HW

const

static

initialisation list

padding & greedy

alignment

Rec

Pending

Deep & Shallow copy

OOPs Class-2

Special class

4 pillar of OOPs

Encapsulation ✓

Inheritance ✓

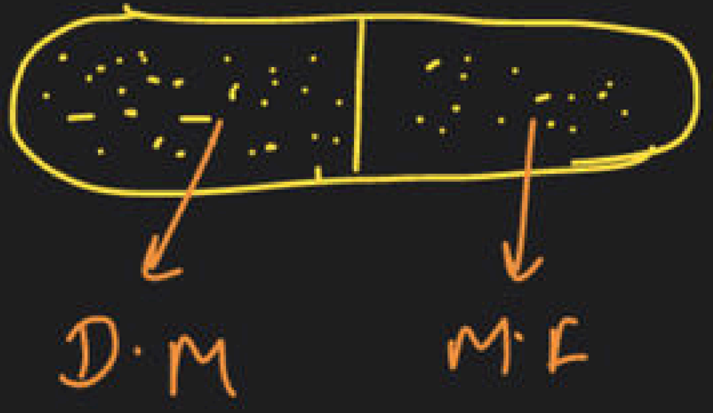
Polymorphism ✓

Abstraction ✓

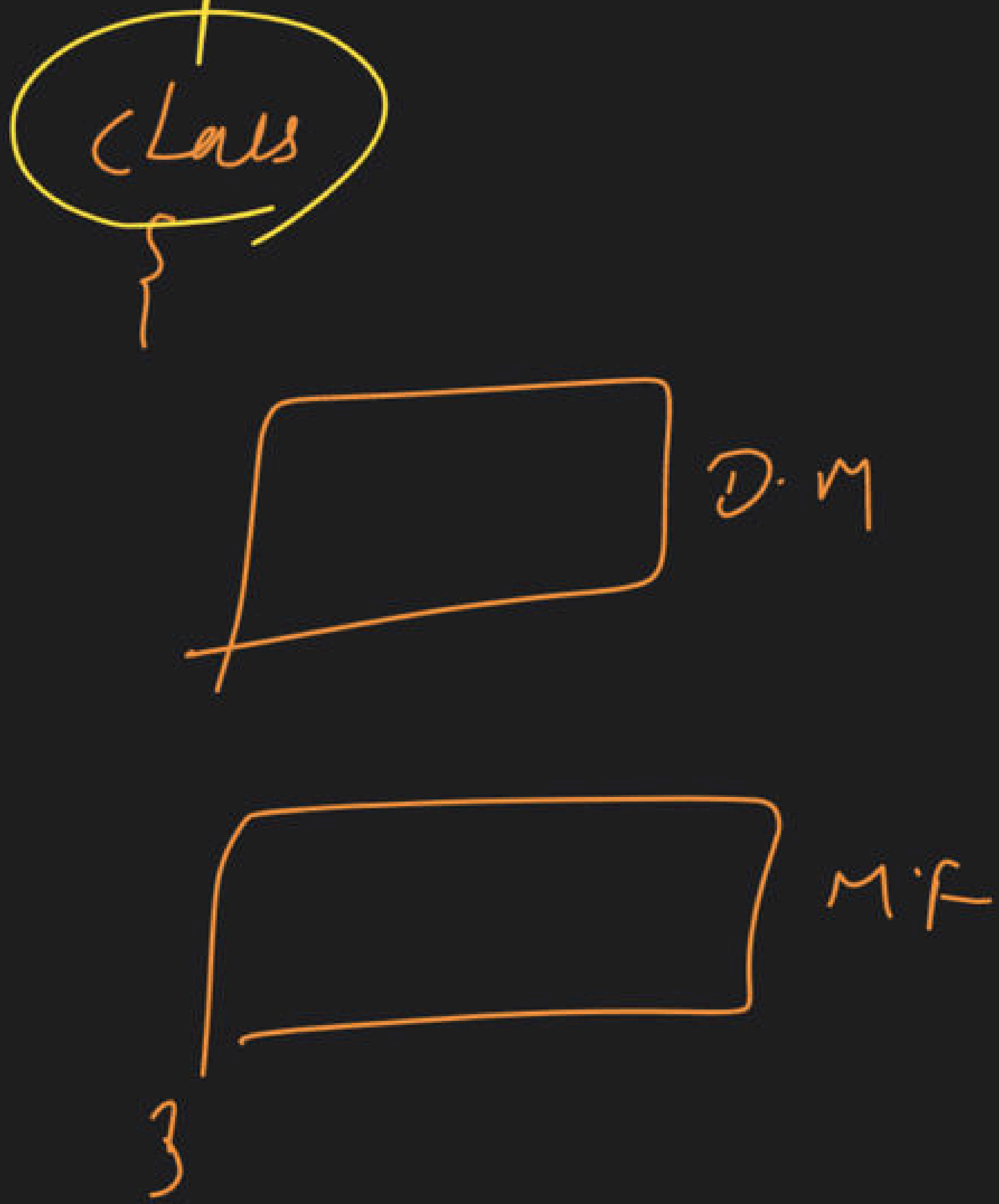
Love Babbar • Apr 2, 2023

Encapsulation :-

(Data Hiding)



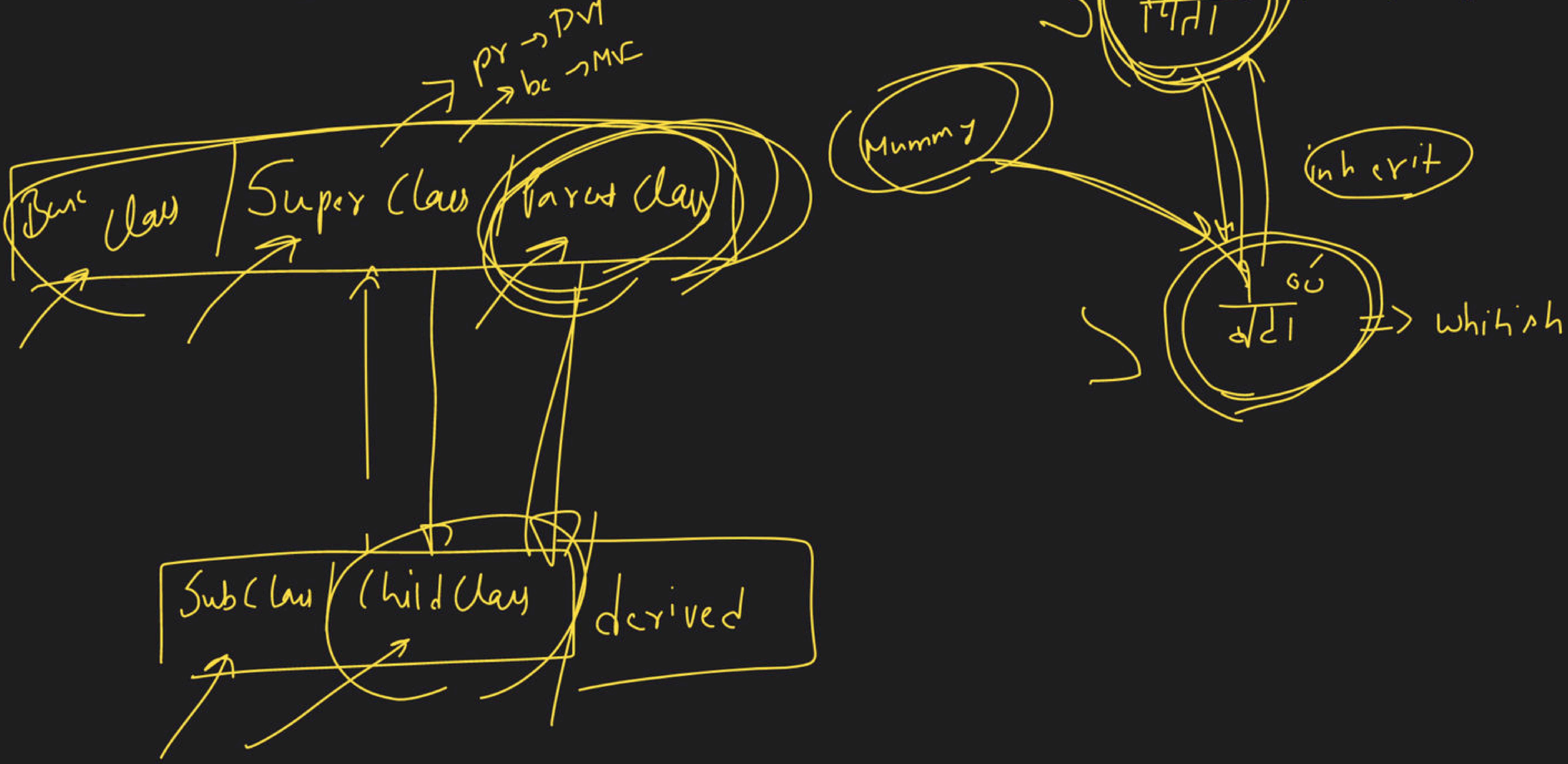
as a capsule

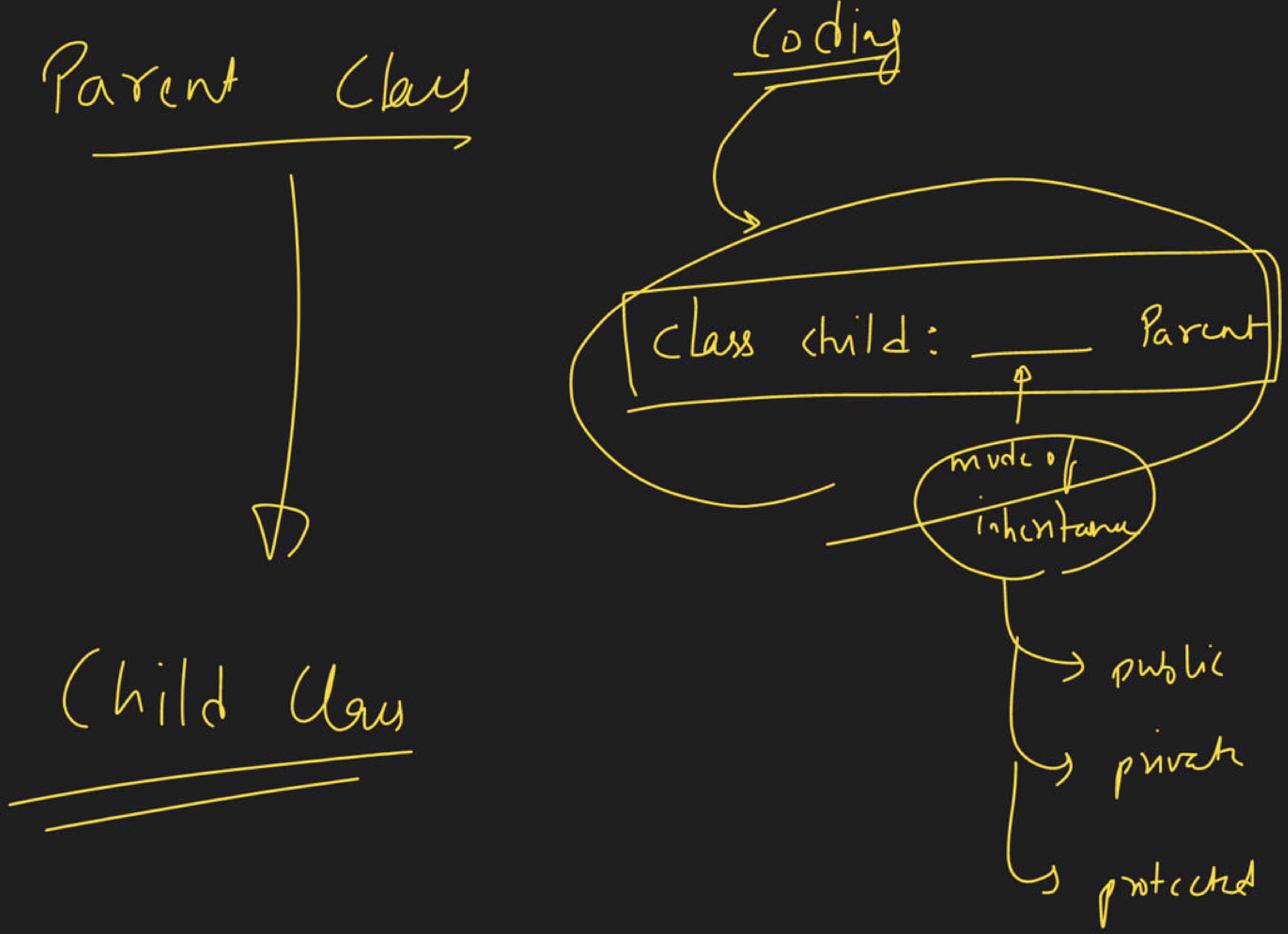


Perfect Encapsulation → ?

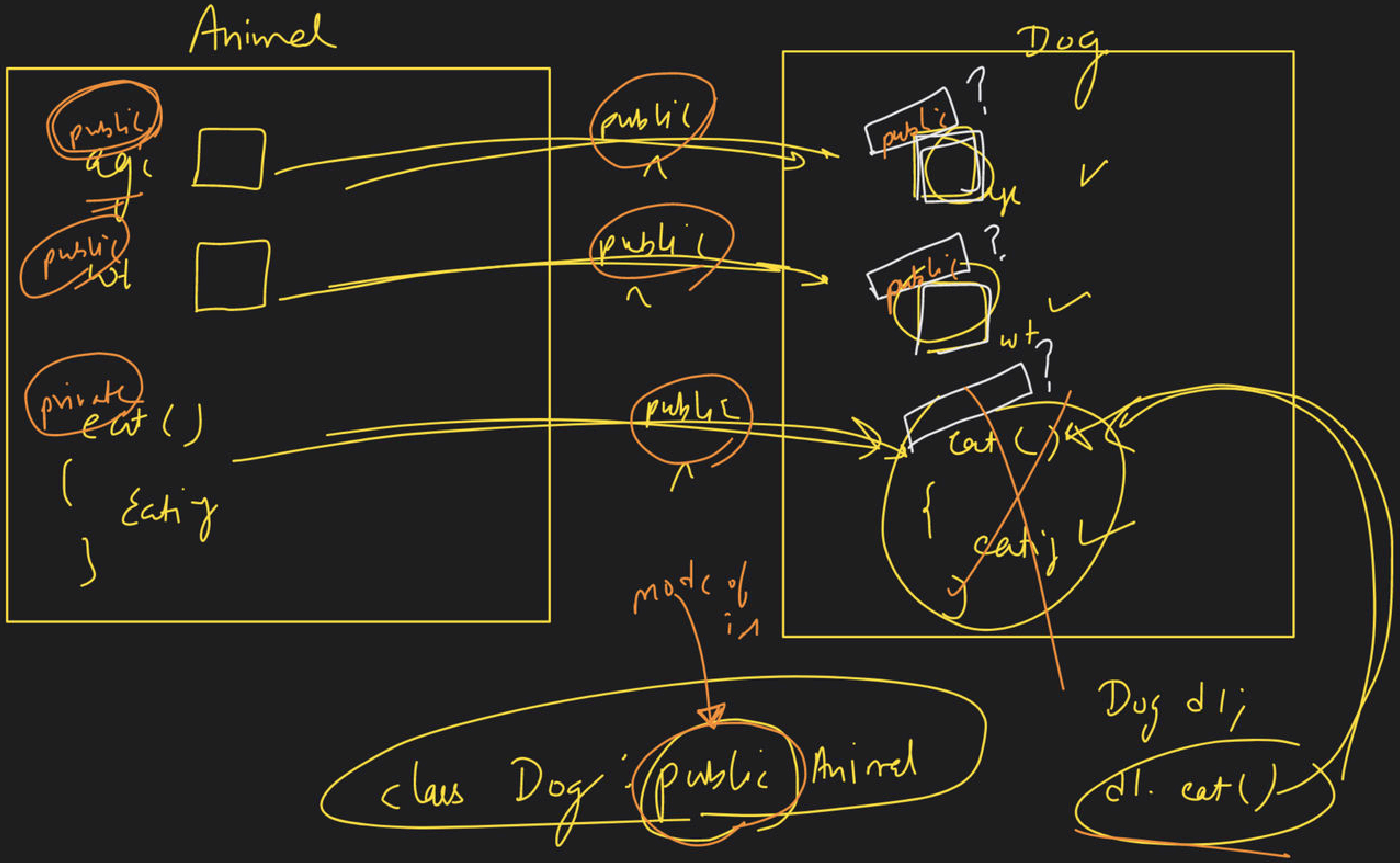
private

Inheritance

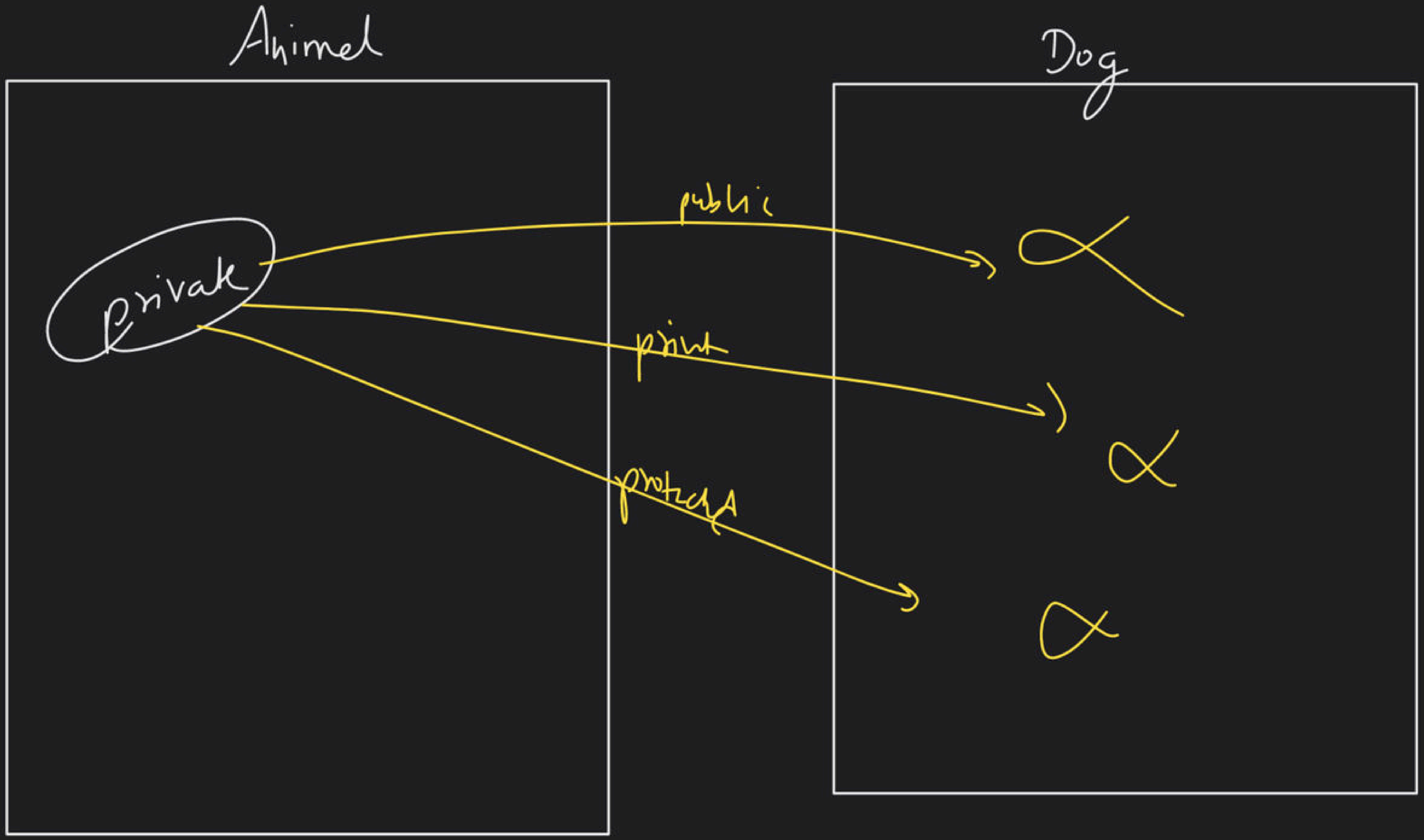


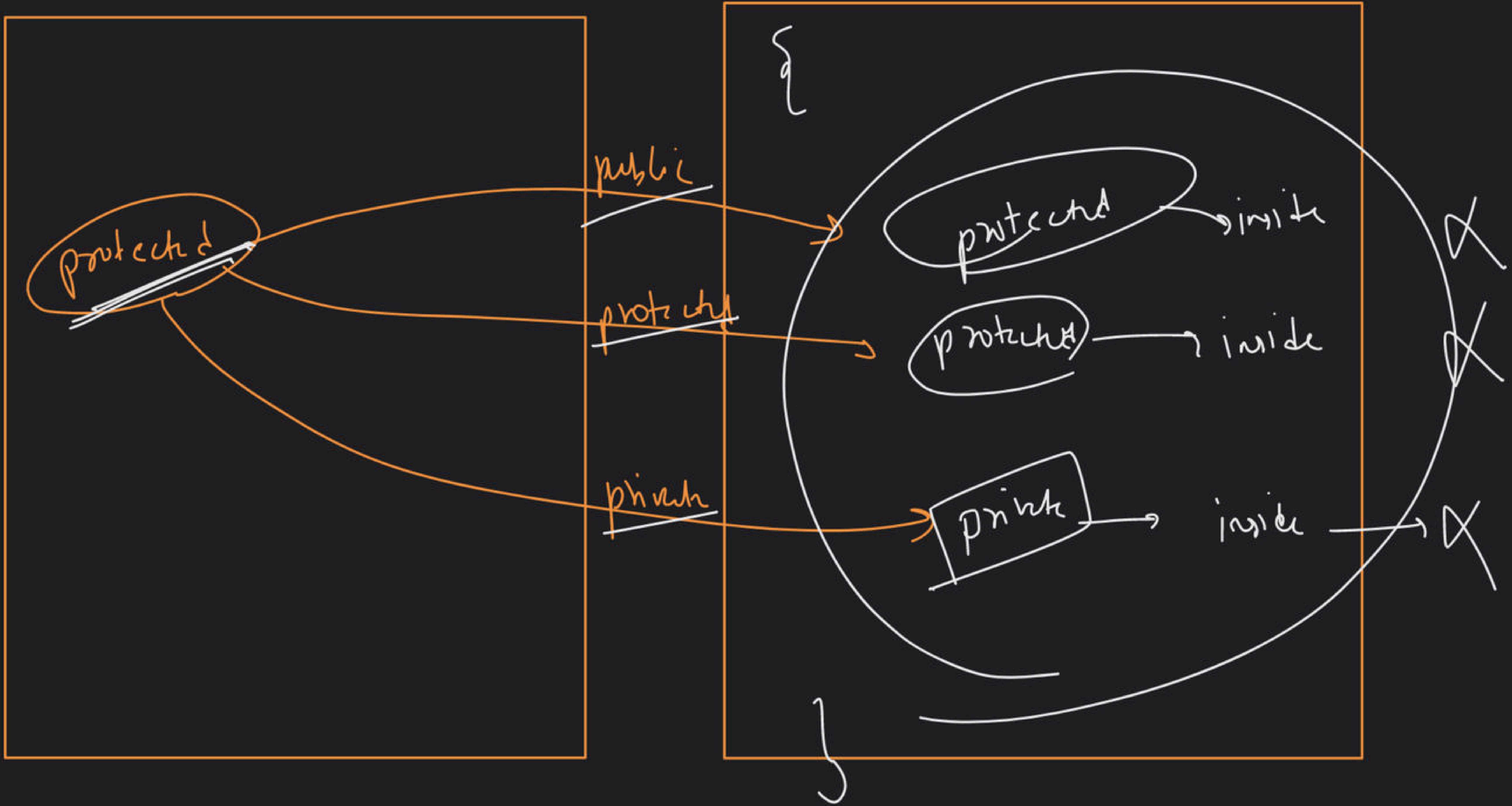


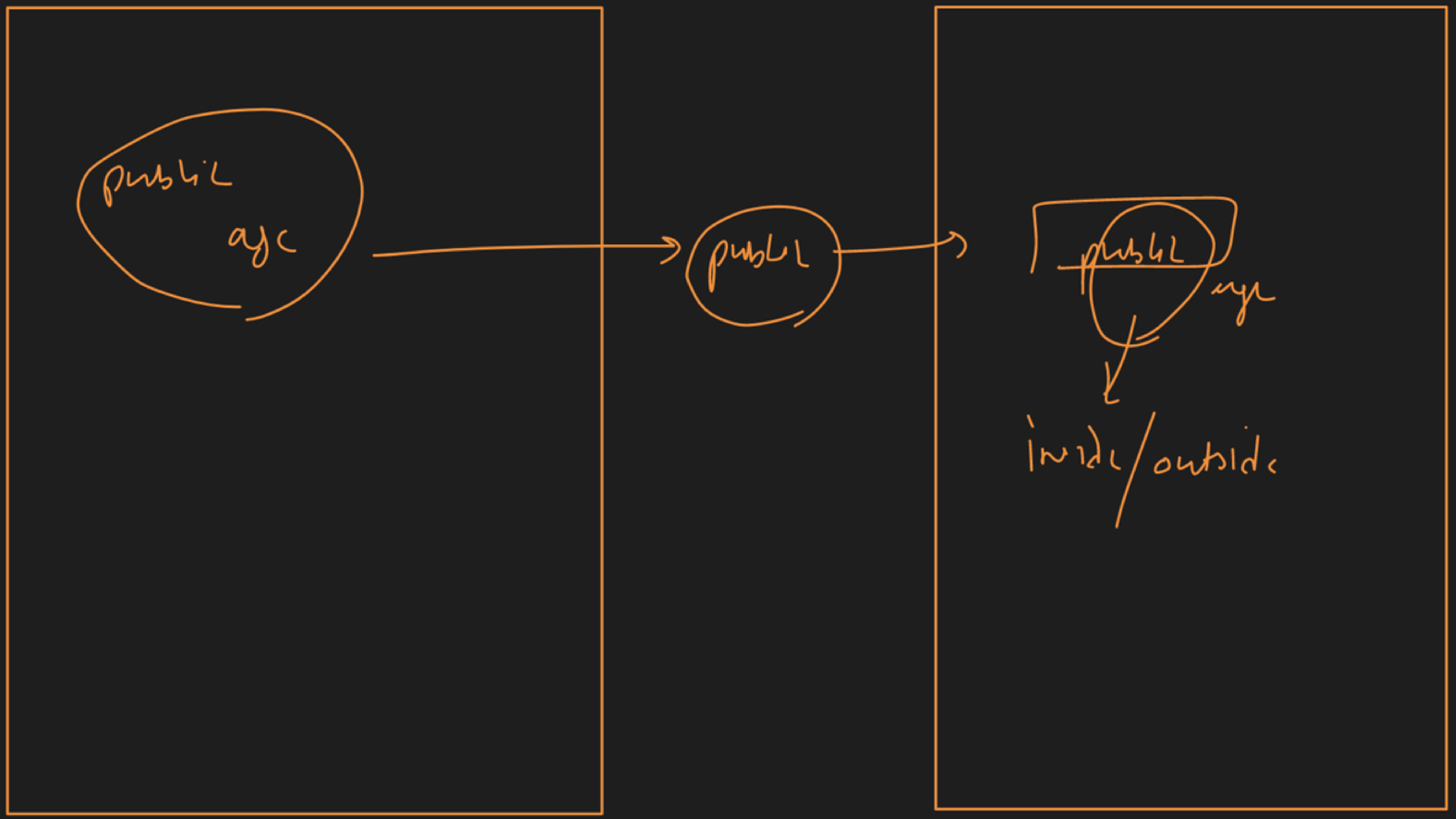
```
class child: ______ Parent
```


Animal
public age
public wt
private eat()
{ Eating }
public
public
public
Dog
public age
public wt
eat() { eating }
mode of in
class Dog : public Animal
Dog d1;
d1. eat()

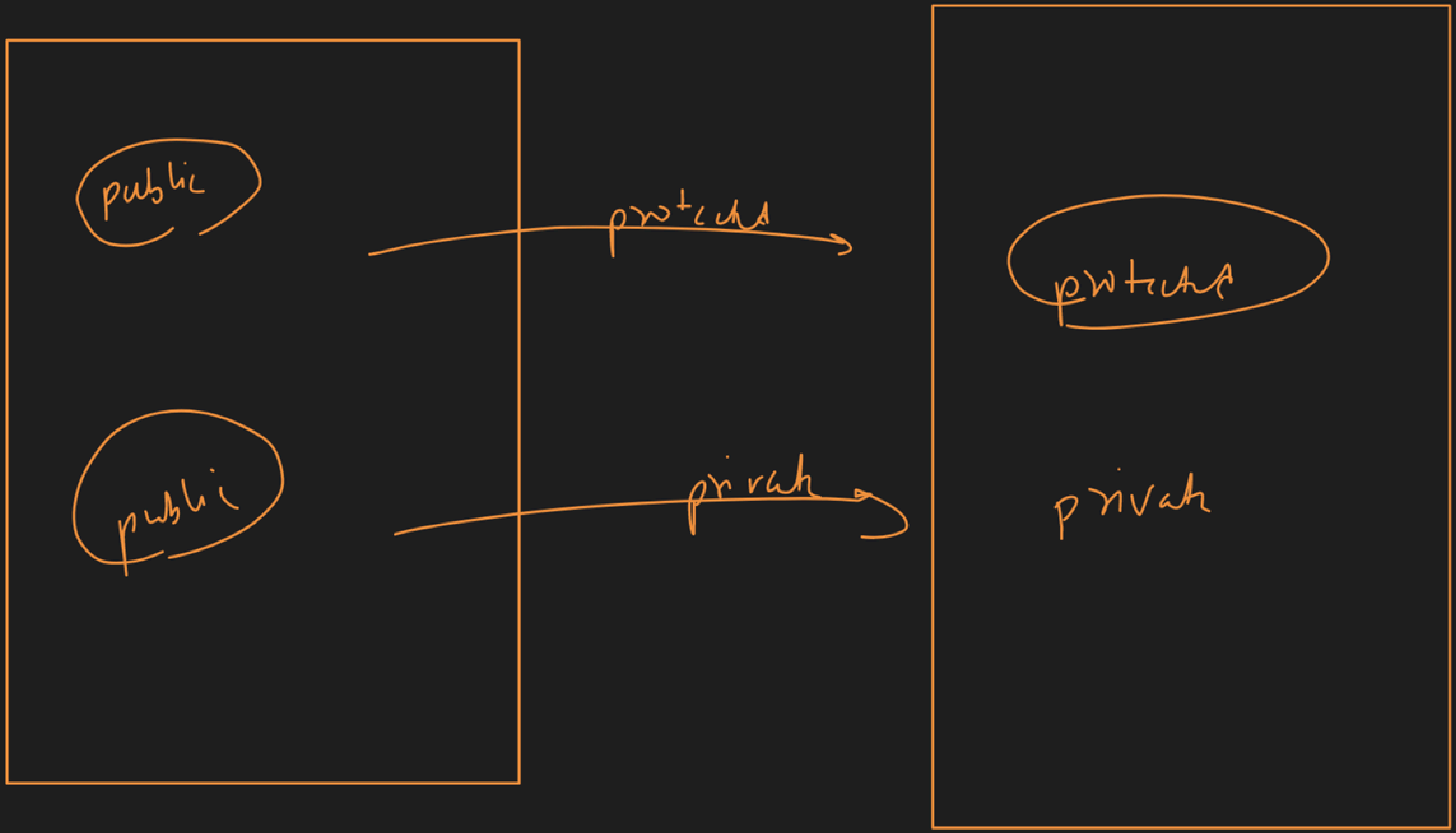
Base class ke Access Modifier	Mode of Inheritance		
	Public	Protected	Private
Public x	Public	Protected	Private
Protected	Protected	Protected	Private
Private	NA	NA	NA

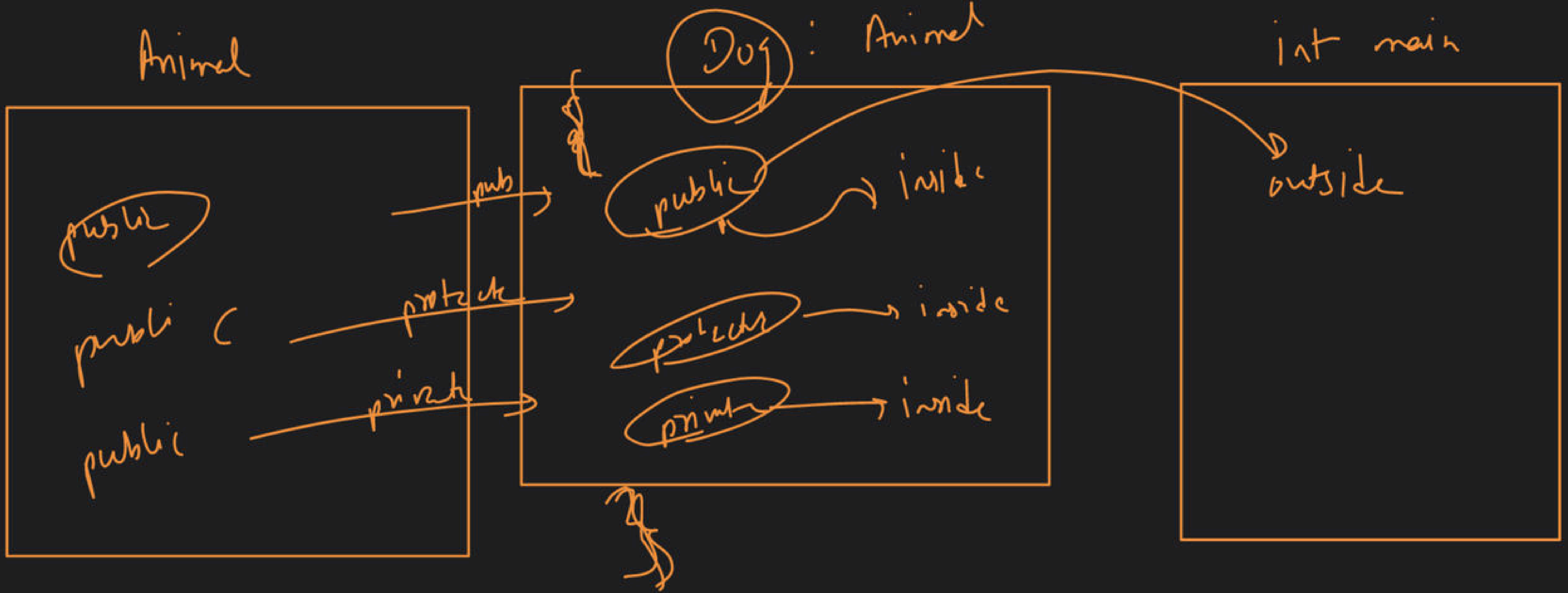

Animal
Dog
private
public
private
protected

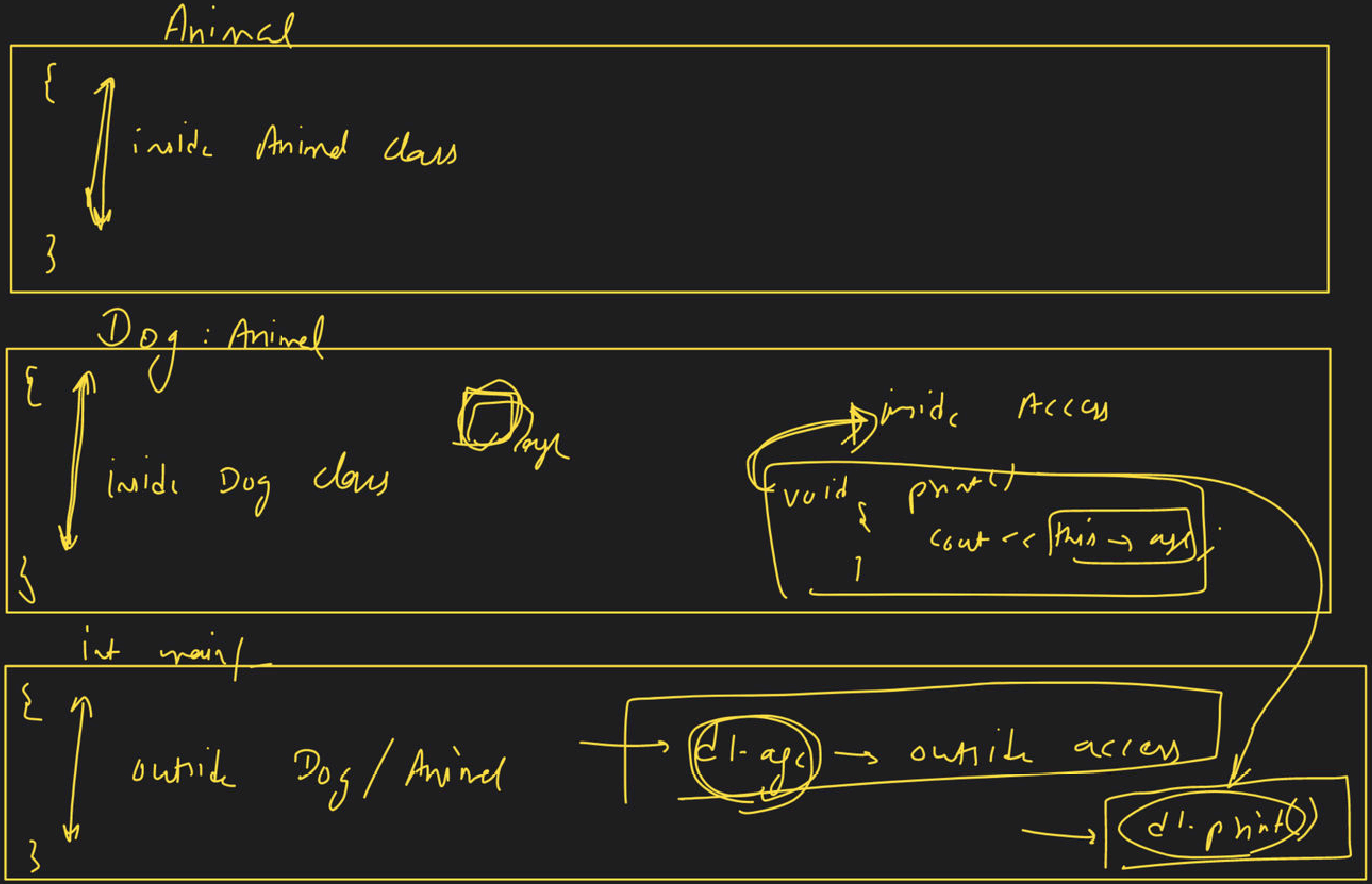

protected
public
protected
private
{
protected
inside
protected
inside
private
inside
}



Mode of Inheritance


public
public
protected
private
protected
private


Animal
public
public
public
pub
protected
private
Dog : Animal
public
protected
private
inside
inside
inside
int main
outside



```
void print()
{
    cout << this->age;
}
```

Inheritance

→ ✓

class $\boxed{\text{child}}$: ______ $\boxed{\text{Parent}}$

Mode

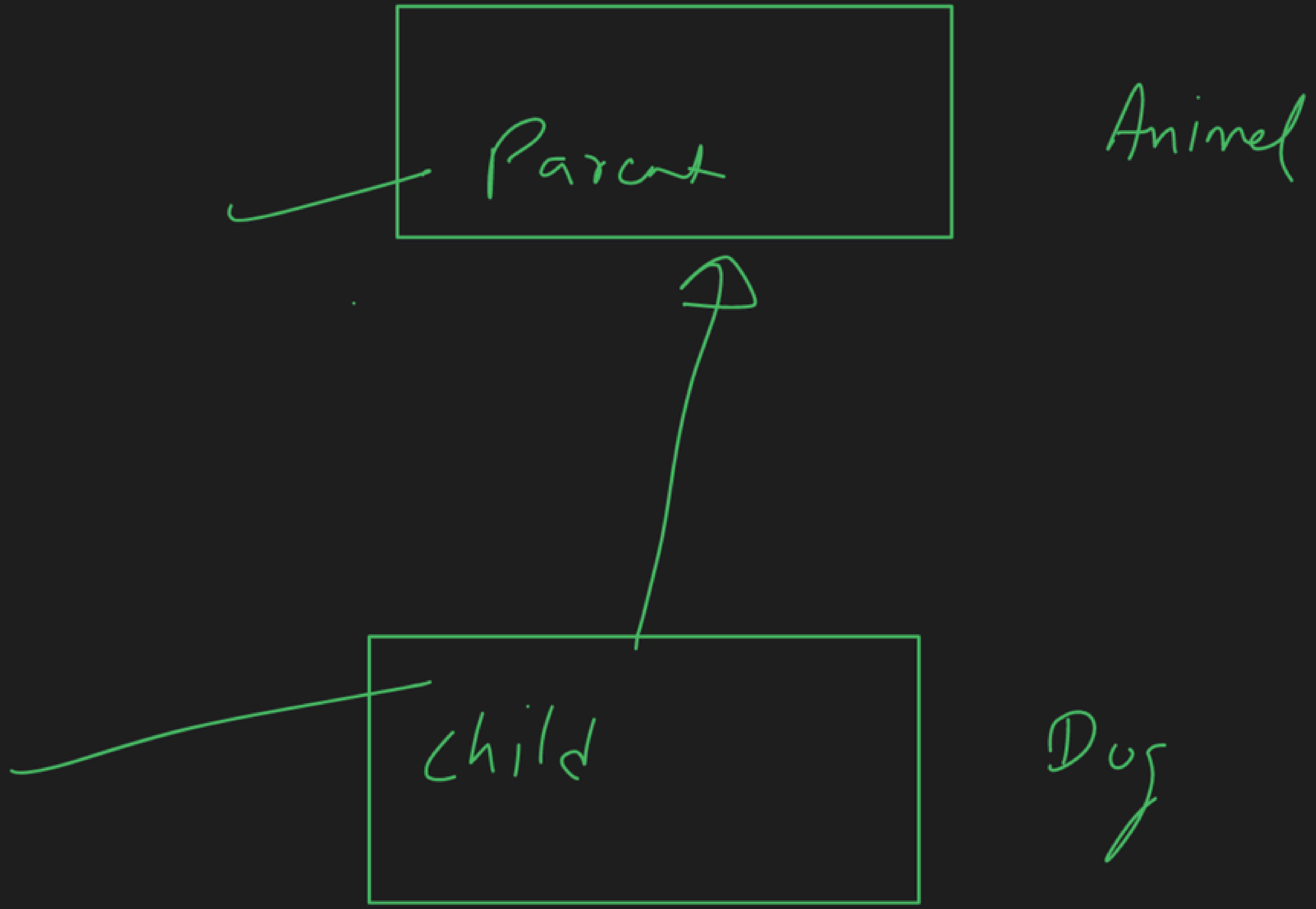
{

}

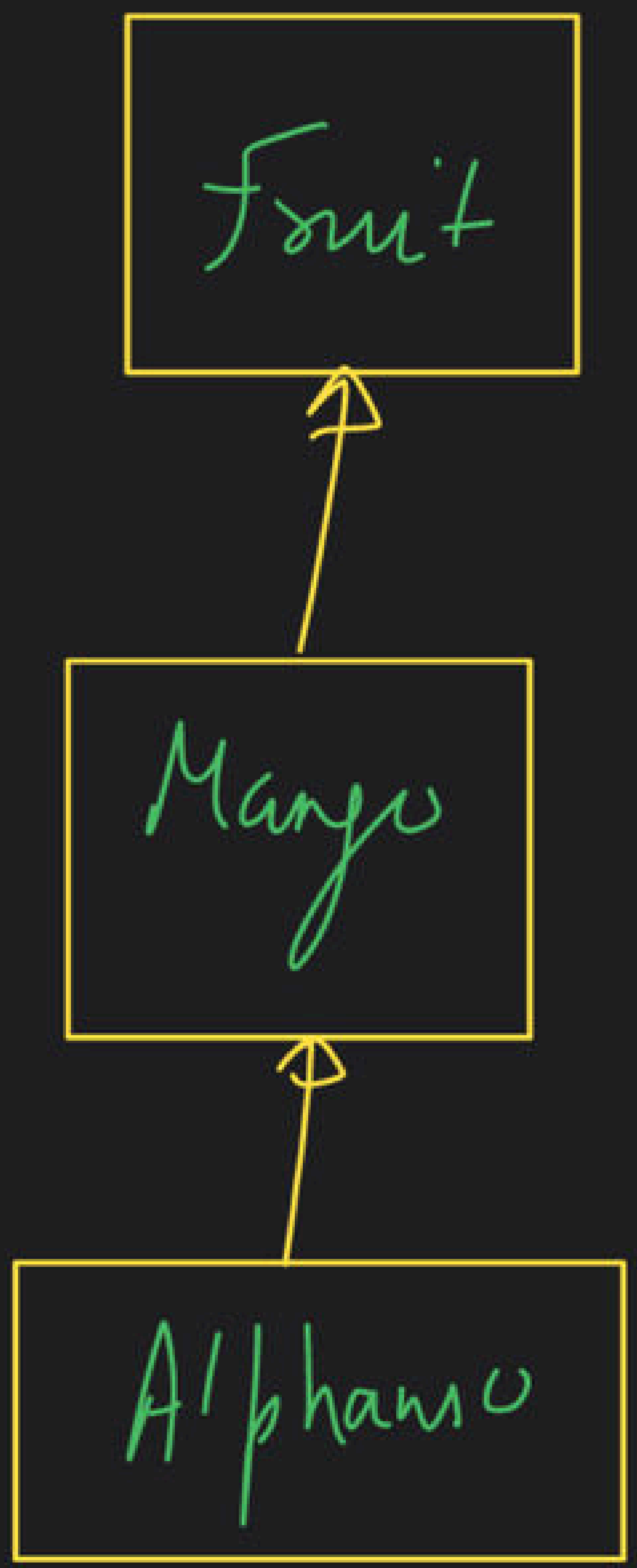
→ Type of Inheritance

- Single
- Multi-level
- Multiple
- Hierarchical
- Hybrid

Single



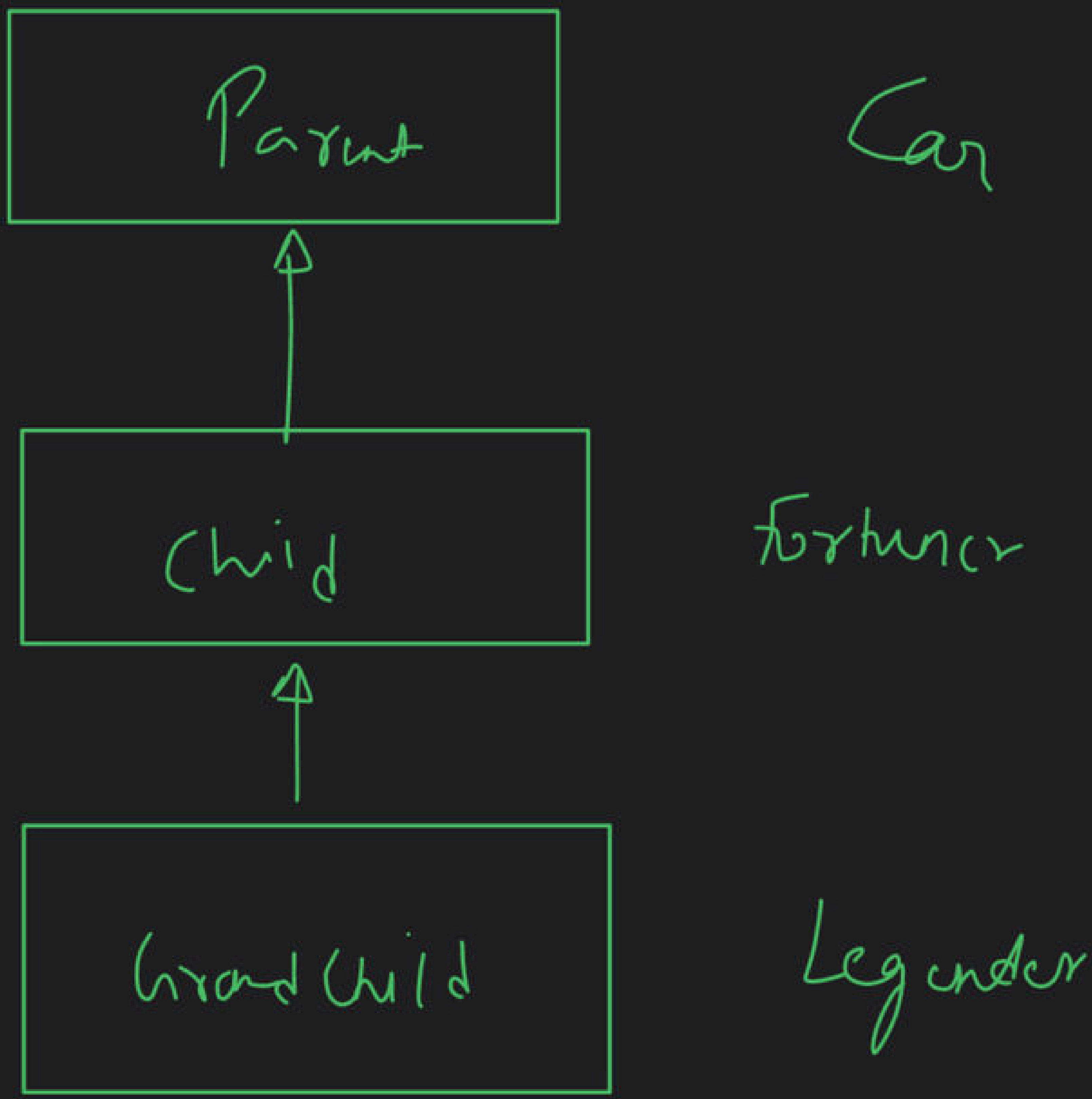
$$\begin{tikzcd}\boxed{\text{Car}} \\ \boxed{\text{Scorpio}} \arrow[u]\end{tikzcd}$$

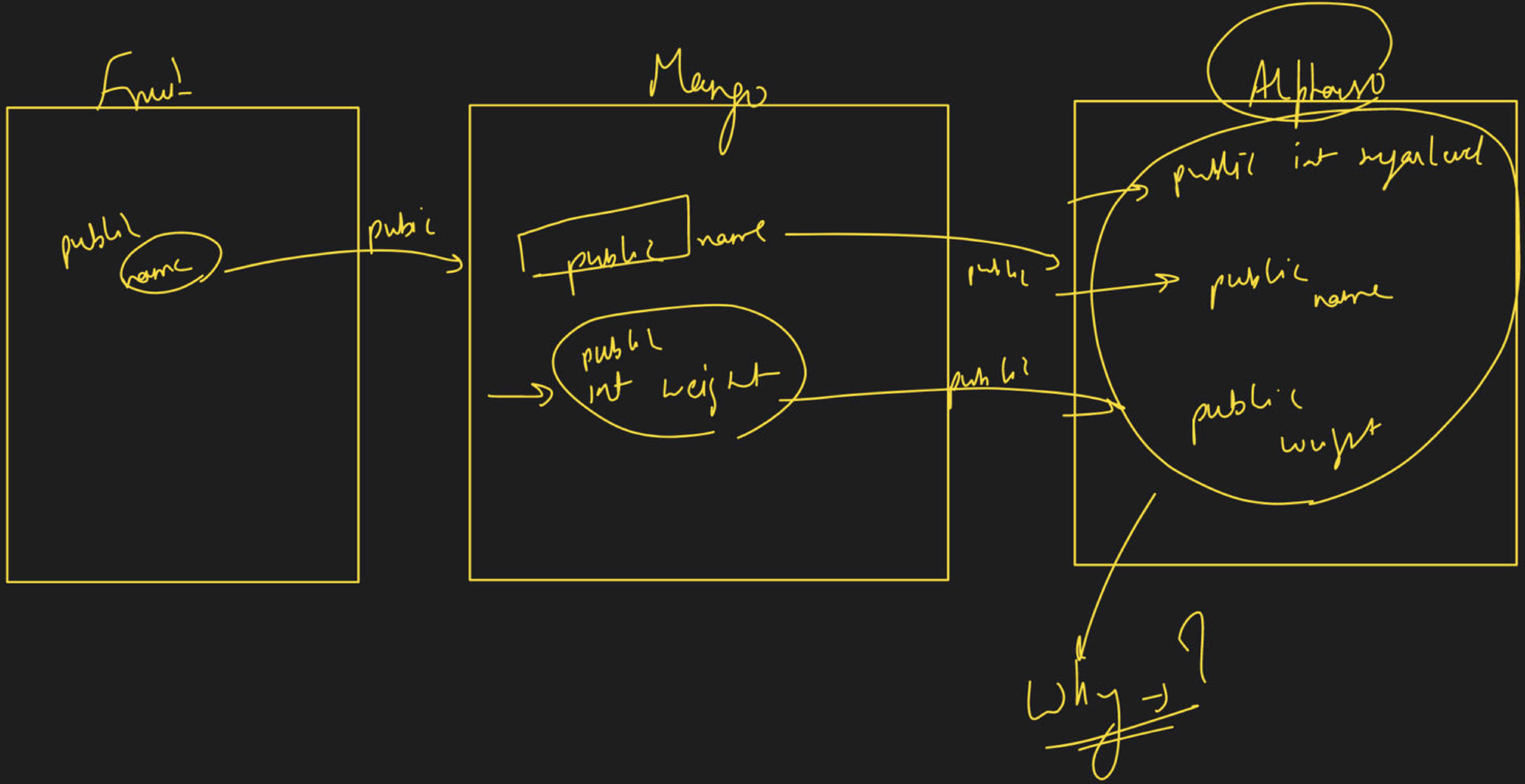


~~Car~~

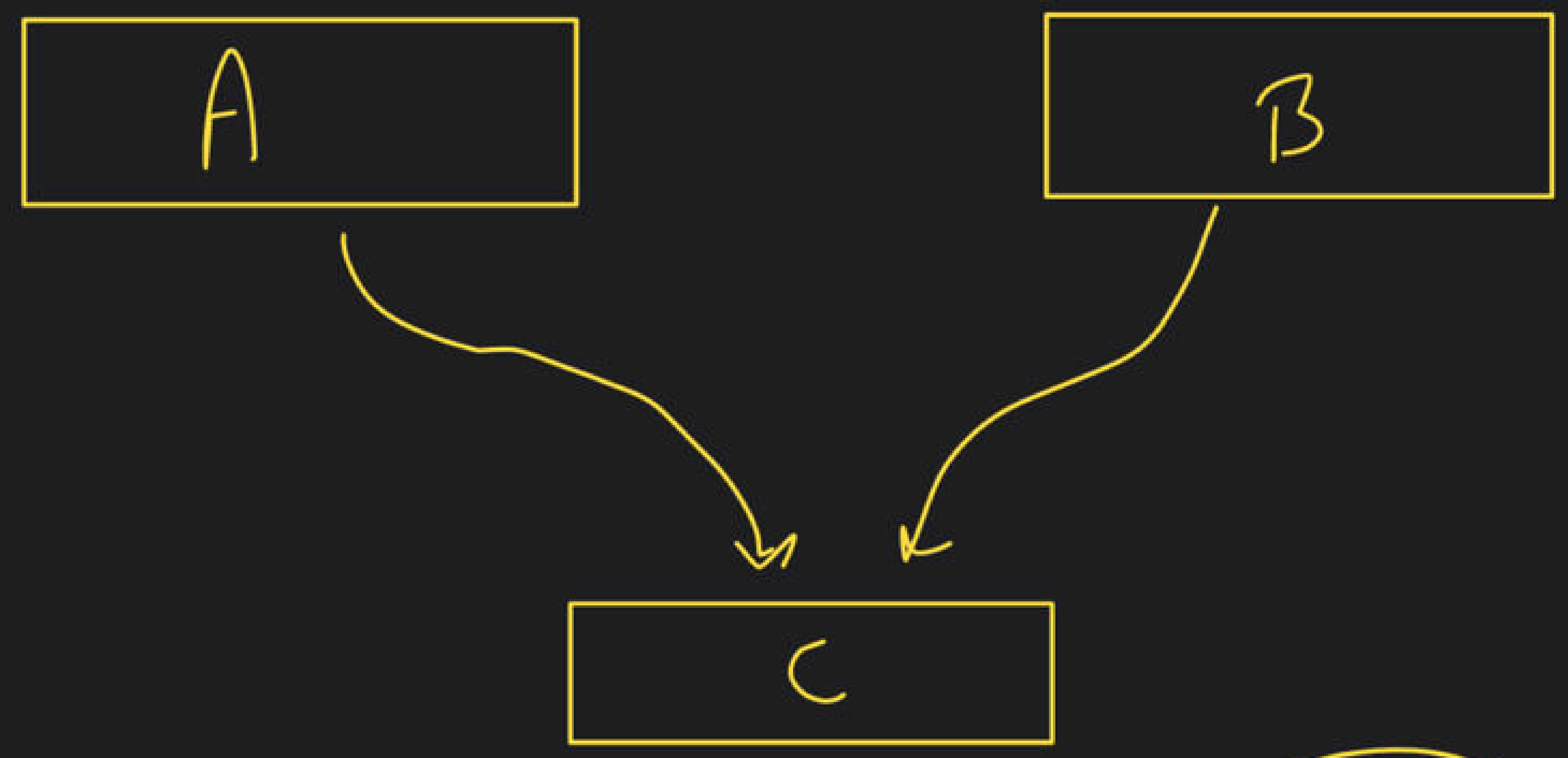
~~Mahindra~~

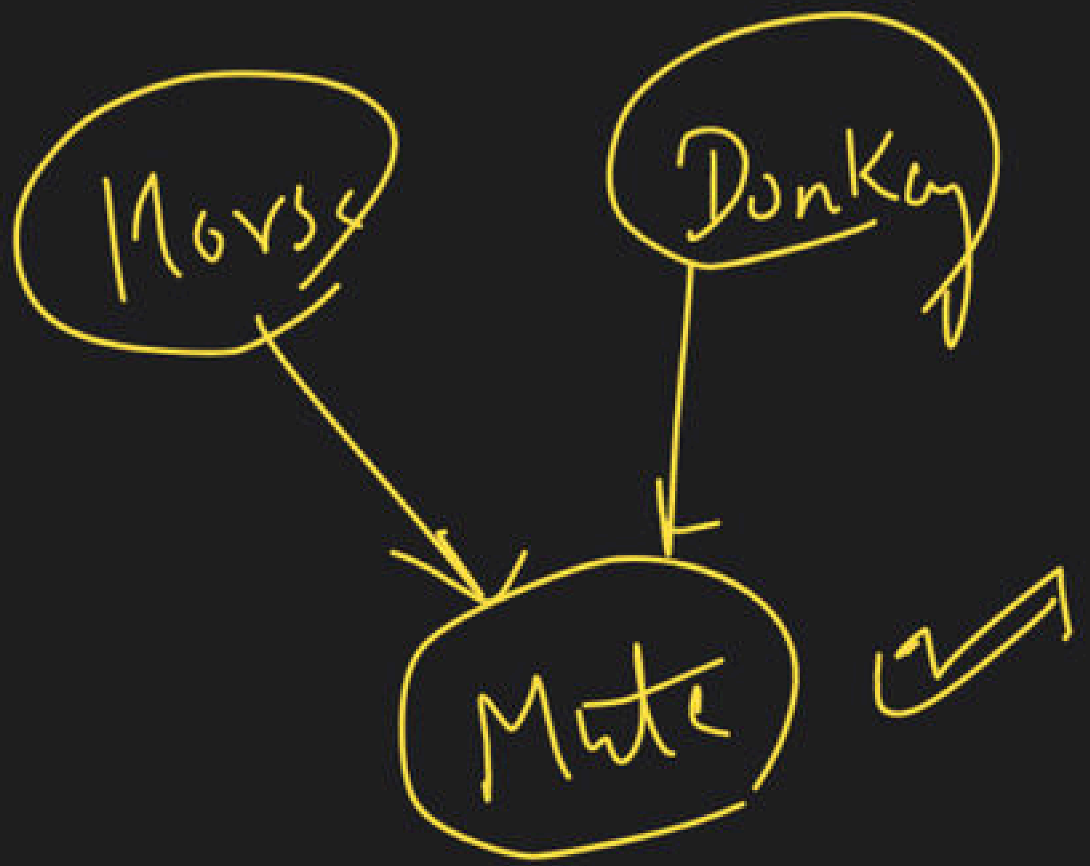
~~Scorpio~~

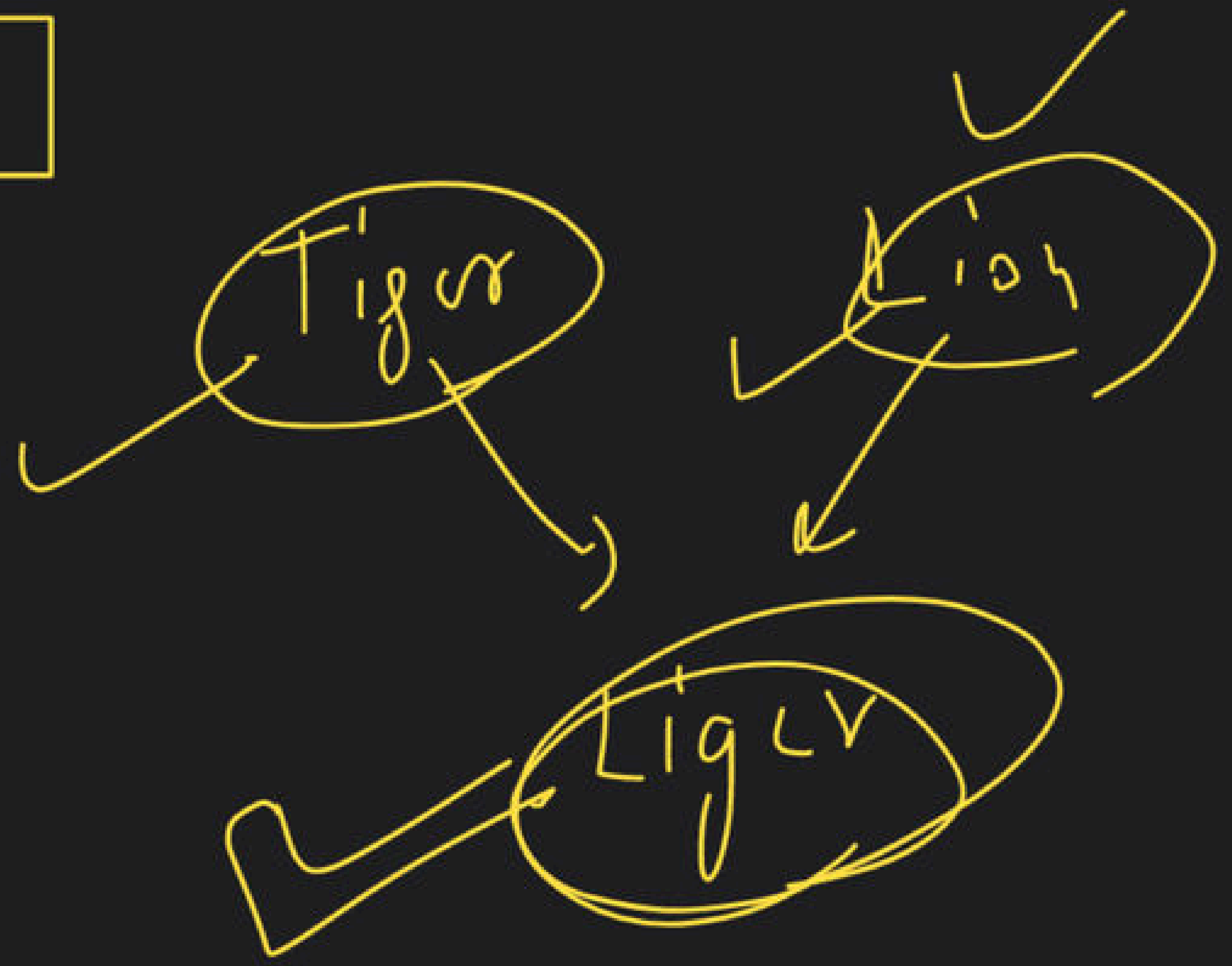


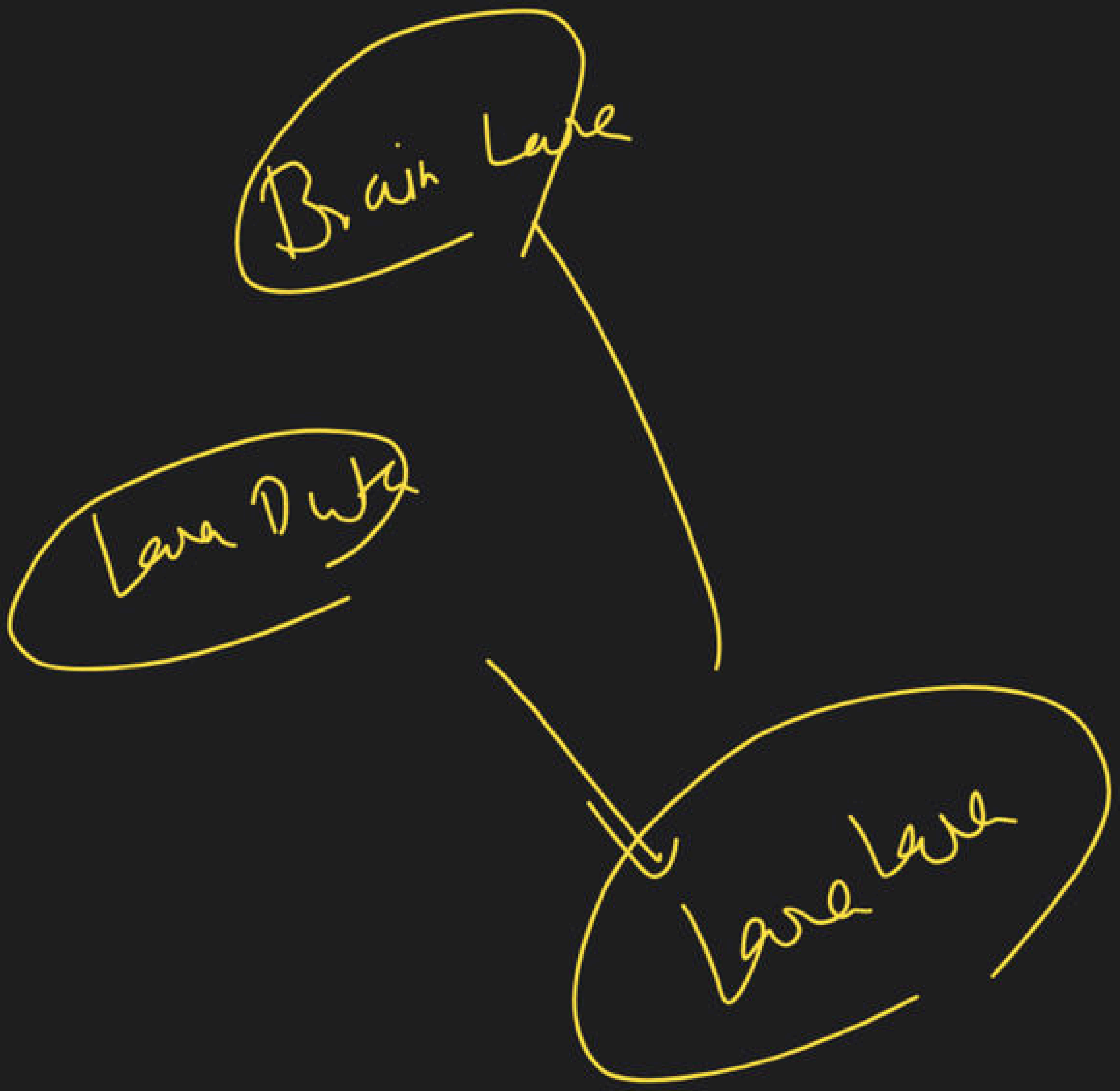

Fruit
public
name
public
Mango
public name
public
int weight
public
public
Alphonso
public int sugarlevel
public name
public weight
Why → ?

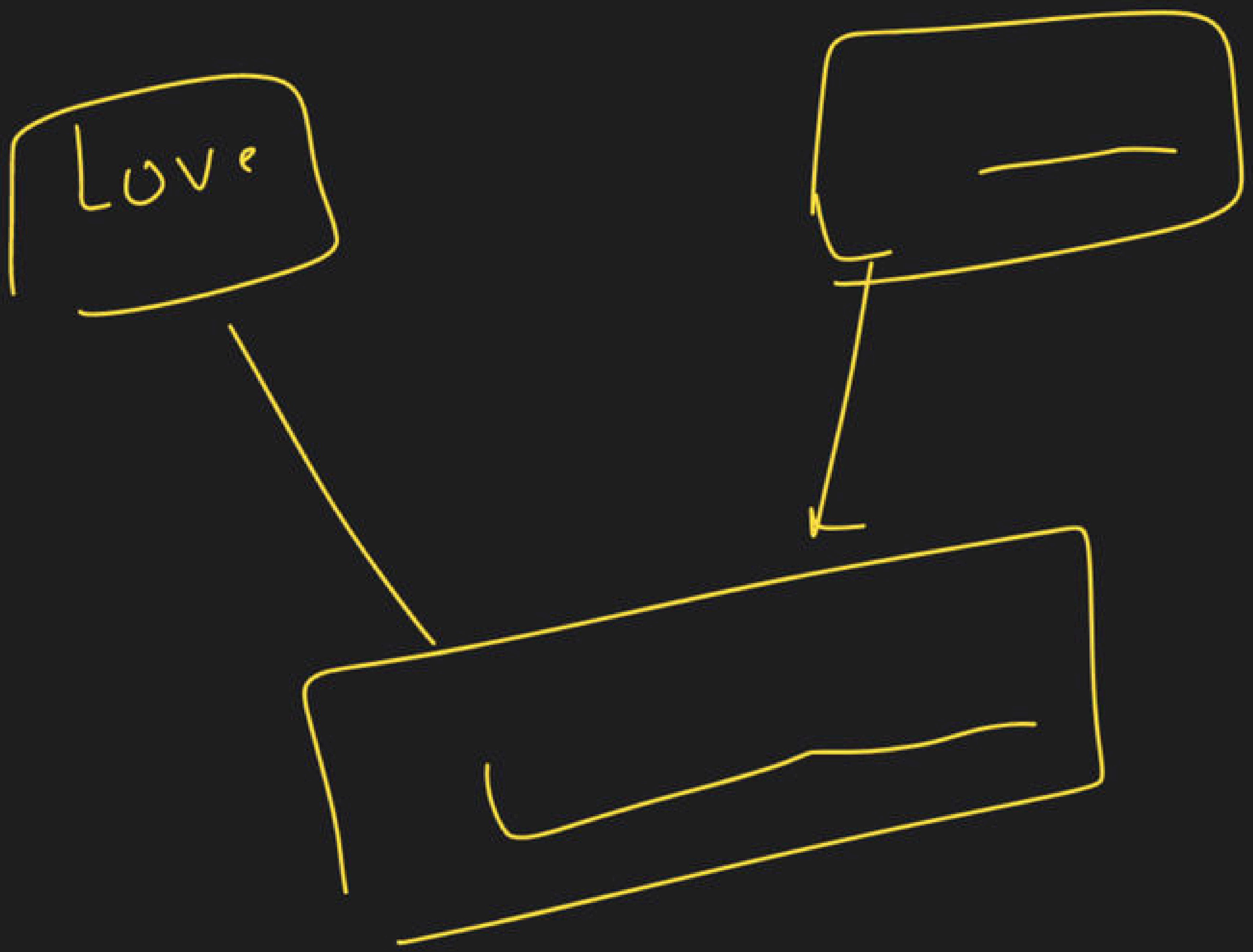
→ Multiple

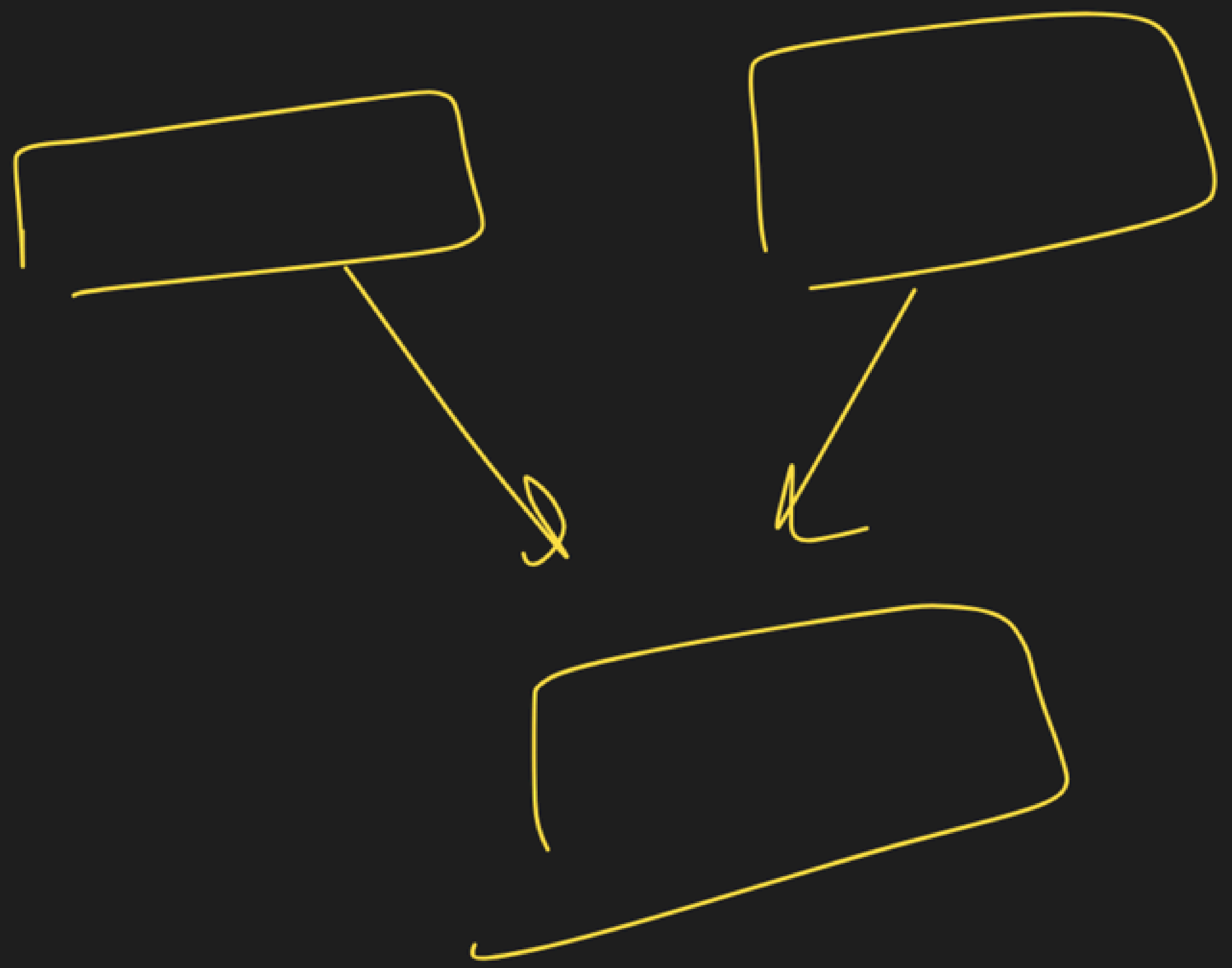


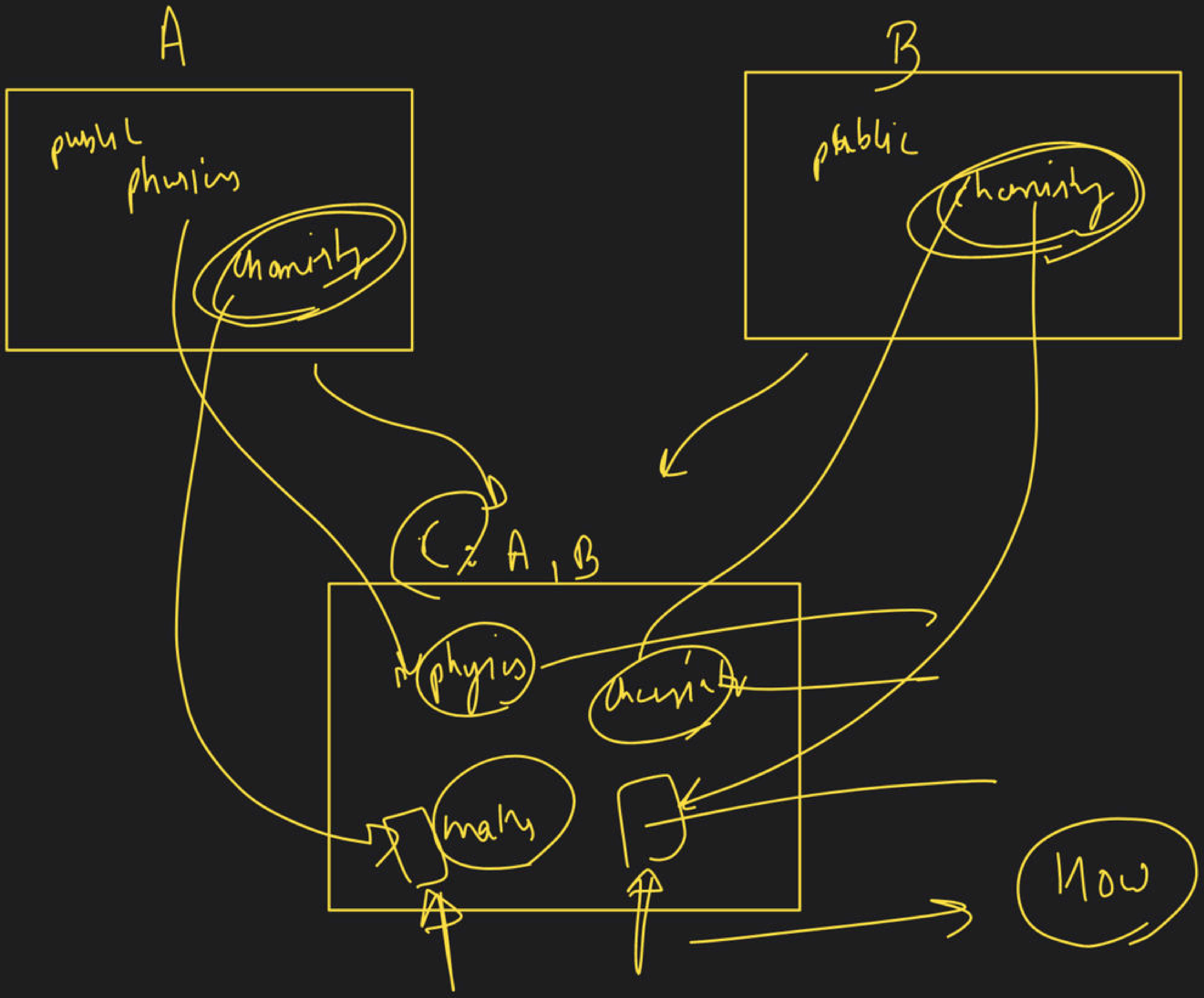


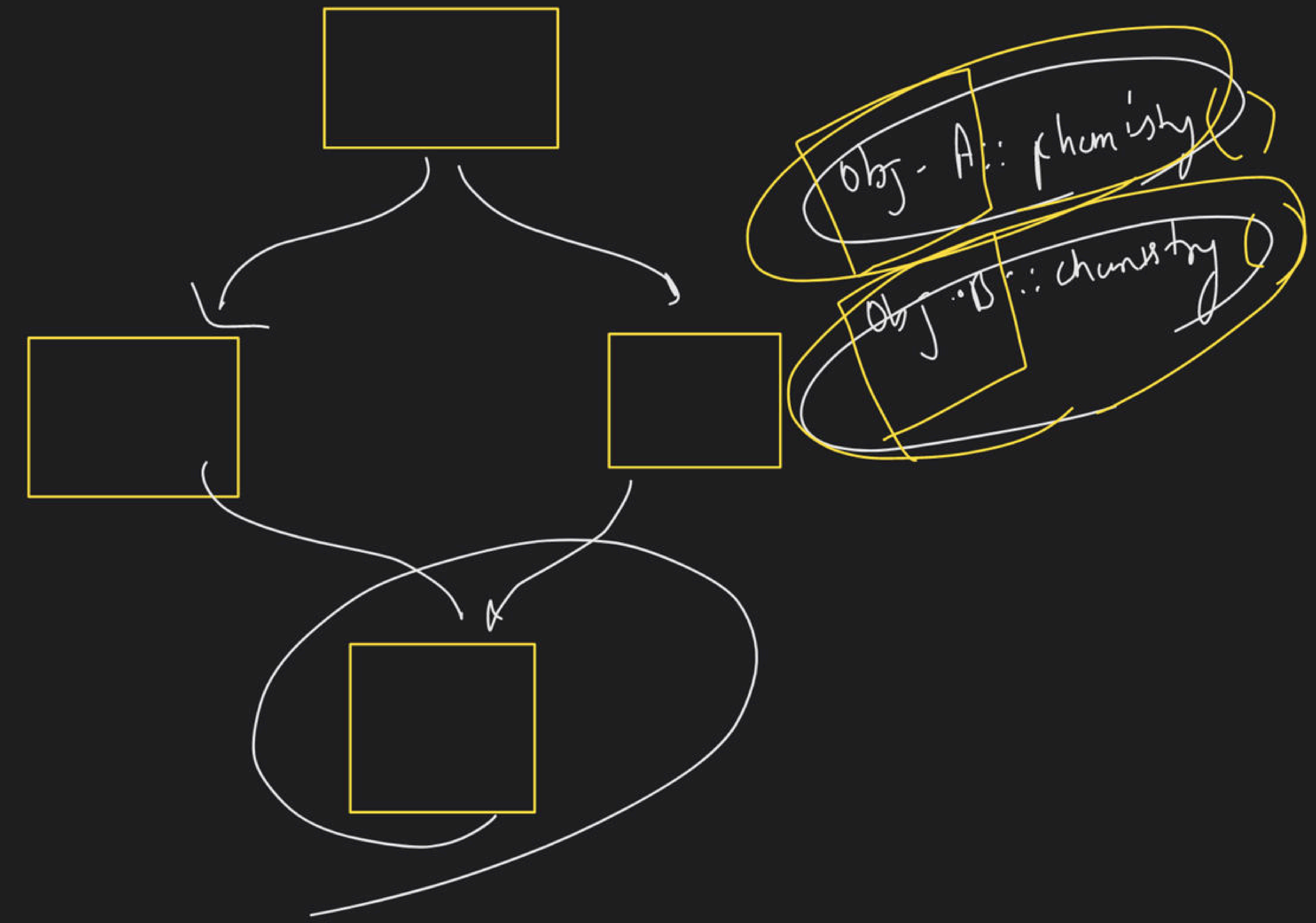



Brain Lara
Lara Dwta
Lara Lara


Love

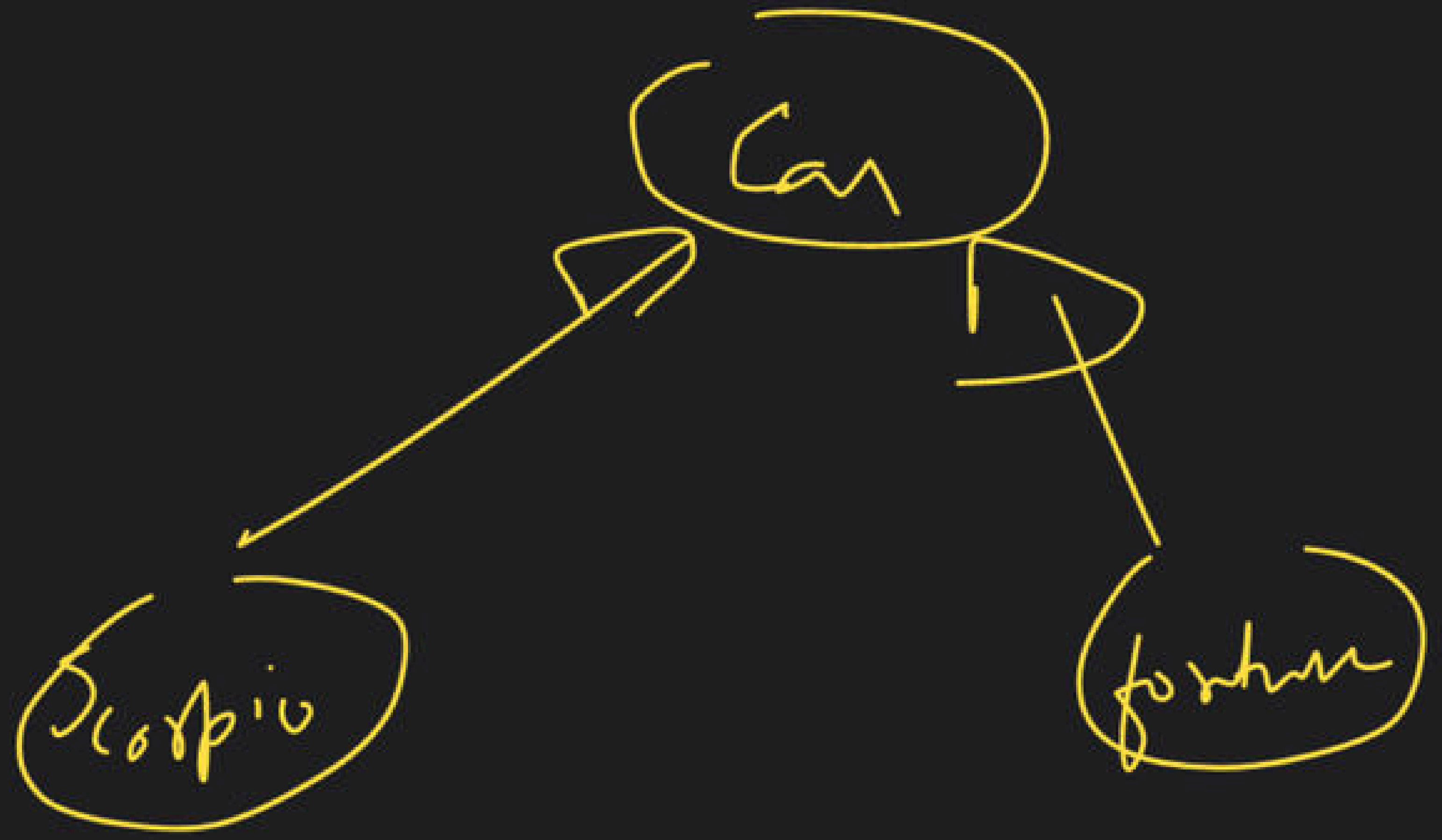



A
public
physics
chemistry
B
public
chemistry
C : A , B
physics
chemistry
maths
How



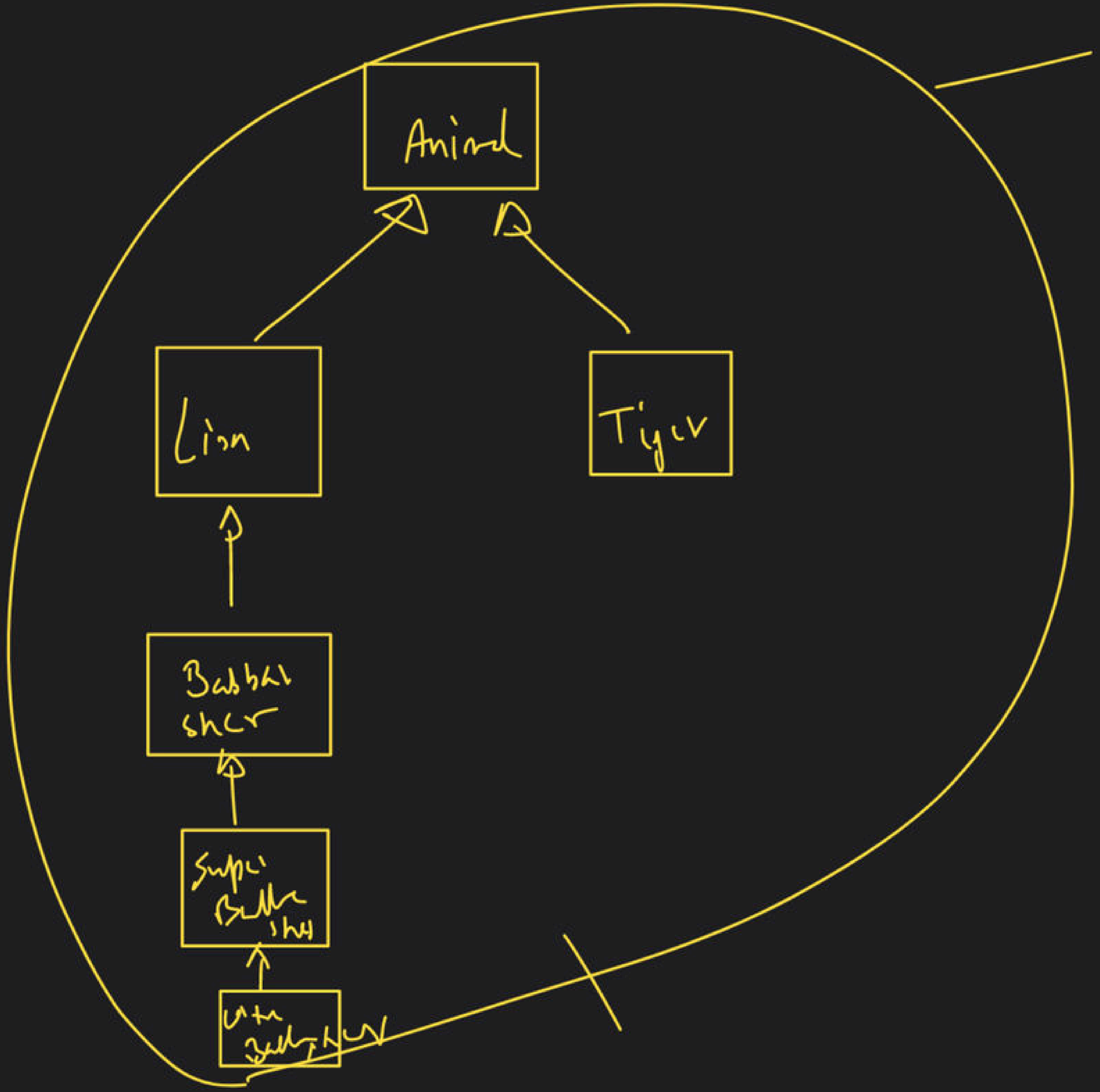
Obj - A :: chemistry ()
Obj . B :: chemistry ()

→ Heirarchical



Hybrid

Logical



2min

Polymorphism

Poly → many

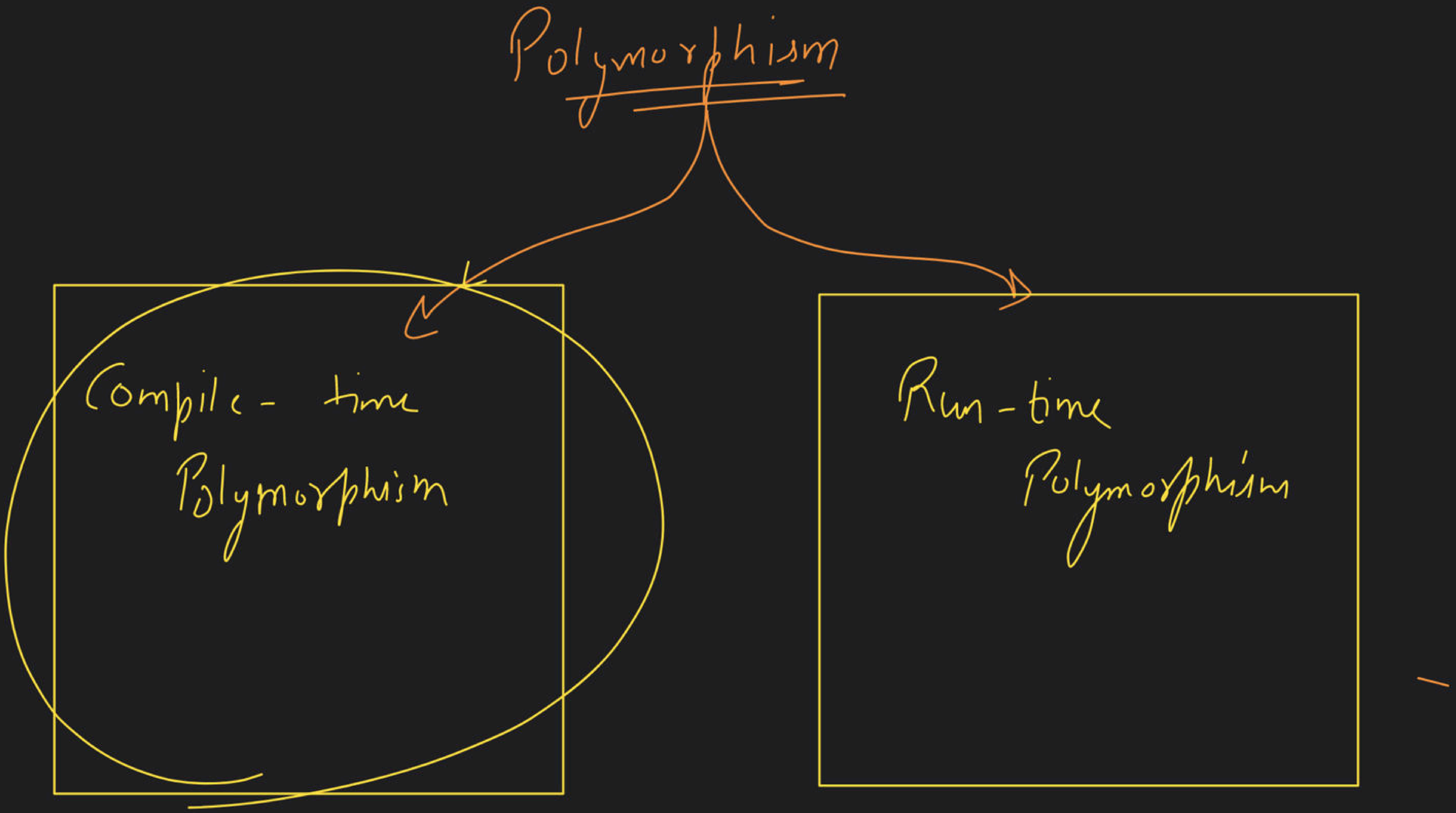
morph → form

→ existing in many forms

How?

```
int    print( )
 {
 }

int    print( )
 {
 }
```


Polymorphism
Compile-time Polymorphism
Run-time Polymorphism

$$\begin{tikzcd}
\text{Compile-time Polymorphism} \arrow[r] \arrow[d] & \text{function Overloading} \\
\text{Operator Overloading} \arrow[r] & \text{Operator} \overset{(+)}{} \arrow[d] \\
& \text{Overload}
\end{tikzcd}$$

obj → 5 value

print

Operator Overloading

obj.print()

overload

<< , >>

cout << obj; ✓

+ → sum
+ → concat

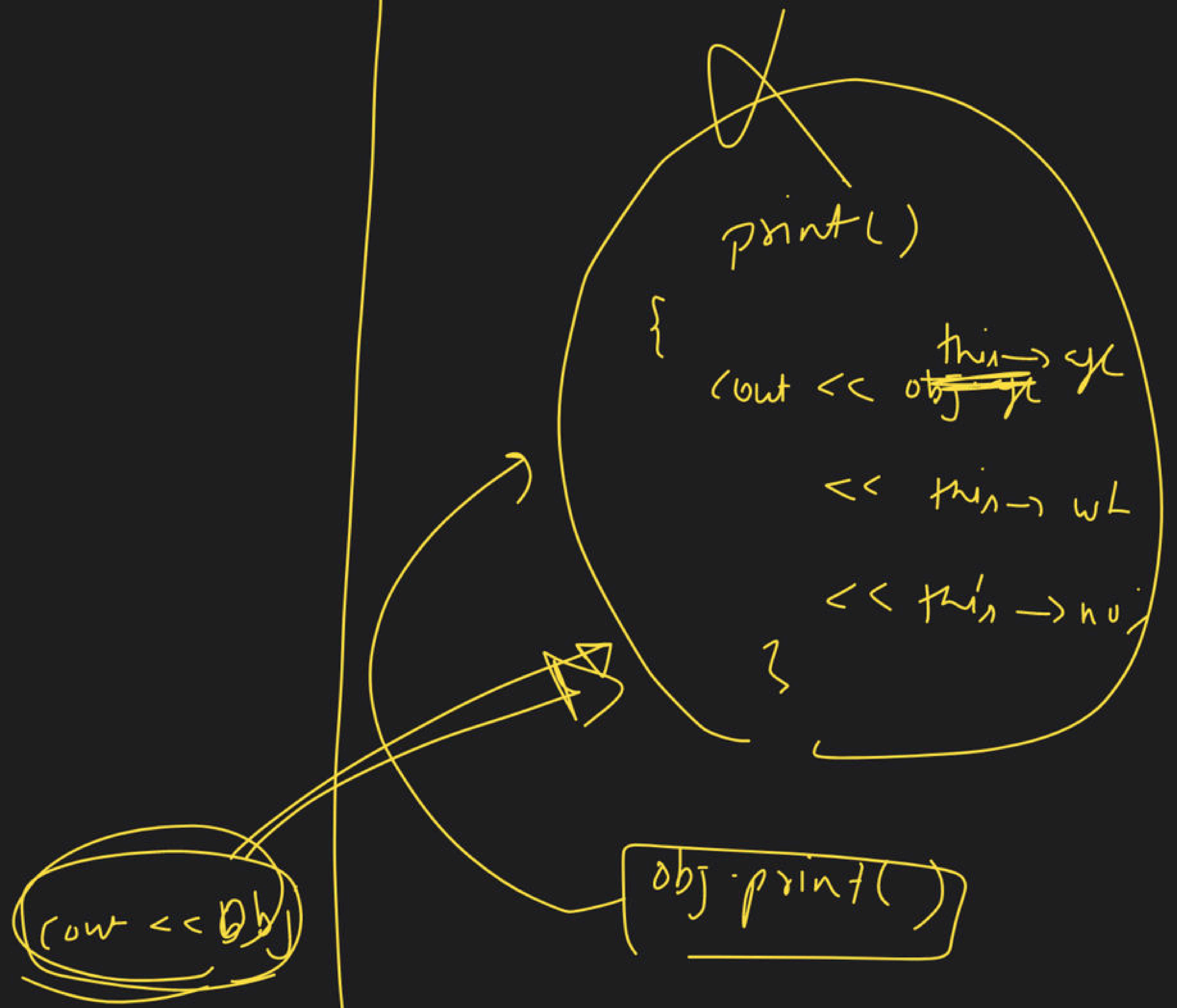
minus

H/W

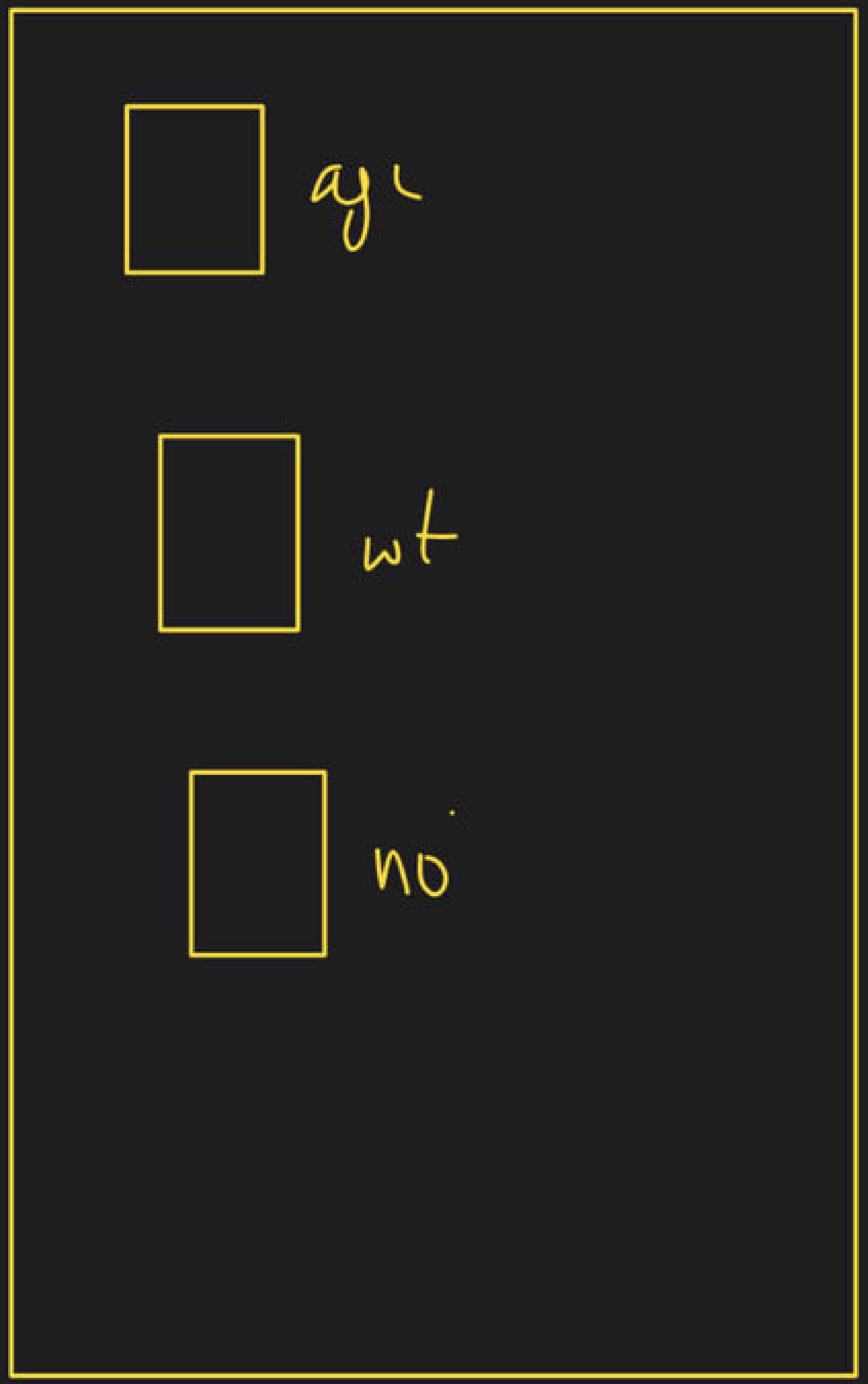
Syntax

```
return_type operator+ (     )
{
}
```

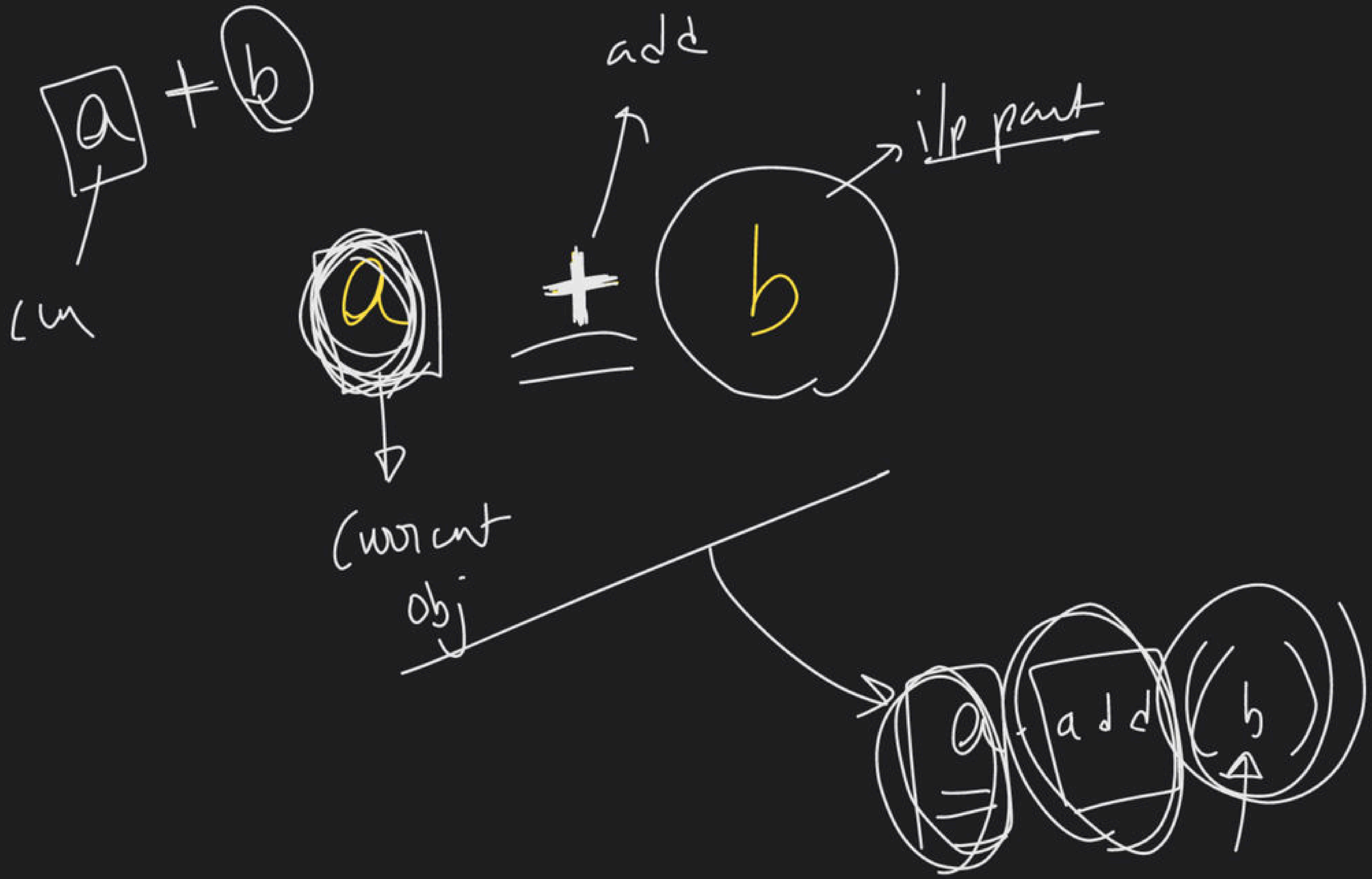
find out
↓
Google
↓
Kon Kon se Operator overload Karna allowed hai



print()
{
cout << ~~obj.age~~ this -> age
<< this -> wt
<< this -> no;
}
obj.print()
cout << obj



age
wt
no



a + b
cur
add
i/p part
a + b
current obj
a . add (b)

$$\begin{tikzcd}
 & \text{M.F} & \\
a \arrow[d] & . \ \text{add} \arrow[u] & (b) \arrow[d] \\
\text{current obj} & & \text{i/p param}
\end{tikzcd}$$

$$a \underset{=}{+} b$$

a → current obj

+ → function call → M.F

b → i/p → parameter