9th Feb DSA

Types of Languages.

Procedural Functional Object-Oriented

- Procedural

① Specifies a series of well structured steps and procedures to compose a program.

② Contains a systematic order of statements, functions and commands to complete a task.

eg:- Java, python, C++

- Functional

① Writing a program only in pure functions i.e. never modify & variables, but only create new ones as an output.

② Used in situations where we have to perform lots of different operations on the same set of data, like ML.

eg:- Python.

- Object-Oriented

① Revolves around objects.

Class is collected group of different data types.
Object is an instance of that class.

② Code + Data = Objects.
③ Developed to make it easier to develop, debug, reuse & maintain software.

eg. Java, Python, C++.

Static vs dynamic Languages.

Static

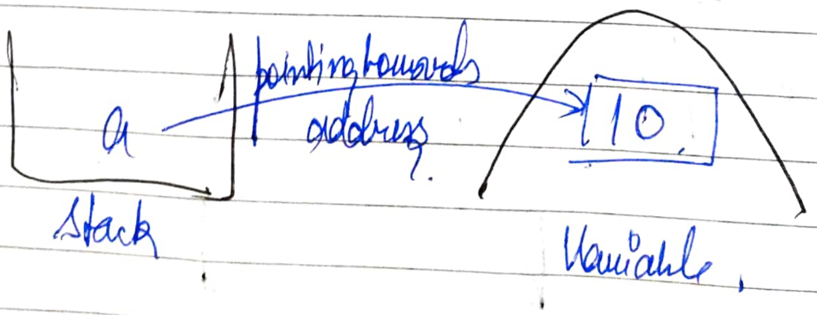
- Perform type checking at compile time
- Errors will show at compile Time
- Declare datatype before you use it.
- More control.

Dynamic

- Perform type checking at runtime
- Errors might not show till program is run
- No need to declare datatype of variables
- Saves time in writing code but might give error at runtime.

Source code $\xrightarrow{\text{compilation}}$ Machine Code. → compile Time.
when this machine code is running → Runtime

Memory Management:



a = 10.
Reference Variable → Object.

- several ref. variables can point towards the same object
- If one ref. variable changes the object then the original object is changed (changed for all its ref variable).

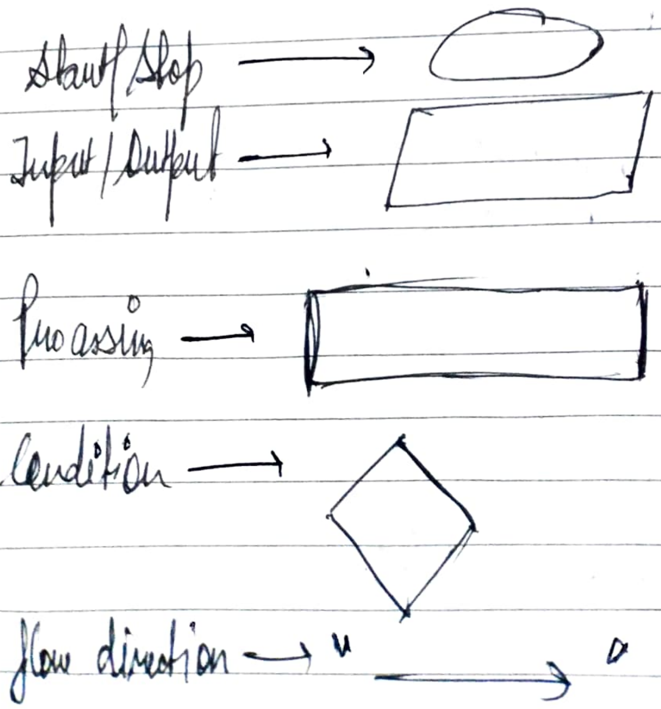
- java only has pass by reference rule. It doesn't follow pass by value.

eg.

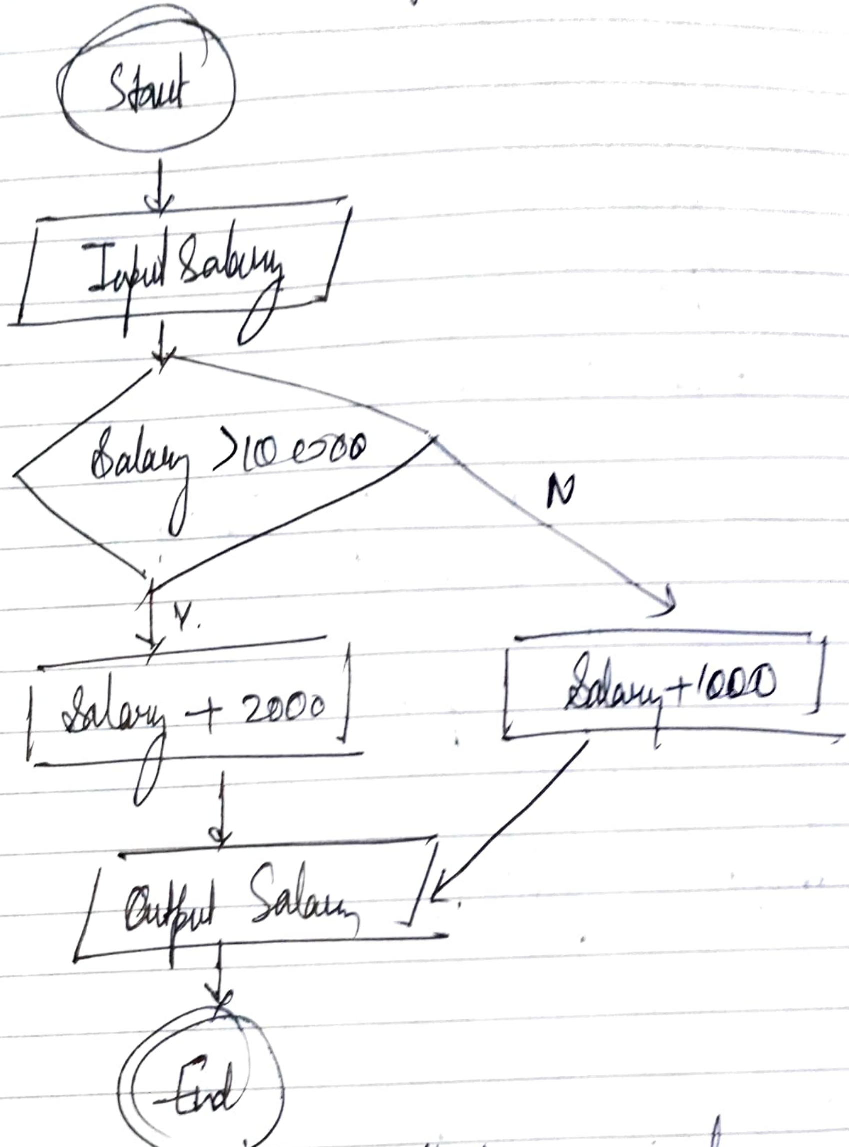
```
a = [1,3,5,9]
b = a
a[0]=99
output (b).

⇒ [99,3,5,9]
```

- An object with no ref variables is deleted by the garbage collection

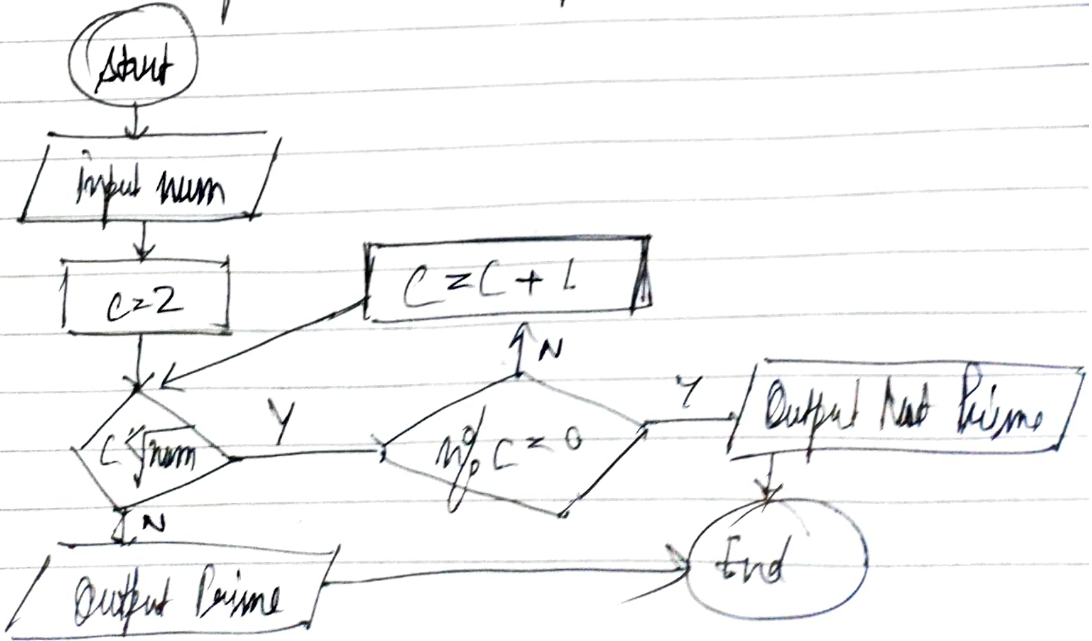
Flowcharts



Q. Take input of a salary. If salary >10000 add bonus of 2000 otherwise add bonus of 1000.



Q. Input a no. & print whether its prime or not.

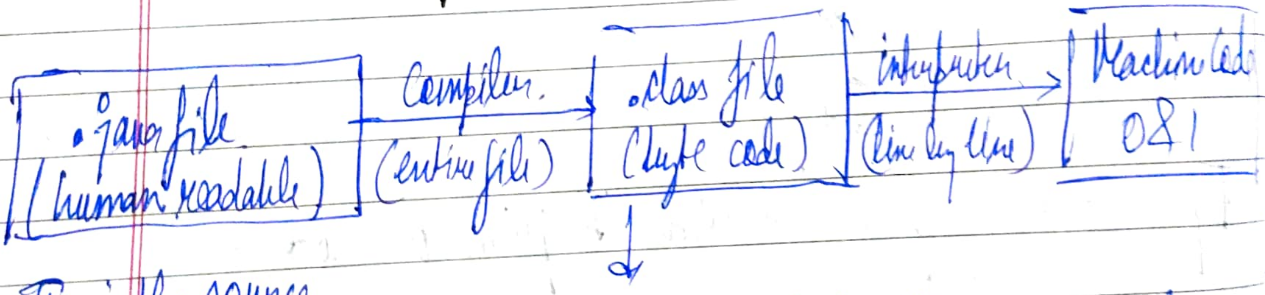


Pseudocode we explain the algorithm of the program without caring about syntax or whatever.

eg. Prime check.

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

Architecture of Java

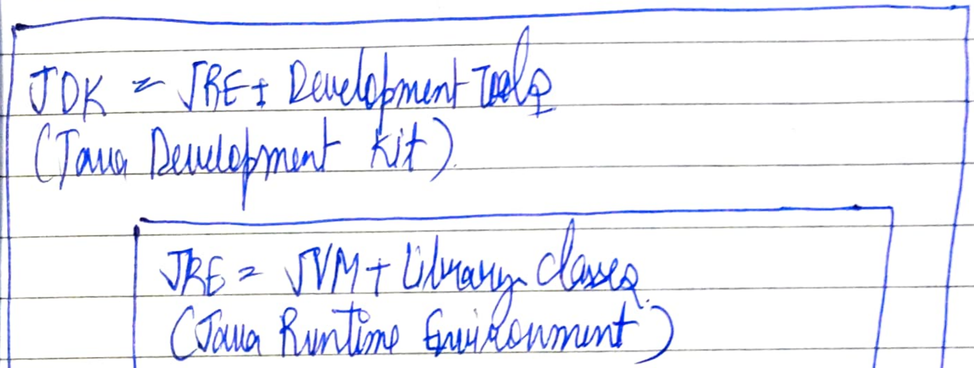


This is the source code.

- this code will not directly run on a system.
- we need JVM to run this
  (Java virtual machine)
- Reason why Java is platform independent

Platform independence

- it means that byte code can run on all operating system
- 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 instructions for the computer.
- 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 = JRE + Development Tools
(Java Development Kit)
JRE = JVM + Library classes
(Java Runtime Environment)
Java Virtual Machine (JVM)
JIT
Just in time.

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

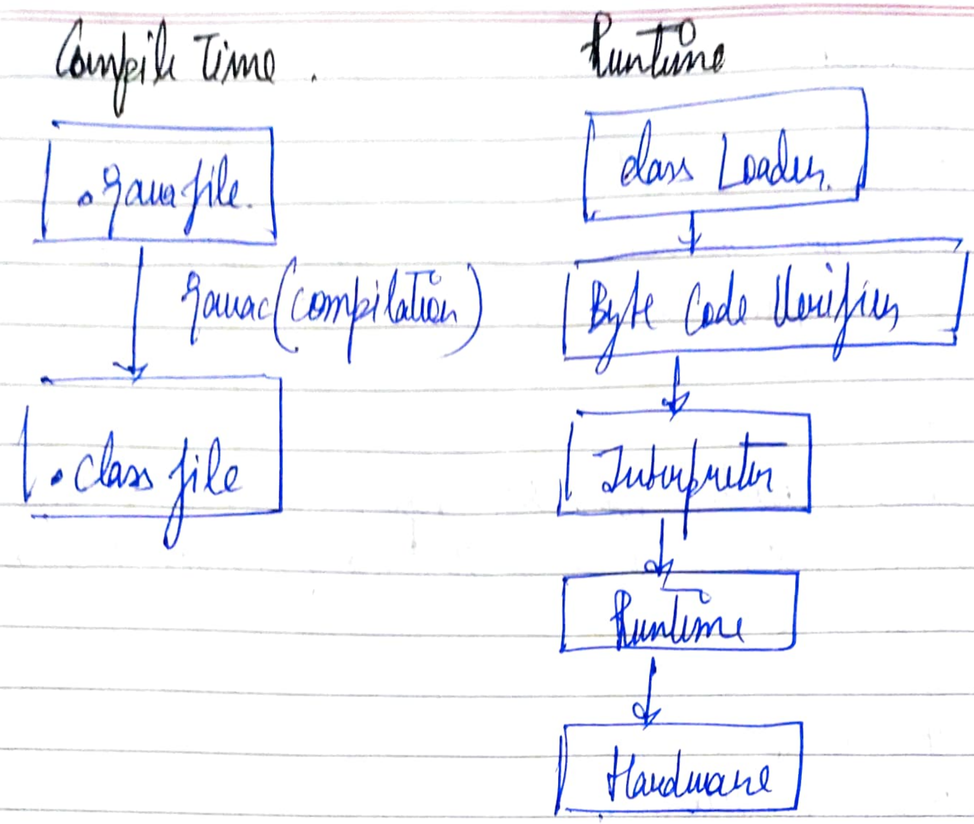
2) JRE - to execute your program
3) A compiler - javac
4) Archiver - jar
5) docs generator - javadoc
6) interpreter / loader.

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 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.



How JVM works (class loader).

- Loading:
  - reads .class file & 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.

JVM Execution -

Interpreter -

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

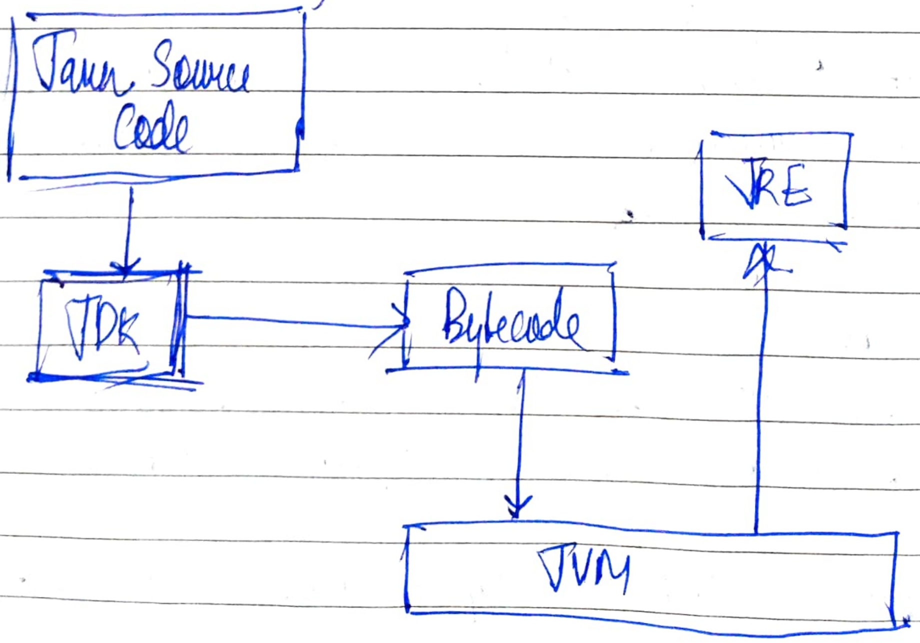
JIT -

- Those methods that are repeated, JIT provides direct machine code so re-interpretation is not required.
- Makes execution faster.
- Garbage collector.

-(Static variables are variables that do not depend on the object of the classes)

eg. Population variable in human classes -

- It doesn't ~~#~~ depend on the object of human class



Installation.

- Download JDK. oracle.com.
- Download an IDE (Eclipse / Intellij Idea).