

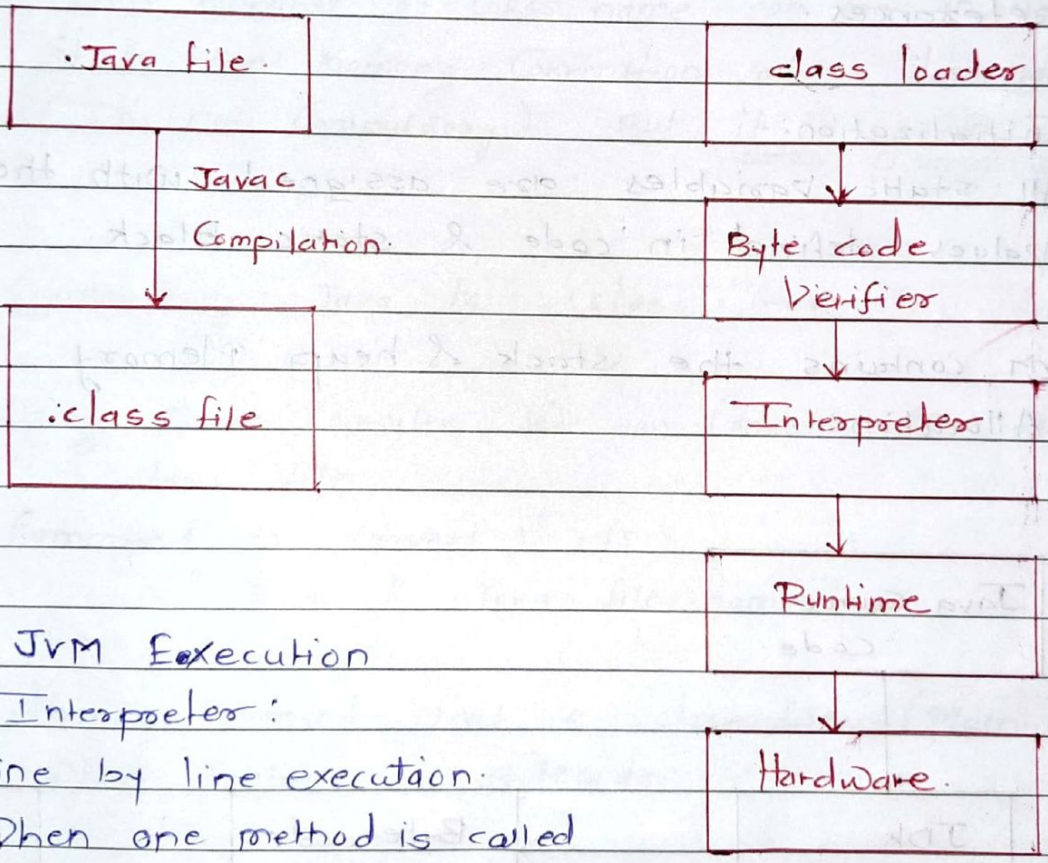
① Deployment technologies.
② User interface toolkit.
③ Integration libraries.
④ Base libraries.

⑤ JVM (Java Virtual Machine).

- After we get the .class file the next things happen at runtime.

① class loader loads all classes needed to execute the program.

② JVM sends code to "Byte code verifier" to check the format of code.



JVM Execution

Interpreter:

- line by line execution.
- When one method is called again and again it will interpret again & again.

JIT:

- Those methods are repeated JIT provides direct Machine code. So re-interpretation is not required.
- Makes execution faster.
- Garbage collector.

* (How JVM works) class loader :-

* loading :-
- reads .class file & generate binary data
- object of this class is created in heap.

* Linking :-
- JVM verifies the .class file.
- Allocates memory for class variables & default val
- replace symbolic from the type with direct references.

* Initialization :-
- All static variables are assigned with there values defined in code & static block.

* JVM contains the stack & heap Memory Allocations.