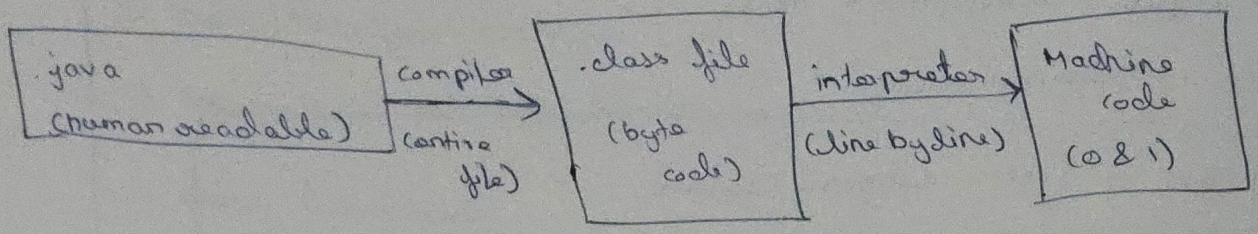
## How Java code executes:



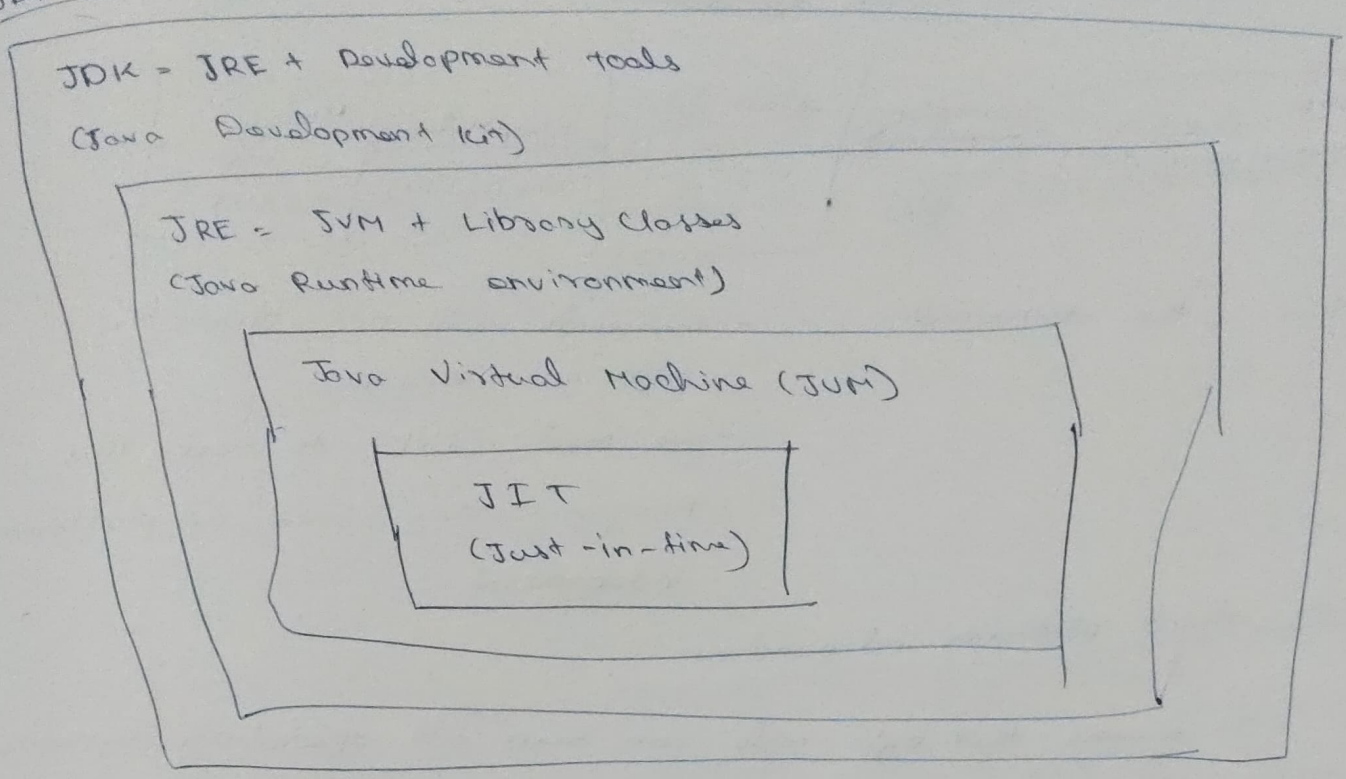



this is the source code
- this code will not directly run on a system
- we need JVM to run this
- Reason why Java is platform independent.

### + more about platform independant:

- It means that 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 executable code.
- this executable code is a set of instruction for the computer.
- After compiling C/C++ code we get .exe file which is platform dependant.
- 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:




* JDK:

- Provides environment to develop and run the Java program.
- It is a package that includes:
  1. development tools - to provide an environment to develop your program,
  2. JRE - to execute your program.
  3. a compiler - javac
  4. archiver - jar
  5. docs generator - javadoc
  6. interpreter / loader

* JRE:

- It is an installation package that provides environment to only run the program.
- It consists of,

1. Deployment technologies.

2. User Interface toolkits.

3. Integration ~~tool kits~~ libraries.

4. Base libraries.

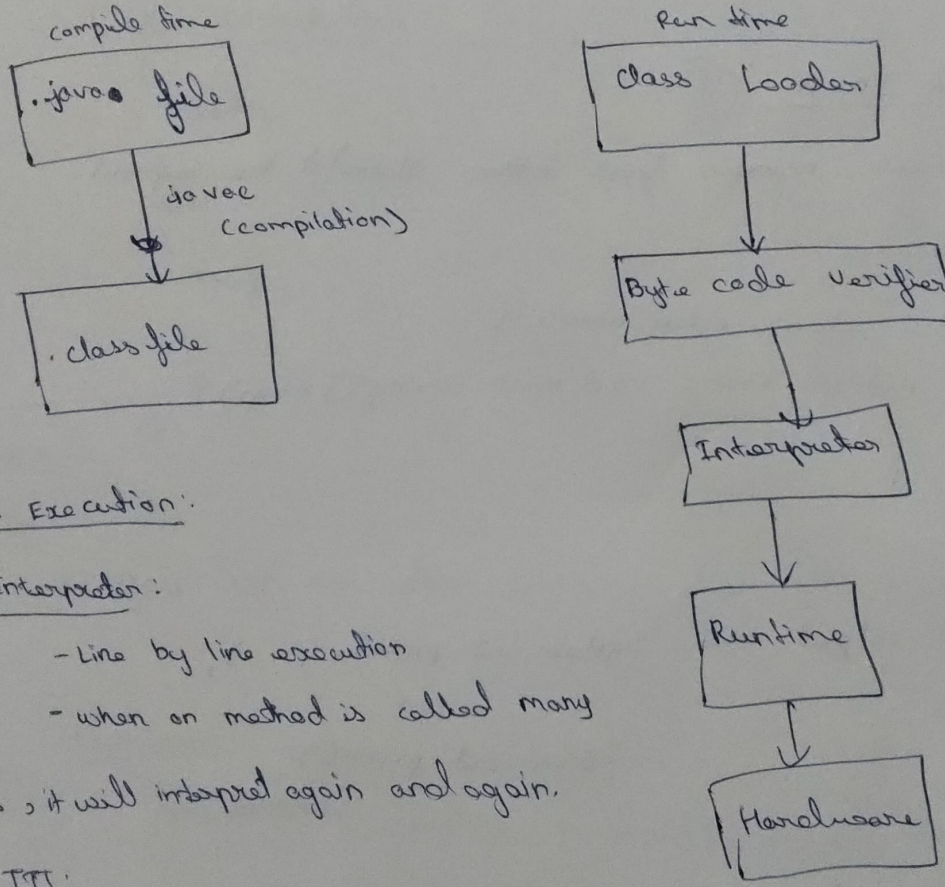
5. JVM.

- 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.

Flowchart of Java file execution:




JVM Execution:

Interpreter:

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

JIT:

- those methods that are ~~not~~ repeated

JIT provide direct machine code, so re-interpretation is not required.

- makes execution faster 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 byte with direct references.
- Initialization:
  - all static variables are assigned with their values defines in the code and static block.

JVM contains the stack and Heap memory allocation.

First Program in JAVA: (8)

- Java class name's first letter should be capital.

Main function: