Date Jun-2
Page No.

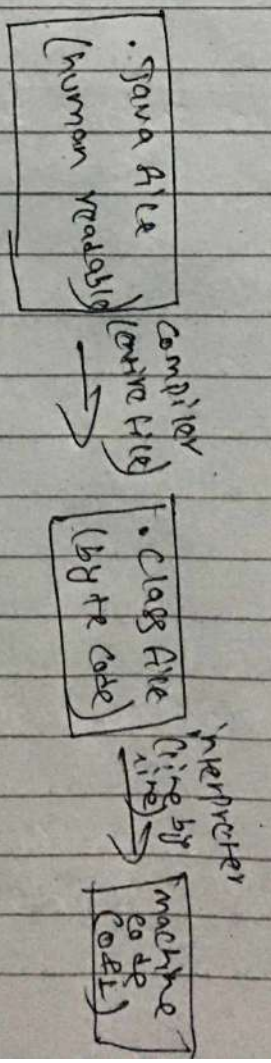
Introduction to Java - Archiecture

- java is used to save Java file.

Java working :

Compiler compiles the java file into byte code ~~as first~~ (.class file) at first most it is converted to machine code (0 & 1) by using Java virtual machine (JVM).

Diagram



Date
Page No.

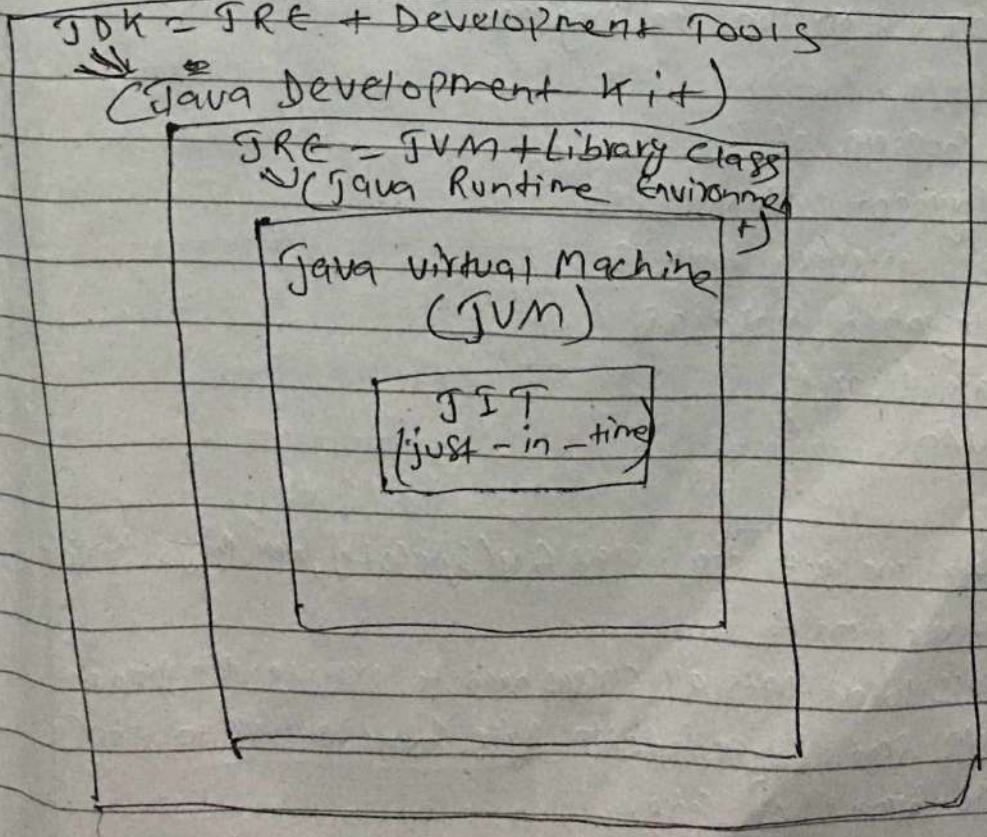
ii) Java is platform independent.

It is an platform independent means that byte code can run on all operating systems.

- In java we get byte code, JVM converts this to machine code. but in other like c/c++ we get .exe file which is platform dependent.

- But JVM is platform dependent.

Archiecture of Java



1. JDK (Java Development Kit)

=> It provides environment to develop and run the Java program.

- It is a package that includes:
(i) development tools -> to provide an environment to develop your program.
(ii) JRE -> to execute your program
(iii) a compiler -> javac (converts .java file to .class file (byte code))
(iv) archiver -> jar
(v) docs generator -> javadoc
(vi) interpreter / loader

2. JRE (Java Runtime Environment)

=> It is an installation package that provides environment to only run the program.

- It consists of:
(i) Deployment technologies
(ii) User interface toolkits
(iii) Integration libraries
(iv) Base libaries
(v) JVM

- After we get the .class file (byte code), next things happen at run time:
(1) Class loder loads all classes need to execute the program.
(2) JVM sends byte code to Byte Code verifier to check the format of code.

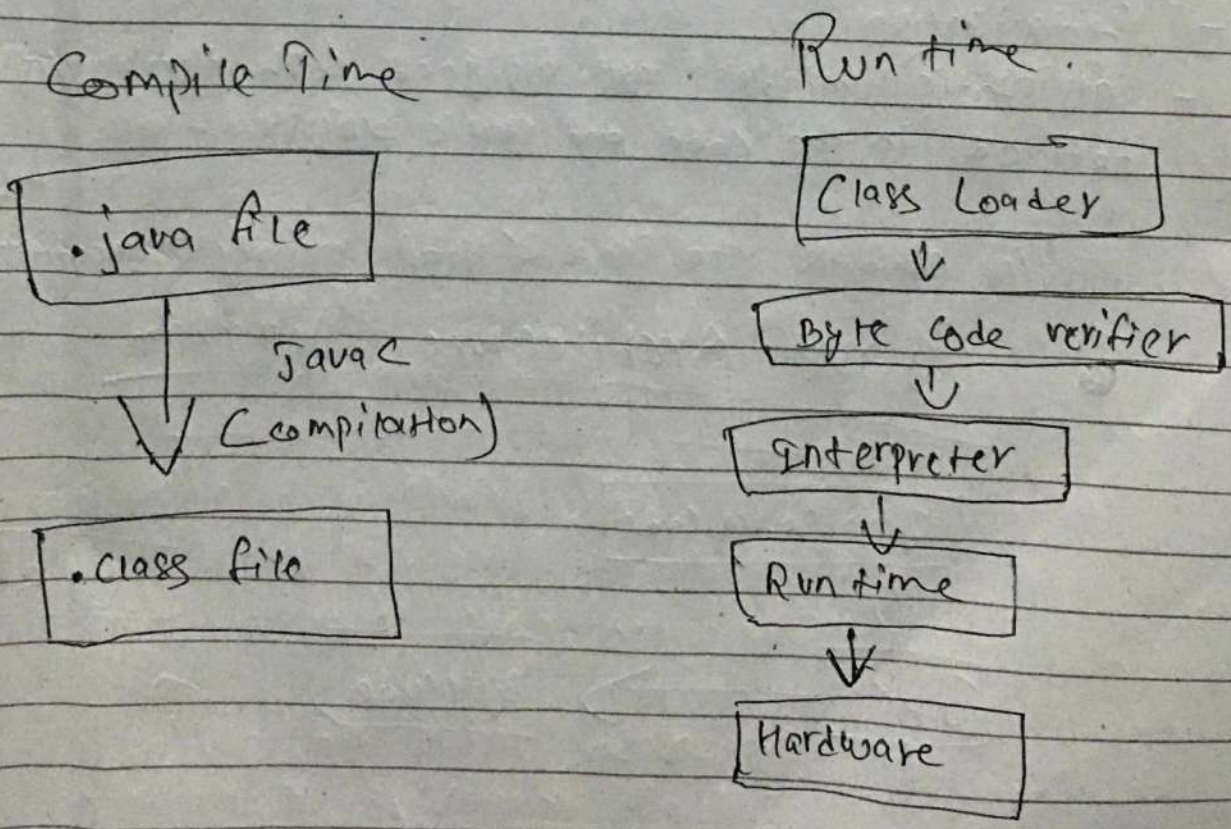
3. JVM Execution

Interpreter :

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

4) JIT

- Those methods that are repeated, JIT provides direct machine code. So re-interpretation is not required.
- makes execution faster ~~Garbage~~
- Garbage Collector.



- (How JVM works) Class loader

- Loading:
  - reads .class file and generate binary data.
  - an object of this class is created in heap.

- Linking
  - JVM verifies the .class file
  - allocates memory for class variables & default values.
  - replace symbolic references from the type with direct references.

- Initialization
  - all static variables are assigned with their values defined in the code and static block

JVM contains the stack & Heap memory allocations.

- Java Architecture Diagram