OOPs Concept - Part I

Course on OOPs Basics

Love Babbar • Lesson 1 • May 15, 2022

→ what -> ?

→ why
- → Interview
- → SDE -1 -> Expectation
  - 90% -> Task xyz -> Implement -> Code

→ How to study -?
- → interactive [Code -> Ans]
  - HW -> solve
- → Discord
- → Exploration / Spoon-feeding

D]

SDE-1 =
SDE-2 =

~~SDE~~

System Design

OOPS
↓
? X

Object Oriented Design

Low Level Design
- Machine Coding
- Rounds

→ what is OOP ? → Procedural programming ?

Programming tech

class, Object

4 pillars

- Inheritance
- Polymorphism
- Encapsulation
- Abstraction

functions

6 mons

→ Class (Blueprint)

Babbar abc = "501"

int, string

Enum map

design / type → state / property / field / data member

→ behaviour / function / data function

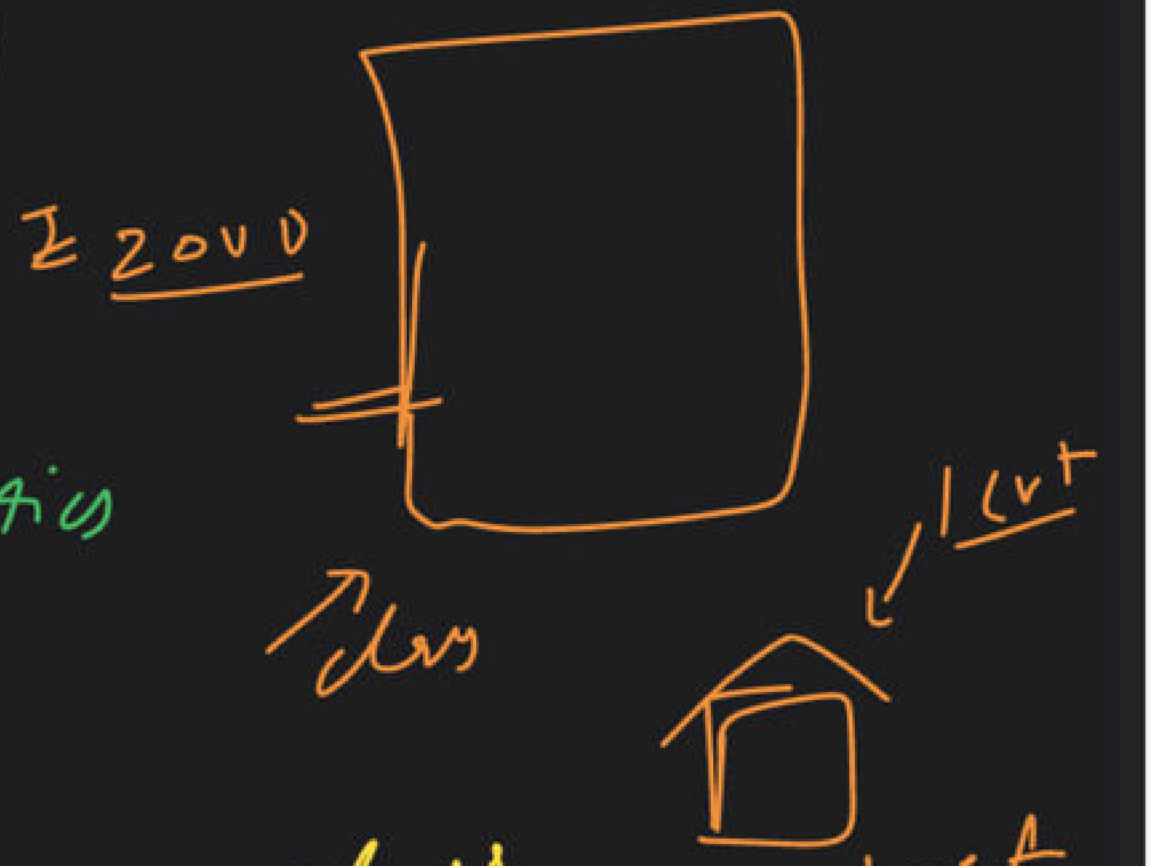
template

Ex → Humans → 2 Hand, 2 Leg, 2 eyes } → state/properties

↓

func / behavior

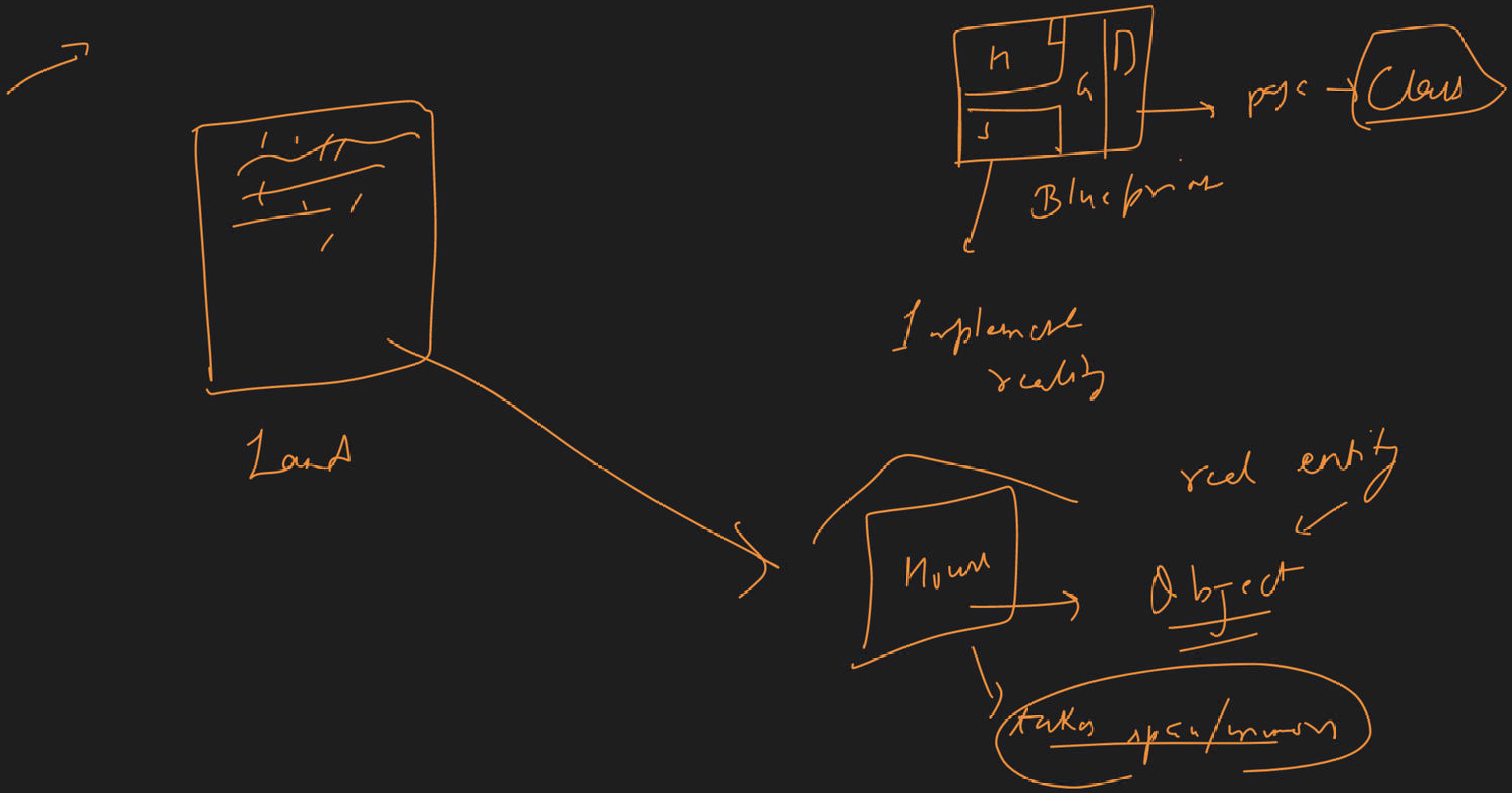
walk, eat, speak,



Human → class

→ state → hand, leg, eye

→ behaviour → walk(), eat(), speak()


H
G
D
S
Blueprint
prog
Class
Implement
reality
Land
real entity
House
Object
takes space/memory

-> class

-> object instantiate

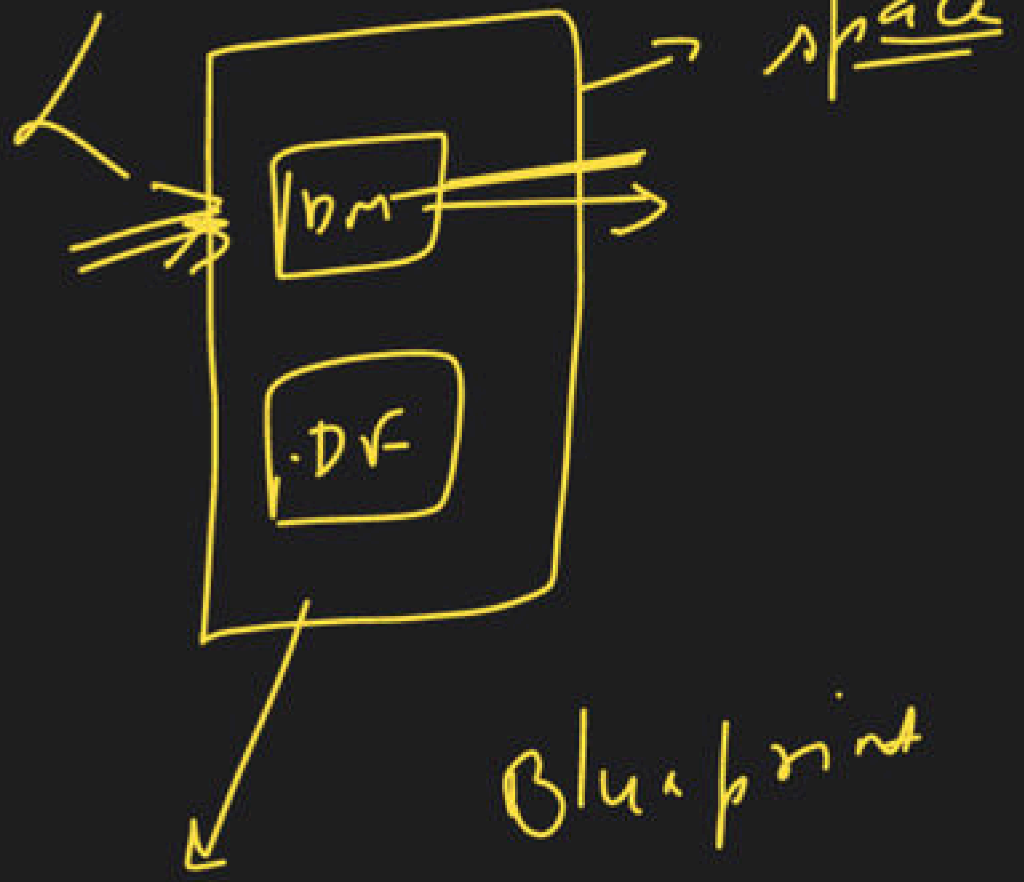
-> data member

-> data function

access

u
&

class don't have memory ?



Blueprint

logical component

age
wuj

Object
hus
int
wuj

Object is the instance of class

→ Constructor

Stack

Heap

stack

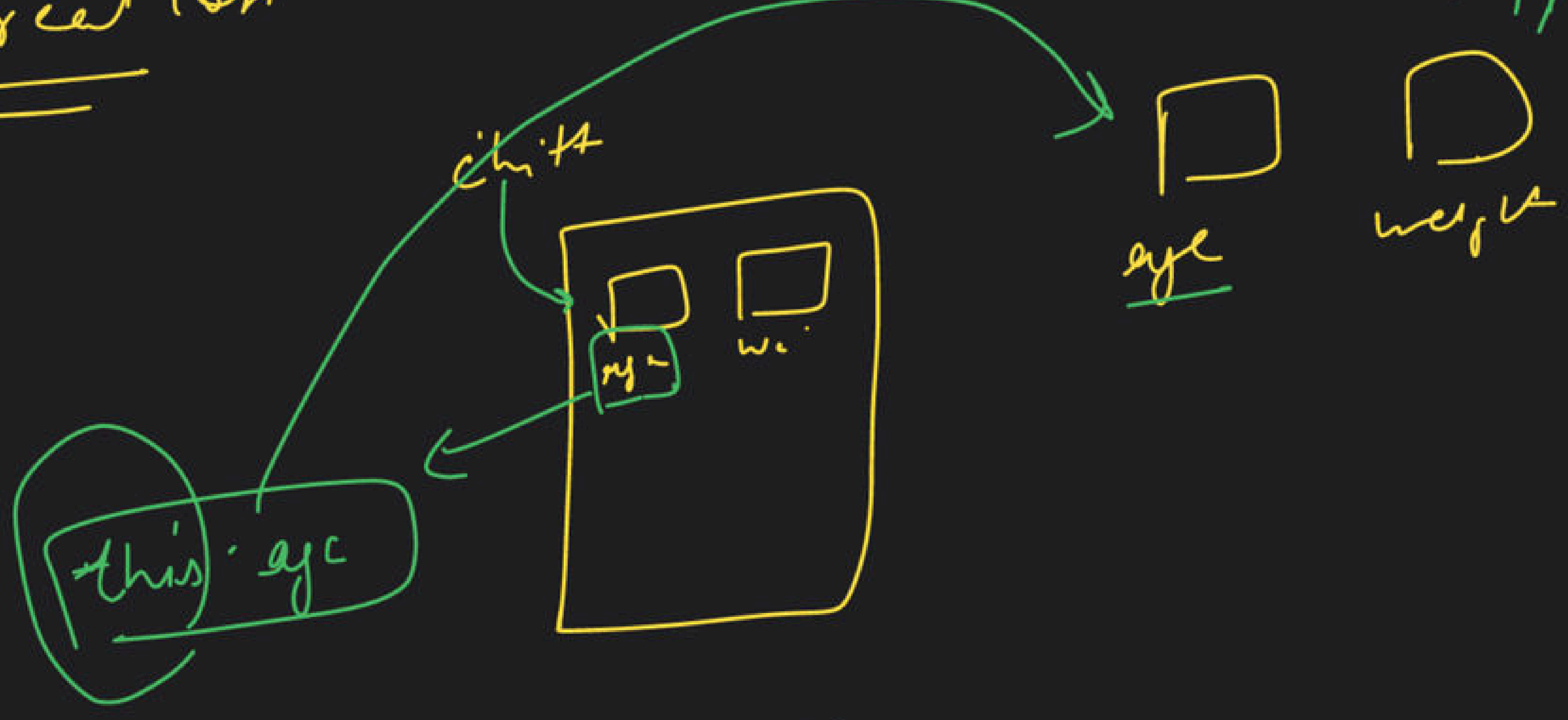
Heap area

1/0K

→ constructor :- (initialisation)

↳ object creation

| private Human → } → H/W



age = age

this.age = age

Rules

↳ no return type

↳ name → same class name

↳ abstract / static / final / synch

→ Constructor Overloading

→ Diff → Constructor vs Method

	Constructor	Method
1	N. r T	R T
2	same name → class	∝
3	~~∝~~	static

→ Copy constructor → Yes or NO

flex

- how →?
- → constructor
- → assign value
- → clone → object cloning → how

obj 1

obj 2 → □

obj2.age = obj1.age

obj2.w = obj1.w

→ Do constructor return any value? → Explore

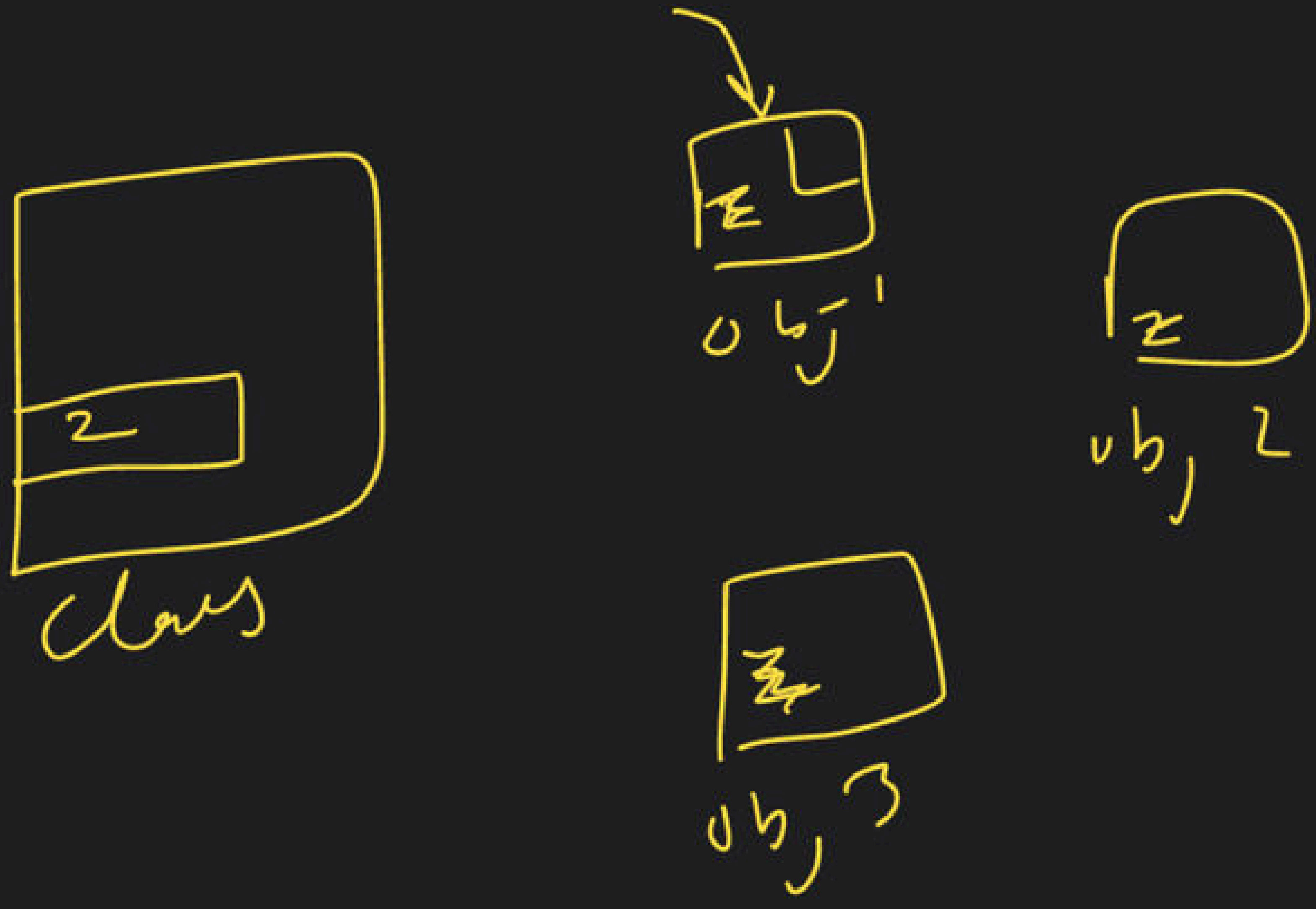
→ static keyword ?

static data member

obj ✗

class se belong

significance



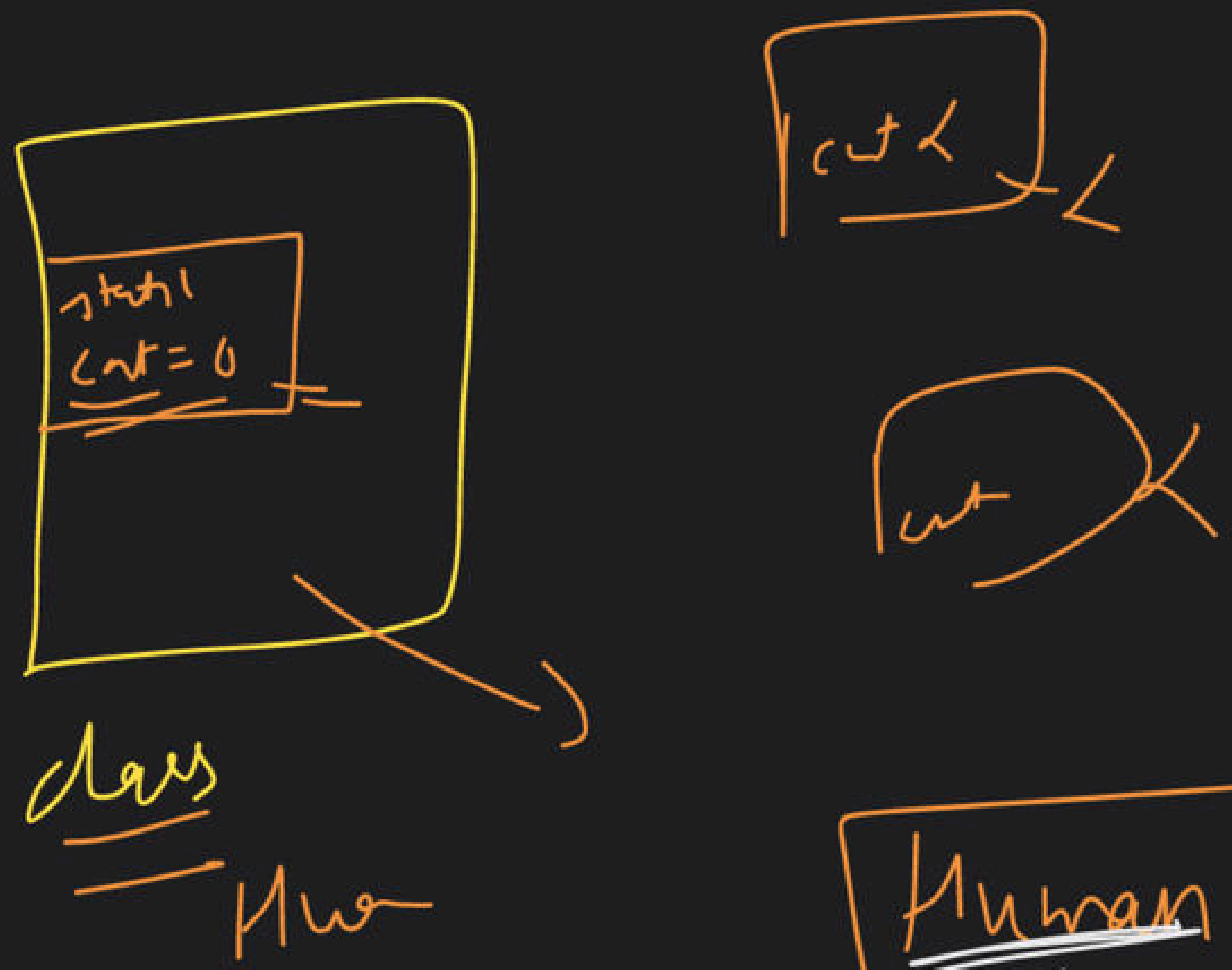
1000 obj

4 byte

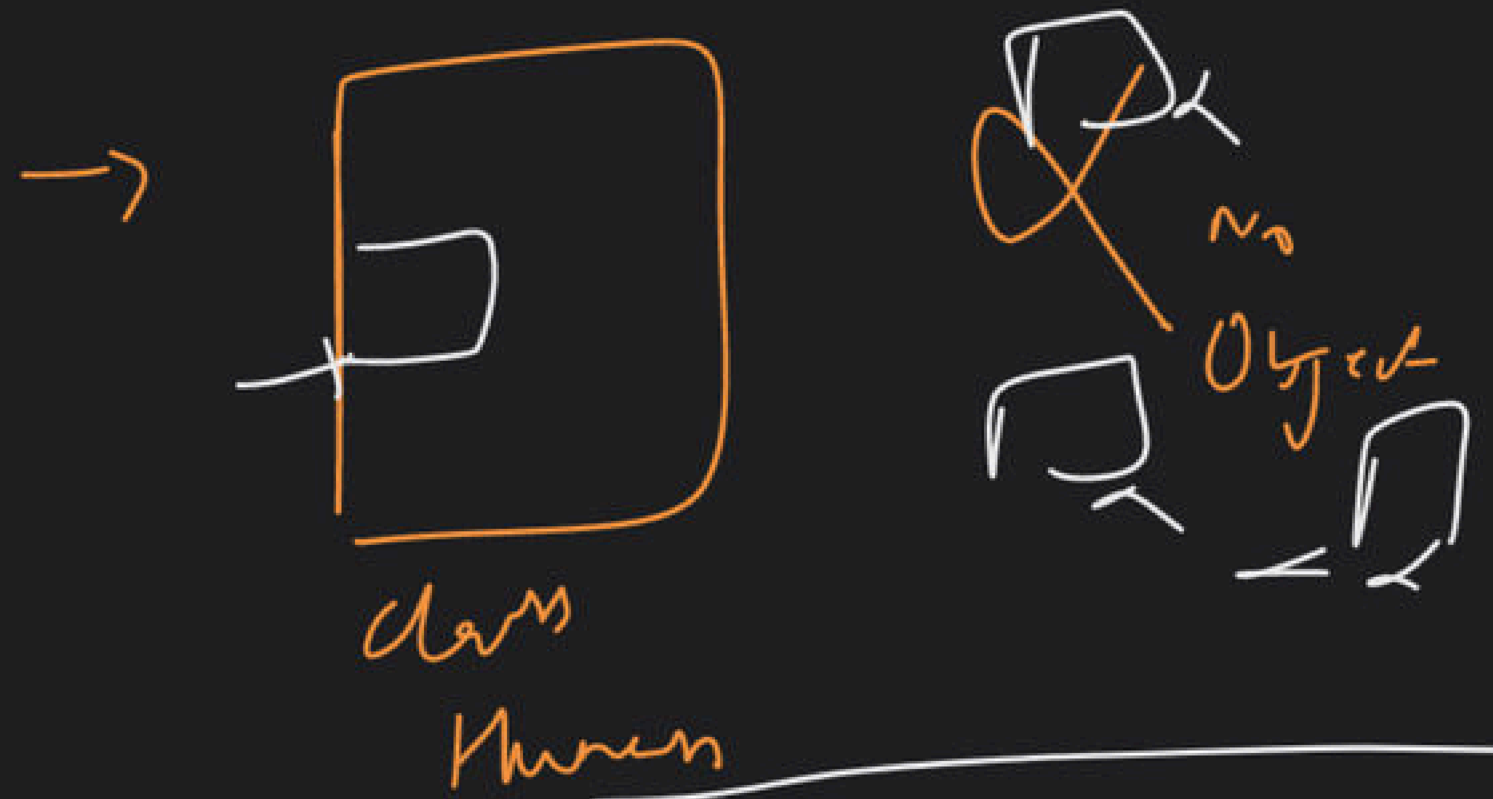
2 × 1000

4 byte → 4000 Byte

static DM:-



Human. count → access



→ static member access without making object

Yes

static DM → one for all objects

→ represent class property

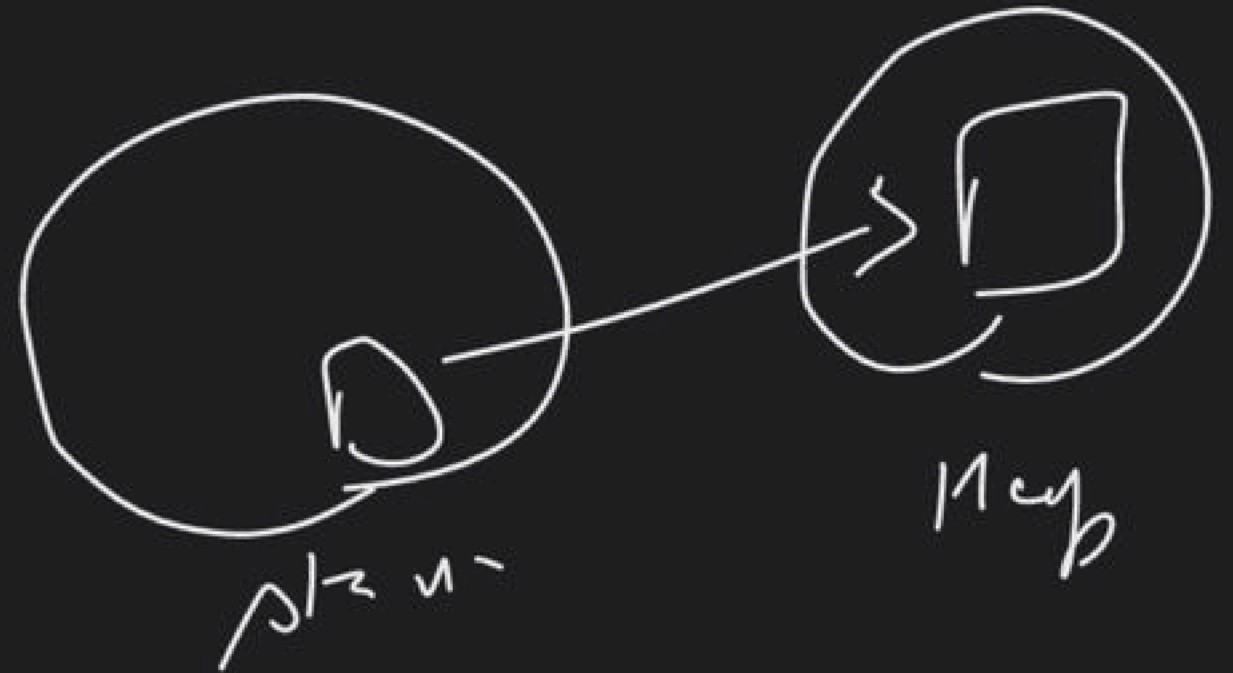
→ require less space

Human obj = new Human();

---

static method :-

- → belongs to class rather than obj
- → invoke without creating object
- → SIB → static DM
  - → value change



static func → static DM (change) Yes

→ Non static DM (change) No

what will be the o/p of this p/g ?

→ static

why main method is static ?

→ memory

This()
Super()

→ this Keyword

- → DM → this.age;
- → DF → this.sleep();
- → Constructor → this(); →

(3)

this(___ , ___) → Explicit

P

Co

↓

LB

→ Inheritance → is-a relationship

super

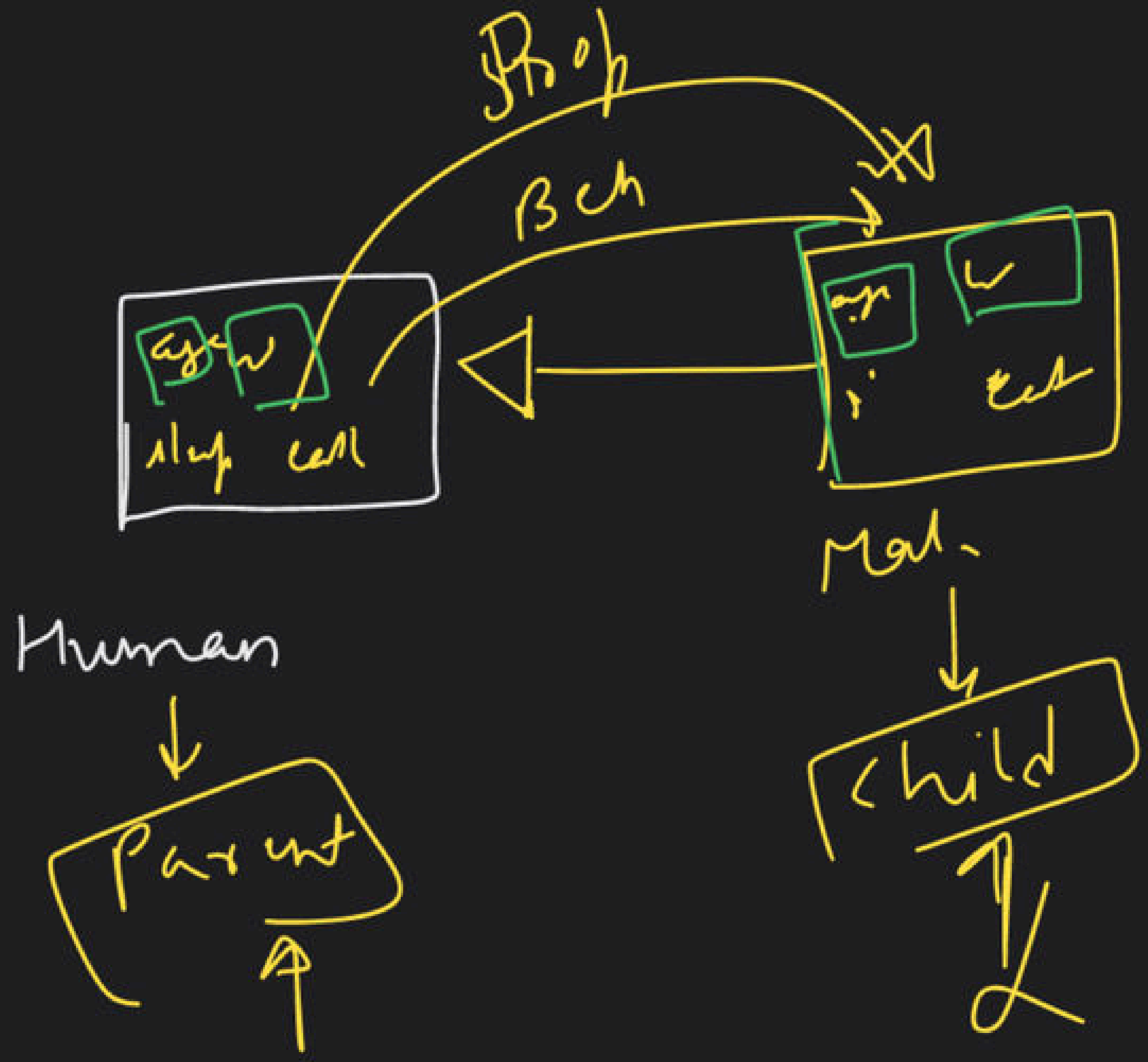
child — Male is a Human

Female is a Human

2hr

↓ 4hr

2.5 hr



Babbar is a Teacher

Babar

Parent class/super class

child class/sub-class

Adv:-

→ Reusability

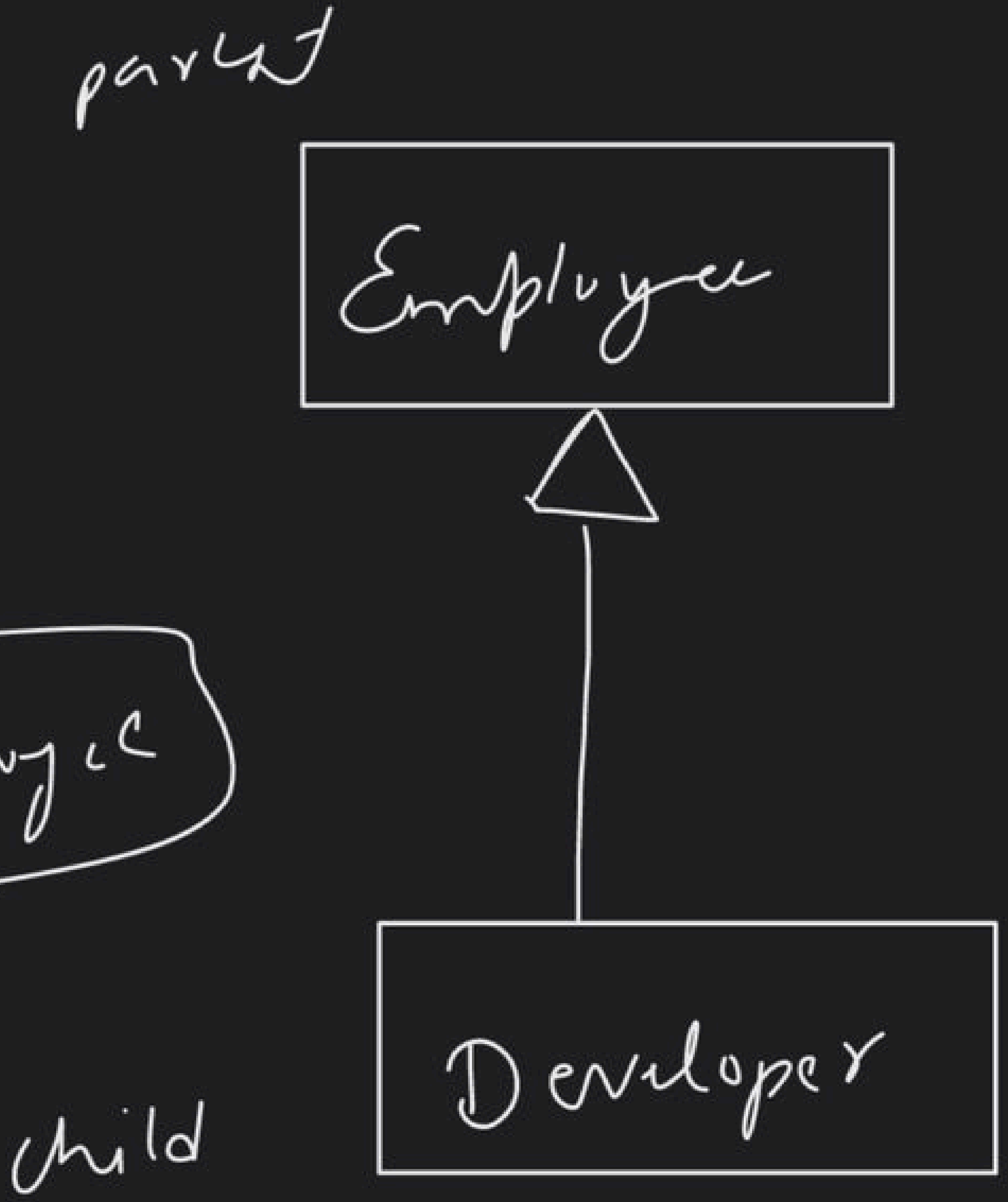
Goldy is a Dog

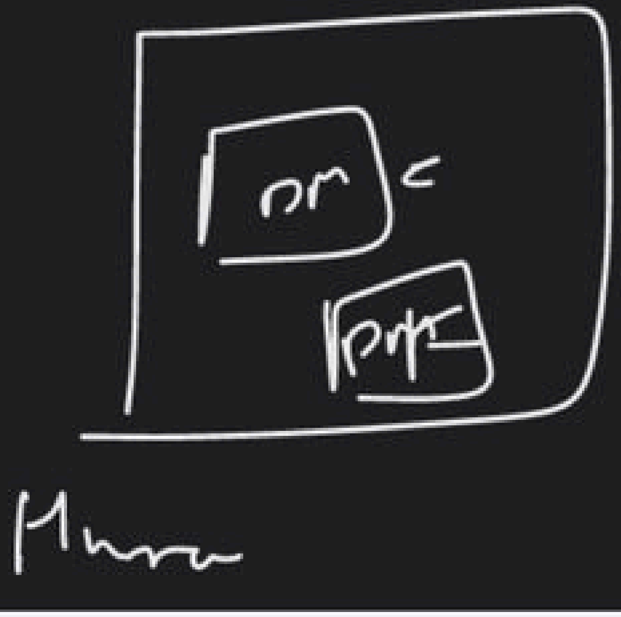
mercedes is a Car

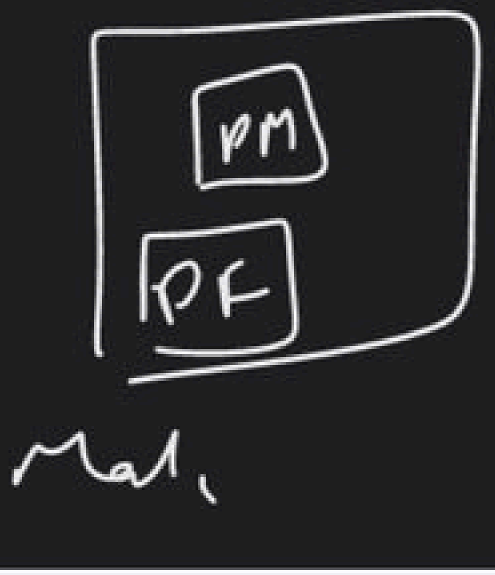
→ **Types**

① Single

Developer is a Employee







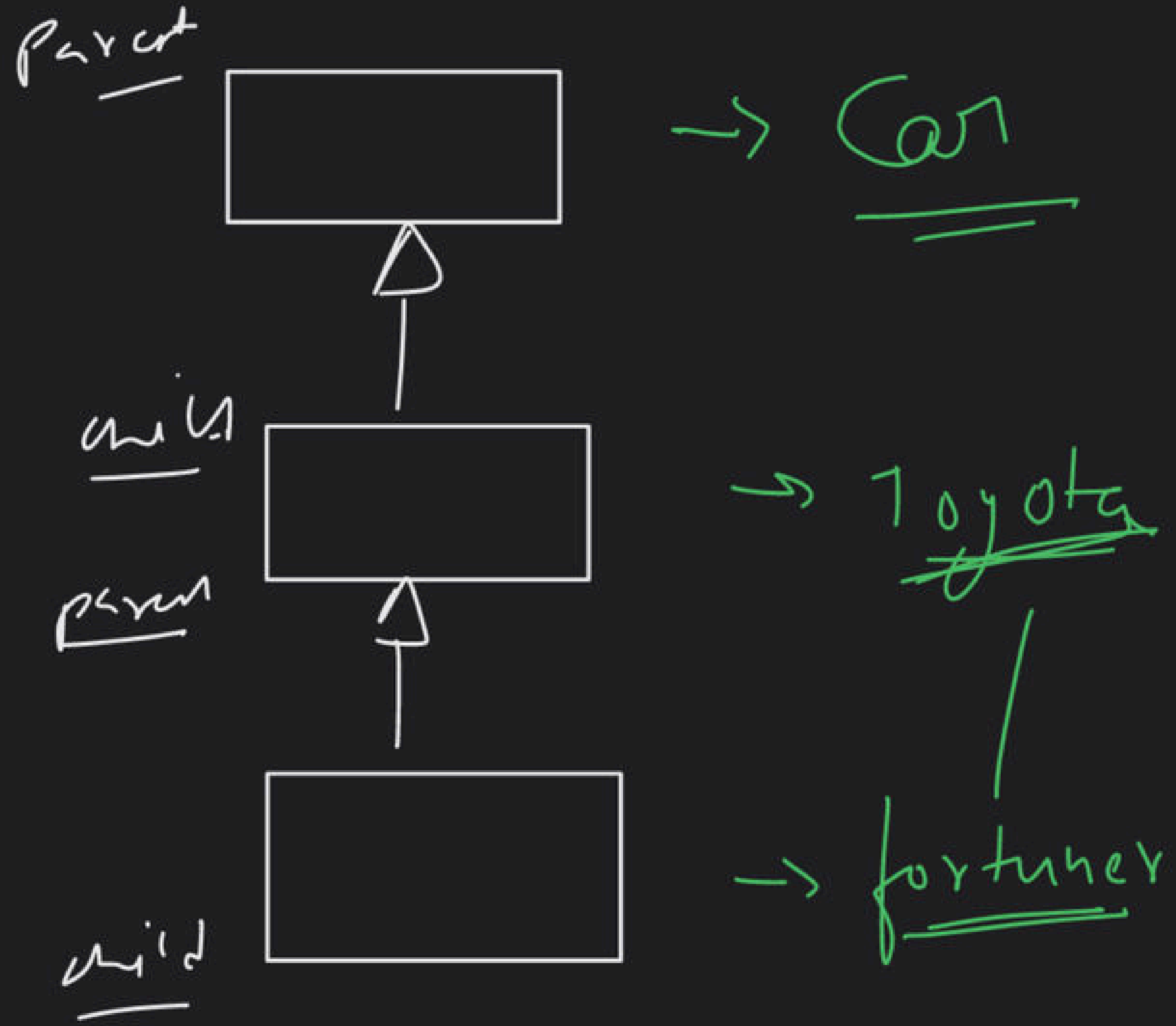
Male extends Human

② Multi-level

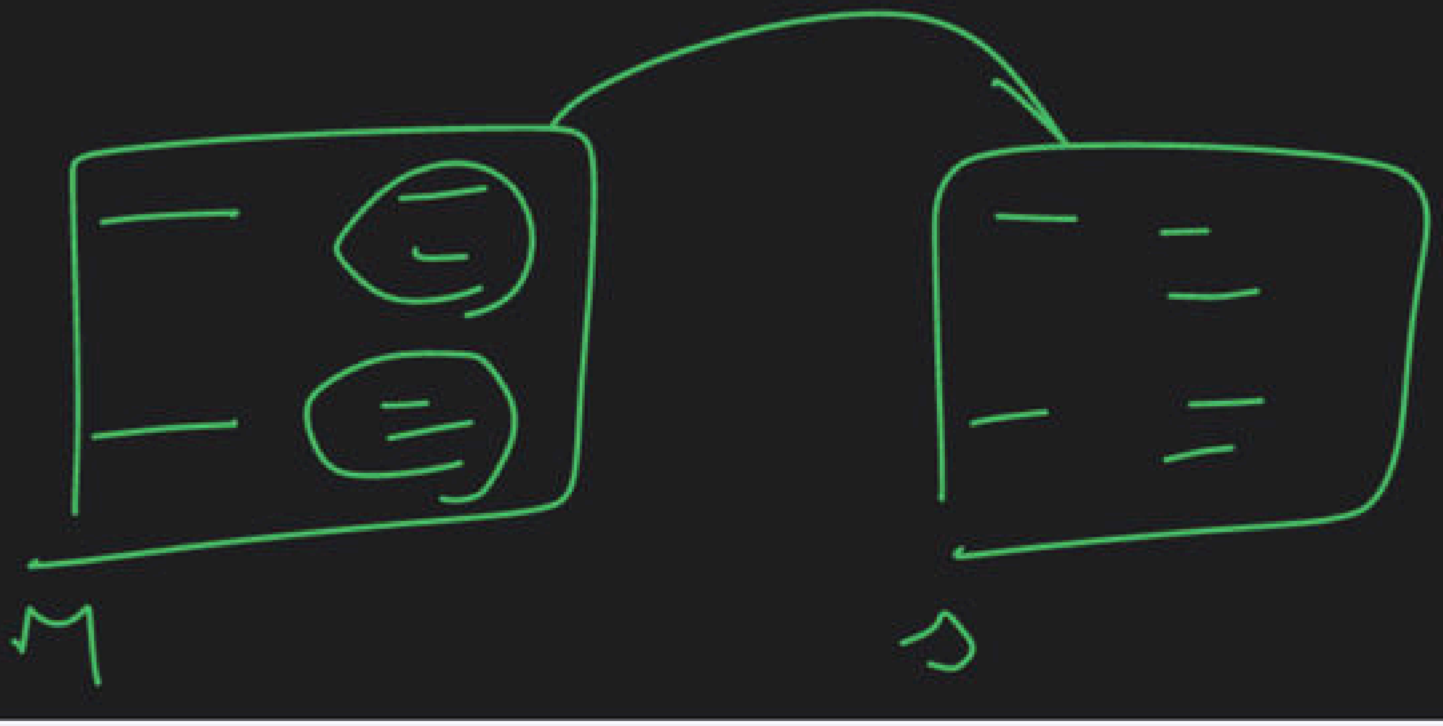
Car

mahindra

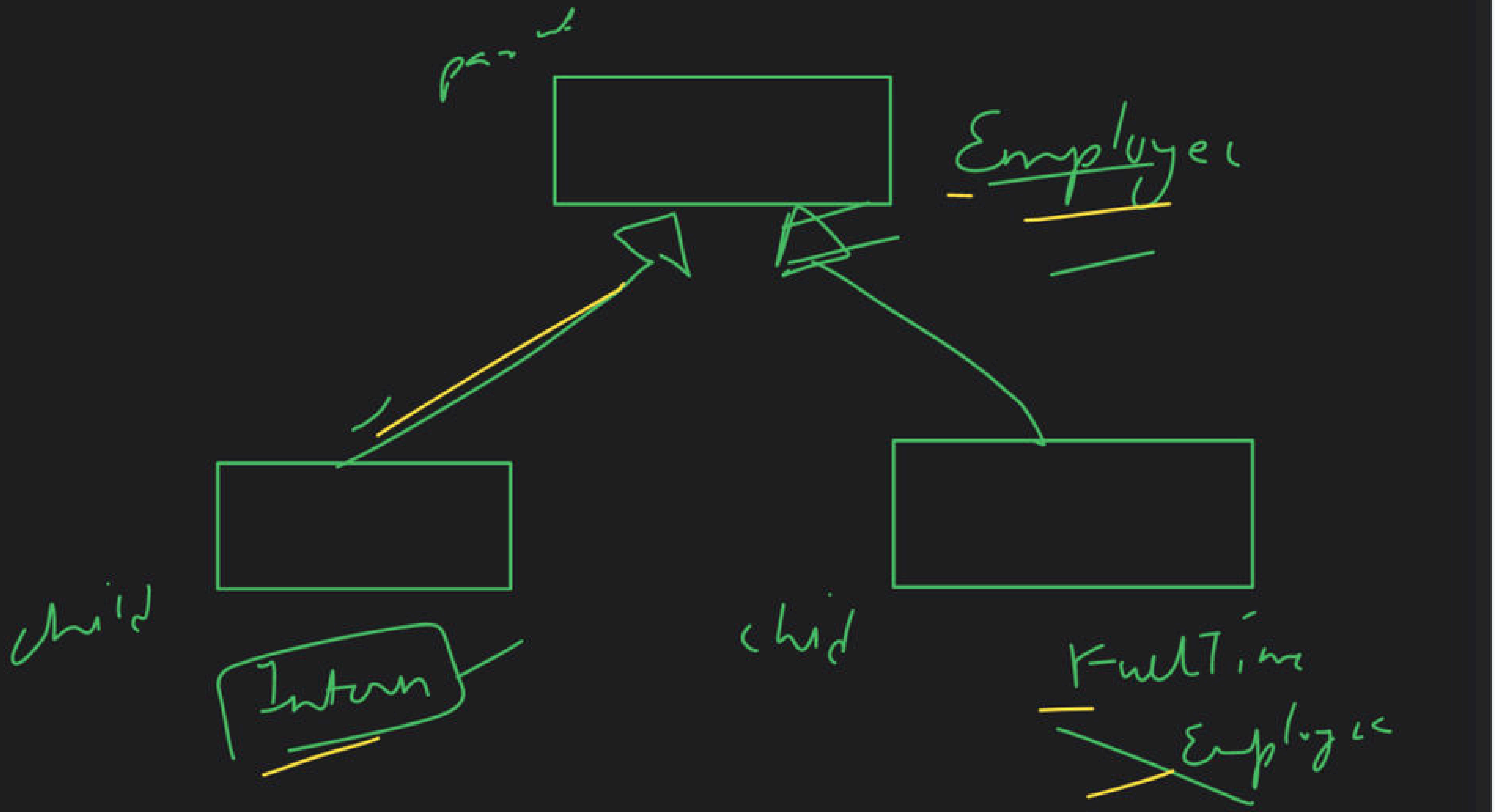
THAR / Scorpio



Car



③ Heirarchical



```
class Employee
{
}
```

```
class Intern extends Employee
{
}
```

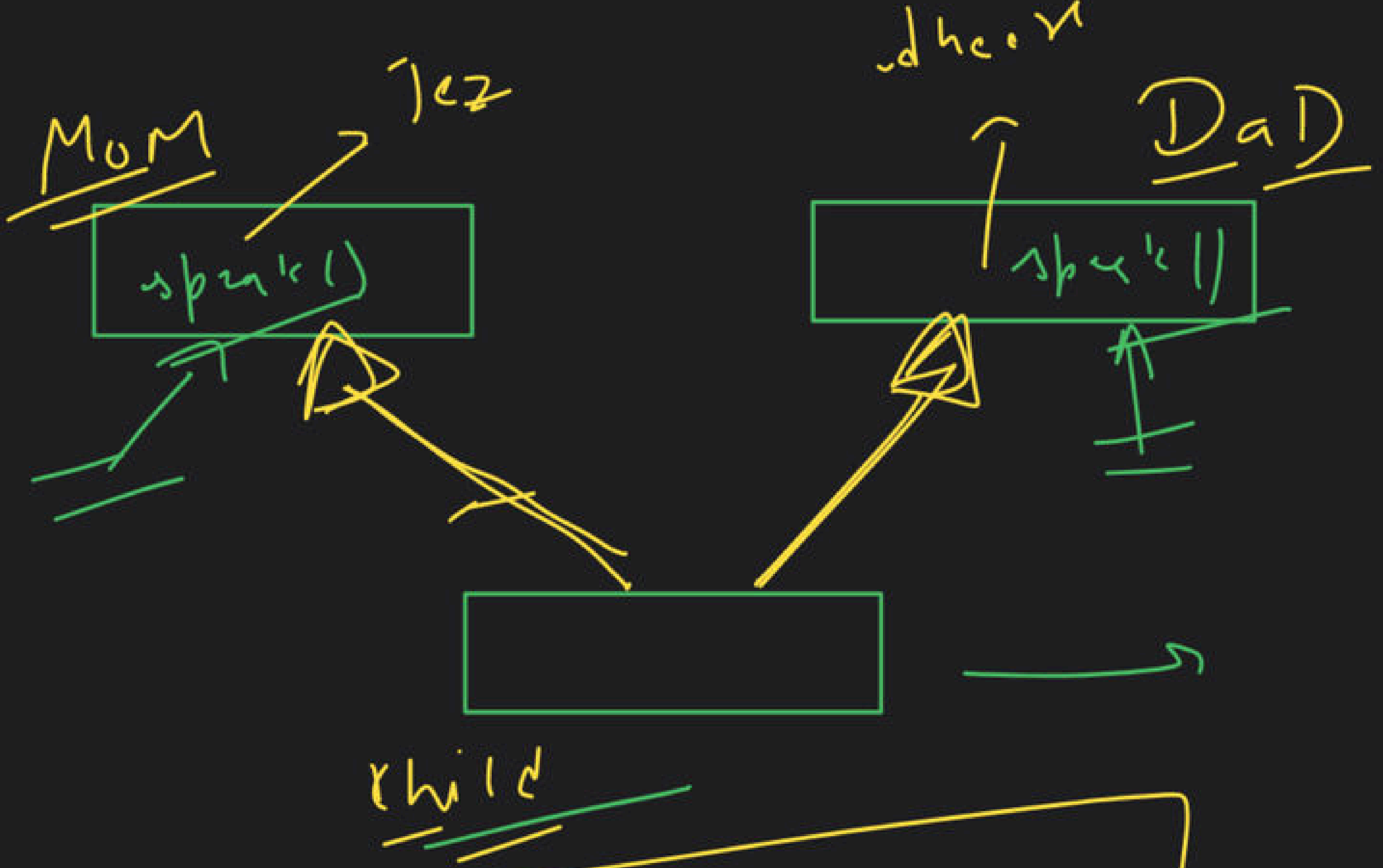
KHUD

```
class FTE extends Employe
{
}
```

(2) Multiple, Hybrid

→ Diamond problem

↳ explore



→ child.speak()

child is a mom

child is a DaD

→ ambigous

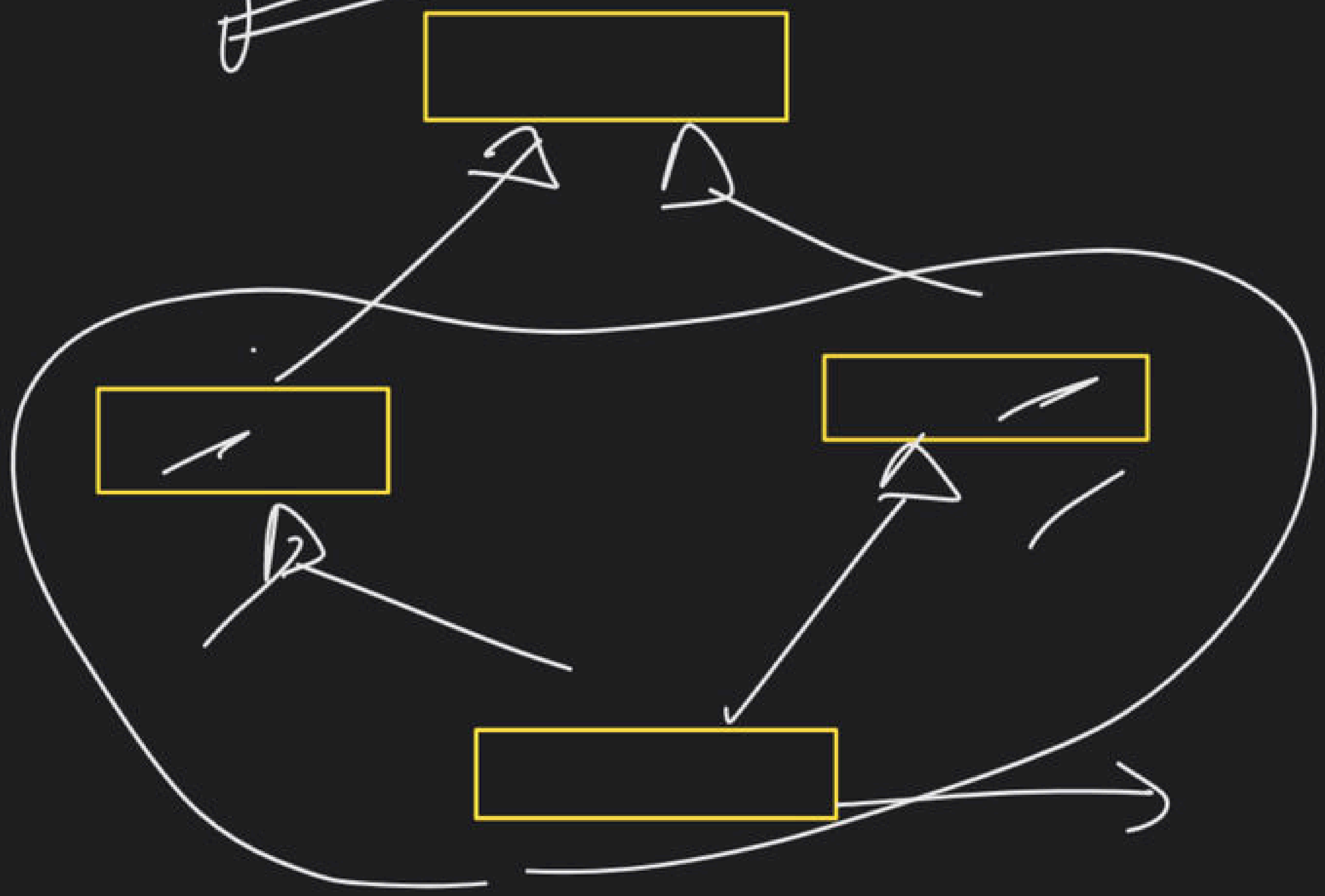
→ cpp → multiple

→ how?

→ explore

→ mi

Hybrid



30 min

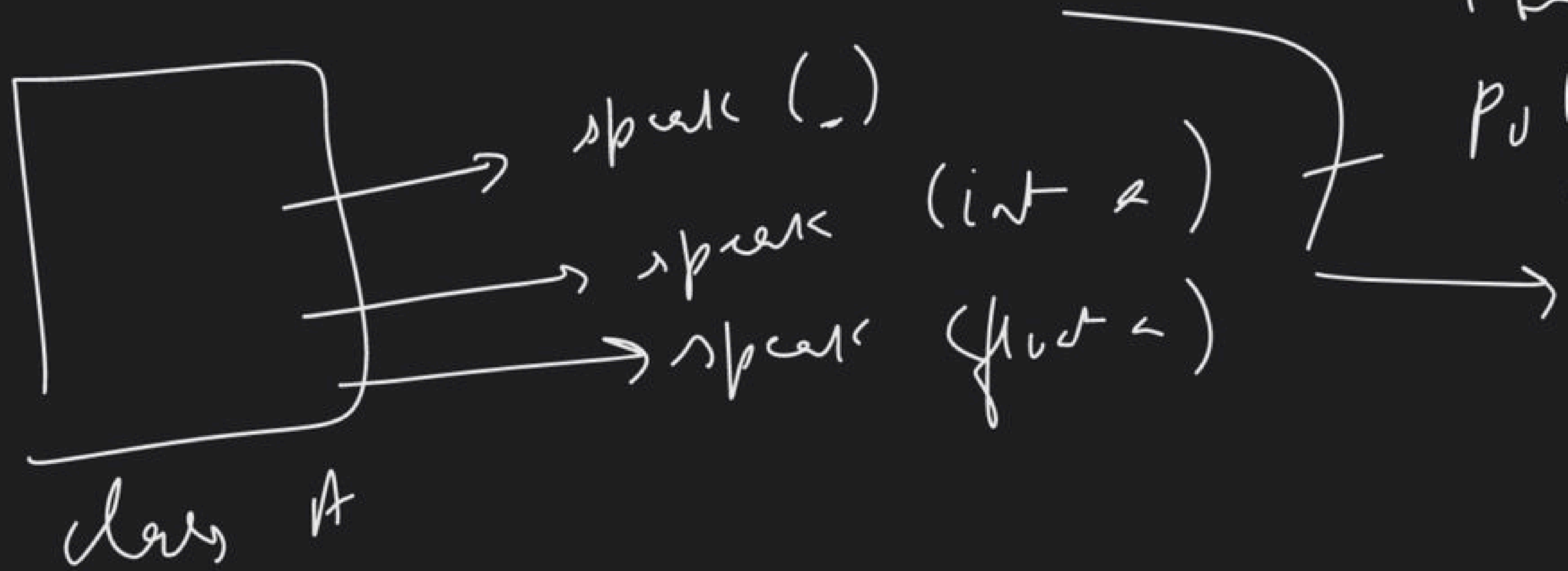
Adv:-

Reusability

2nd Pillar → Polymorphism → existing in many forms

Poly ↓ many

morph ↓ forms



Compile time Poly-

Method Overloading

→ Method Overload

→ Compile time polymorphism

```
Calculate {

    int add (int a, int b)
        { return a+b; }

    int add (int a, int b, int c)
        { return a+b+c; }

}
```

→ Can we overload main method?

No → ~~explore~~

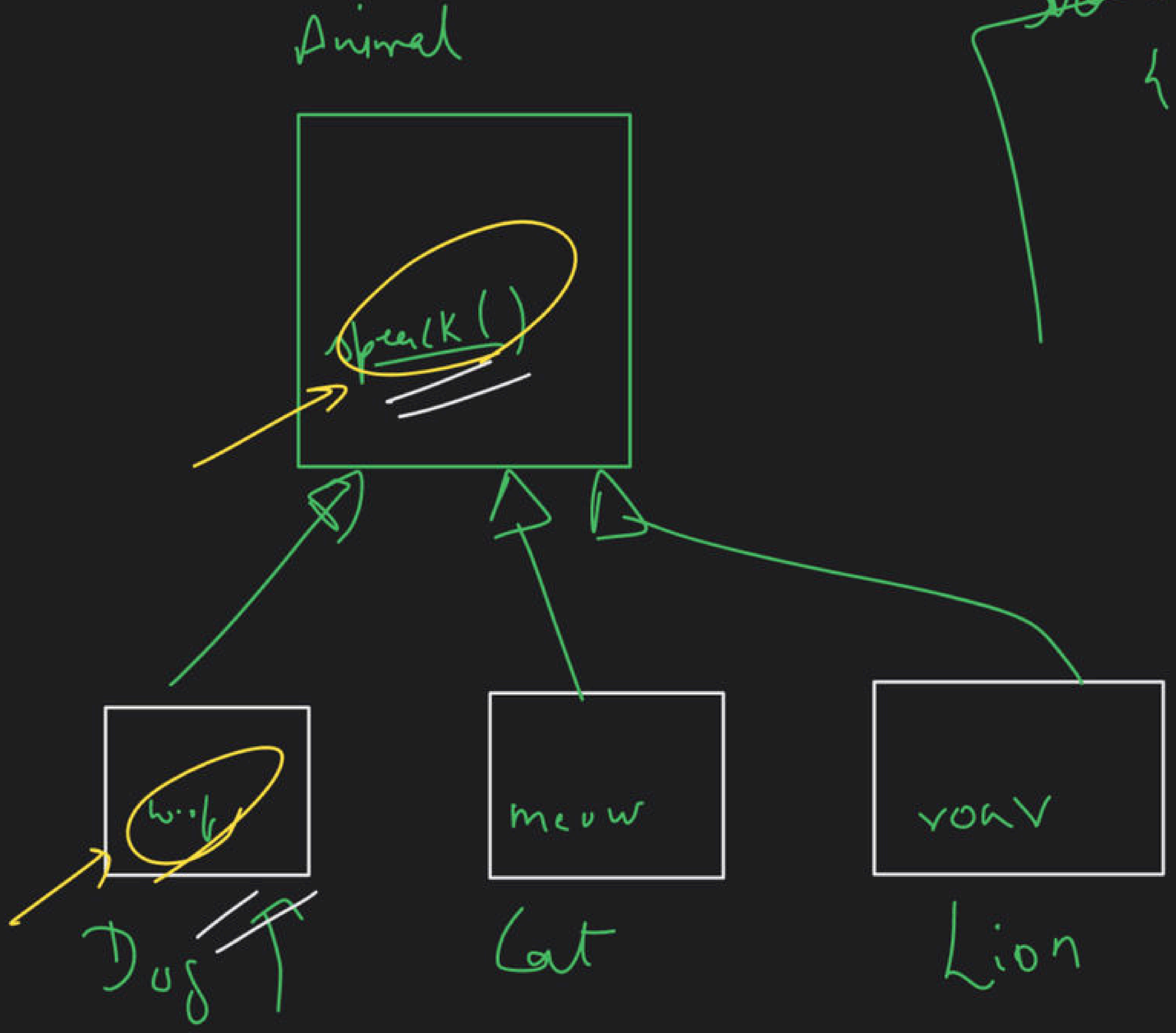
→ Method Overriding → Run time Polymorphism

Animal

Dog → woof

Cat → meow

Lion → roar



```
~~void~~ speak()
{
sout("Maaye")
}
```

2hr +11min

↳ Done → T

N - F

Method override

Runtime

Runtime Polymorphism

why →

→ Rules

→ same func name / same parameter

→ must do inheritance

→ Can we override static method?

↓ NO → why?

H/W → Explore

→ Overloading vs Overriding → H/W

Assignment

- Cohesion
- Coupling
- Association
- Aggregation
- Composition

Access modifier

Encap

Abstr

OOPS → I

Imt vs → 5 → 1hr