Introduction to Java.

1) How Java Code Executes:

- Machine only understand 0's and 1's but it will very difficult to use write in 0's & 1's.
- In order make human readable, structure manner we use programming lang.

Java File extension : .java file (we write our code here)

Now after writing we give this .java file (source code) give to compiler (javac) it will convert .java file to byte code (.class) file.

.Class file (byte code)

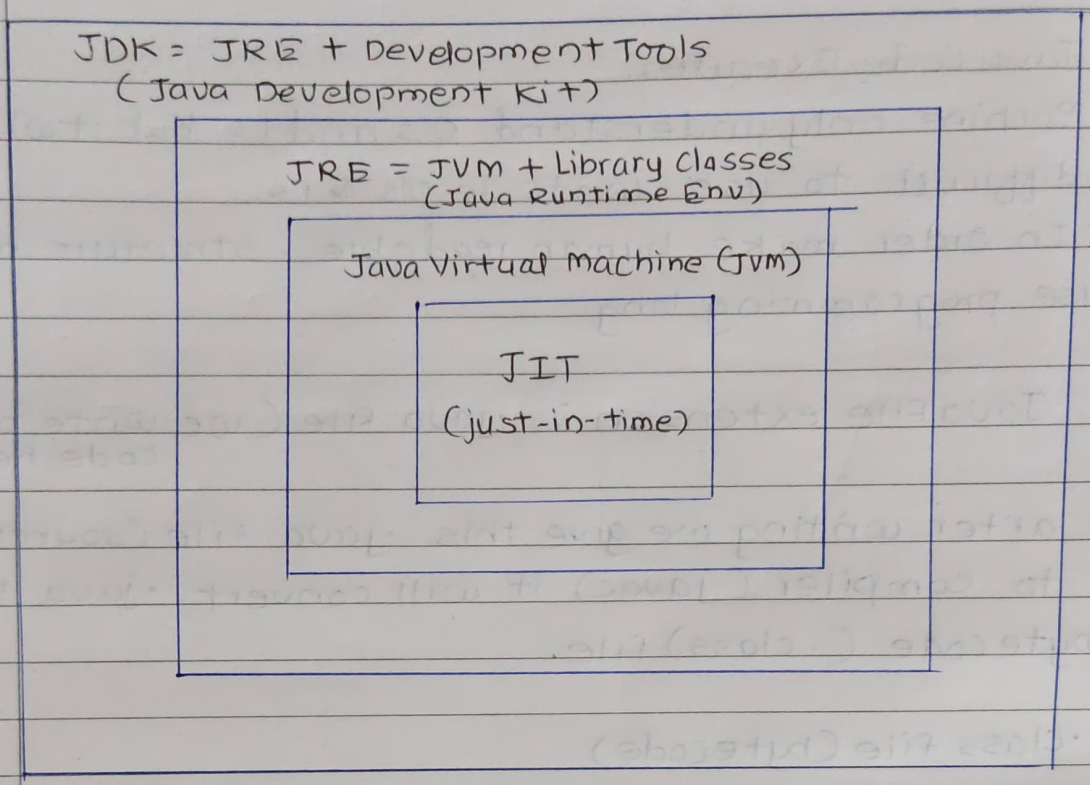
- this code will not run directly on a system.
- We need JVM to run this
- Reason why Java is platform independent.

Byte code is interpreted using JVM and converts to machine code (010101)

More about platform independent?

- That bytecode can run all on OS. (Windows, Linux, macos)
- ~~the~~ After compiling C/C++ code we get .exe file which is platform dependent.
- In Java we get bytecode, JVM converts this to machine code.
- Java is platform independent but JVM is platform dependent.

JDK vs JRE vs JVM vs JIT



1) JDK: provides env to develop and run the Java Program.
   - It is package that includes
     1. development tools : To provide env to develop your program
     2. JRE : to execute your program
     3. a compile - javac
     4. archiver - jar
     5. docs generator - javadoc
     6. interpreter/loader

2> JRE: It is an installation package that provides env to only run the program.
   - It consists of :
     1) Deployment Tech.
     2) UI toolkits
     3) Integration libraries
     4) Base libraries

5) JVM

- 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.

Compile Time	Runtime
.Java file	classloader
↓ (javac compilation)	↓
.class file (bytecode)	ByteCodeVerifier
	↓
	Interpreter
	↓
	Runtime
	↓
	Hardware

(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 var
  - Replace symbolic references from the type with direct references
- Intialization: all static variables are assigned with thei values defined in the code and static block

JVM contains the stack and Heap memory allocations.

JVM Execution:

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

JIT:

- those methods that are repeated, JIT provides direct machine code. so re-interpretation is not required
- makes execution faster.