DSA BOOTCAMP (KUNAL KUSHWAHA)

L-1 Introduction to programming.

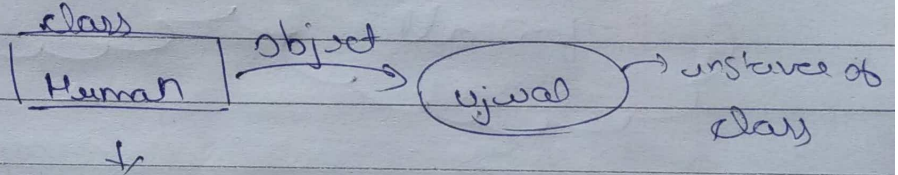
* Types of languages

Procedural
eg: C, C++, java, python

functional
eg: python

Object oriented
→ code + data = object
→ classes: named grp of prop & functions
→ object: instance of class

*



① Static vs Dynamic languages.

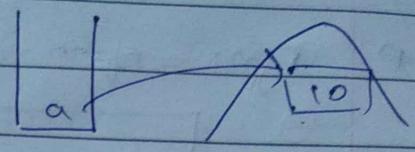
	Static	Dynamic
type check	compile time	run time
error	compile time	run time
datatype →	declare before use more control	No need to declare saves time
→	int a = 10	a = 10
→	int a = 10 ✗ a = "ujwal"	a = 10 ✓ a = "ujwal
→	'a' + 10 ✗	

② Object (non primitive) and reference variables.

a = 10

↓ ↓

ref variable obj



Stack heap

→ more than one variable can point to one (same) object

→ If any one of ref var change obj, original obj is to be changed, changed for all

eg: a = [1, 3, 5, 9]

b = a

→ a ⟶ [1, 3, 5, 9]

b ⟶

L-2 Flow of Programs.

① Flowcharts

Start / Stop :

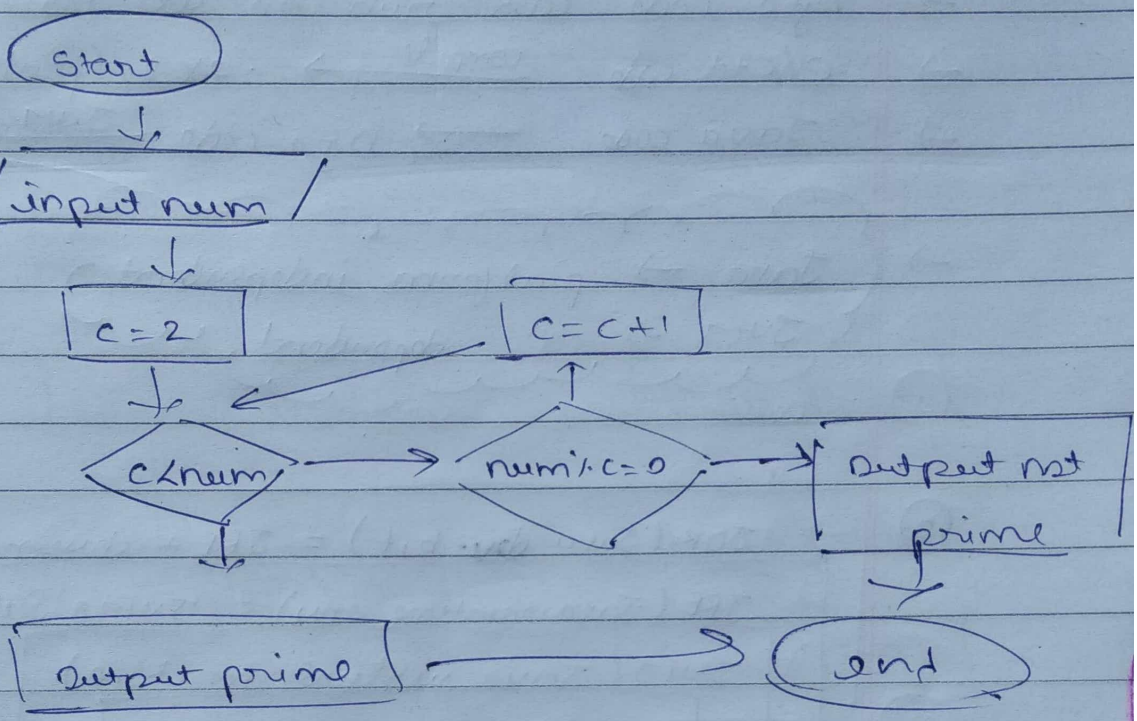
Input / output

Processing :

Condition :

Flow :

eg) Input num and print whether it is prime or not-



L-3 Introduction to Java.

.java file (human readable)	compiler (entire file)	.class file (byte code)	interpreter (line by line)	machine code (0 and 1)

→ this is the source code

→ this code will not directly run on system
→ we need JVM to run this
→ Reason why Java is in dependent.

(1) what is platform independence?

→ byte code can run on all OS

→ C/C++ code —compile→ .exe (platform depen.)

→ Java code → byte code —JVM→ machine code

→ Java → platform independent
JVM → " dependent

(2)

JDK (Java dev. kit) = JRE + development tools

JRE (Java runtime env) = JVM + lib classes

JVM (Java virtual machine)

JIT (Just in time

(3) JDK

→ Provides environment to develop and run java

.java —javac→ .class

→ It is a package that include

↳ development tools

↳ JPE

↳ compiler - javac

↳ archive - jar

↳ docs generator - javadoc

↳ interpreter / loader.

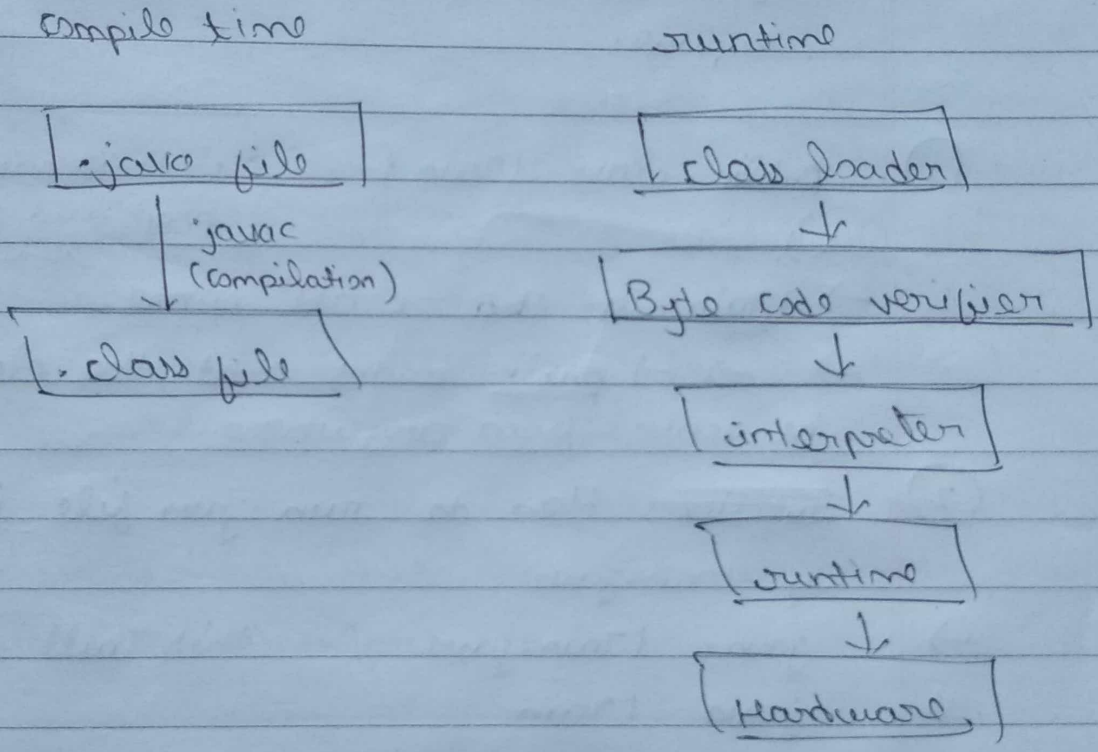
JVM takes help of JRE

(4) JRE

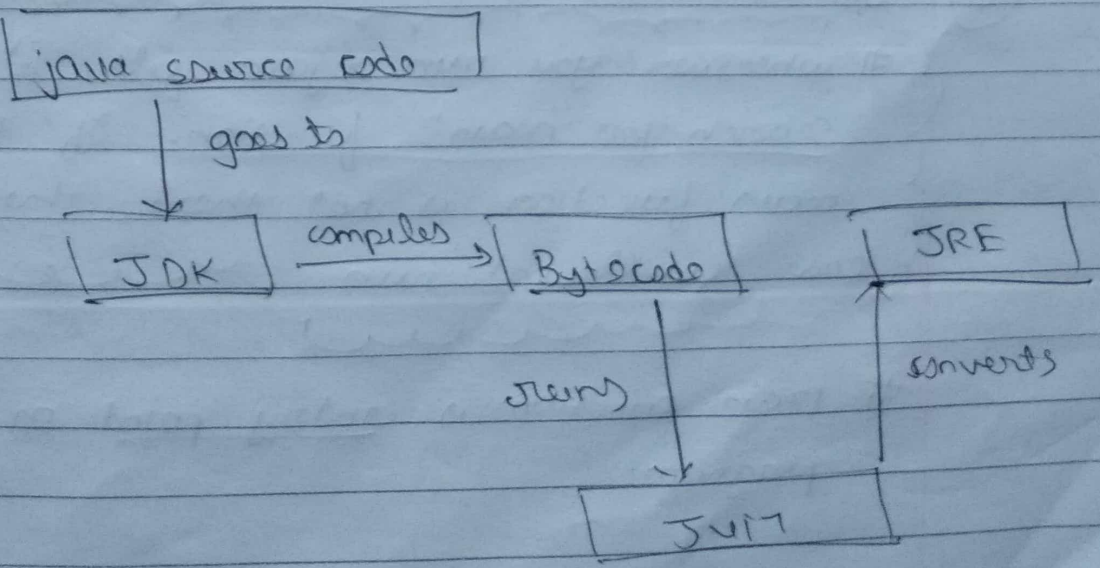
→ installation package

→ it can only run the java program and not create.

(5)



(6)



L-4

First Java program.

Every file that that ends with .java is a class itself.

→ first letter of the class should be capital.

(1) public class Main {
}

class is named grp of properties and fn

→ Main is the class name.

→ → → public means that the class can be accessed from anywhere.

(2) ~~Function~~. How to run java file in cmd p.

eg: Main.java

→ javac Main.java ... this will create bytecode
java Main

(3) Function:

whenever you run a java file, it will search for "main" function. If the main function is not there, then the code will not run

main function is entry point of java program.

Static is used when we want to run the function without creating the object of parent class

Page No. 7
Date

```
public static void main (String [] args) {
    System.out.println ("Hello world");
}
```

(1) (2) (3) (4) (5)

(2) To make main function available everywhere public is used.

(3) Static : main function is a part of "main" class. If we want to run the 'main' function without creating object of 'main' class. (Static variables / function do not depend on the objects)

(4) void : In this we dont want to give value. Therefore void.

(5) String [ ] args → means array of sequence of characters ("string") that are passed to main function

[ ] → array

→ Whatever arguments are given in terminal as an array it will be stored in args.

(4) After compiling, .class file is saved in current location where you are in.
If you want to change the location, use -d (destination) while compiling.

```
javac -d <path> Demo.java
```

(5) echo $PATH:
(every command looks for this location before executing (environment variables))

→ whenever you ~~write~~ / execute a program using command, then the path will be checked.
↳ of env variable

21:25

class Name and file name should not be same. but if we dont want to make class name as file name then it should be not public
eg: class Divide

(6) this means print output on standard o/p stream (here, terminal)

```
System.out.println("Hello");
```

System → class
out → variable (type of printstream)
println → function / method. (takes String)

41:44

available in java.util package.

(7) Scanner input = new Scanner (System.in)

- Scanner → creating input variable of type scanner
- new → creating object
- System.in → takes input from standard input (here, keyboard)

System.out : Command line
System.in : Keyboard

Hey object (new), you are a type of Scanner and you have a value of System.in. When ever input asks for something, take it from keyboard.

(8) Primitive → any data type that cannot be broken further. integer, char, etc

String
↓
Vijwal
↓
V i j w a l
↓ ↓ ↓ ↓ ↓
x x x x x

integer
↓
19
↓
x

int rollno = 64 → 4 bytes

char letter = 'r'

float marks = 98.6f → 4 bytes (without f → double, with f → float)

double large Decimal = 456789.12345 → 8 bytes

long large Integer = 12345678910L → 8 bytes

boolean check = true;

String is written in double quotes while char we write in single quotes

All decimal value that we give are by default of double datatype, ∴ if we want to store in float, we use 'f', same for 'int' & long.

Wrapper class → provides additional functionalities
↳ converts primitive datatype to object.

int a = 10

↓ ↓

identifier literal

→ Literal : ~~Syste~~ syntactic representation of boolean, character, numeric or string data, here 10 is an integer literal.

Type casting → Widening : lower → higher
→ Narrowing : ~~low → high~~ high → low

Page No. 11
Date

→ Identifiers : names of variable, methods, classes, packages & interfaces.

→ int a = 234_000_000 —o/p→ 234000000

→ 564.12345678 —rounds off→ 564.12345

if we give float very big, than it rounds off to the value which gives floating point error.

↗ implicit conversion

(9) Widning or ~~Automatic~~ / Automatic type conversion :/
→ two datatypes are automatically converted.
→ happens when we assign value of smaller datatype to bigger datatype & datatype must be compatible

int i = 100
long l = i 100
float f = l 100.0

bigger → smaller
float → int
19.21 → 19

byte → short → int → long → float → double

(10) Narrowing or explicit conversion.
→ happens when we want to assign a value larger datatype to smaller.

double → float → long → int → short → byte

Eg: double d = 100.04 → 100.04
long l = (long) d → 100
int i = (int) l → 100

(11) Automatic type promotion in Expressions:

→ while evaluating expressions, the intermediate value may exceed the range of operands and hence the expression value will be promoted.

→ Some conditions of type promotion are:

(1) Java automatically promotes each byte, short, char to int

(2) ~~long, float, double~~ → as it is

eg) After solving expressions:

```
(f*b) + (i/c) - (d*s)
  ↓       ↓       ↓
float + int   - double = double
```

converted to biggest one

(12) Explicit type casting in expressions.

→ If we want to store large value into small data type

eg: byte b = 50

b = (byte) (b*2) → type casting int to byte