Today Agenda :-

1) Interfaces
2) Abstract
3) Static

Interfaces :-

Class :- blueprint of an entity / idea

↳ Attributes & behaviours

variable methods

↳ along with dfn of the method

A concept which is not exactly a real entity

↳ Not really have any attribute / dfn " of method.

but that concept is categorized by the type of behaviour supports.

Concept :-

Anyone who can eat, walks, runs is an animal for me.

Interface :- Blueprint of a behaviour.

↳ Defines a set of behaviours that must be implemented by a class to considered belonging to a category.

```
interface Animal {          → checker
    void eat();
    void walk();            → method
    void run();                declaration
}

class Cat implements Animal {

    void eat(){
        s.out("cat is eating")
    }
    void walk(){
        s.out("walking")
    }
    void run(){
```

} S.out ("cat is running")

void meow() {
} S.out ("meow")

class Dog implements Animal {

void eat() {
}
void walk() {
}

class — imp A {

:- compile time

}

Stack :- LIFO

↳ push ()
pop ()
peek ()
isEmpty ()

Clear :

```
interface stack {
    void push (int x);
    int pop ();
    int peek ();
    bool isEmpty ();
}
class Arraystack implements stack{
    int top;
    int [] arr;
    int maxsize;
    void push (int val) {
```

```
        arr[top] = val
    }  top++
    void pop(){
    }
    int peek(){
    }
}   bool isEmpty()
```

Class LL stack implements stack {

// 4 methods

}

HW :- Implement ArrayStack & LL stack classes which implements stack integers.

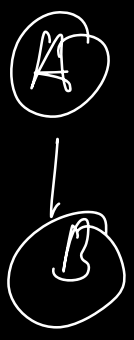
Polymorphism :-

class → Dog

Stack

is a → Array stack

is a → LL stack



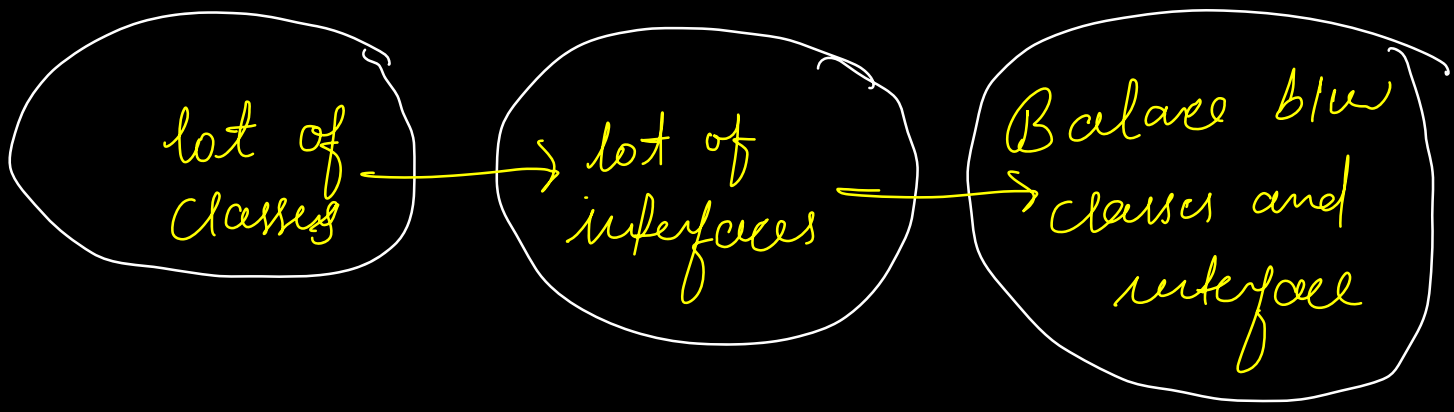
Animal a1 = new DOG()

stack S1 = new ArrayStack()

Stack S2 = new LLstack()

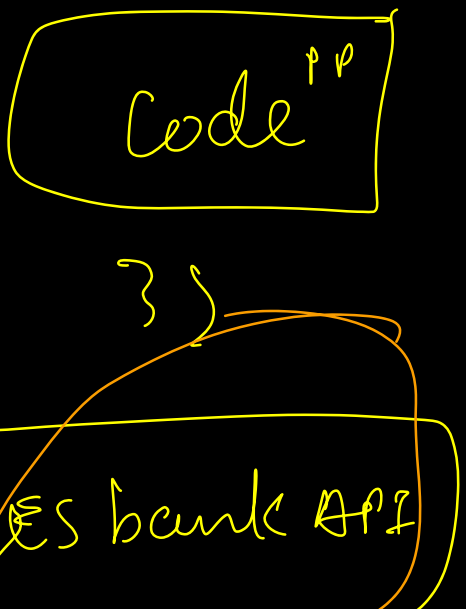
To the principal → anyone

Program to interface, not Implementation



PhonePe :-

phonepe partnered with YES Bank RBI banned YES Bank for all online transaction.



within a 1 day, phonePe was back. changed to

Icic

Provider to icici bank APIs.

Interfaces :-

Class phonepe {

YesBank api

icici bank

add to wallet () {

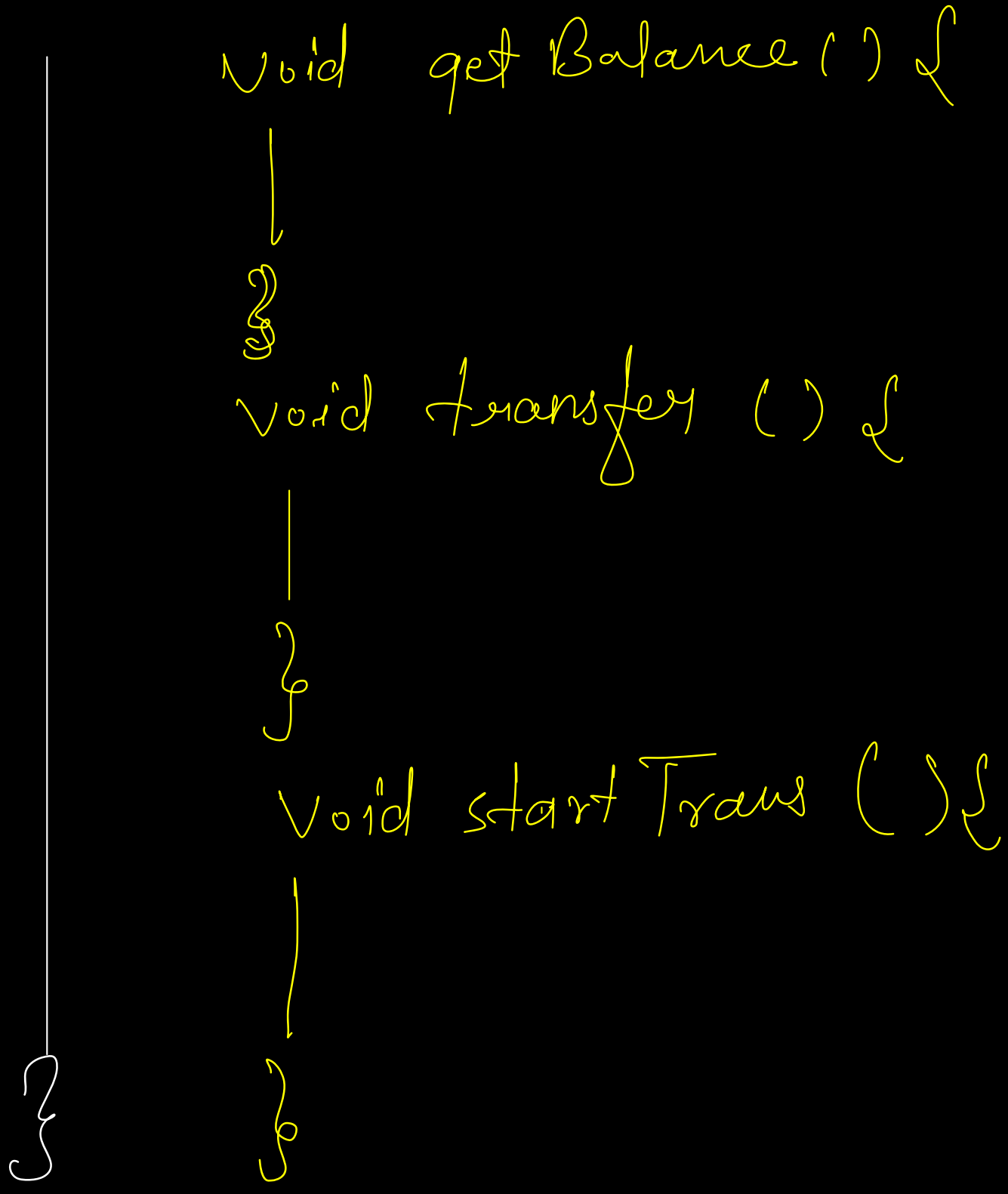
api. cheak balance ()

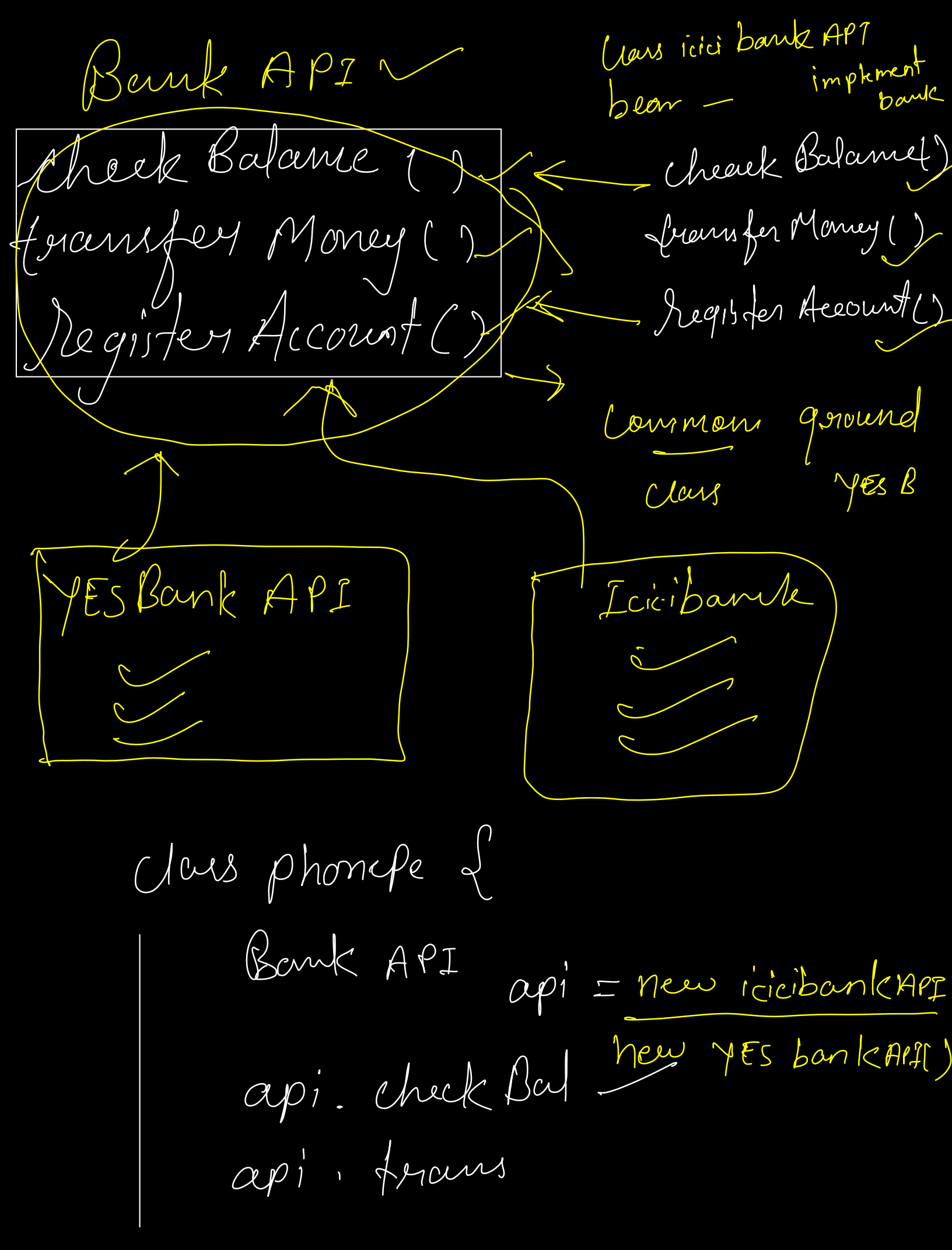
if (—) {

api. send money ()

}

api. register ()

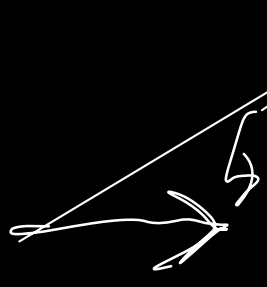

Class IciciBank API {
Void getBalance() {
}
Void transfer() {
}
Void startTrans() {
}
}


Bank API
checkBalance()
transferMoney()
RegisterAccount()
class icici bank API
bean —
implement
bank
checkBalance()
transferMoney()
RegisterAccount()
common ground
class
Yes B
YESBank API
Icicibank
class phonepe {
Bank API
api = new icicibankAPI
new YES bankAPI()
api.checkBal
api.trans

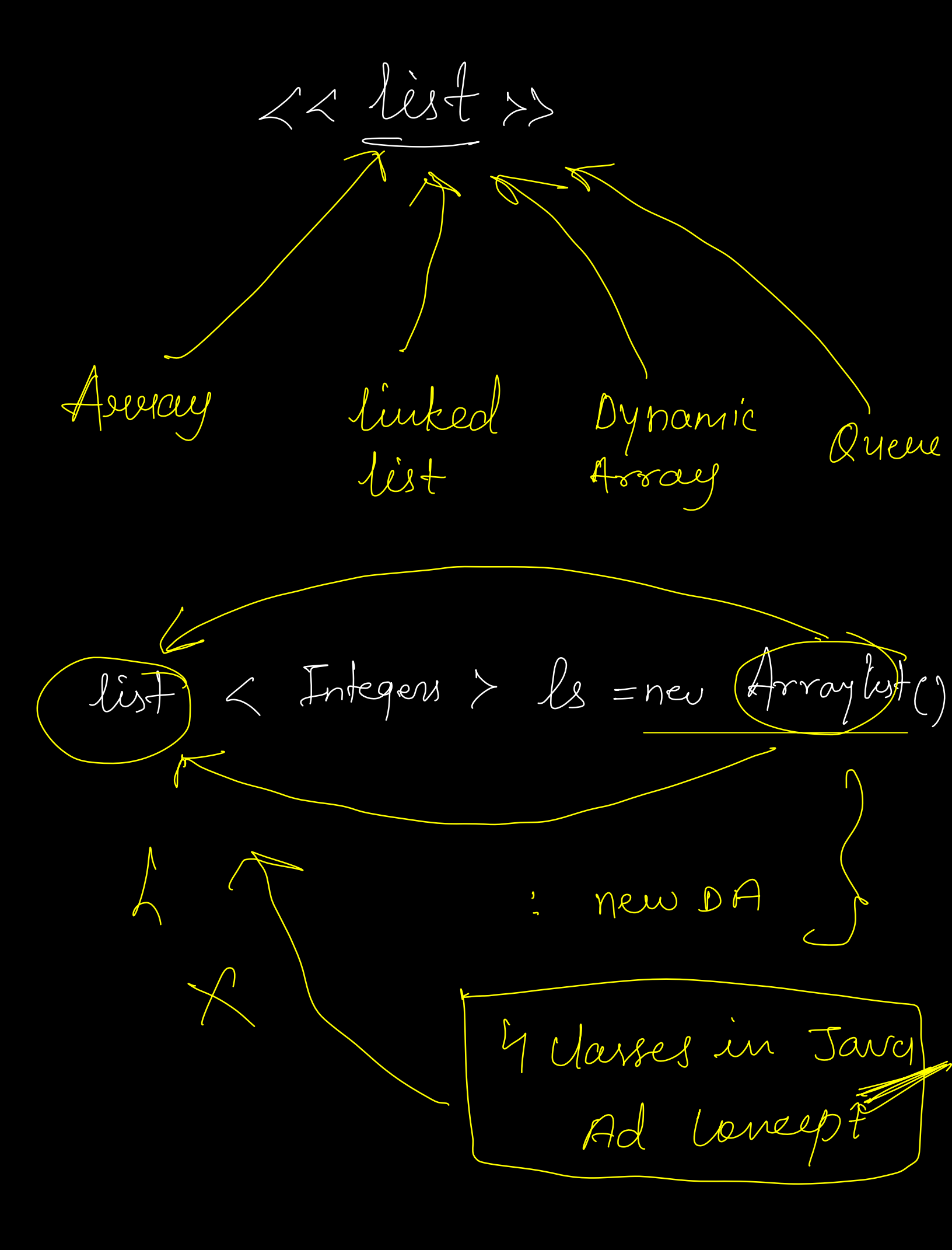
Life statement :-

> Never get attached to a person, if they leave, you'll be Messed up.

System should be looseley coupled avoid tight Coupling



2 classes have direct dependency on each other.


<< list >>
Array
linked list
Dynamic Array
Queue
list < Integers > ls = new ArrayList()
: new DA
4 Classes in Java
Ad concept

Abstract classes :-

↳ Entity { Att, Beh

Interface { behaviour / Method

↳ But I don't have 100% clarity on how those behaviour (will work

```
abstract Animal {
    String name
    int age
    abstract void walk();
    abstract void eat();
    void printName(){
    }
```

We cannot create object of Abstract classes.

Any child class of animal will be forced to implement these abstract method.

```
class Tiger extends Animal {
    Tiger t = new Tiger();
    t.name
    t.age
    {
        void walk() {
            System.out.println("Tiger walking");
        }
    }
    void eat() {
        System.out.println("Tiger eating");
    }
}
```

Class
↑ Extended by
Class

Interface
↑ implemented by
Class

interface

↑ Extend

interface

abstract class

↑ extend

class

abstract Animal

Sound

abstract

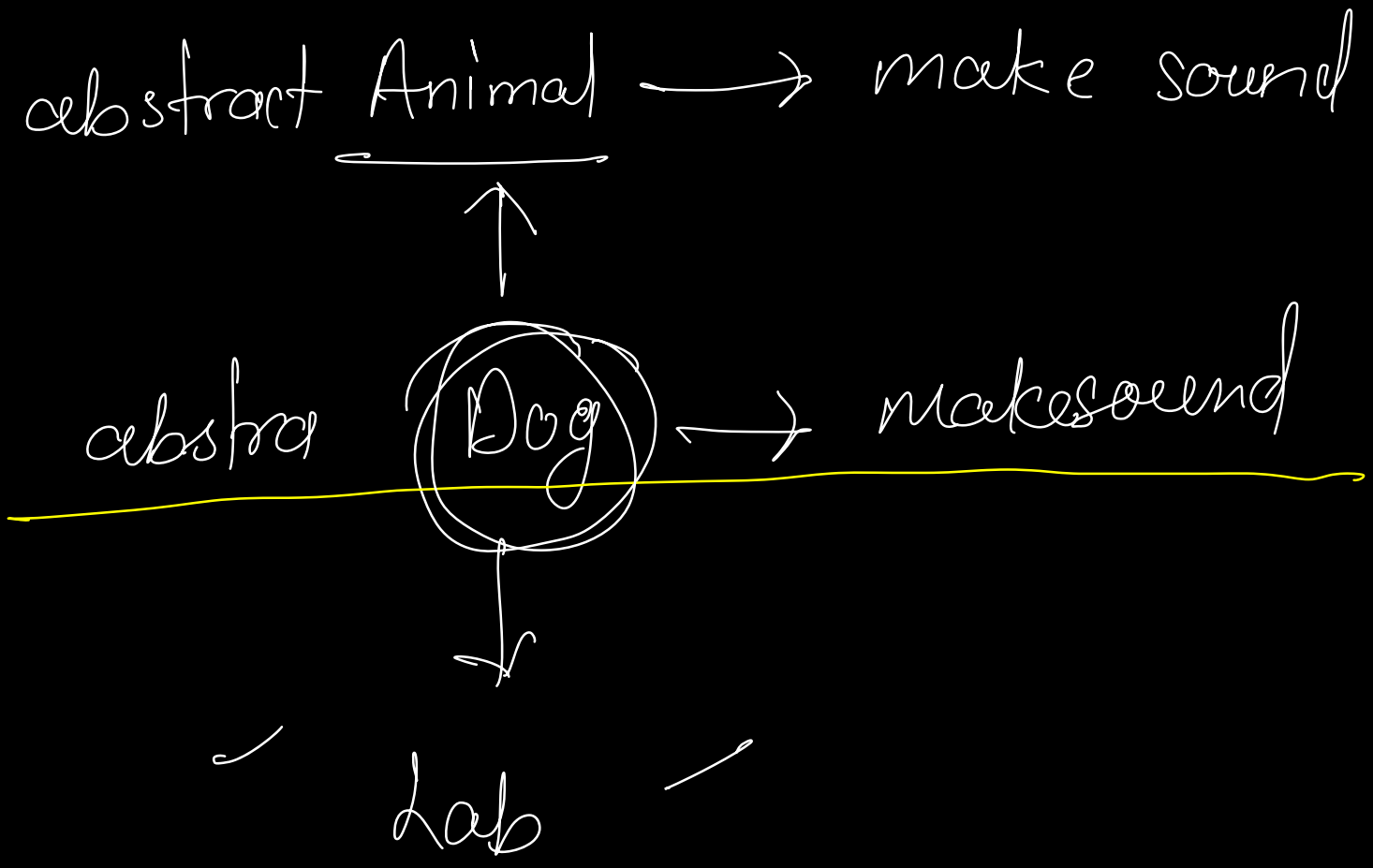
Dog

Sound

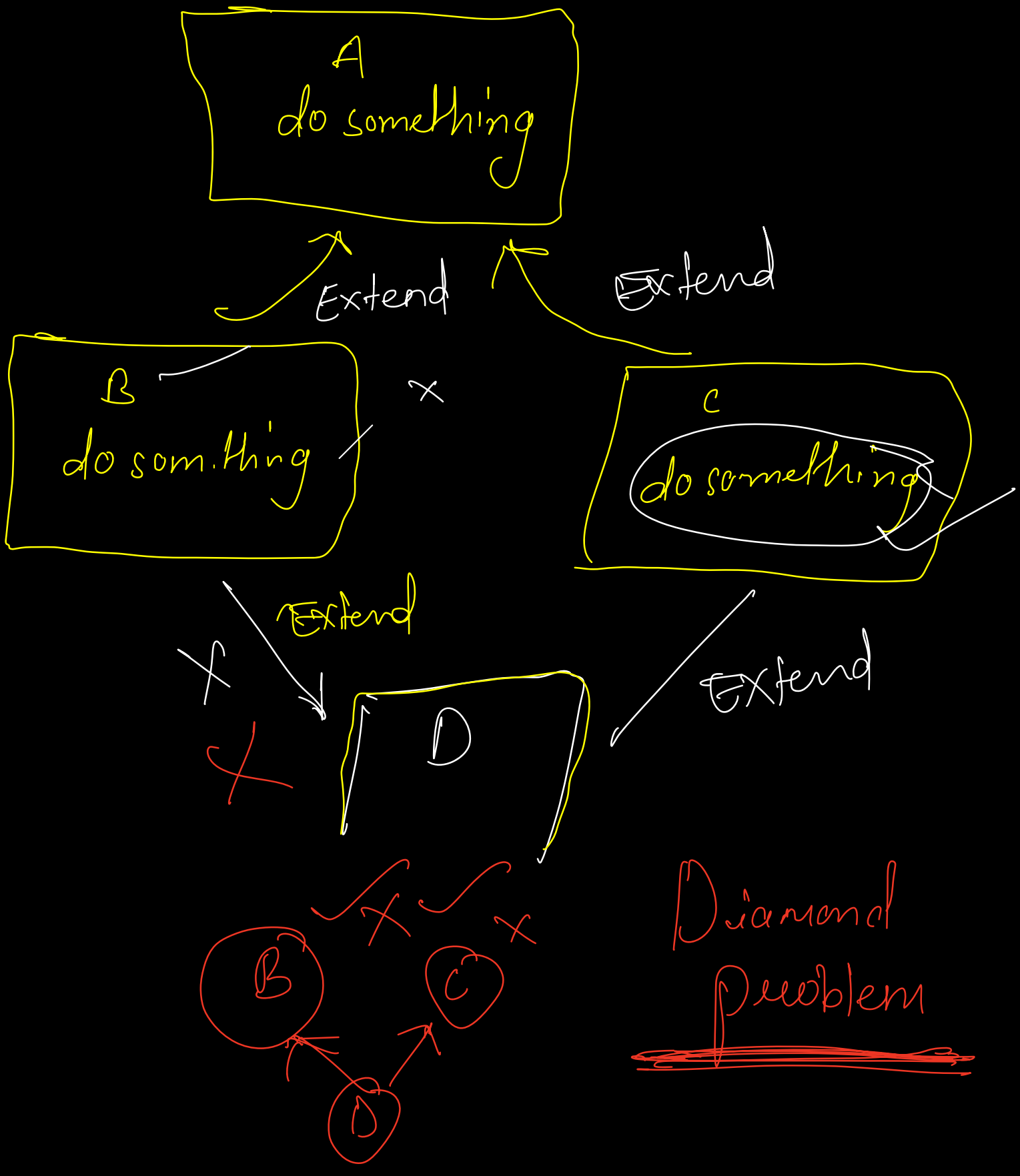
An abstract class with no attributes ⇒ interfaces

Only abstract method

Only method defn



- One class can extend only one class
- One class can implement ∞ interfaces


A
do something
Extend
Extend
B
do something
C
do something
Extend
Extend
D
B
C
D
Diamond
problem

Static ⁰—

client {

access modifier

psvm

Stand for code execution

public static void main ( ) {

↳ method name

Return type

you can access static method / variable without an object of the Corresponding

Class

Class Var

Class Var

static Var

static method

method