Note-1

Method signature

Public int sum(int a, int b);

signature

⑧ for-each

n = 17 | i=3 3<5 | i=4 4<4 | i=5 5<=5

i=2 2<9-5

⑨ square root in java

21/9/21

Day 1 >

Concept of programming

① General purpose language →
② class based → applicatn development using class & object
③ object oriented programming language :-
④ WORA → write once Run anywhere
⑤ Java is high level language
⑥ Syntax Orientation only.
⑦ Product language not a Research language

int num = 50;
char ch = 'A';

num: 70 // will not work in Java

this is compulsory in java (statistically typed) :-

WORA :- Project ==> Java

play store → Applicatn

Bank
- Bank project developed by developer → consisting of code by 10 people

② JSL

standadizing java language Specification is job of

JDK 1.0 → january 23 1996
JDK 1.1 →
J2SE 1.2 →
J2SE JDK 1.3 →
J2SE JDK 1.4 →
J2SE 5.0 →
Java SE 8 → → march 18, 2014
9 →
10 → → LTS (long term support)
Java SE 11 → → can be used for long time.
↘ sept 25 2018
⋮
Java SE 16 → march 16 2021

strongly typed :-

Java platforms

Java SE :- ⊗ Java platform standard edition
⊗ also called core Java.

Java EE :- Java platform enterprise edition

Java ME :- Java platform micro edition

Java card : Java based application.

Component of

API : (application program interface)

work with memory

pointers ✗

develop one applicatn by useing C programming

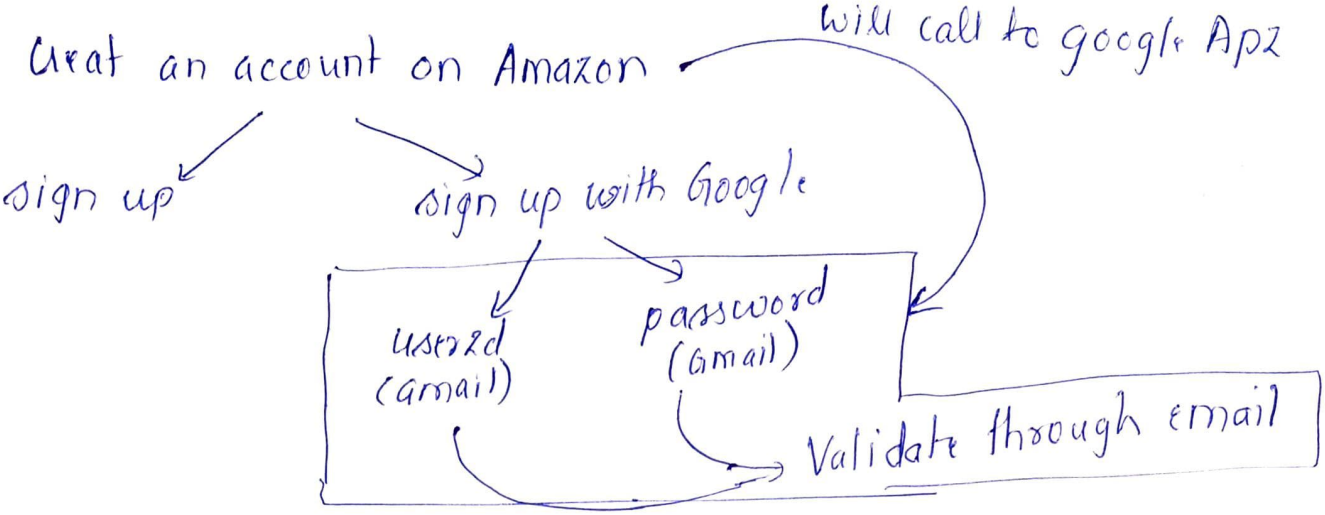
To run on Java

JNI →

Java application

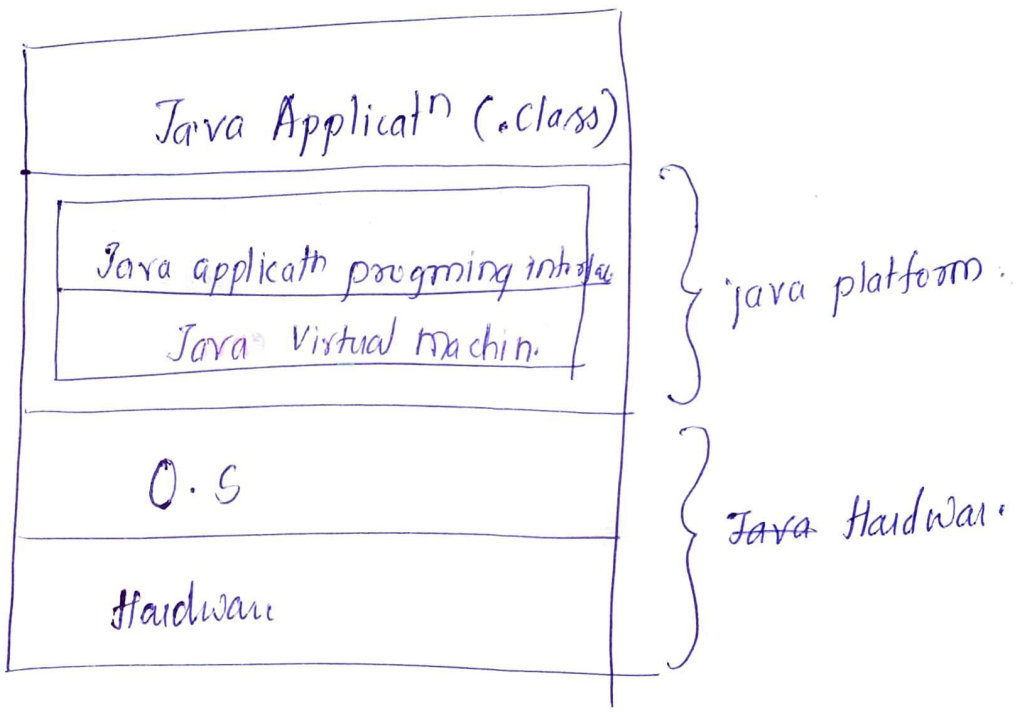
Interface

API :-



* Reuse any other application (facebook, google)

Java platform



JDK ⟶ JVM →

gets installed ← Execution engineer

automatically

gcc → bin
→ stdio
→ string
→ math
→

Java program	C program
.java	.C
.class (byte code)	.exe

⊗ Java is both compiled & interpreted.

SDK, JDK, JRE, JVM ~~interpreter~~

⊗ SDK → Development tool + Documentatn + Library + Runtime environ

- → Comment
- → #includ hear
- → stdlib.h

```
int main (int argc, char *argv[], char *env[])
```

↳ gives environment Infoomatn of system (OS, memory, processor file directory Ram, core)

⊗ JDK → Java Development tool + Java Docs + JRE (rt.jar + JVm)

↳ Java developm kit → Java runtime environment (rt.jar + JVm)

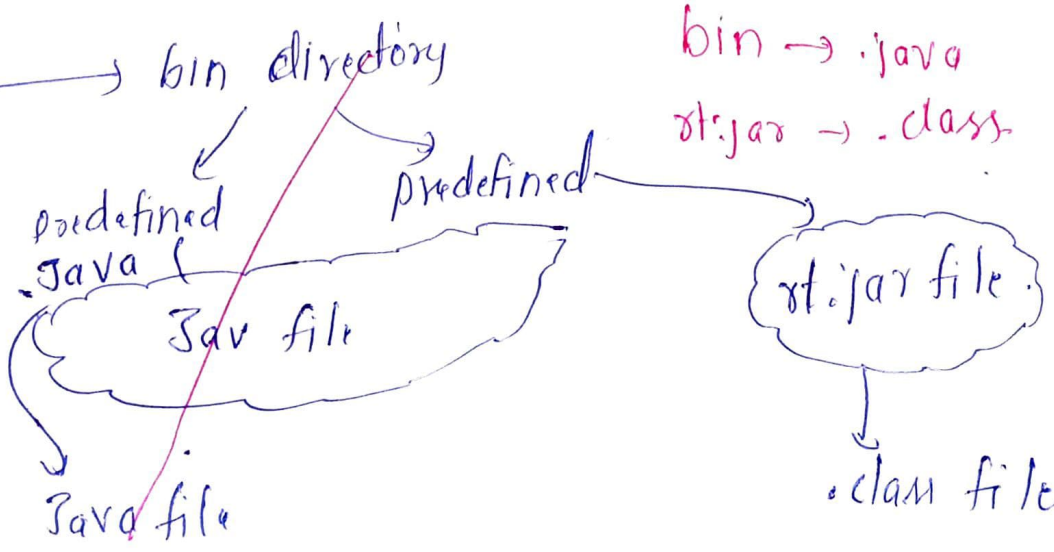
Java gives predefined classes →

Codes (.Java files) → Compiled → Compiled code (.class files.)

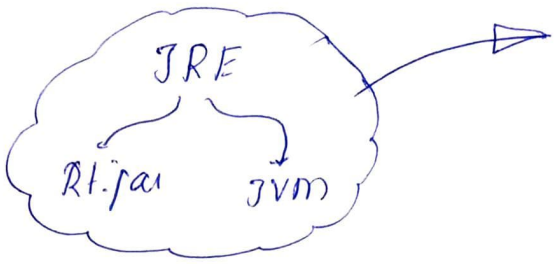
JDk → bin directory

bin → .java

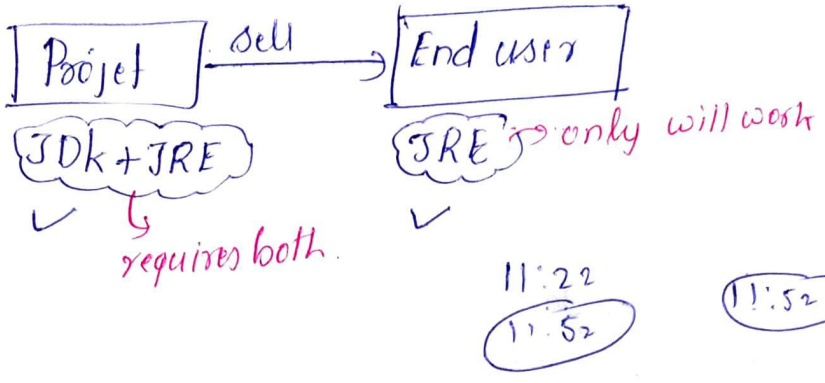
rt.jar → .class.



rt.jar → contains core java Api in Compiled form



JDK → developer
↓
JRE → D+ End user.



11:22
11:52

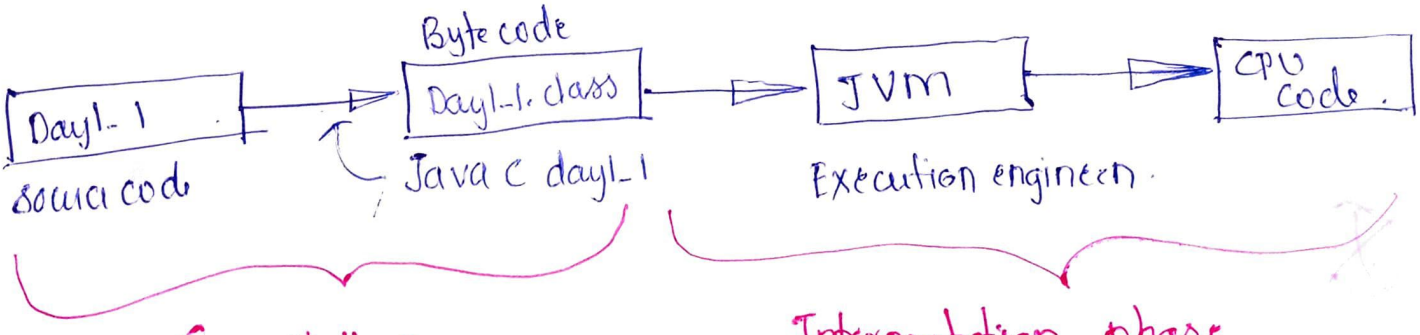
11:52

Java useing VScode

```
class day1_1 {
    Public static void main(String [ ] a){
        system.out.println("Hello world");
    }
}
```

① Compile → Javac day1_1.java
→ java day1_1

Phases of execution

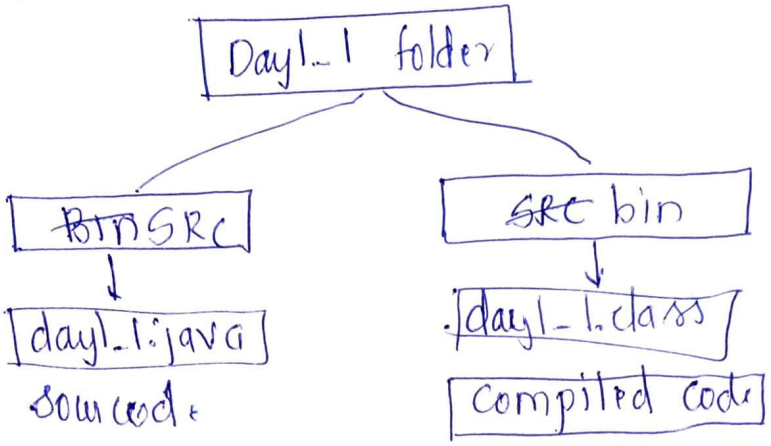


6

Compilation | Interpretation

day1_1
windows
JDK
→ Ubuntu
JDK .

Execution on STS



Package → class

User program

day1_1.java
- SRC

day1_1.class
→ bin

JDK program

String.java
- SRC (JDK)

String.class ← rt.jar ← bin

Compulsory

Public static void main(String[] args) {
}.

entry point function

C/C++ int main(void){ }

Void main( ){ }

no return concept from main it is always void

Why static & public?

Static :— ① OS calls it
② OS is outside class / program /
③ hence declared public static
④ outside means

Compulsory

→ as we have not created object of ~~main~~ class
→ To call without creating object we mak it "Static"
→ In general we dont create object of class where main is present

---

→ String [ ] args :—

int main (int argc, char *argv[ ]) Void main(String[ ]a);

Println :- used to print the data on o/p screen

System. out :-

Javadocs
PrintStream class → println → Return type → void.

it is non-static declaratn hence always called useing class

Println
↓
java.io.printstream (class) → o/p [OStream] out (object) i/p [IStream]

> To call println → out.println( ).

(*) public static Void main( );
↑ because outside class 'OS' will call | ↑ called w/o creating object | ↑ main does not Return anything

System.out.println(" ")
↓ class ↓ object ↓ function.

Why System.out.println("");

System class ⟶ public static void main(String[] args)

hence System.out.println( )

as printstream is void static hence it is skipped from sequence

---

when there are multiple classes containing "main" ⟹ than main should be named same as file name

class name = file name

Wrapper Class :-

- wrapper class provides a way to use primitive data type as object
- some of the collection object stores only objects and as primitive data type are not object hence cannot use ⟹ use wrapper class
- tostring() ⟶ use to convert wrapper class object to string

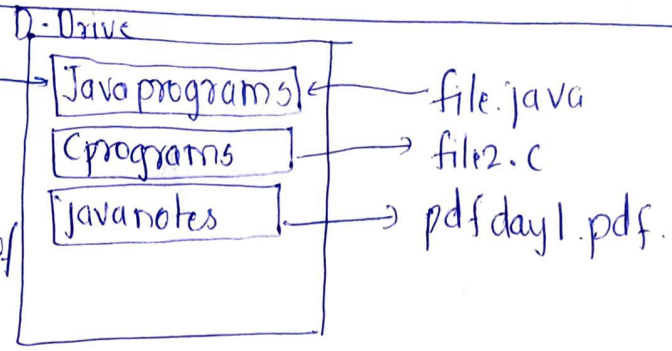
- we generally don't creat object of the class where main is present.
- class name = file name where there is main.
- To call a function w/o making object than make it static
- instance variable :- variables belonging to the class,
- local variable :- variables in block method,

9 Day2
22/9/2021

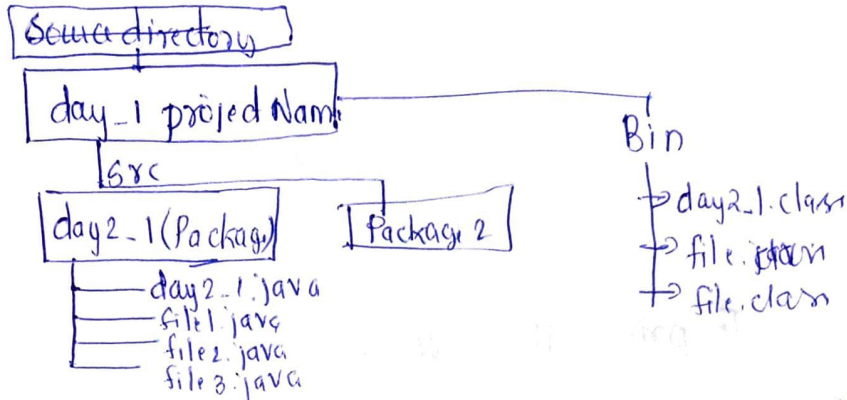
Q day02_1 → hello world
day2_2 →

Doubt → why we cannot create object of main containing class.
⊗ Can we write 2 package in one java file.

Package →
Consist of similar kind of data in one place.



⊗



⊗ every package can have main or every package can represent one project

⊗

⊗ main() overloaded :- main function can be overloaded by changing signature

⊗ ending with 'f' for floating point is mandatory

⊗ 'e' is used to take power of 10 = 2e2 = 200, 3e2 = 300

⊗ for char :- char b = 'c' / char b = 99; → both are equal.

10 day2_2 → ① variable declaratn
day2_3 → ② scanner (taking i/p)

Variables

```
Public static void main() {
    int num;
    syso(num)
}
```
// error

- ⊗ C if we declare any variable it is assigned with garbage value
- ⊗ but in java garbage value is not assigned initialisatn is compulsory

```
int num=5;
syso(num); → // will work.
```

To print Number = 50 → Concatenatn.

```
syso("Number = "+num);
```

To print 50

```
syso(num);
```

To print V1=20 V2=60

```
syso("\n V1: "+V1+" V2 = "+V2);
syso("\n addition = "+(V1+V2);
```

print / Print ln → will take to new line
↳ Print in same line

40%2 → 0 operator overloading.
%d → print.

Scanner :-

To take i/p from user

① command line
② scanner class
③ Console class
④ Data Input Stream
⑤ Buffered Reader class.

⊗ in java initialisation is must* if we are not taking it as i/p

Command line argument

```
int main(int argc, char *argv[3]){

}
```

argc = number

argv : Vector(string)

./a.exe sun pune ⟹

arg[0] [1] [2]

→ String

Command line in 'C'

./a.exe 30 20

string

30+20 (not possible)

→ because of string.

atoi(argv[1])

atoi(argv[2])

↓

used in 'c' to convert to int.

Command line in java

(*)

```
main(){
    int num1, num2;
    num1 = args[0]; //args[0] will hold string
    num2 = args[1];
}
```

error :- type mismatch cannot convert to string to int.

soln :→ string to int. ParseInt()

parseDouble()

```
num1 = parseInt(args[0]);
num2 = parseInt(args[0]);
```

Data types :- (primitive)

int, double, float, char, byte, boolean.

non-primitive (class)

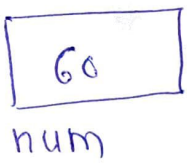
Integer, Double, Float, Char, Byte, Boolean.

Wrapper class / non. primitive

Package → java.lang → class → Integer → Object method → wraps value of int into Object
Short → ParseInt()
Boolean

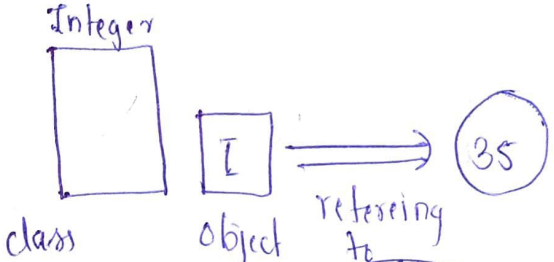
Primitive

int num = 60;



non primitive

Integer a1 = new Integer(35);



⊗ To achive call by reference we use wrapper class

Java.lang → package
↓
Integer → class
↓
parseInt() → function

⊗ To achieve call by reference we use wrapper class.

2_3 Need of wrapper class

→ main {

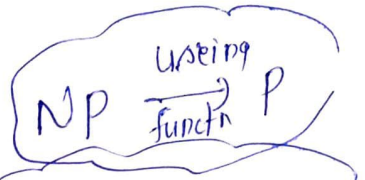
string
int num = args[0]; // primitive ← non primitive.

}

String is non-primitive ☆☆

Soluⁿ:- wrapper class

by useing wrapper class ⇒ NP → P



By ~~useing~~ functions ⇒ NP → P

parseInt( )

NP → P

function of wrapped class

num = parseInt(args[0])

☆☆

⊛ package will always contain class

To call function

→ object.fun( );

→ classname.fun( );

static int parseInt(s); ← string

int

→ Returns primitive

function is static

num = Integer.parseInt(args[0]);

⊛ string class is non-primitive class.

⊛ wrapper class is used to convert primitive to non primitive & vice-versa.

⊛ To call a non-static function we use object where as for static class we use class name.

2_3, 2_4 → printf.

```
main(){
    n1 = Integer.parseInt(args[0]);
    n2 =  "          (  "[1]);
    syso(num1);
    syso(num2);
    syso(num1+num2);
}
```

How to pass argument in Comman line In Java.

Run → Run argumem → program argument | 20 30 |

To print ASCII Value

```
syso('a');
syso((int)'a');  ⟵ print ascii value
```

%10d → width specifier (Printf());

%9.2f → 9-width
2 → after decimal '2' digits.

```
System.out.printf(" X =  ", x);
int num=55;
```

2_4 :- `syso printf("num = %d", x);` →

`syso printf("\n num = %20d", num);` →

`syso printf("\n num = %0.3f", fval);` →

55

.....20..55

→ 55.000

2_4 → final key word, format specifier | 2_5 → scanner class

PI cannot be resolved to Variable

⊗ syso printf("PI = %f", PI);

Java.lang → math → (PI) → field

final = Const

Math.PI.

```
main(){
  syso printf("PI = %f", Math.PI);
}
```

// ss ..... %-10d

2_5 Scanner Class :-

Java.lang Package
Java.util → class Scanner → method → nextByte() ← to read string
Object → super class
int nextInt()
returns int

```
main(){ Scanner sc=new Scanner(System.in)
  int n1, n2;
  n1 = nextInt() → sc.nextInt();
  n2 = nextInt() → sc.nextInt();
  syso(n1+n2);
}
```

class → function.
non-static function
↳ scanner class object.

⊗ To call π = Math.PI

⊗

16 2_6 → narrowing & widening control statements;

```
class Test {
    void f1();
    static void f2();
}
```

2_5 :- parseFloat();

```
main() {
    int i;
    float fval;
    double d;
    i = Integer.parseInt(args[0]);
    fval = float.parseFloat(args[1]);
    d = Double.parseDouble(args[2]);
    sysо(i, fval, d);
}
}
```

Menu driven (switch case) / dowhile

2_6

```
main() {
    Scanner
    int n1 = scann;
    int n2 = scan;
    switch(ch) {
        case +: '+' a+b;
        case -: '-' a-b;
        case *: a*b;
        case /: a/b;
    }
}
```

1 2 3 4

Integer.parseInt() used to get Int out of string.

2_6 → upper case ⇒ ch >= 65 && ch <= 90
lower = ch >= 97 && ch <= 122
ch >= 48 ch <= 57.

Scanner for character : (next();)

2_6 ⇒
```
main() {
    Scanner
    sysso(enter char);
    ch = sc.next();   // Deepak
                      charAT(0) → D
    ch = sc.next().charAT(0);
    if (ch >= 90 && ) → upper
    if (          ) → lower
    if (          ) → Digit.
}
```

2_6 accepting name.

```
{
    str = sc.next();
    sysso("Hello " + str);
    sc.close
}
```

3:-3:30
4-6 → lab-

(*) sc.nextLine() → reads till line change → used to accept string (accept spaces)

Math.abs() → To calculate absolute of number.

fill space ← sc.next().charAT(0) → To scan character (~~accept~~ do not accept spaces)

2_6 → per case ⇒ ch >= 65 && ch <= 90
lower = ch >= 97 && ch <= 122
ch >= 48 ch <= 57.

Scanner for character : (next();)

2_6 ⇒
```
main() {
    Scanner
    sysO(enter char);
    ch = sc.next();   // Deepak
                      charAT(0) → D
    ch = sc.next().charAT(0);
    if ( ch >= 90 && a ) → upper
    if (           ) → lower
    if (           ) → Digit.
}
```

2_6 accepting name.

```
{
    Str = sc.next();
    sysO("Hello " + Str);
    sc.close
}
```

3:-3:30
4-6 → lab-

④ reads till line change
sc.nextLine() → used to accept string (accept spaces)

Math.abs() → To calculate absolute of number.

fill space ← sc.next().charAT(0) → To scan character (~~accept~~ do not accept spaces)

Assignment

①

```
main(){
  int a=0;
  a=sc.nextInt();
  addupto(a);
}
```

```
addupto{
private int a; Private int res;
  public addupto(int a){
    this.a=a;
    result=0;
    for(i=0;i<a;i++){
      a result=i
      (result=result+i)
    }
```

4

1+2+3+4

10 $Sn=\frac{n}{2}(2a+(n-1)\times d)$

```
add up (int a, n)
  if(a=n)
  return(   int(   add(int a, max){
                     if(a==max+1)
                       return
                     else {
                       a=a+1;
                       add(a+1,max)
                     .(a+ add max(   )s)
```

(1+2)+3+4

1

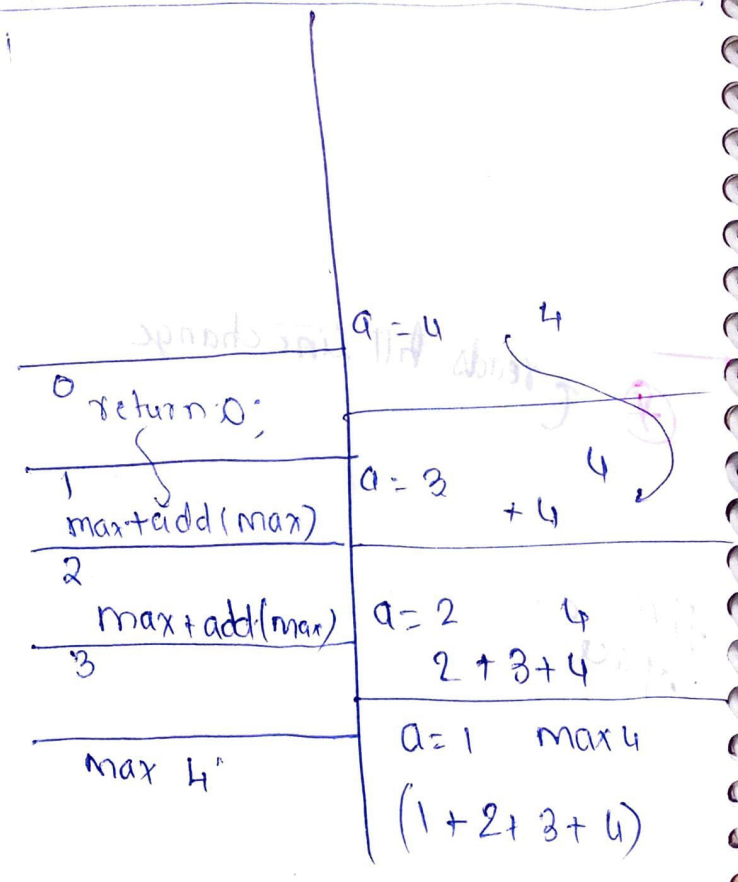
1+2

1+2+3

a=1

```
add(a, 4){
  add(a+1,4)
}
```

```
if(a==max)
  return a
return(a+add(a+1,4)
```



(5)

```
Circle {
private int radius;
private public circle() {
        radius = 0;
    }
public void area(int r){
        syso(area = + (PI*r*r);
Public void circumference(int r){
        syso(circumference = + (2πr)
        }
```

```
circle main() {
  circle c=new circle();
  scanner sc= new scanner()
  int radius = sc.nextInt();
    c.area(radius)
    c. circumference
```

(6)

```
int i;
Public void ifelse(int i) {
  if (i=1) {
      syso("one")
  } else {
      if (i=2)
        syso(two)
      else (
          if(i=3)
            syso(three)
```

```
Public void switch(int a)
  switch (a)
  {
  case 1 : syso(one): break
  case 2
  .
  .
  .
  default :
  }
```

Main

1) if else syso(choose the option 1) ifelse \n 2) switch)

```
int ch = sc.(  )
if (ch==1){
    syso enter the no)
      ifelse ( ):

else
  { enter the no
    switchcase( )

  }
```

Q4) sum first and second last.

```
sumfSL(String st){
    Syso( sum :- (st.getchar(1)+st.getchar(l-2)
}
main( ){
    ① Scanner
    ② String st = sc.next( );
    ③ sumfSL ( st);
        st = a= Integer.parseInt (st).
        for( i=0 ; i<=5; i++){
```

0 1 2 3 4
1 2 3 4 5

int a=
a = a/10
int d = a%10;
(a/10)
(a/10)
a/10)

12.34/10 *2
128/10 3
12/10 1
1

1234
1=4
123
123

1230
100. int = 1.

(12)
10) 123
10
23
20
3
23

⑤ main( ) { scanner;
int

Public void calculate (a, b, c){
switch(c)
{
case / :
- -
- - -
}
}

⑥ ⑦ main ( ) {
int quant = sc.r)
if ((quant*100)<1000)
syso(cost : (quant*100)*0.90)
else
syso (cost :.

982 add digit (int max)

(max+

return max/10

0		0
9	9+0	add(max/10) return (max+
98	9+8	add(max/10) return(
982	9+8+2	add(max/10)

Day 3 23/9/21 Need of primitive & non primitive.

⊗ Command line i/p → stored in string form → unboxing

salary 6000 → "60000" → for increament 1000

"60000" + 10,000; // not allowed

NP P

option 1: NP → P

String → int

Integer.parseInt( )

60,000 + 10000 = 70000

P + P

→ Boxing.

Op-2 premitive to non-primitive:

Narrowing & Widening (Typecasting)

int data; 4bytes

double d; 8bytes

data = d; → type caste

int 4b ← data = d → double 8b

data = int(d);

narrowing

Primitive to primitive → typecasting → To save memory.

d = 12

1100

double (12)

data = 32bits 0000 0000 0000 0000 0000 0000 0000 0000

d = 64bits 0000 0000 0000 1100

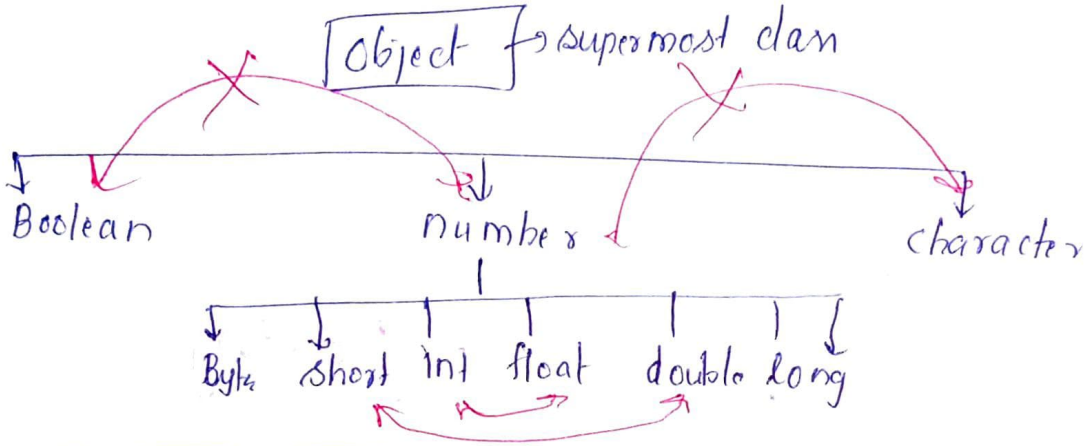
Widening Concept (no need of typecasting)

```
int num;
double d;
```

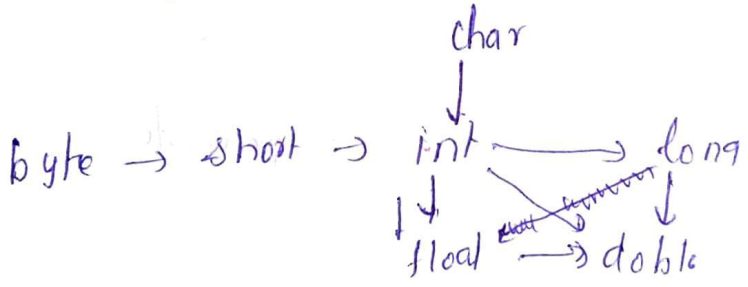
d = num // allowed

d = (double)num // allowed.

** Type casting is done on primitive type only.



float/double/~~char~~int → char



(ctrl + +) → to increase font size.


```
main() {
    int num=50;
    double n1;
    n1 = num; // int to double widening.
    sysO(n1, num) = // explicit typecasting is optional
```

o/p → 50, 50.

```
    n1 = (double)num;
```

o/p → 50.0 50

↳ widening
// explicit type casting.

```
main() {
    double num1 = 60;
    int Val;
    Val = num1; // double to int
    sysO(num1, val);
    Val = (int) num, //narrowing
```

o/p ⇒ compile time error

o/p 60.0, 60.

```
main() {
    int num;
    float fval,
    double d;
    num = Integer.parseInt(args[0])
    fval = (args[1])
    d = (args[2])
    sysO(num, fval, d);
    int result = num + fval + d; // type mismatch error
    int res = int (num + fval + d)
    sysO(results, res)
```

o/p → 50, 20.5, 30.7

o/p 101.2 101.

change to double (1)

int

double

(2) option

Day3_2

Sizeof (i+f+d) = 8 byte.

Primitive Type Conversion
↓
type casting
↓
narrowing & widening

Interview
What is primitive
data type with specific size is called primitive

---

non-primitive type Conversion :-

Day3_1

Boxing & unboxing

~~main() {~~

class.

java.lang → String → Constructor

np → p → unboxing;
p → np → boxing;

main() {

anything in (" ") is string.

String str = "55";

is name of class: ∴ np.

int num = 35; //primitive)

int ~~var~~ result = str + num; // cannot convert string → Int.

result = Integer.parseInt(str) + num; // ok

Syso(Result) ⟶ o/p = 90.

copied

String str = args[0]; // command line.

int result = str + num; // not allowed

↓

Int.parse(str) **

Non-p → primitive →

with wrapper class with parseInt( )

ParseInt( ) → Returns Int only if "number" is passed or else it will produce error

non primitive → it is np because its size is not fixed

Int → method → Int Value

main( ){ String to Int → ParseInt( );

int / Integer — Int — Integer

p — np — return type of parseInt( )

Int to Integer Conversion

Int ⇒ Integer (boxing)

⇐ (unboxing)

ParseOf

String → Int. Integer.parse.

→ float — float.parsfloat( );

→ Double — Double.parsefloat( );

Day3_3

```
main() {
    Integer i = new Integer("55"); //np
        ↳ i is object of integer class
    int num = 5; //p
    int Result; //p        np to p => unboxing
    Result = num + i; //
```

IntValue() → Return 6 ~~Integer~~ int.

```
    Result = num + i.intValue();
    Syso(Result);
}
```

o/p :- 120

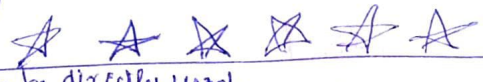
- Integer —— int
- Float —— float
- Double —— double
- Long —— long

intValue.

String → Int → unboxing

~~Integer → Integer → Value~~

Integer → Int → intValue



Day3_3

ToString → toString() → nonstatic Return string

↓ toString(int i) → static type " "

int/Integer → string

(can be directly used)

ValueOf() → string.valueof() → int to string

Integer.valueof() → int to Integer

Day3_3

Boxing (To String)

Day3_3

(X) To stri

```
main(){ int num=50;
        String str=Integer.tostring(num)
        String str=num; //error  Solution to this problem.
```

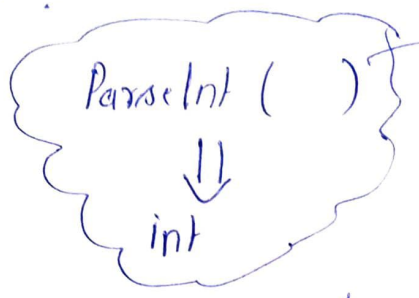
```
main(){
    String s1=s
    Int num1=40; //p
    float fval=50.55 //p
    double d=60.5; //p
    String s1=num1 // String.valueof(num1); // p → np
    string s2=fval // String.valueof(fval); // p → np
    string s3=d // String.valueof(d); //
    Integer i;=new Integer(300); //np.
    Float f1= " " (40.52; //np
    Double d1= " " (80.8); //np
    String st1=1.tostring(); //np
    String st2=f1. " ; //np
    String st3=d1. " (); //np
```

Valueof → ~~called on string class~~ return string (p → string)
tostring → called on string class. (p → string)

Parseint → string → p (string to int) (np → p)
Valueof → Integer → string (int → string) (p → np)
tostring → np → np (Integer to string) (np-np)
Intval → Integer → int (np → p)

OOP

abstractions :-



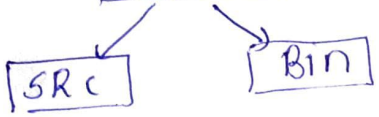
→ all internal details are hidden.

⊗ Implementation details is not available

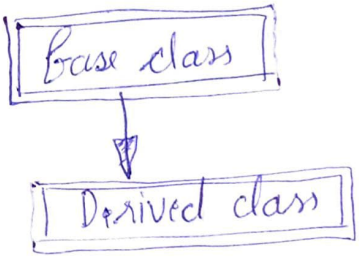
Implementation ≈ encapsulation :- (binding).

Implementatn of abstraction ⇒ encapsulation

Day 3-4 = modularity.



Inheritance ⟶ Reusability :

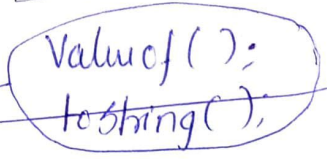


minor pillars

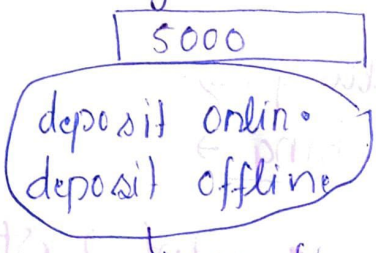
① polymorphism

⊗ Fuction overloading

Concurrancy :- & persistence



Banking data



↳ same time

↳ one operation is hold and other is continued and after complet one only deposit will take plan.

⊗

⊗ state of variable is maintained all time, one day, month years.

Day4_1

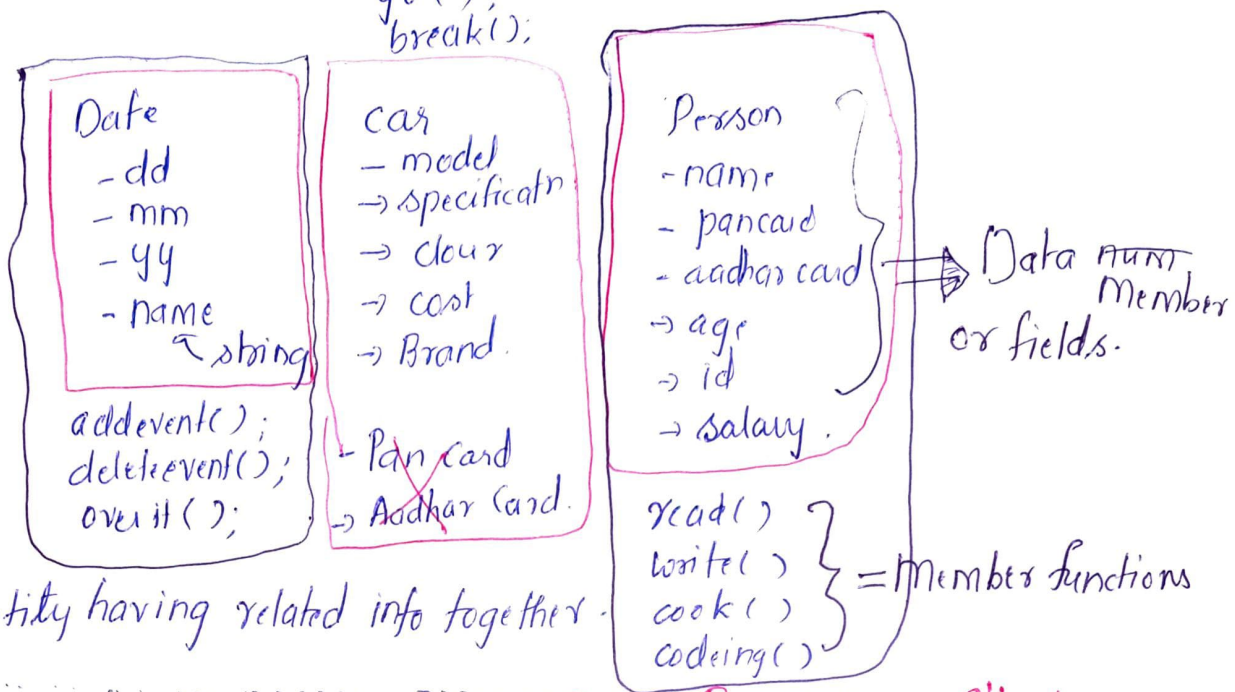
Architectural Nuetral

@ as byte is interpreted on all the machines, hence it is architecture Nuetral.

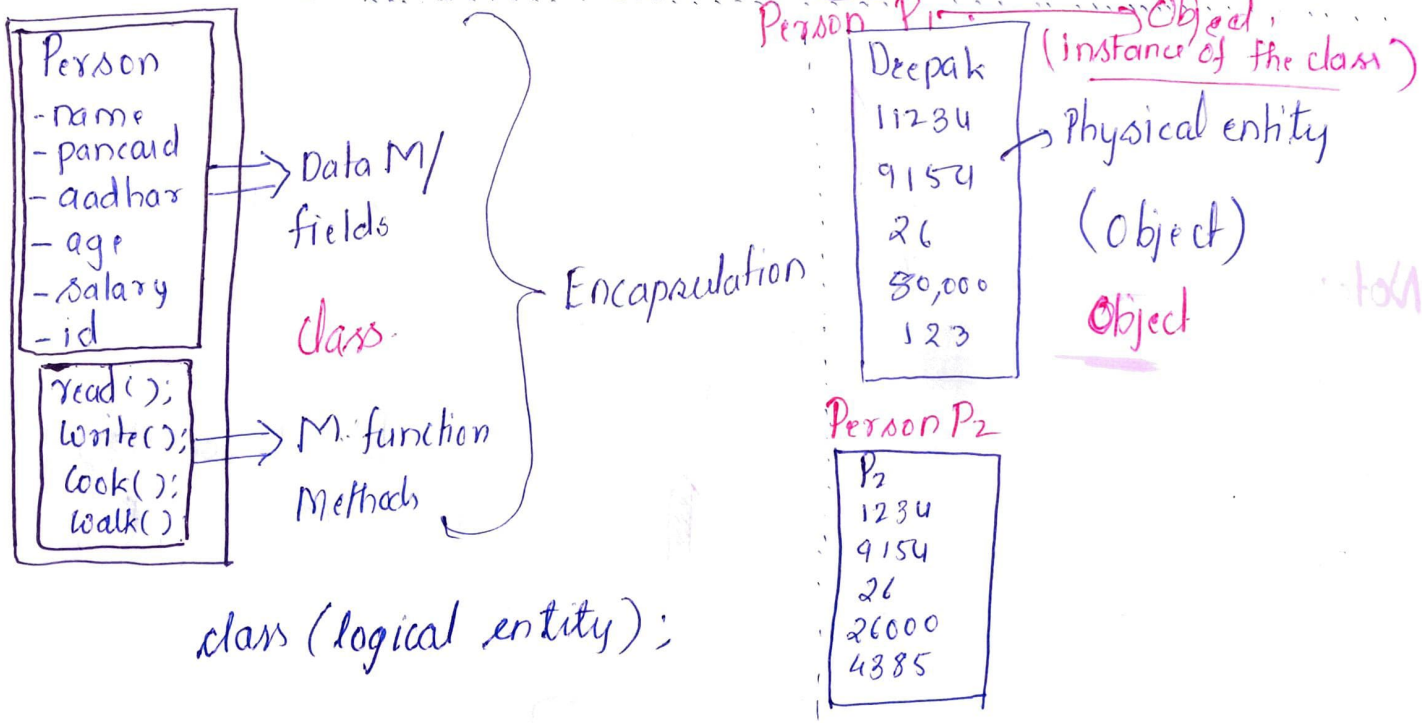
Day-4:-
24/9/21

Day4_1

run();
stop();
go();
break();



class entity having related info together.



String to Int/Float/
ParseInt();

Integer to int
intValue();

Int/Float/ → String.
toString(), toString(int i/float),
String.valueof(int)

30 Day 4-1 → class.

✱ When we create instance of the class than only Data members will get memory, Data function don't get any m

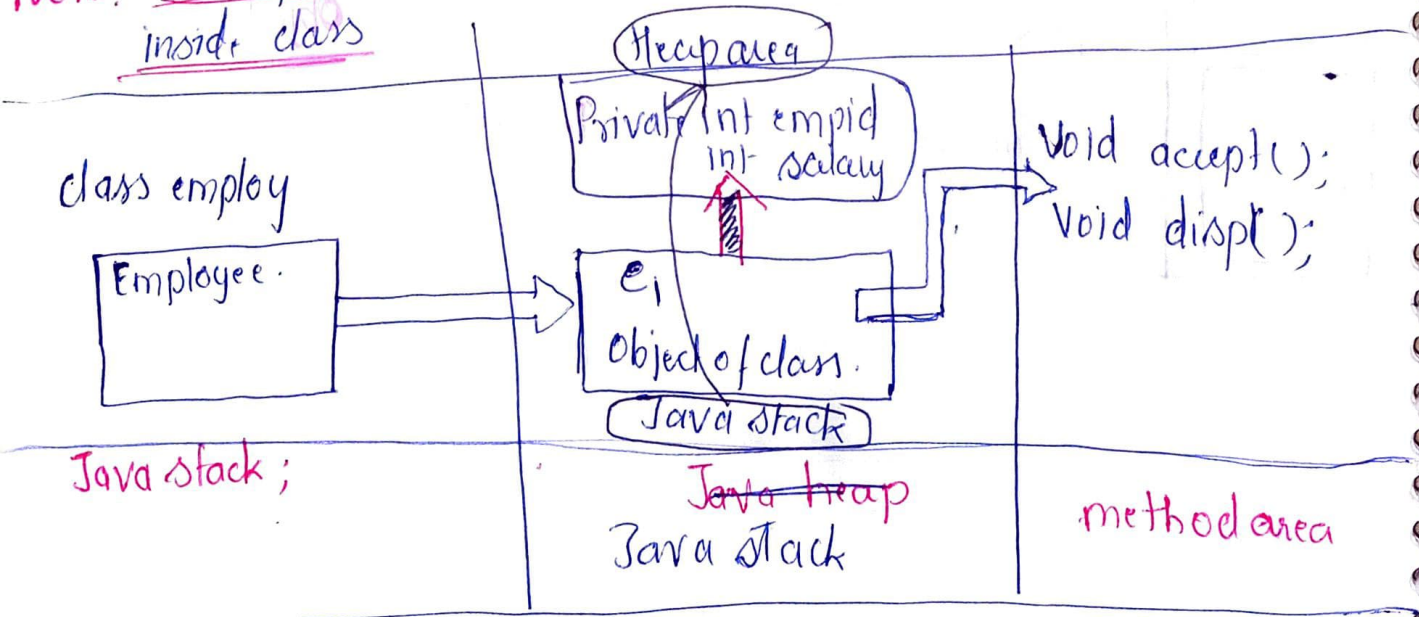
→ class Employ {

```
Private int empid;
Private in salary;
Public void accept() {
        sysO(enter empid);
    }   empid = sc.next();
        sysO(enter salary);
        Salary = sc.next();
}
Public void display() {
        syso (empid, salary);
}
}
```

Private:

Public:

Note:- No scope resolutn operator hence compulsory to define function inside class



✱ In java declaratn & definatn in class is compulsory
✱ when object is created only data members get memory

Syntax of object creation -

```
Employee e = new Employ();
```

new → dynamic Memory allocation

Employ e1; → ~~object~~ reference

e1 = new employ(); → ~~reference~~ object

Reference Variable

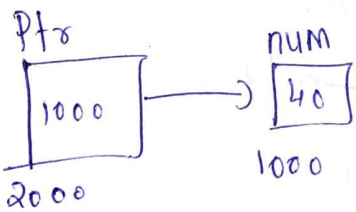
```
Employ e1 = new Employee();
sysо (e1);
int num;
num = 50;
```

o/p → sunbeam.Employee@d761517

address of object

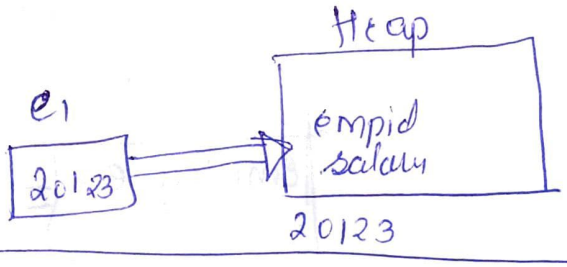
Package · class · # code.

getclass().getName() + "@" + Integer.to Hexstrin (hashcode)



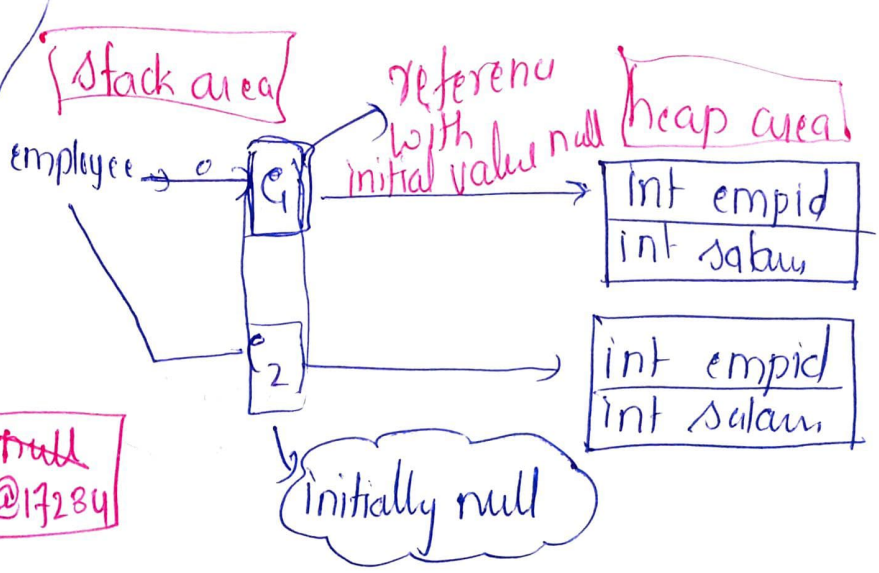
int * ptr

ptr = ~~num~~ & num;



Hash table

	Hashcode
e1	@72315
e2	@20123



Method area

Void accept()

Void disp()

~~null~~ @17284

11:23 recorded session

e1.disp()

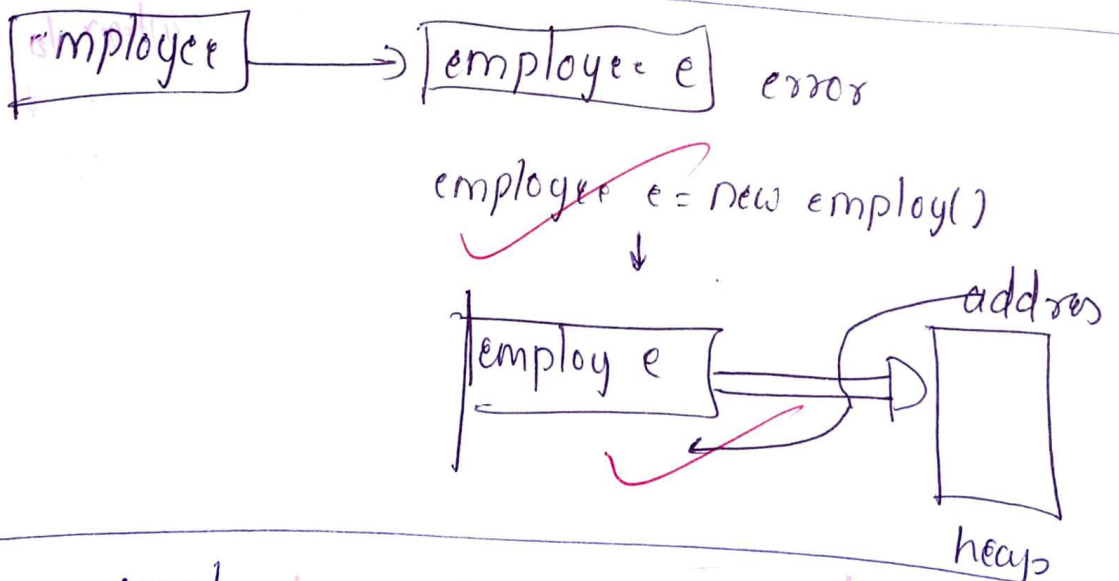
11:18

e1 . disp()

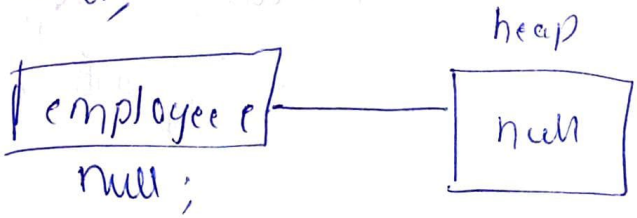
@52738 . disp(); → system.out( )
syso(empid),
syso(salary);

null object :-
employee e=null; → It does not point to any memory

We cannot only reference i.e declare new or null
employee e= new employ();



employee e=null;



Valid

To make null reference :- employee e= null;
employ e :→ reference
e=new employ(); → assigning value

class Bank

Bank account
* customer id
* account num
* balance
* name
* dob
* id
* pan
* aadha

To string function

```
class employee (){
        Private int id;
        Private int salary;
        Private String name;
                char grade;
                int number;

        disp( ){
                syso(id)
                syso(salary)
                syso(name)
                syso (grade)
                syso (no)
                }
        Accept () {
                sc. (id)
                sc. (salary)
                sc. name)
                sc. grade
                sc. no
                }

@override.
Public String toString()
 {
   return "employee".
```

```
main () {
        employee e1 = new employee();
        e1. accept( )
        syso( info of e1)
        }
```

@7212

after to string

[name, salary]

34

object

syso(e1)

S.OP(e1)

↓

no tostring

↓

hashcode

syso(e1)

↓

with tostring

↓

will display function defination data.

---

① toString → display hashcode

② toString → as per user requirement.

To Display :→ disp()

syso(e1) with to string

⊗ syso(e1) gives internally call to toString

Employee →

id ;

salary ;

name ;

grade

number

Display Complete

Void display()

user defined

Partial Information (as per end user need)

Override toString function

id

name

Partial info

id

salary

name

One more time toString ✗

not allowed.

Fetch indivisual data from object

Getters (M.F of class to fetch indivisual element)

get id, get salary get name.

↓ ↓ ↓

sysol() syso syso.

Tostring :- used to fetch details of an object. [@17323

→ when overidden it gives user required details [name : DOB :]

Setter / mutators (to modify data members individually)

```
class employee{
    ....
    int getID(){
        return id;
    }
    int getno(){
        return no;
    }
```

```
main(){
    employee e1=new Emp
    e1.accept()
    sysO("ID = "+e1.getID
    +" "+e1.getno);
```

---

this :- stores address of current object

array of object

```
class name variable[] = new class name[10];
Class A
student s[] = student[10];
```

Copy Constructor :-

```
student s1 = new student(8, deep);
student s2 = new student(s1);
```

Hash map

```
HashMap happy = new hashMap();
happy.put("a", 10);
happy.put("b", 3);
```

key, value

```
HashMap<String, Integer> happy = new Hashmap();
```

27/9/21 Day 5

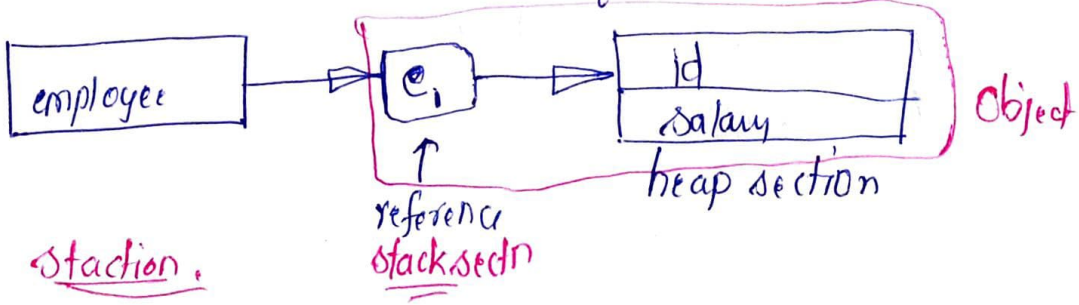
Day5_1 :-

```
class employ {
    private int id;   // all are accessible only by MF
    private int salary; //  ''   ''   ''   ''   ''
}
```

getter(id); ——→ setter/Getter : / Inspectors / mutators.
(sal);

```
void accept(){ ——→ facilitators : user defined.
    syso(enter ID)
    sc.nextInt()
    syso(enter salary){
      sc.nextint();
}
void disp(){
    syso(ID);
    syso(salary);
}
```

```
main(){
    Employee e1,e2; ——→ reference variable.
    e1 = new employee();
    e2 = new employee();
    e1.accept(); //accept is called upon e1
                 // address of e1 is stored in this.
```



Method area

Void Accept( );

this

main( ){

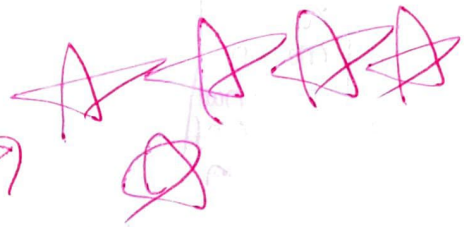
Employee e1 = new employee( );

Initialize(40, 70000); → user defined function.
→ we are initialising object.

Void init (int id, int salary){

not allowed ✗ e1.id = id; // this.id = id;
✗ e.salary = salary; // this.id = id;

e1 is not accessible in employee class.



Constructor

→ used for initializing the object.

Explicit calling :- If we call any function with object name.
eg init / disp / accept

⊗ whenever we create object, object must get initialize.
⊗ we create constructor

Day 5_2 :- class → ~~compto~~ student
int rollno;
int id;
String name;
float per;
Void disp(){
sysо(student data)
sysо(complete record)

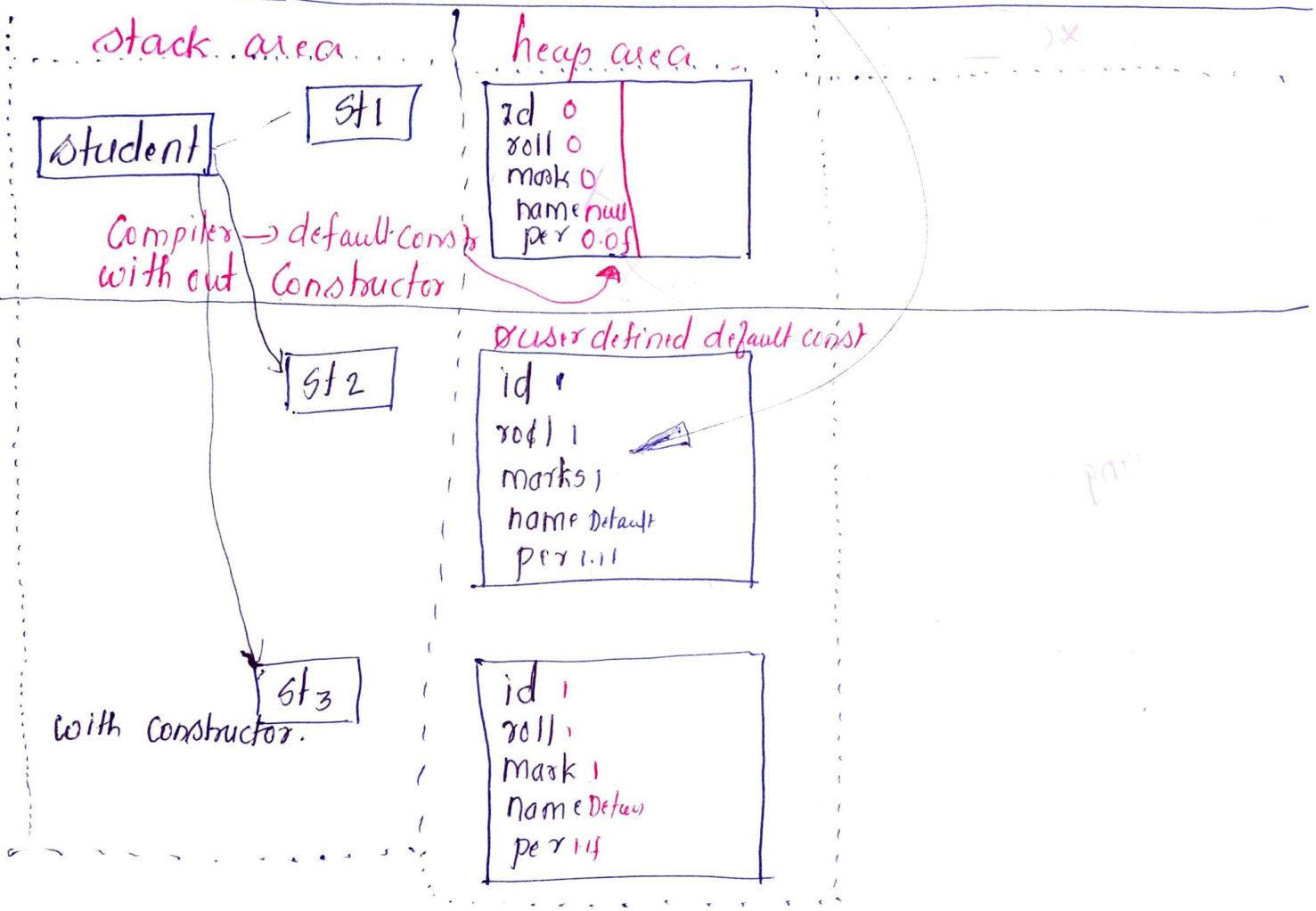
main( ){ student st = new ;
st.disp ( )explicit call;

Constructor Name :- same as that of class

* special mf →

⊗ it does not have any return type.

```
student(){ ——→ parameterless constructor.
    this.id = 1
    this.roll = 1          // user defined default constructor
    this.mark = 1
        name = Default;
    this.
```



Parameterized Constructor :-

this(1, 1, 1, "Default", 1.1f); ≈ this.id = 1
this.Roll = 1
this.~~mark~~ Name = "Default"
this.percent = 1.1f;

id ← Roll ← mark ← Name ← percent.

Calling one constructor from another constructor is called Constructor chainning

programmer effort are reduced.

this(1, 1, 1, "Default", 1.1f);

Sti addr

~~will give value to~~ will call already existing parameter

id
roll
mark
Name
perc.

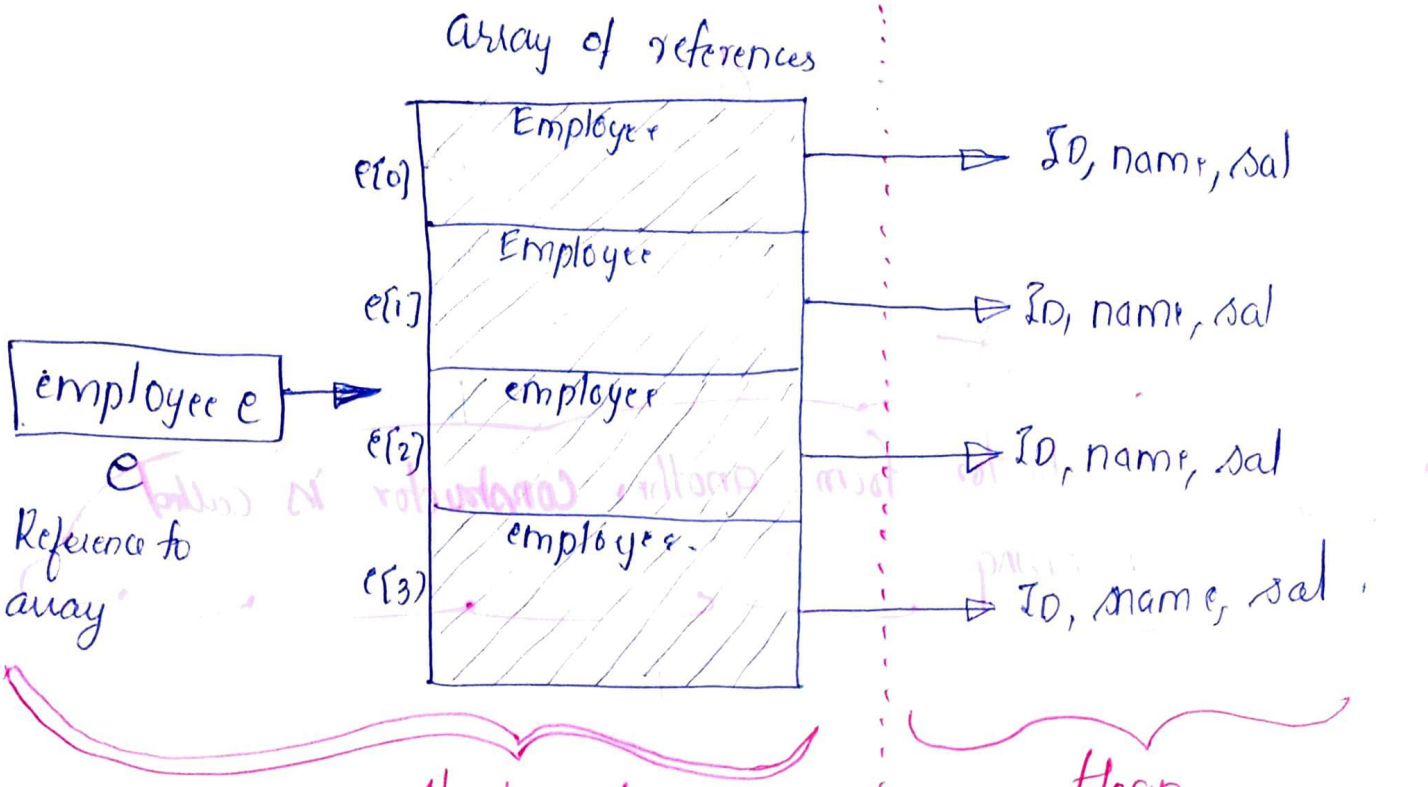
array of Object

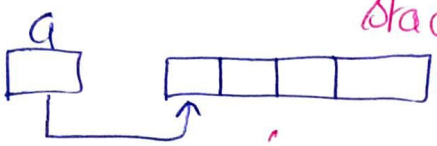
```
class employee {
        id, salary, age;
    }
employ(9, 5600
this(9, 56000, "IT");
}
```

five employ records to be maintain;

Employee e[]; int a[4];

e[] = new employee[4]





employee e[4) = new employ[4)

~~e[0] = new employ[4)~~

for(i=0; i<4; i++)

e[i] = new employ();

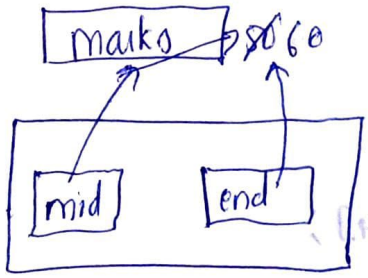
employee e[] = new employ[4];

Day 5_6 Static field & methods

static : initialised only once:

(X) ~~gets~~ ~~reatted~~ retains last updated value.

Static variabl.



① Static function

② static fields.

③

student, teacher, principle.

```
class demo{
    static void sum(a b):
        {
    static void sum (a,b,c){
class main {}
    main ( ) {
      void sum X
      class demo d:
      d.sum ( )
      Demo.sum ( )
```

→ creation of object not required.

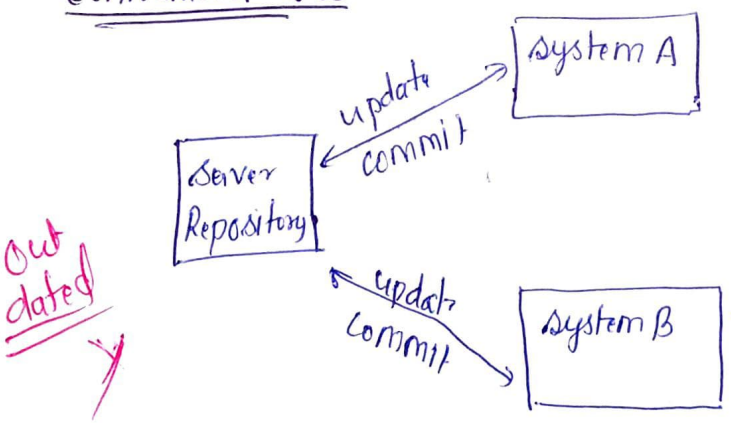
→ static way to call static M.F

Introduction to Git.

Version Control system.

- Tool management of documents / source code
- also called as Revision control system.
- source code control system.

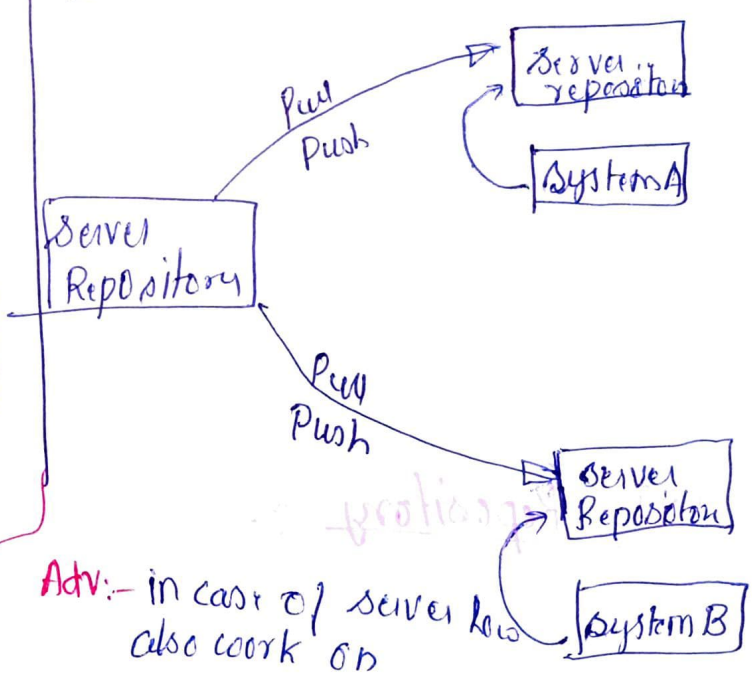
Centralised VCS



Adv :- all codes in control system

dis :- what if server is low.

Distributed VCS



Adv :- in case of server low also work on

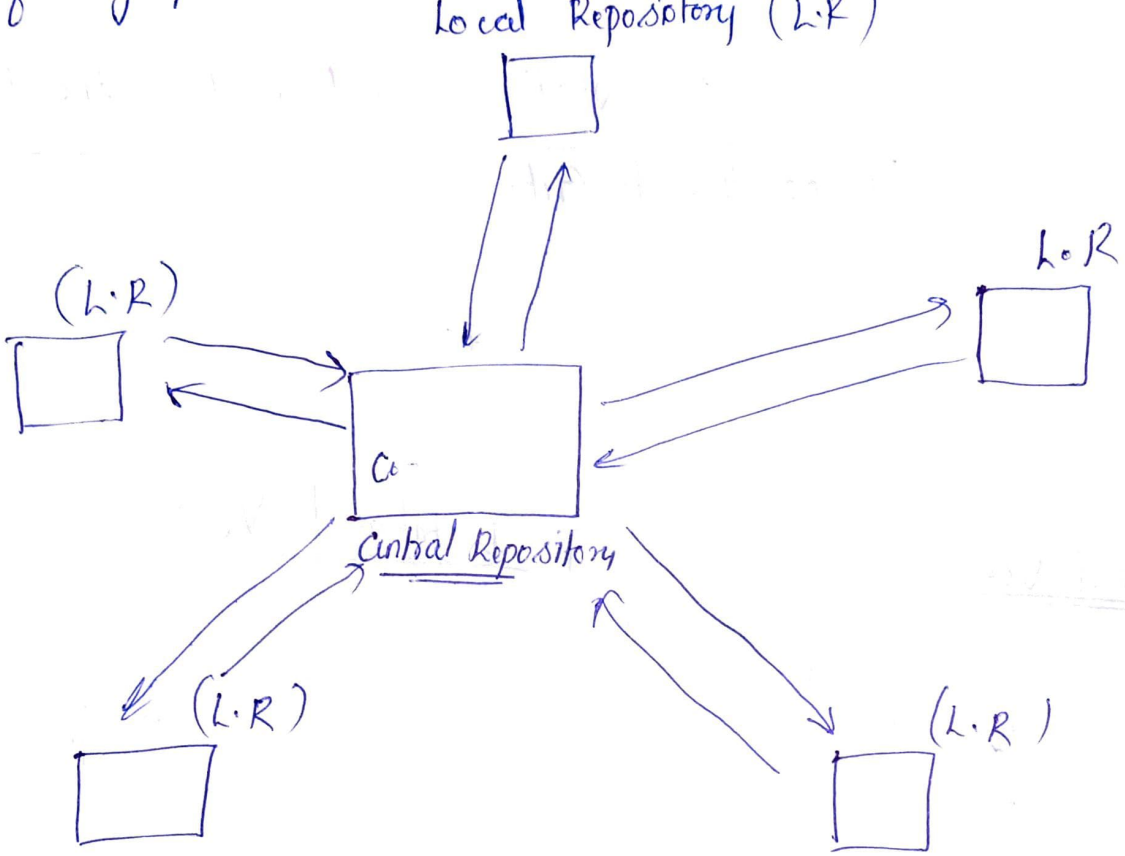
Repository :- where we keep project Data (alias folder)

GIT

- free
- open source

} becomes popular in 2T.

- Developed by Linus Torvalds to mange Linux kernel source code.
- Began to code in 3-Apr-2005 & on 6th day published and finally published on 16 Jun



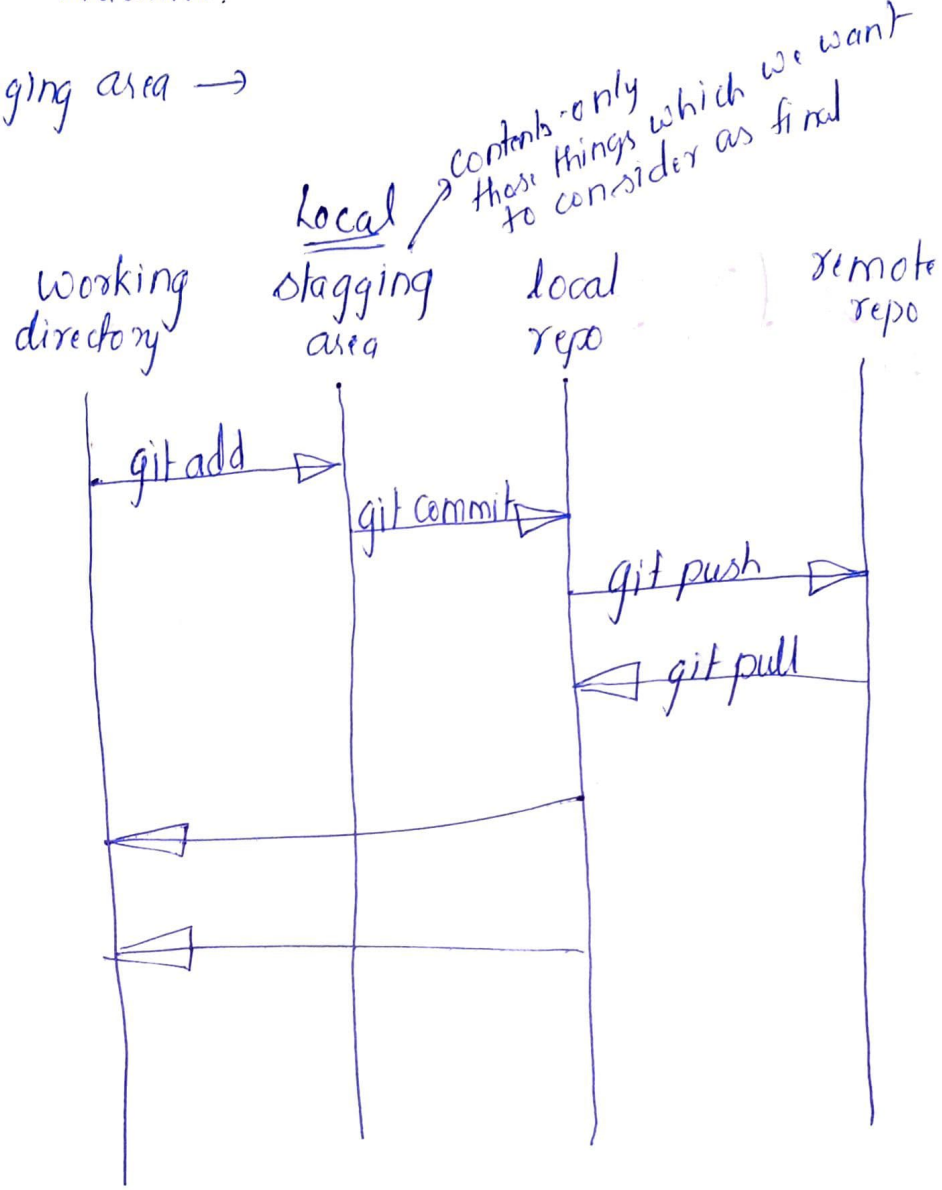
- Free under "GPL" (General Public Lince)

GIT Repository :- any directory structure with code & ~~metadata~~ its metadata (.git).

- .c/.h/.java files
- developer, time, id, Location difference from prev version

Working: working area → directory, subdirectory, visible on machine.

Staging area →



⊗ Folder → Git Bash here → `git init` (1 .git folder created.

Vscode → f1.txt f1 putin folder → `git status`

* git add f1.txt → (staging area) → git commit -m "first commit"

⊗ git status.

when to commit → one logical change → always complete one and commit next seperately.

⊛ devide functionally

* first commit.

git config --global user.name "Deep"

user.email "ddeepak2294"

• gitignore → special file (notepad / vscode)

* *.jar
* *.class

To know change

git diff

• gitignore (vscode)

* *.exe
* *.obj
* *.class
* ~~*.bin~~
  bin/

To add directory

git add . → all moved to staging

git commit -m "gitignore added".

Where we want to send

⊛ ~~git hub~~ free vendor → gitlab.

How to publish

Group name (your organisatn)

Username = @nilesh-sipl.
don't consider @.

Subeam →

Menu
① project → your projects → new project → blank project → demo → ~~pub~~ privat

→ Create project → my java classwork → push an existing folder.
→ programing assignment →
clone with HTTPS →

① git clone paste → creates new folder.

② Ellipse → Assign1.java –

To commit ① .gitignore file (*.class)
bin/

② git status.
git add.
git commit -m "assignment complete";
git push

To make it public or private. github →

28/9/2021 Day 6

Initialising Static Variable

```
Public class {
        Pint num1;
        Pint num2;
        Pstatic int num3;
        Test ( ){
            this (20,30);
                or
            this. num1 = 20;
            this. num2 = 30;
        }
        static initialiser block.
        {
        static
          {
            this.num1 = 20;
            this. num2 = 30;  num3 = 500;
          }
        void print_record ( ){
            syso ( "  " + this. Num1);
            syso ( "  " + this. num2);
        main ( ){
            Test s1 = new Test ( );
        }
```

[data m/m.F]

static initialiser block
num3=500

gets loaded when class gets created

class test

int num1;
int num2;

obj 1 → num1, num2 — obj1 = new class(test)

obj 2 → num1, num2 — obj2 = new classtest.

Day 2_6 Getter & setter for static

```
Public static int getnum3(){
    return num3;
}
```

static can access static only

static ⟷ static

public ⟷ public

Constructor chaining ——→ used in parameterless.

```
Test(){
    this(30,40);
}
```

✗ if

static function cannot access non-static members

✗ if you want they are class level fields and need to access by class only

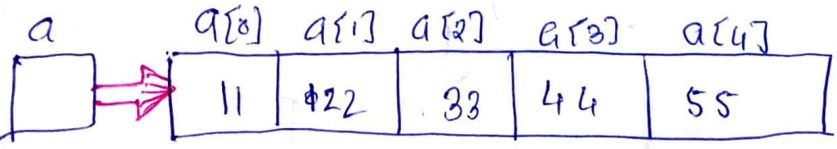
✗ !

Day 6_5 Array

```
int a1;
int a2;
int a3;
```

Array collection of similar kind of data.

datatype ars[] = new datatype[];

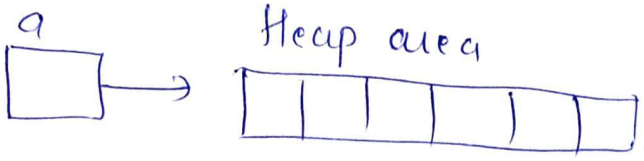


int a[] = new int[4];

↳ subscript

int a[] = new int[4]. → size of an array

↗ Compiler will understand it is integer.



Valid / Invalid syntaxes

int arr[] = null; ✓

int @[arr2] = null; ✗

int @]arr = null; ✓

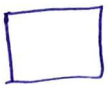
int arr4[] = new int[3] ✓

int[] arr5 = new int[4] ✓

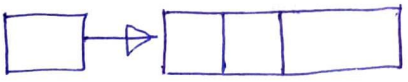
int arr[] = new int[-3] ✗ negative size array exception.

int arr[5] = new int[] ✗

int a

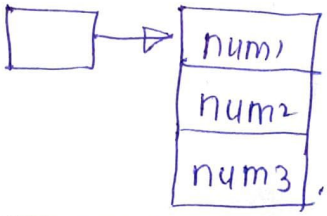


int a[] = new int[3]



Heap.

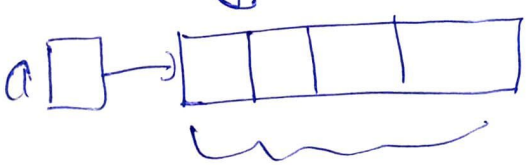
Demo d = new Demo();



Taking i/p of array

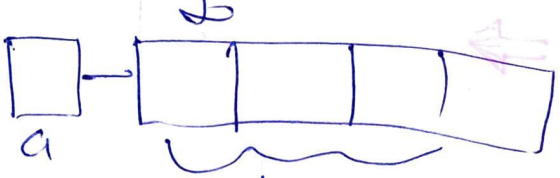
Non primitive array :-

int a[] = new int[4];



Memory space

Integer a1[] = new Integer



Object

code is generated only when object is passed

array of non primitive & primitive

```
for(int i=0; i<4; i++); {
    syso(a1[i]); → non primitive Integer a1[];
    syso(a1[i] → primitive.
    op → null null null
    → 0, 0, 0, 0.
```

Arr.to string

syso(arr) → Internally give call to to string function. hence returns address.

syso(arr.toString) → arr.toString( ) → gives base address

array.util → arrays.toString(arr); → only primitive accepted

[1,2,3,4,5];

Array sort

⊛ To search a element

arrays. binary search