KUNAL KUSHAIAHA

Types of Languages.

procedural — functional — Object Oriented

programing language → computer understands only 0s and 1s so PL and because we can give instruction to computer in human readible format. (internal it covert in bytes of 0s & 1s)

PROCEDURAL →

- Specifies a series of well structured steps and procedures to compose a prgm
- contains syntactic order of statements functions & commands to complete a task. Eg sum of 2 → take inp1 - take inp2 → add → print.

FUNCTIONAL

- A particular branch of code that can be reused again and again

Eg I have 10 files in which I have to do sum so within one function I can call that (and do the sum)

Adv → If any changes require I would have to make changes only once.

Use → used when we need to perform same operation on different data.

→ Never modifies variables, but only creates new one as output.

Date
Page

FIRST CLASS FUNCTIONS

a = 20
b = 30
c = b

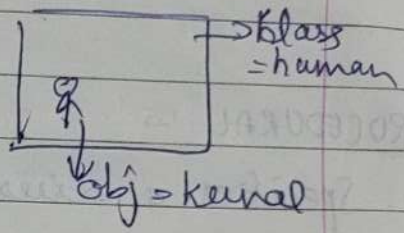
} Here how we assign c to b so if we print c it will give 30

similarly same be done with functions

Java → doesnot supports FIRST CLASS FUN
Python → supports FIRST CLASS FUN.

3) OBJECT ORIENTED :=

code + data = object.



Eg) take ~~marks~~ rollno of all stud (int)
~~take~~ take name of all stu (string)
take percent of all stu (decimal)

Now if someone asks to take ~~all~~ for all stud collectively & print their rollno, marks, name then what would be the datatype ?
ie CUSTOM DATA TYPE.
⇓
we can specify it using classes

Eg 2) car → need to just change its lights
(we donot chage entire car).

CLASSES → Named group of property and functions.

Seperate data type for student (not int, not str) its collection

OBSECT → instance of class

a = 10

variable obj

Object oriented → it was developed for ease of develop, debug, reuse and maintain software

engine → needs to change from diesel to petrol

it doesn't makes sense to change entire car instead

create separate ~~class~~ obj for car & each part and do the changes whenever required

---

procedural	functional	object oriented
c++		c++
java		java
python	python	python

STATIC	DYNAMIC
1) variable type checking at <u>compile time</u>	variable type checking at <u>run time</u>
int rollno = 56 int name = "string" → error. int a = 5 → while converting source code into machine code, it checks the type of a	(entire code is converted into machine language and then it is running) a = 20 a = kunal we dont worry about specifying the type previously ⇒ converts whole code into machine code & then checks type of 'a'
⇒ Errors will show at compile time	⇒ errors might not show till program is run
⇒ need to declare datatype before we use it	⇒ donot need to declare datatype of variables.
⇒ more control, takes time to declare variable	⇒ saves time in writing code but might give error at run time
⇒ Eg Java, C++	Eg Python

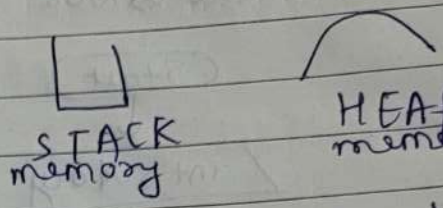
COMPILE TIME → Source code (Human understandable format) into computer understandable lang (0's & 1's) is called compile time

MEMORY MANAGEMENT

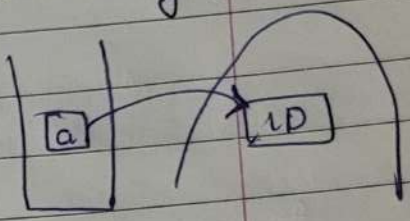
a = 10

reference variable → object

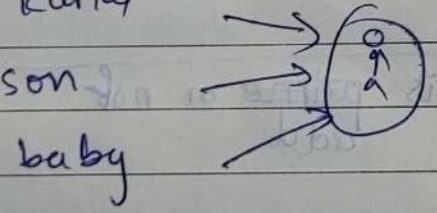
2 Types of Memory →



1. reference variable is stored in stack = a
2. object is stored in heap = 10
3. reference variable is pointing towards object



1. more than one reference variable can point to a Object
2. if any one of reference variable changes the object the object would be effected for all

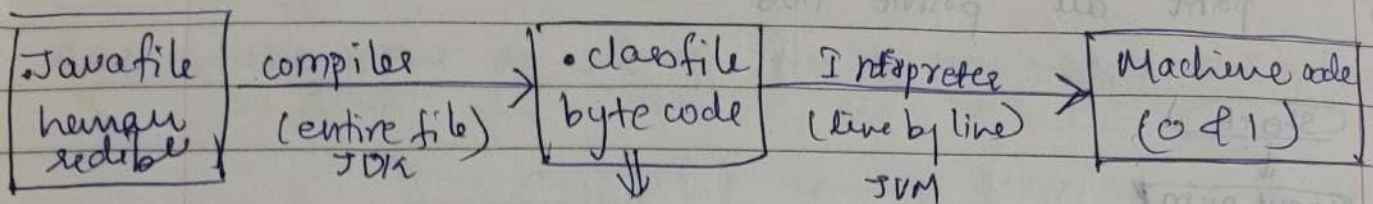


son gets haircut
kunal gets haircut
↓
son gets bald
baby gets bald

JAVA → Pass by reference value

Eg
```
a = [1, 2, 3]
b = a
a[0] = 99
sop(b) = 99, 2, 3
sop a : 99, 2, 3
```

How JAVA CODE EXECUTES



This is source code

we cannot read it

This is Machine code

→ Java file will have extension of .Java
→ in .Java we write our code (sum of 2 nos)
→ COMPILER → takes .Java file (human readable) and converts into byte code and extension of that file is .class

.Java file → .class file

byte code ⇒ is some intermediate code of Java.

→ JVM → the .class file (has byte code) code needs to run. In order to run it we require Java Virtual Machine

JVM interprets the code of .class file (byte code) line by line and converts it into machine readable format (0's & 1's)

In C++ the mid part is skipped. Directly by compiler the code is converted into Machine code

Qu) Why Java is platform independent & C++ not ?

Platform Independent and Dependent

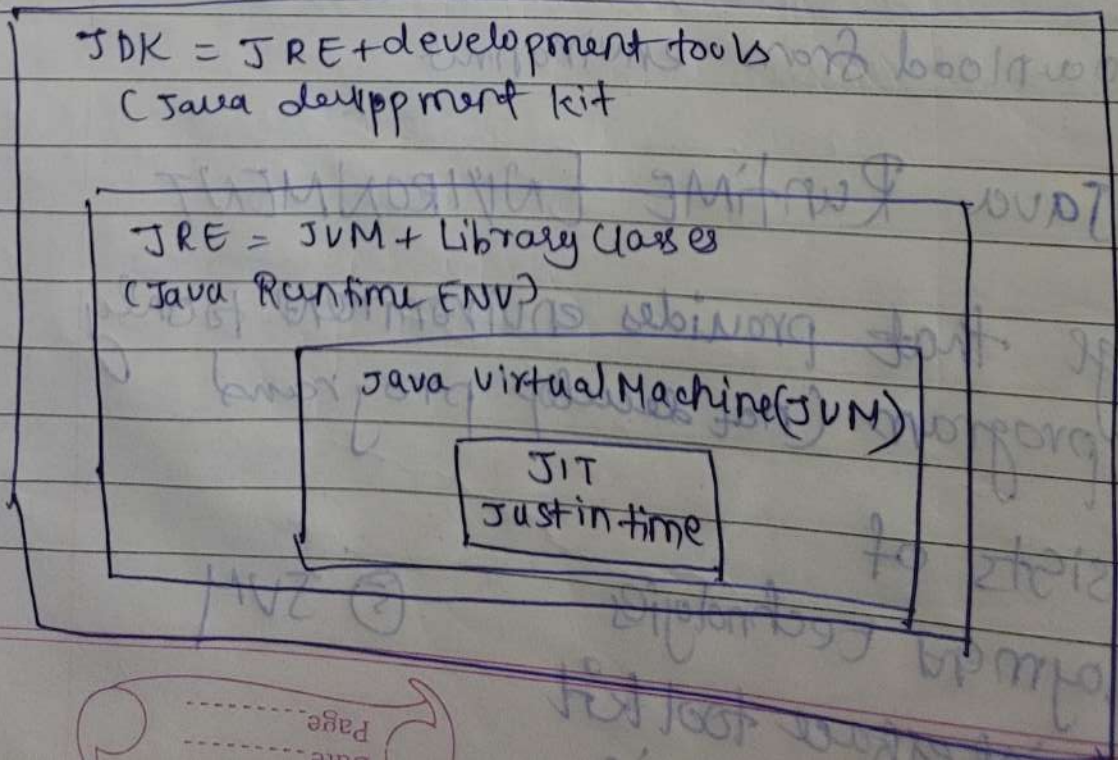
• Java file (sum of 2 nos) → compiler (JDK) → • class file (byte code) → JVM → Machine code (0's 1's binary lang)

→ byte code can run on all os
→ compiler converts our code to • class file
→ we request JVM (prgm) to convert byte code to Machine code

⇒ So I will share "byte code" with everyone mac, linux ubuntu and they can run this code on any system (all you need is JVM on that Machine)

→ Java is platform independent
JVM is platform dependent.

→ where as in c++/c after executing the code.
we get • exe file that is platform dependent



① JDK - Java Development kit.

→ we download it from internet

→ compiler that converts .Java file to .classfile ie (i code to byte code)

→ It provides environment to develop and run Java program

→ To we want to write java app we need JDK

→ Its package includes

① Development tool → to provide an environment to develop program

② JRE → to execute prgm

③ a compiler - Javac

④ Archiver - Jar

⑤ docs generator - Javadoc

⑥ Interpreter / loader

→ we can download from environment

② JRE - Java RunTiME ENVIRONMENT

→ a package that provides environment to only run the program (not develop program)

→ It consists of

① Deployment technologies

② User interface toolkit

③ Integration libraries

④ Base libraries

⑤ JVM

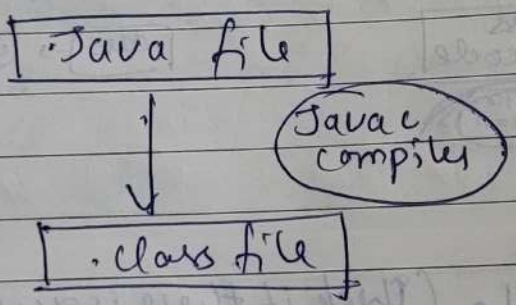
→ after we get the .class file, the next thing happens at run-time

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

② JVM sends code to "Byte code verifier" to check format of code. (to check error)

With JRE - we cannot develop a code, we can only run the code

COMPILE TIME ⇒ ~~Java~~



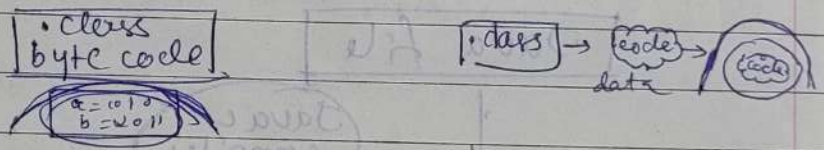
RUNTIME

[class Loader]
↓
[Byte code verifier]
↓
[Interpreter]
↓
[Runtime]
↓
[Hardware]

Qu How JVM Works (class loader)

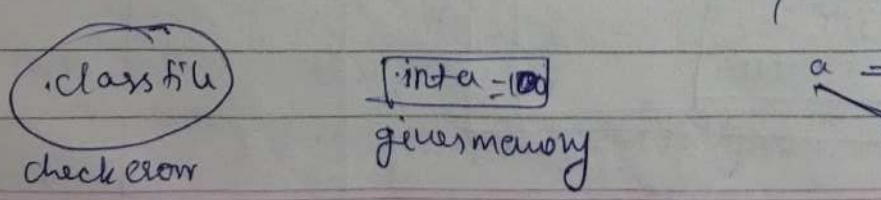
↳ ① Loading
↳ ② Linking
↳ ③ Initialization

① Loading
→ It reads .class file & generates binary data
→ an object of the class is created in heap



② Linking
→ JVM verifies .class file (check if there is any error)
→ it allocates memory for class variables and default value
→ replaces the symbolic reference with direct reference

a = 20 b = 30
sum = a + b
sum = 20 + 30 → this replaces



③ Initialization
all static variables are assigned with their values defined in static block.

⇒ JVM contains stack and heap memory allocation

Static variables → object independent variables
(variables that doesn't depend on objects of a class)

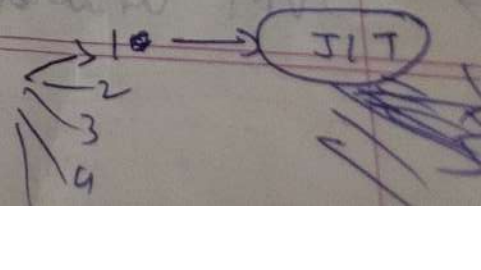
~~5)~~ whenever a new prgm will run stack memory is created

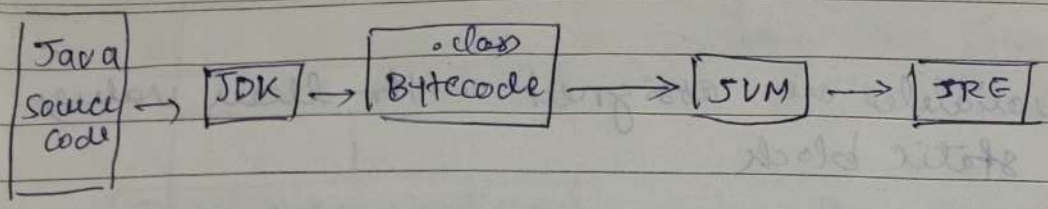
JVM EXECUTION

④ Interpreter :
→ line by line execution
→ when one method is called many times, it will interpret again & again

→ JIT ⇒ JUST IN TIME
→ In interpretion suppose we have a fun that prints hello world. If the function is getting called 10 times then intrepreter will convert byte code into machine code 10 times. This is not efficient so JIT is exist

→ Those methods that are repeated JIT provides direct machine code. So intrepreter is not required 10 times
→ makes execution faster
→ Garbage collection





public class Main
(1)
this can me small as well main. but it is not a good practice.

JDK → Java Development kit.

→ It has Java c that is Java compiler that converts .Java file (java code) to .class file (byte code)

→ It provides environment to run and develop Java program.

→ It has development tools that provides environment to develop program.

JVM

→ JVM is a virtual machine that provides runtime env to execute the byte code.

→ JVM doesn't understands the Java code directly (.Java) so it has Java interpreter that converts byte code to machine codes that are stored in Java.

→ Because of JVM the byte code can run.

→ JVM understands only byte code

JRE - Java Runtime Environment -

→ It has libraries, JVM and other components to run applets and applications. written in java prgm

→ It helps only in running prgm, not in developing the prgm

→ It contains ~~only~~ tools and utelities such as compiler and debugger for developing tools & applns

doesnot

JDK

it is superset of JRE. contains everything that is in JRE, plus it has compiler & debugger that are necessary for developing applications

---