9/9/21

* Java Course *

Types of Languages

Procedural | Functional | Object-Orient

1) Procedural :- It specifies a series of well-structured steps and procedures to compose a program.
It contains a systematic order of statements, functions and commands to complete a task.

2) Functional :- Writing a program only in pure functions i.e never modify variables, but only create new one's as an output.
Used in situations where we have to perform lots of different operations on the same set of data, like ML.
First class functions. eg :- a = 10 , b = 30 , c = b

3) Object Oriented :- It revolves around a objects.
Code + ~~Object~~ Data = Object
Developed to make it easier to develop, debug, reuse and maintain software.

→ Static Vs Dynamic Languages

1) static :- a) Perform type checking of function at compile time.
b) errors will show at compile time.

c) declare datatype before use it
d) more control

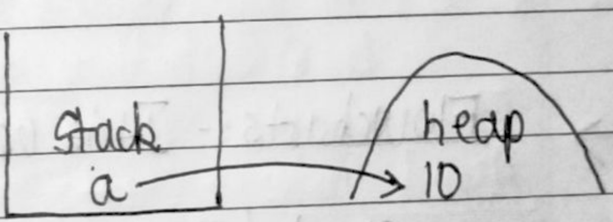
eg: int a =10
    str b = "Akash "

2) Dynamic:- a) It performs type checking of function at ~~run~~ time.
b) Error might ~~now~~ not show till program is run.
c) No need to declare datatype of variables.
d) saves time in writing code but might give error at runtime.

eg: a =10
    a= "Akash"

→ Understanding of Memory Management

2 Types of memory:



eg: a = 10 → Object → It is stored in heap.
↙ reference variable
↳ It is stored in stack

⇒ More than one reference variable can point towards ~~that~~ the same object.

⇒ If the object ~~is~~was changed by a one reference variable than this change will also be visible to other to all the other ~~of~~ variables which are pointing towards the same object.

eg: $a = [\overset{0}{1}, \overset{1}{3}, \overset{2}{5}, \overset{3}{9}]$

$b = a$

$a[0] = 1$

Let $a = [99, 3, 5, 9]$

<u>Output</u>: $b[0] = 99$

⇒ Object with no reference variable will be removed from the memory. i.e garbage collection

eg: $a = 10$
$a = 37$

a → 10 (X) ↓ this is now garbage in memory.
→ 37

→ <u>Flowcharts</u>:- It is used to visualize the working of programming language.

Start / Stop ⇒ ⬭

Input / Output ⇒ ▱

Processing ⇒ ▭

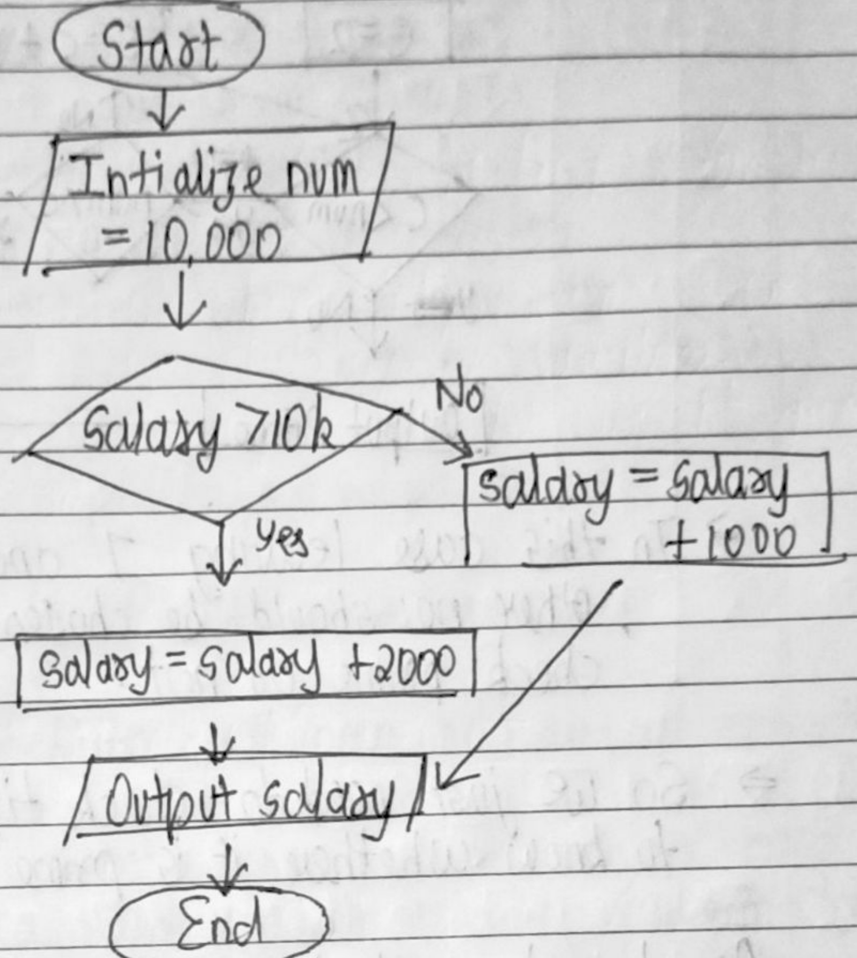
condition ⇒ ◇

Flow direction of program ⇒ ⟶

eg: Take input of ~~input of~~ a salary. If the salary is greater than 10,000 add bonus as 2000, otherewise add bonus as 1000.

Pseudo Code:

```
start
input salary
if salary > 10,000 :
  salary = salary + 2000
else:
  salary = salary + 1000
output salary
exit
```

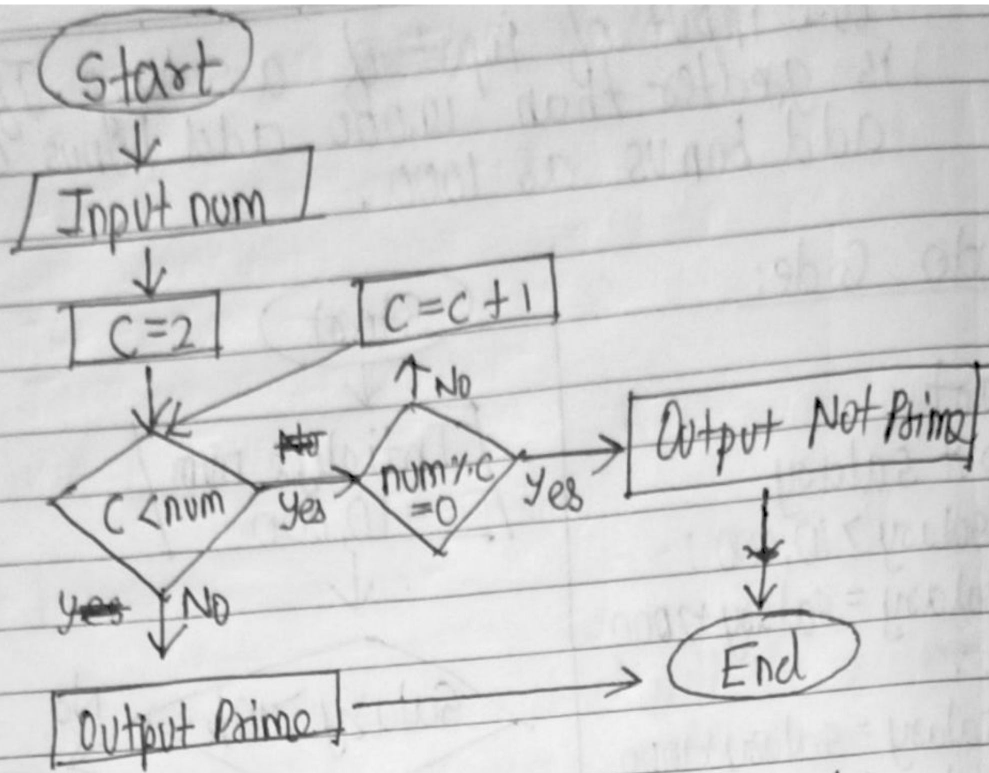


eg: Input a numbere and print whether it is prime or not.

eg: 7
(2, 3, 4, 5, 6)

Pseudo Code:-

```
start
input num
c = 2
while c < num :
    if n % c = 0 :
        output "not prime"
    c = c + 1
end while
output "prime"
exit
```



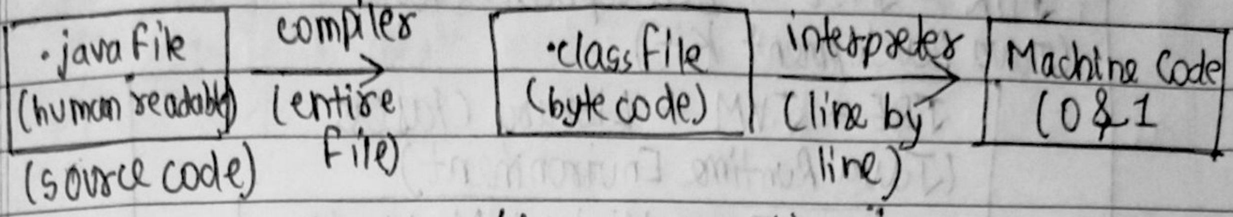
→ In this case leaving 1 and that numbere, other no: should be chosen for range to check prime or not.

⇒ So we just need to check till the √num to know whethere it is prime or not.

Pseudo code:

```
start
input n
if n <=1;
     print ("neither prime nor composite")
  c = 2
  while c*c <=n
       if n % c==0 ;
            output "not prime"
            exit
       c+= 1 (c=c+1)
  end while
  output "prime"
exit
```

→ How Java code executes:



- this code will not directly run on a system
- We need JVM to run this
  ↳ (Java Virtual Machine)
- This is why Java is platform independent.

⇒ What is mean by Platform Independence

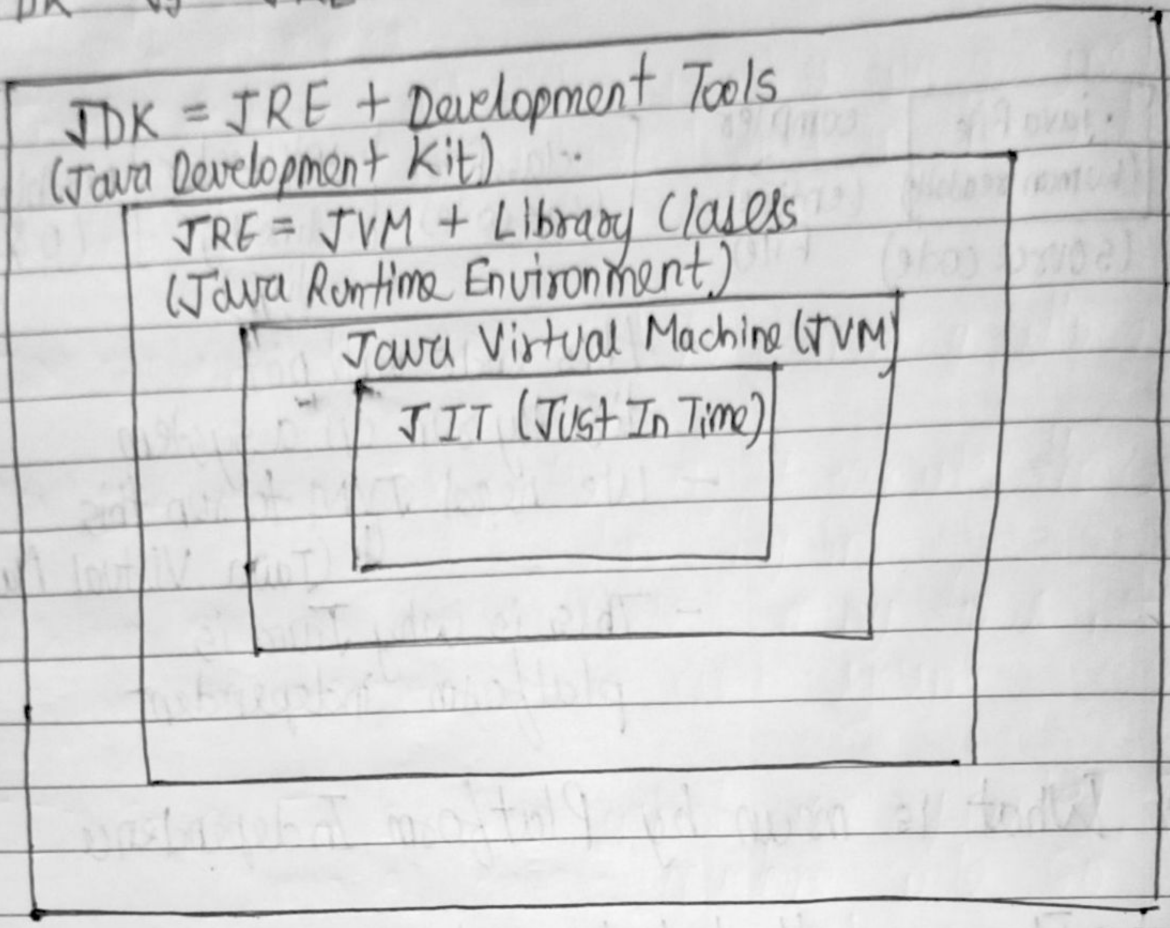
- a) It means that byte code can run on all operating System.

b) We need to convert source code to machine code so computer can understand, compiler helps in doing this by turning it into excequtable code. This executable code is a set of instructions for the computer.

c) After compiling C/C++ code we get .exe file which is platform dependent.

d) In Java we get byte code, JVM converts this into machine code. So Java is platform independent but JVM is platform dependent.

⇒ JDK Vs JRE Vs JVM Vs JIT


JDK = JRE + Development Tools
(Java Development Kit)
JRE = JVM + Library Classes
(Java Runtime Environment)
Java Virtual Machine (JVM)
JIT (Just In Time)

① JDK : It provides an environment to develop and run the Java Program.

It is a package that includes :

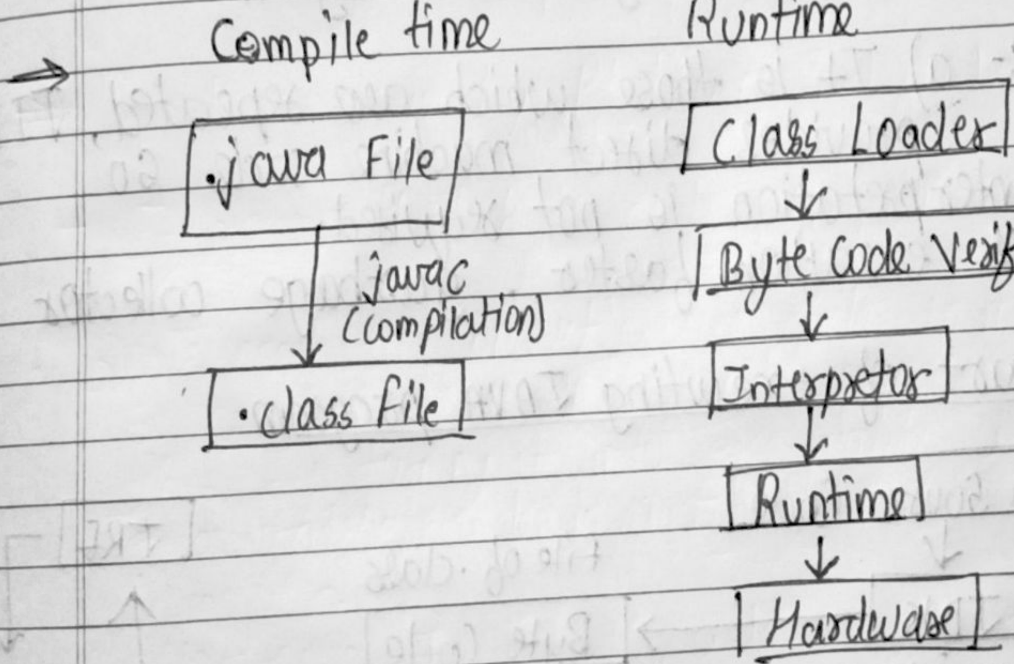
a) development tools : It provides an environment to develop your program.
b) JRE : It is used to ~~execute~~ execute the program
c) a compiler : javac
d) Archiver : jar
e) docs generator : javadoc
f) Interpreter / Loader

② JRE : It is an installation package that provides environment to only run the program.

It consists of :

a) Deployment technologies
b) User Interface toolkits
c) Integration libraries
d) Base libraries
e) JVM

After we get the .class file, the next things at runtime
a) Class loader loads all classes needed to execute the program.
b) JVM sends code to byte code verifier to check the format of code.



→ How JVM works? (JVM - class loader)

a) Loading: ~~it~~ reads .class file and generates binary data
(ii) an object of this class is created in heap.
b) Linking :- (i) JVM verifies the .class file
(ii) It allocates memory for class variables & default values.

(iii) It replace Symbolic references from the type with direct references.

c) Intialization :- (i) all static variables are assigned with their values defined in the code and static block.

→ JVM contains the stack and heap memory allocations.

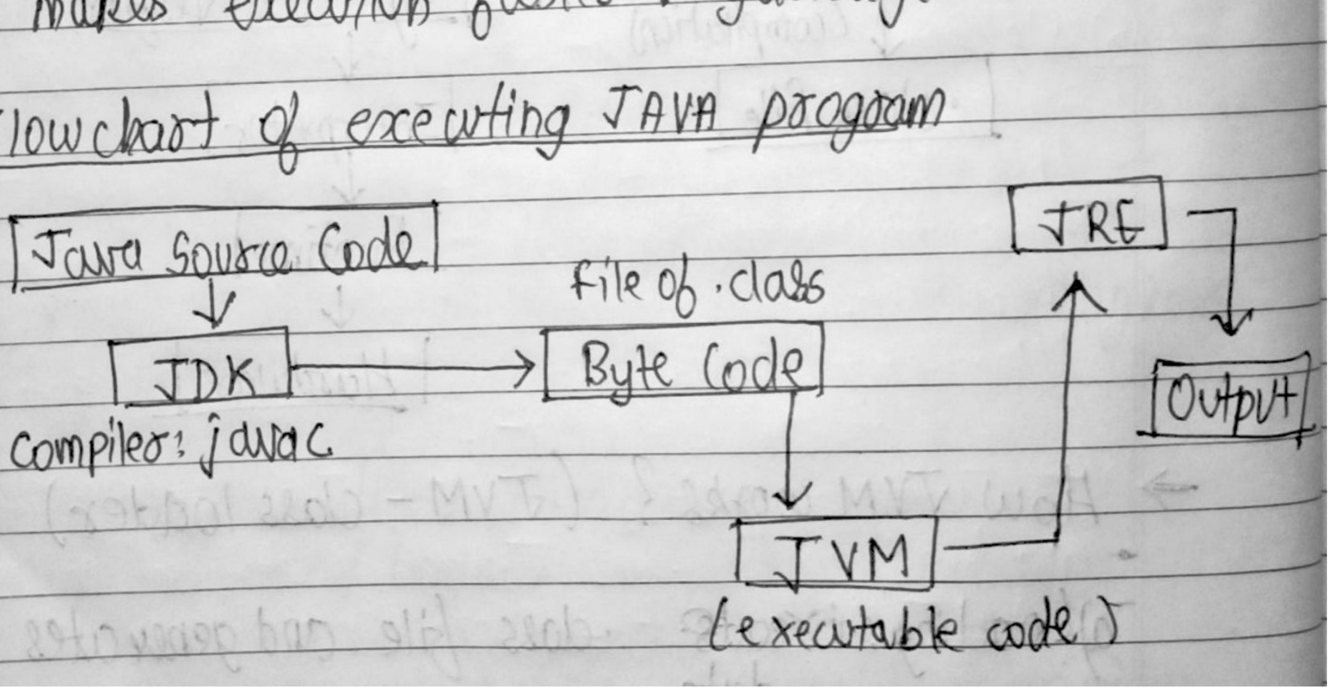
③ JVM Execution Interpreter ; a) It executes the code line by line.

b) When one method is called many times, it will ~~again~~ o interpret again and again.

④ JIT :- a) It is those which are repeated, JIT provides direct machine code so re-interpretation is not required.

b) makes execution faster. garbage collector

→ Flowchart of executing JAVA program


Java Source Code
JDK
compiler: javac
File of .class
Byte Code
JVM
(executable code)
JRE
Output