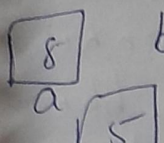
OOPs Class-2

Date- 25/10/2023

Copy Constructor :-

Jab Student s1, s2; → ctor → ① garbage
② Parametrized ctor

```
int a=5
int b;
b=a;
```
→ copy hoti hai




```
Student s1( ---- );
Student s2;  ⇒ copy ctor
s2=s1;
```

Q Student s1; → main me ctor nhi likha tha class ke andar toh by default compiler add default ctor.

s2=s1; ⇒ copy ctor define nhi kiya.

Is case me by default compiler default copy constructor added.

Jab copy karegye toh public hoga

shallow copy create karta hai

int a = b; do int copy hota hai
(a ↓ destination, b ↓ src)

Jab hum s2=s1 → Two students will be copied.
(s2 ↓ destination, s1 → src.)

class ke andar ke obj ko access karne ke liye this pointer kaise hota hai

Jisme assign ho rha hai vo obj ban rha hai

Modification s2 me ho rhi hai this s2

copy constructor creation →

```
Student(const Student& srcobj)
{
this->name = srcobj.name;
this->id = srcobj.id;
this->gf = srcobj.gf;
}
```

* copy constructor ke andar data members copy hogye function copy nhi hogye.

Constructor

Jab ~~aap copy constructor~~ object ko bna rhe hai toh tabhi copy constructor se bna lo.

Student S2 = S1;
Student S2(S1);

copy constructor har baar brana jaruri nhi hai. Jaise default constructor add ho jata waise hi copy constructor bhi add ho jata hai

lekin copy constructor add karne ki habit achi hai

Why we need copy constructor?

Deep copy karni ho.

1. copy / shallow copy] imp concept.
2. deepcopy

Student const declare kargye taki change n kar paaye.
good practice to declare const

let say mai gadi chaluga but meri gadi bani hi nhi hai toh ye possible hi nhi hai

main me se call karegye, toh by copy value pass hogi aur hum keh rhe hai copy banegi and copy constructor ~~declare kiya~~ like rha hai isse nhi ban rha hai

Toh isliye by refernce bhej degye.

```
Student(const Student & srcobj){
{
    this->name = srcobj.name;
    this->age = srcobj.age;
}.
```

```
main(){
    int a;
    a=5;
    fun();
    a=5;
    return 0;
}
```

```
void fun(){
    int b;
    b=5;
    return;
```

koi bhi variable ki life cycle hoti hai

1. init
2. copy
3. Destroy..

Just like in real life Birth death.

life cycle of an object

main()
{
Student S1(- - - -) → ctor

- - -
return 0;

}

Jab yha se return kar jayege toh S1 ka vajud nhi hoga. S1 will be destroyed.

⇒ Object constructor ke through construct hota hai

⇒ toh destroy karne ke liye destructor bhi hota hai

If you don't write dtor, compiler will take the responsibility

default constructor
copy const.
destruct
] agar nhi bnaye toh → compiler bnayega.

Parameterised agar banana hai toh khud se bnana hoga compiler nhi bnayga.

Shallow copy → dangerous.

dest / dtor
kya compiler create kar rha hai toh ye safe hai

Destructor → main se call ho rha hai isliye public hoga.

~Student()
{
cout<<"Student Kcnoll."
delete v;
}

return type nhi hota
Constructor → default
copy "
Destructor.

the memory has to be allocated need to be delete because memory leak ho jaayegi

this→v = new int[10];

toh destructor ki jarurt padegi

Heap me create hui hai

getter setter method
kissi ke liye bhi use kar sakte hai.

① Pillars of OOps :-

1⇒ Abstraction → loose coupling
implementation details are hidden

Abstraction me hum cheezo ko upar upar se use karte hai jo humare matlab ki hai

2⇒ Encapsulation →  → WRAP
Bundling of data & Methods
It is a way to implement abstraction

class ke andar internal details ko hide kar dete hai aur jo neccesary hai vo expose kar dete hai

Why encapsulation ① Easy to handle
② Protect Integrity (security)
friend keyword
private → control | how class data's modified.
③ Maintainability

Perfect Encapsulation
→ If all data member are private
→ Through getter | setter

class data members ko get or set karte hai

Encapsulation Is often to mean, to achieve abstraction by hiding internal details and only exposing what is necessary.

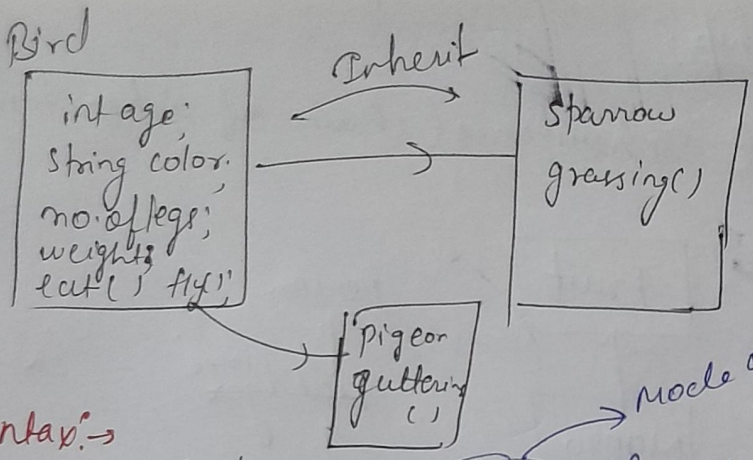
3⇒ Inheritance →
Parent → attributes
↓
son → attributes

class Animal {
void eat()
void sleep()
}

Persian cat → maa se eye orange
↓
eye orange

Super class / Parent class / Base class
↓
Subclass / childclass / Derived class.

cat
→ eat
· sleep
→ gaana()



Syntax:→

class Childname : (- - -) Parent_name {

mode of inheritance

extensibility rules.

① public ② protected ③ private

Experiment karo khud se

Base class access modifier	Mode of Inheritance		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	NA	NA	NA

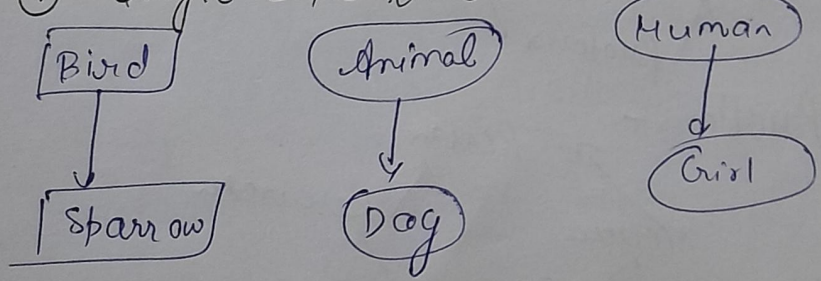
let say protected and private are same

Protected:- Members declared protected are accessible within class itself & to its derived class.

Private:- within class hota hai access
⇒ derived class vissibility not possible.
⇒ Private data can't inherit.

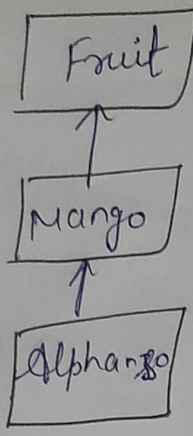
Types of inheritance:-

① Single Inheritance -



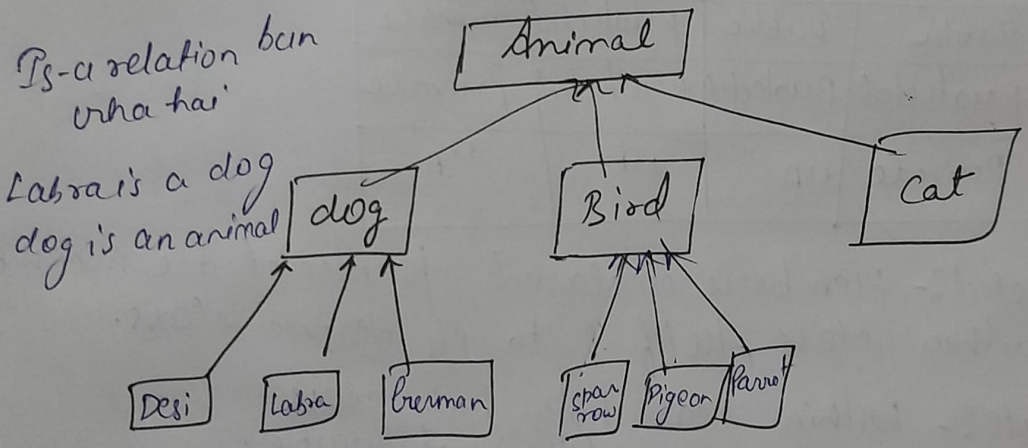
IS-A relationship:

② Multilevel Inheritance (chain of inheritance)



Person
↑
employee
~~manager~~
↑
manager
~~professor~~

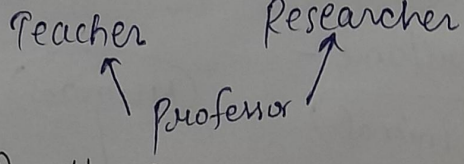
③ Hierarchical
~~Multiple~~ - Inh.



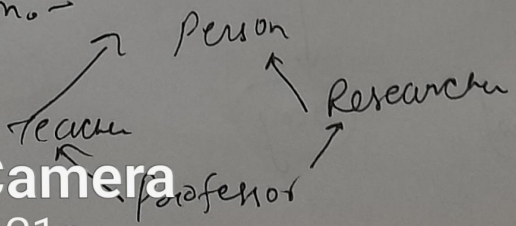
IS-a relation ban rha hai

Labra is a dog
dog is an animal

④ Multiple Inheritance (Not possible in Java **)
Derived class inherits from more than one class



* Diamond Problem:-



Problem ye hai ki humare pass 2 walk hai teacher ka walk Researcher ka walk usko pta nhi lag ~~ith~~ paa rha ki kiska walk use karo.

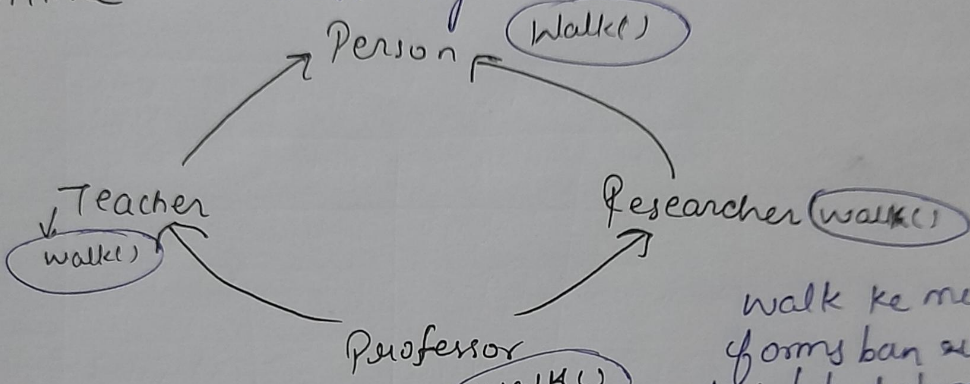
scope ~~Resol~~ Resolution se access karege

let say dono ka ek hi walk hai toh virtual use karo

virtually means Abhi mat karo runtime pe inherit kar dena.

compile time – less logical hoti hai

RunTime – more logical hoti hai



walk ke multiple forms ban rahe hai toh polymorphism hogya.

③rd Pillars of OOps.

Polymorphism :- many forms
↳ existing in many forms.

① compile Time → faster

② RunTime → function overriding.

① compileTime → Static Polymorphism

(i) function overloading → Parametrized constr
↳ Normal funcⁿ

(ii) Operator Overloading (+, −, ++, −−, ()...)

func ka naam same rakha lekin return type or parameter change

let vector

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \Rightarrow \begin{bmatrix} x_1+x_2 \\ y_1+y_2 \end{bmatrix}$$

V_1 V_2.

```
class vector
  { int x, y;
    public:
vector(int x, int y)
  {
this->x = x;
this->y = y;
}
```

parameterised constructor likhne ka dusra tarika hota hai initialisation list.

```
vector(int x, int y): x(x), y(y) {
```

=> same =>