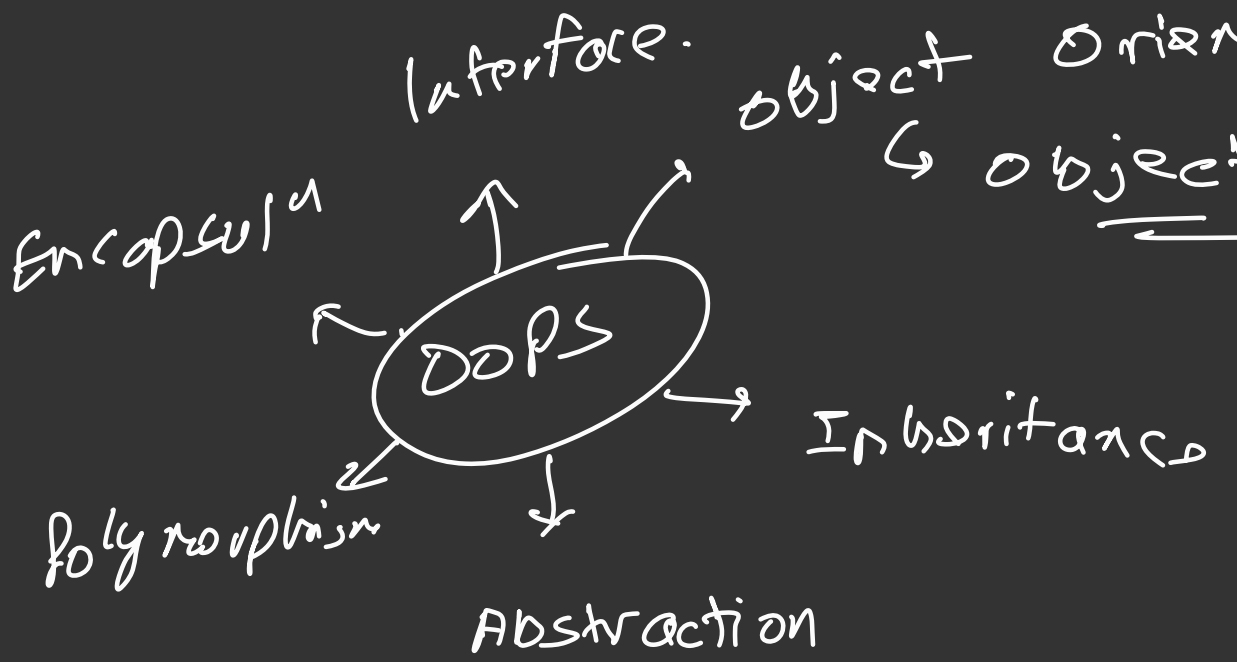
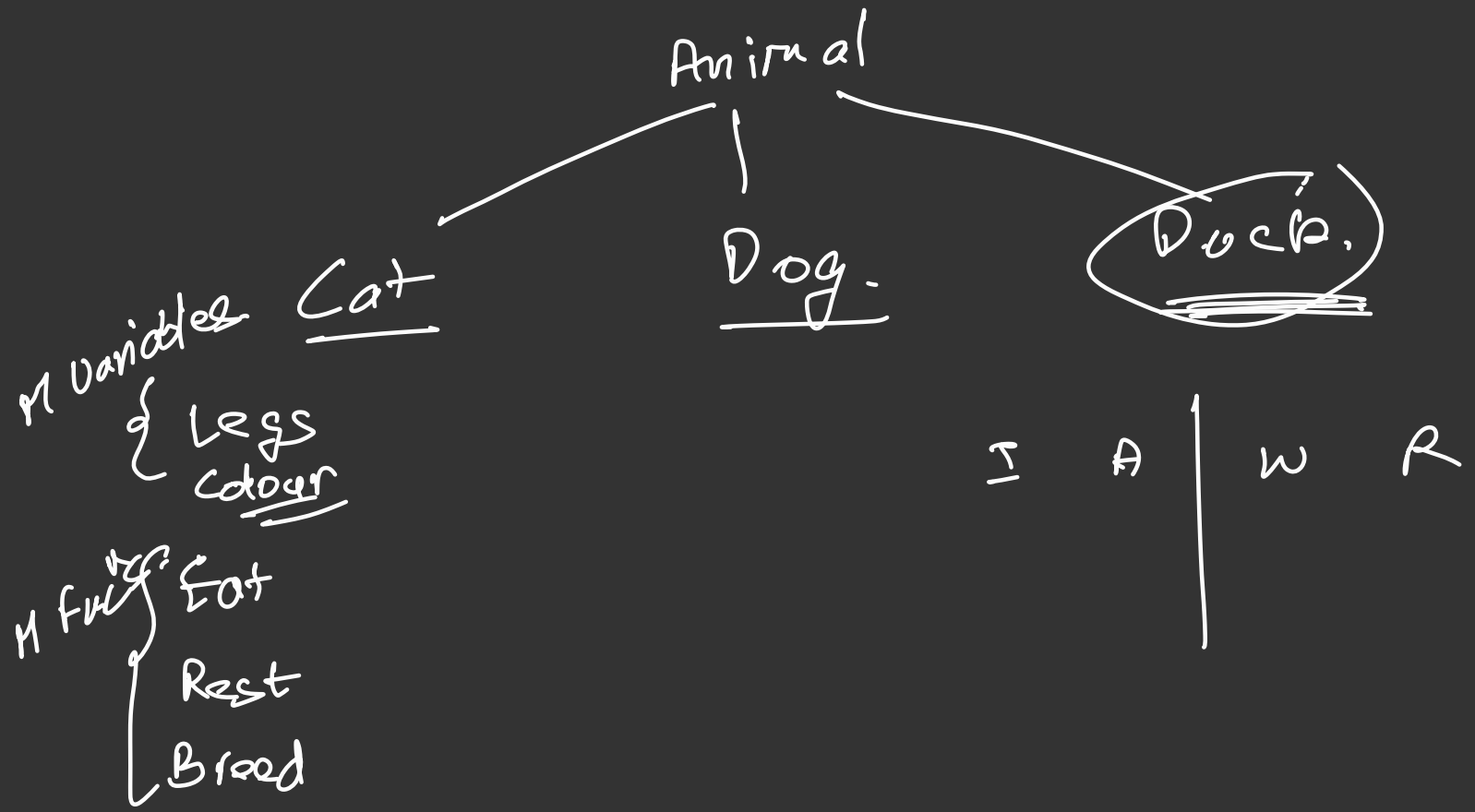
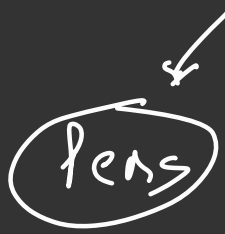

Interface.
object Oriented.
↳ objects
Encapsulation
OOPS
Inheritance
Polymorphism
Abstraction


Animal
Cat
Dog.
Duck.
M Variables
Legs
colour
M fun
Eat
Rest
Bread
I A
W R

DOMS

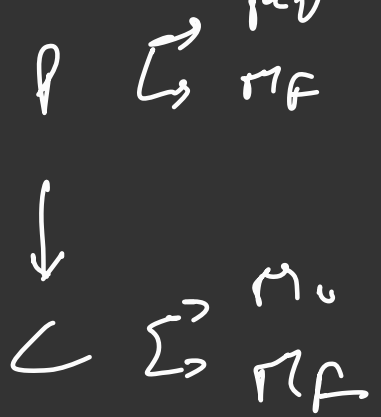
Stationary.



Erasers

Notebook

$\Rightarrow$ Inheritance



Duck

fly → ✓ / ✗

swim → ✓] → reusable

sound → ✓ / ✗

method overriding

class

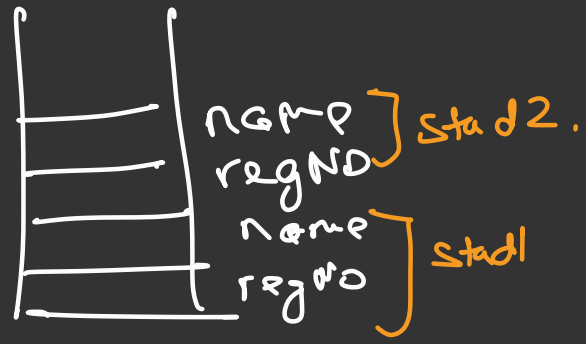
class Student

u → regNo
name

f → details

stud1 { r: 1501, n: ABC }

stud2 { r: 1502, n: BCD }



Task 2:

Books

MV { author
Pages.
name

MF { info() }

method

overriding

overloading

Same name, but diff signature.

sum(n1, n2)

sum(s1, s2)

⇒ Encapsulation

Public Protected Private

Flexible Middle Rigid

class Person {
    name : S
    private userId : S
}

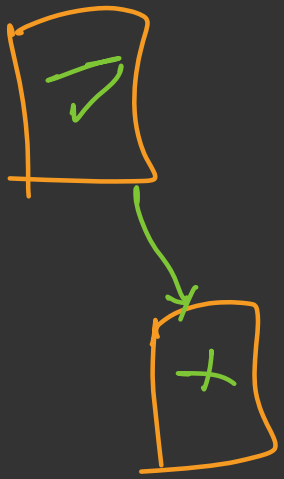
p : P = new
userId ✗

class E extends P {

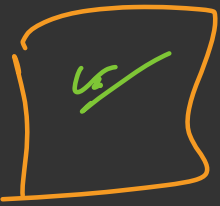
userId ✗

}

Private



Public



Person {

Public name

Protected userId

Private account

}.

Employee {

}.

=

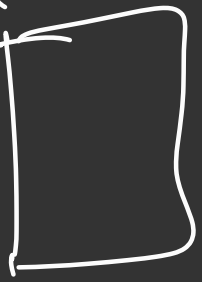
Inheritance

Abstract Class

Interface.

Interface

index.ts



payment.ts

pay1()

{

pay2(){}

```
interface Payment{
    pay()
}
```

```
UPIPay imp Pay{
    pay(){
    }
}
```

⇒ Interface

Problem stat (Limⁿ of Inheri)

```
Duck{
    fly()      fly
    swim()     yes
    sound()    QQ
}
```

```
ID ∈ D{
    fly() — simple
    swim — swim
    sound — In Q
}
```

```
AD ∈ D{
    fly() — complex
    swim — swim ∧ A
    sound — ∈ Q
}
```

Soln: Interface

```
interface IDuck{
    fly() : string
    swim() : string
    sound() : string
}
```

```
ID imp IDE
    f()
}
```

```
AD imp ADE
}
```

Abstraction

↳ Can You Create object of interface ?

No.

↓ why ?

func^n defn incomplete ∴ implm^n is pending.

↳ Abstract (interface) ≠ Concrete → classes

↓ ↓

implm^n x fully implemented

↓ ↓

obj creⁿ x obj creⁿ ✓

Encapsulation

	Inside class	Outside Class	Inside child
Public	✓	✓	✓
Protected	✓	✗	✓
Private	✓	✗	✗

⇒ Polymorphism

(1) method Overriding → Run-time.

(2) —— Overloading → compile time.

M. Overriding

Duck.

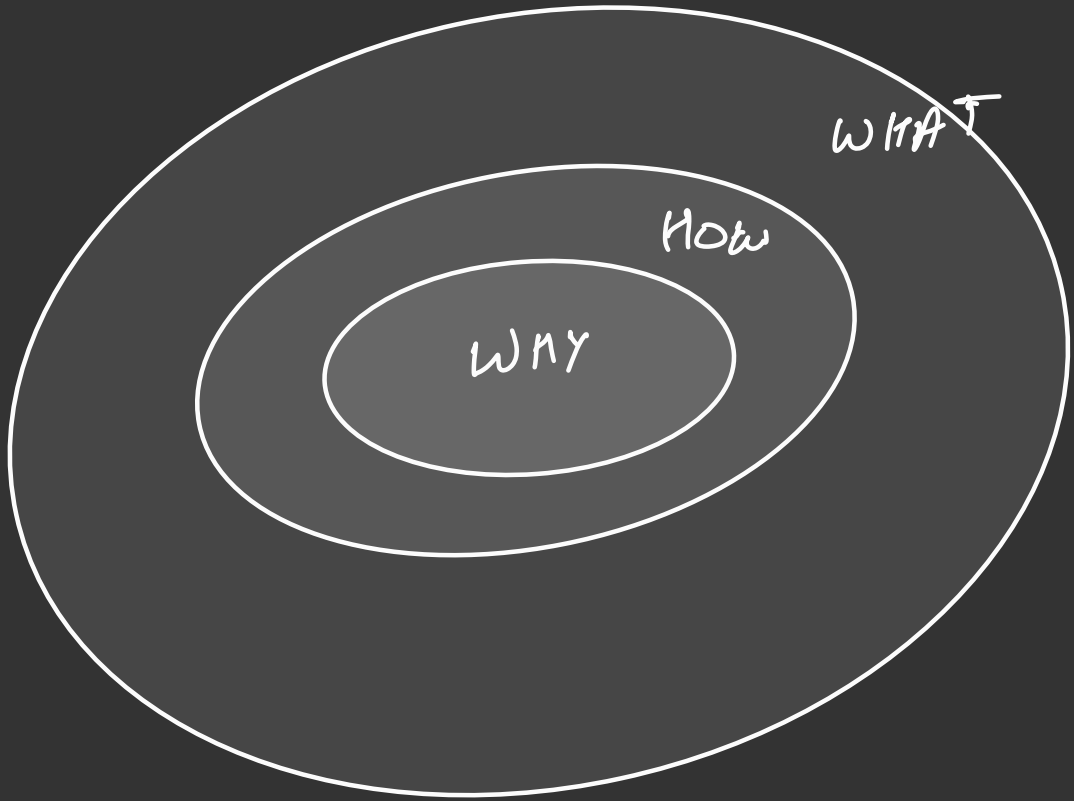
swim

fly

ID

#D

=> Object Substitution


WHAT
How
WHY