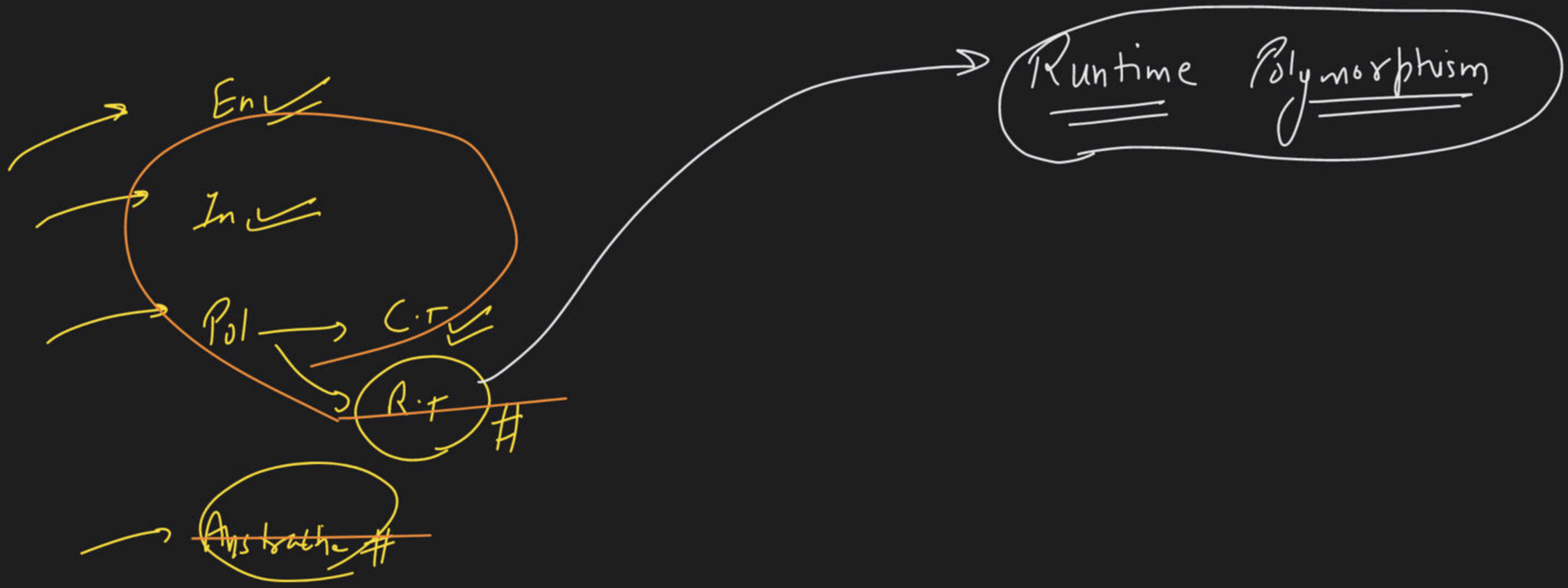
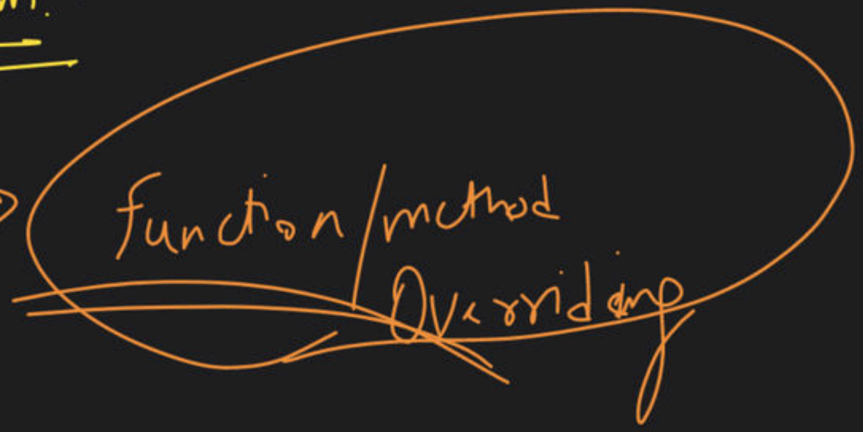
OOPs Class-3

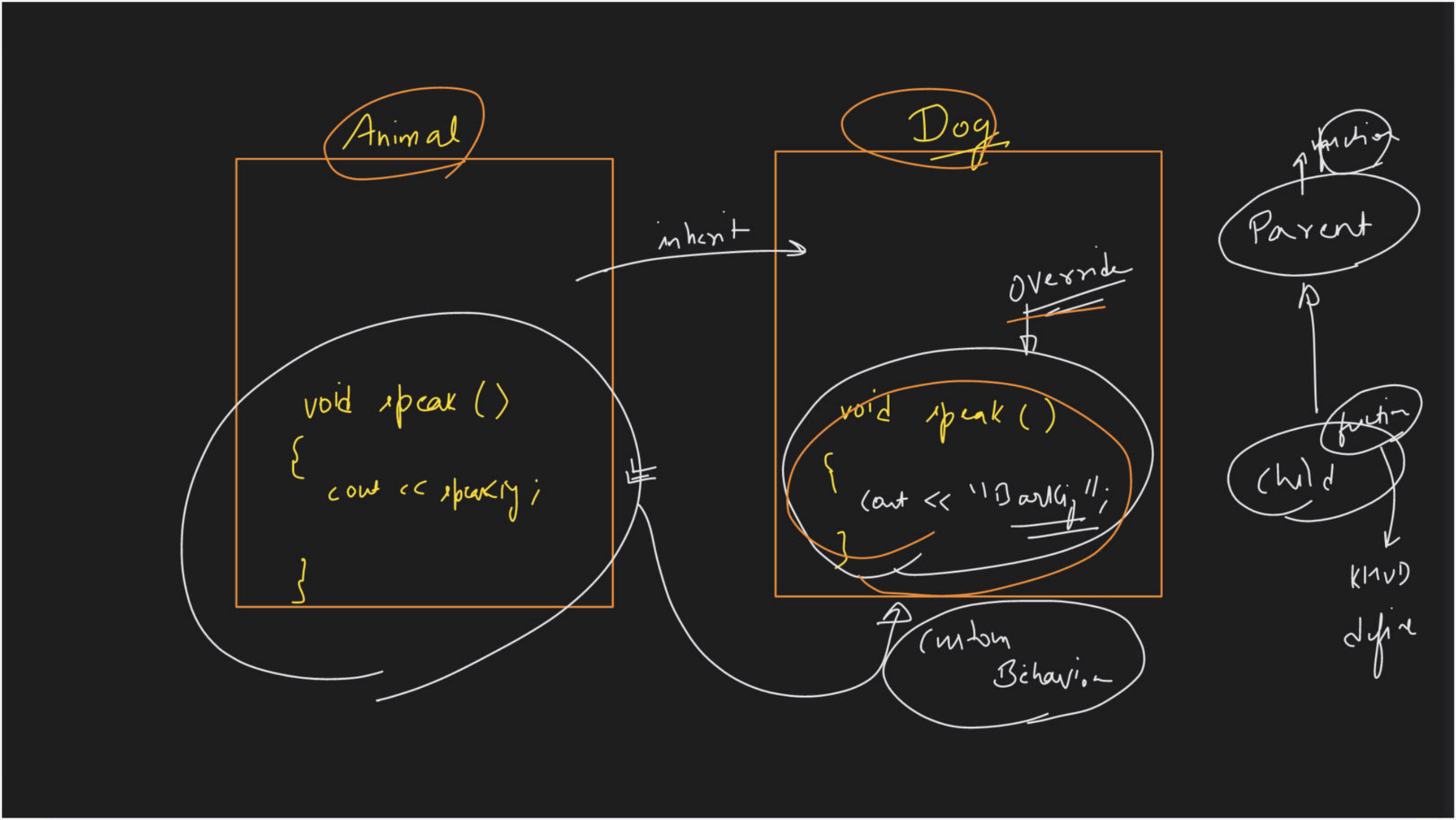
Special class

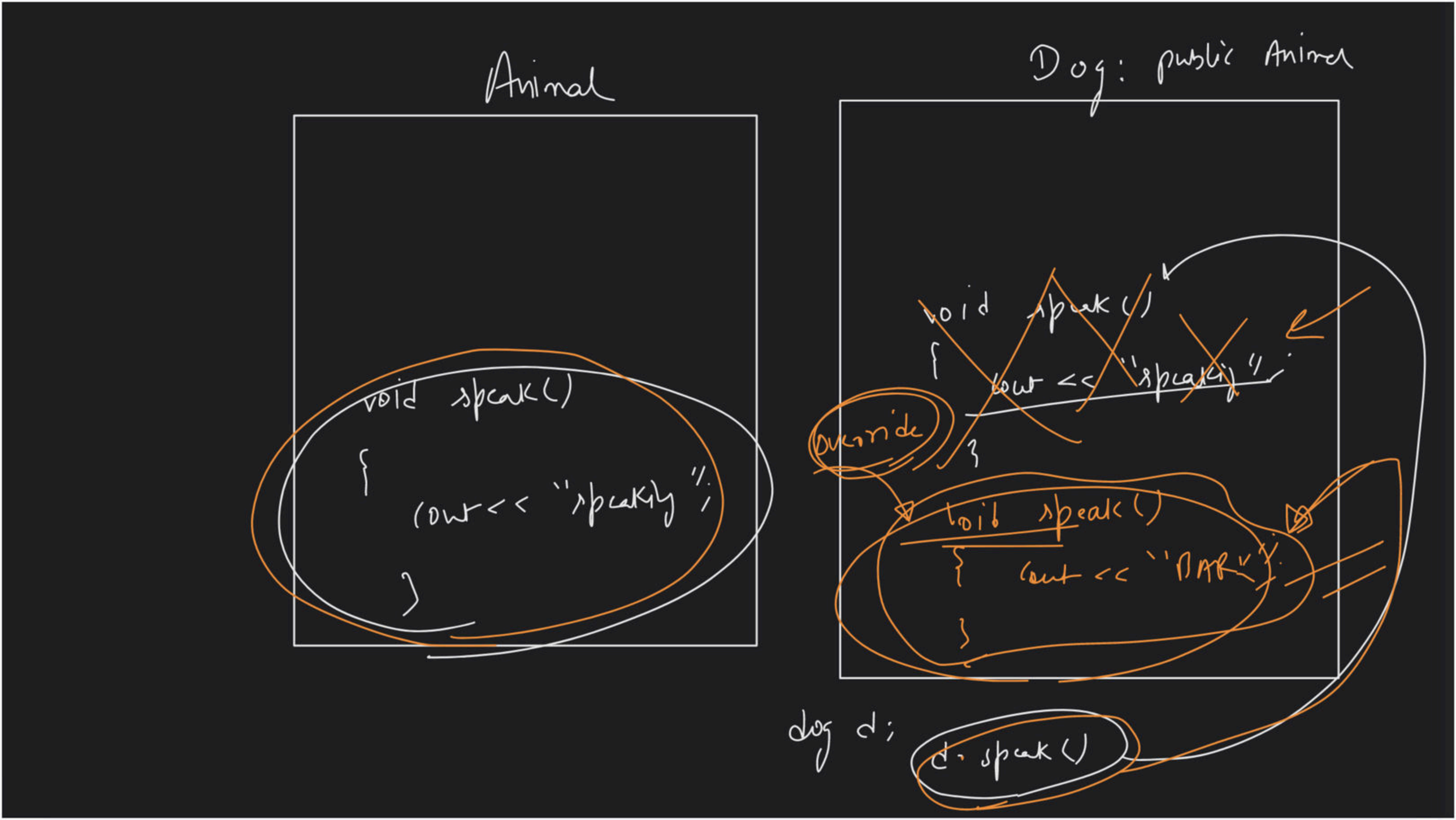
Love Babbar • Apr 7, 2023

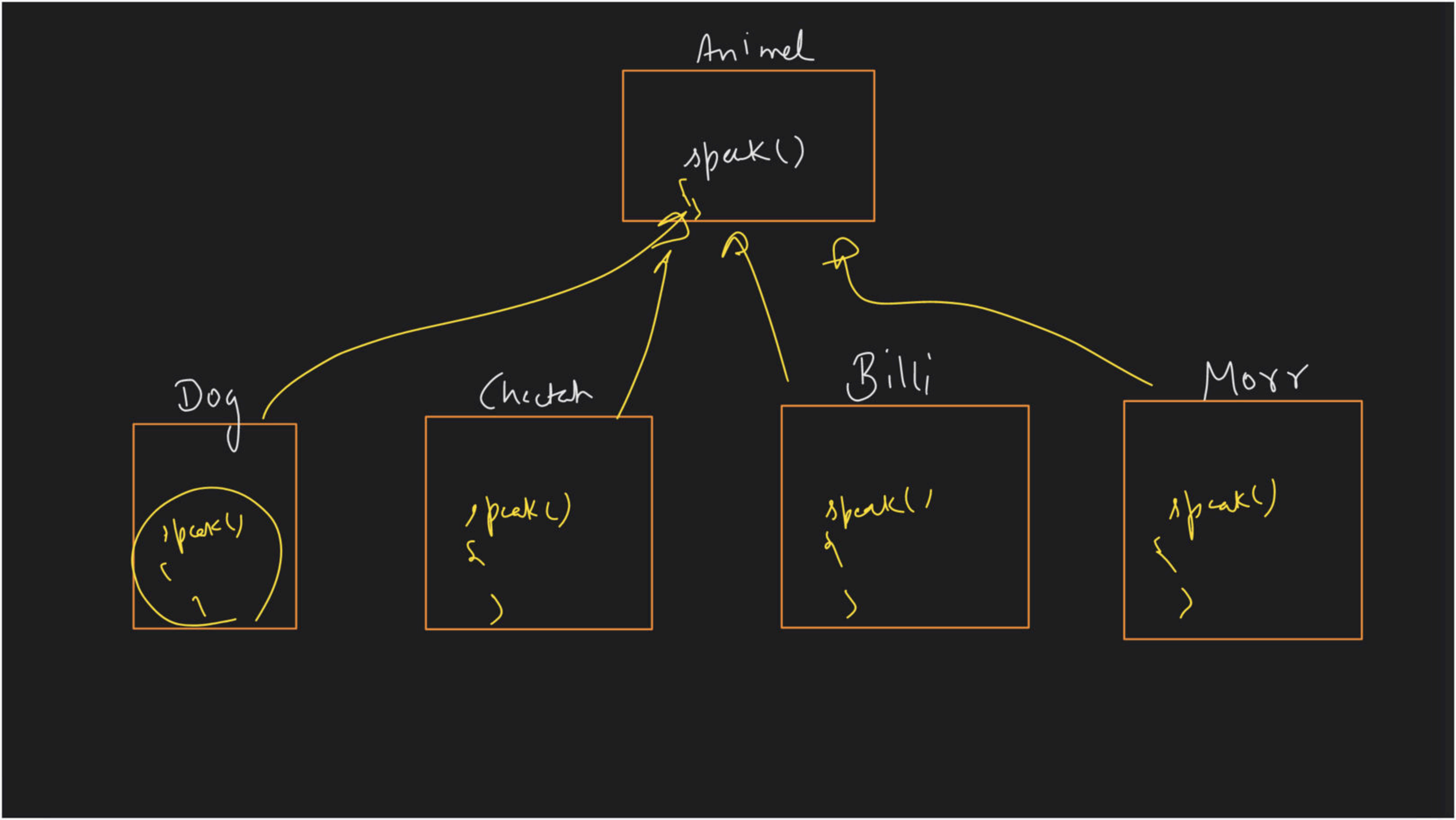

En
In
Pol
C.T
R.T
#
Abstract #
Runtime Polymorphism

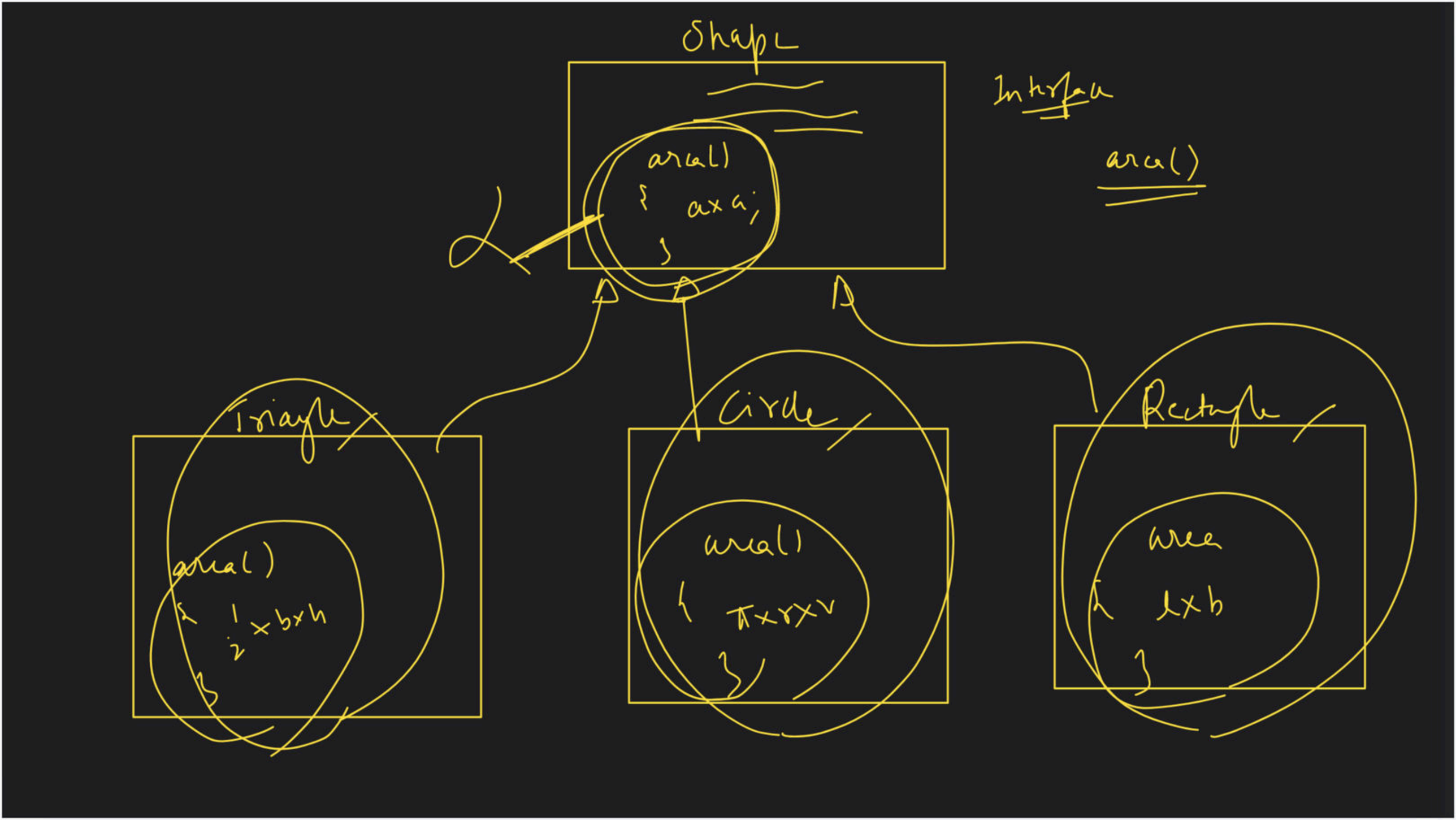
Runtime Polymorphism:-

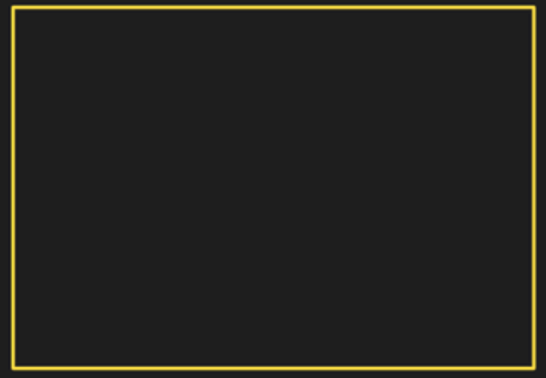


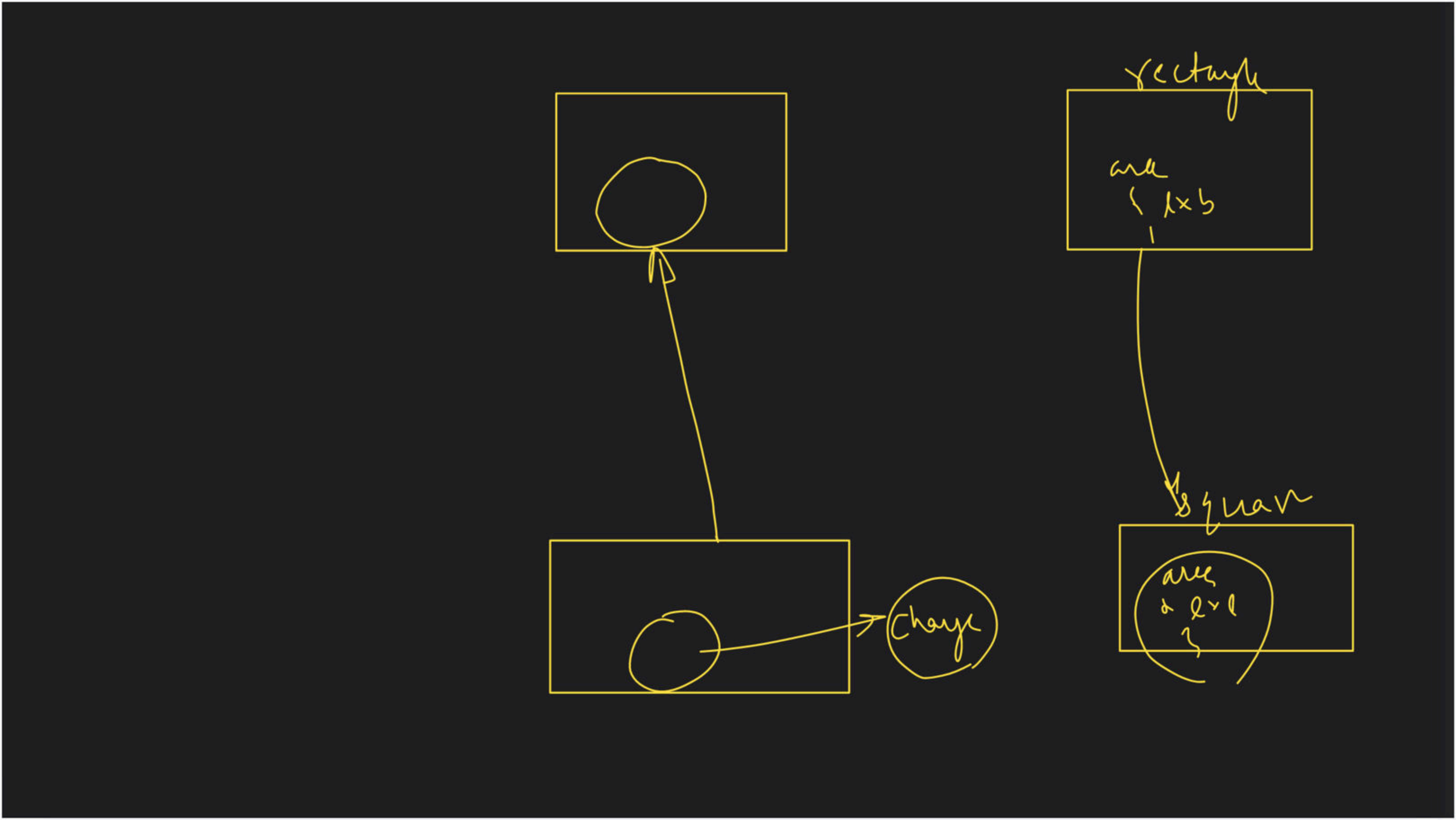

Animal
void speak ()
{
cout << speaking;
}
inherit
Dog
Override
void speak ()
{
cout << "Bark";
}
Custom Behavior
Parent
Child
kind of


Animal
void speak()
{
cout << "speaking";
}
Dog: public Animal
void speak()
{
cout << "speaking";
}
override
void speak()
{
cout << "BARK";
}
dog d;
d.speak()

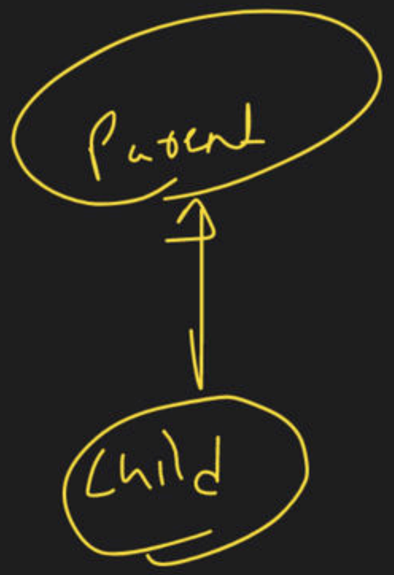

Animal
speak()
Dog
speak()
Cheetah
speak()
Billi
speak()
Morr
speak()


Shape
Interface
area()
area()
{ ax a;
}
Triangle
area()
{ 1/2 x b x h
}
Circle
area()
{ π x r x r
}
Rectangle
area
{ l x b
}




Rectangle
area
{ l×b
Square
area
s×s
Shape

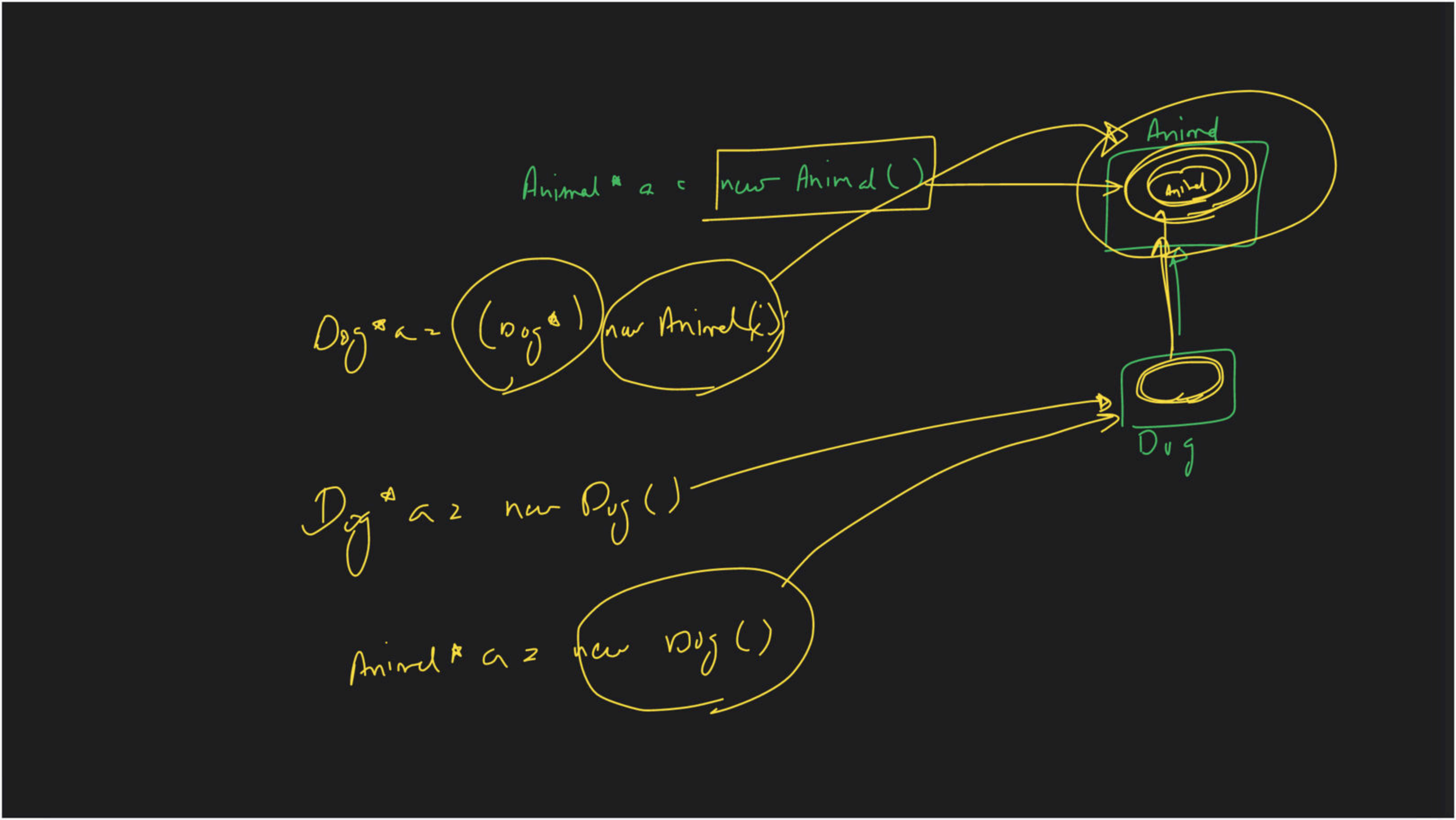
Dynamically obj creation

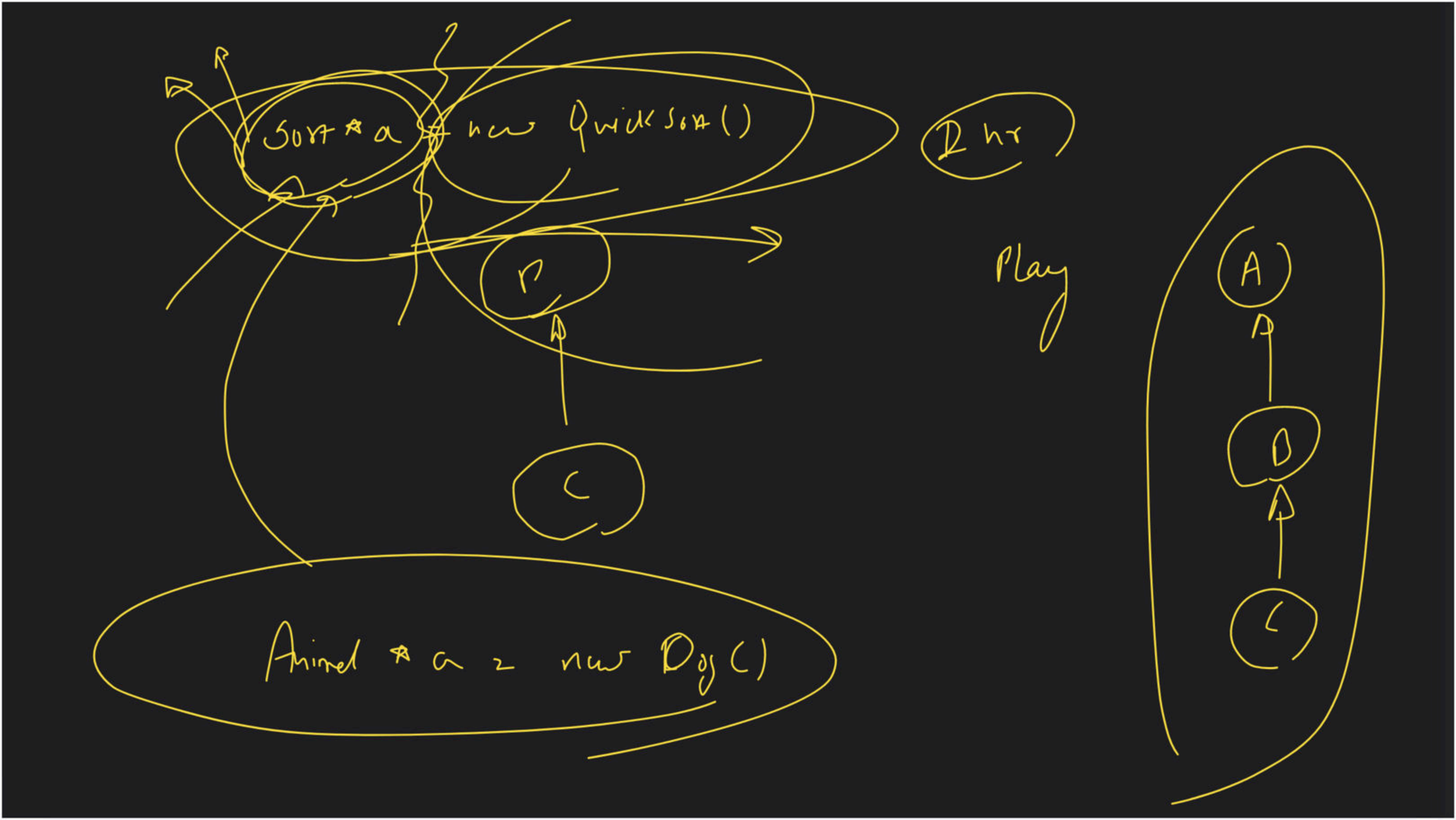


D·M/M·F → array

```
Parent* a = new Parent();
Parent* a = new Child();
Child* a = new Child();
Child* a = (Child*)new Parent();
```

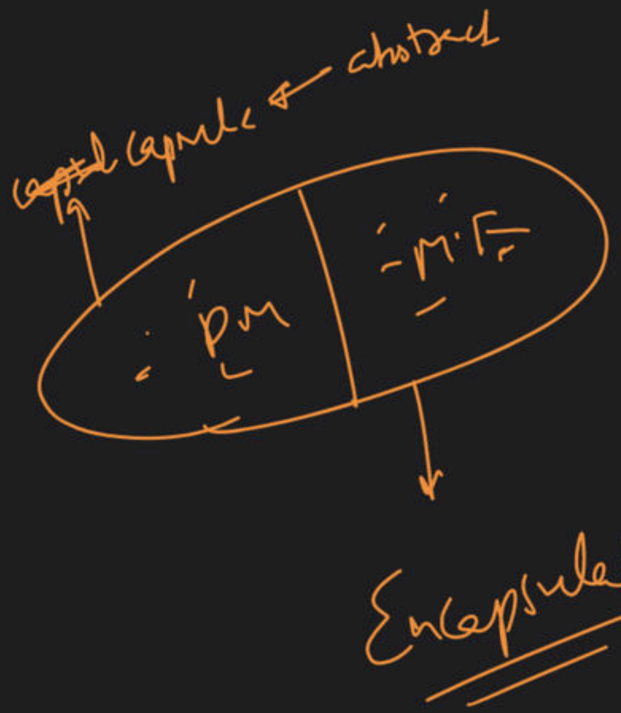
virtual

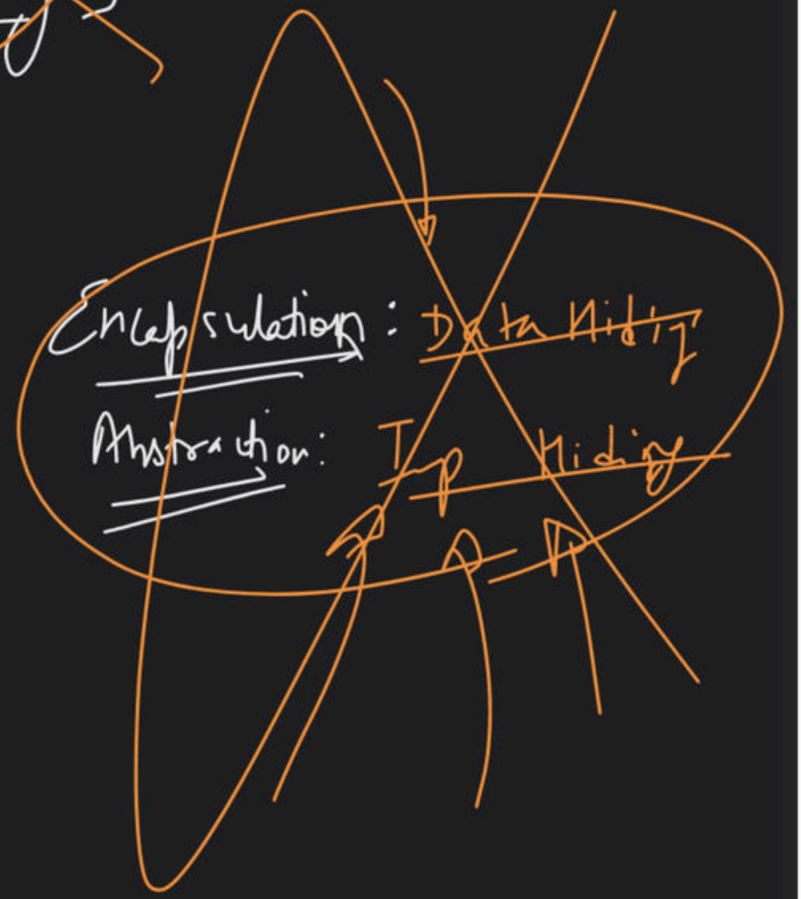

Animal * a = new Animal()
Animal
Dog * a = (Dog *) new Animal();
Dog * a2 = new Dog()
Animal * a3 = new Dog()
Dog


Sort * a = new QuickSort()
2 hr
Play
P
C
Animal * a = new Dog()
A
D
C

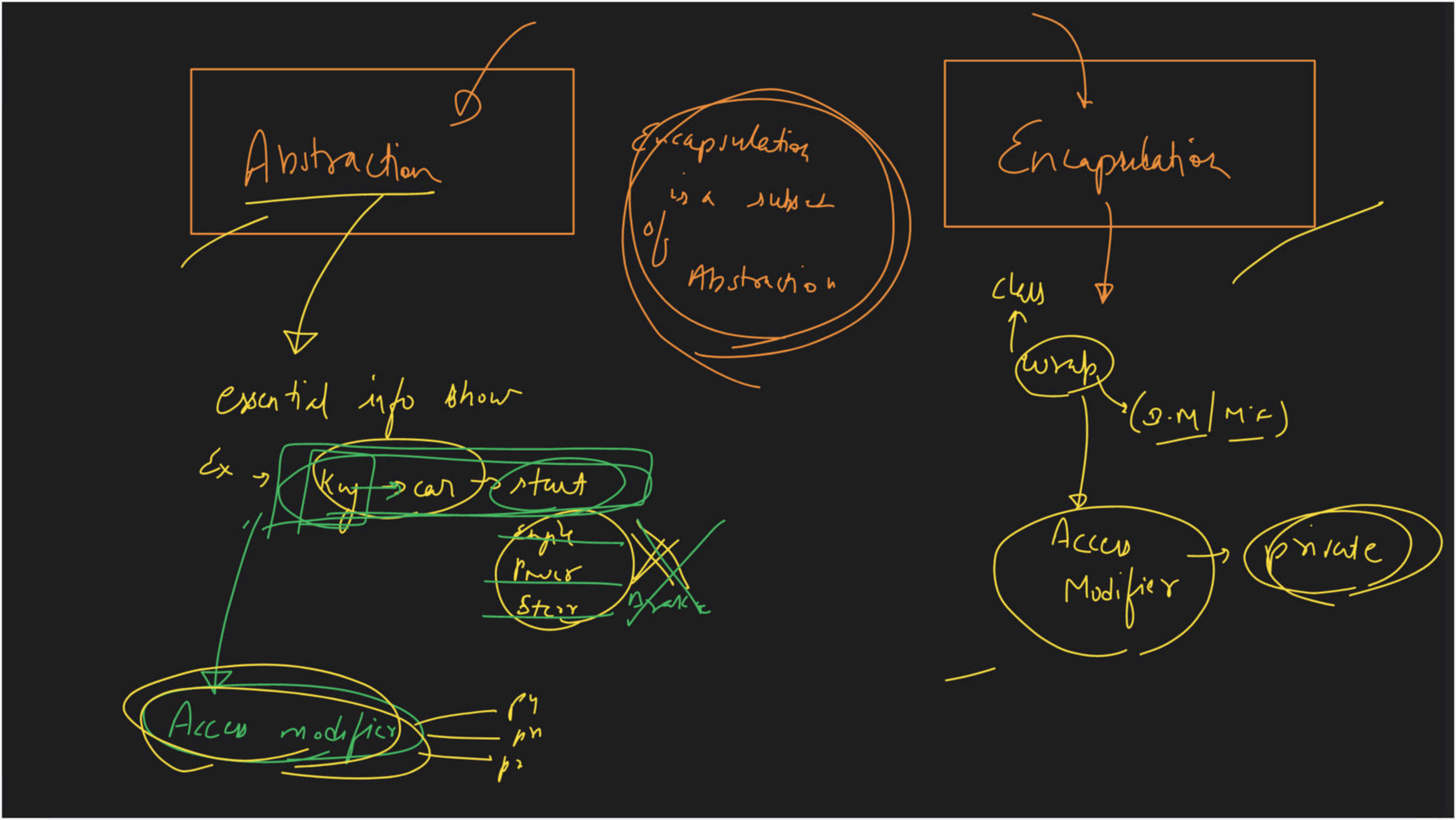
Abstraction: - [Implementation Hiding]

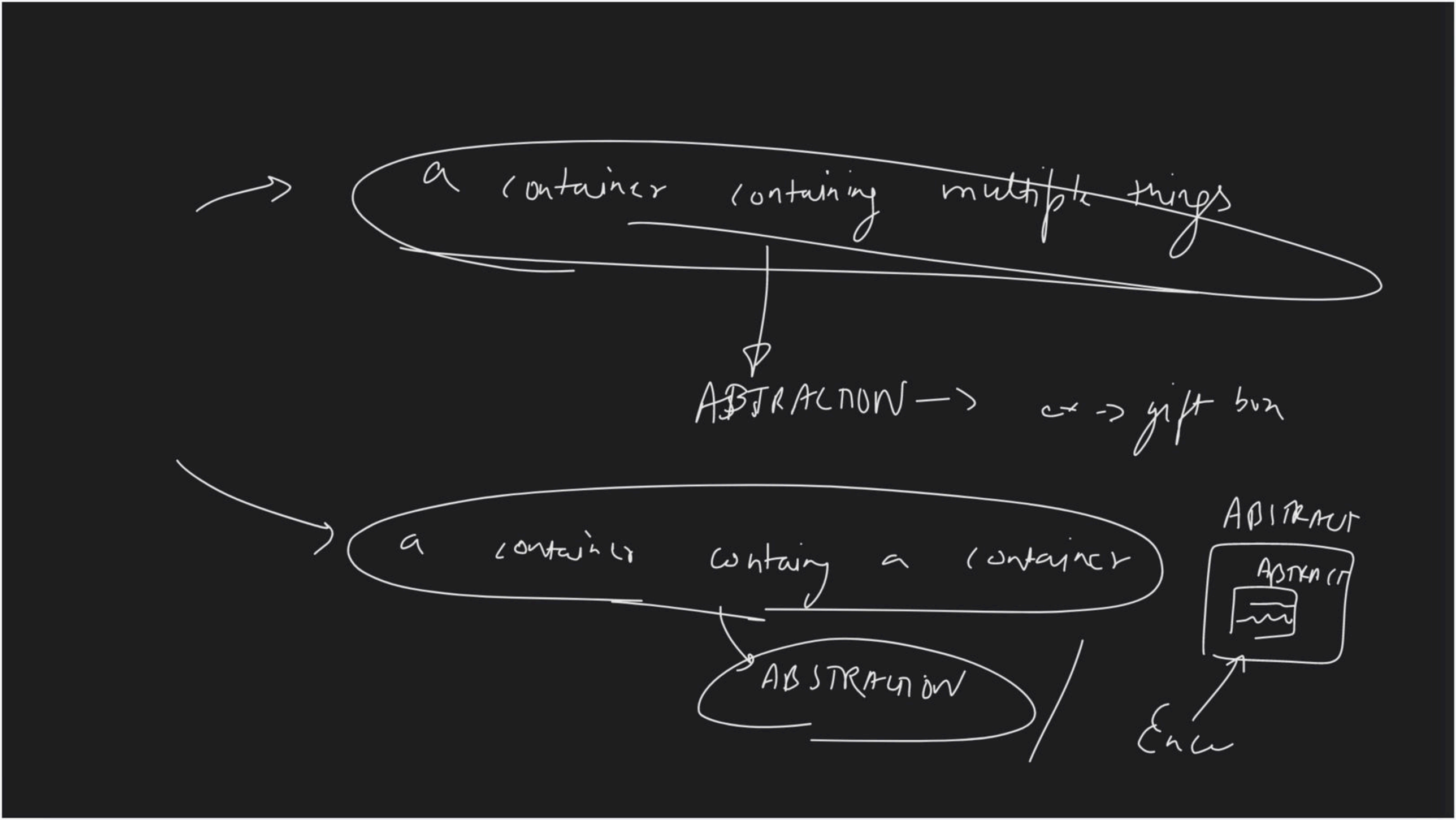
Encapsulation is a subset of Abstraction



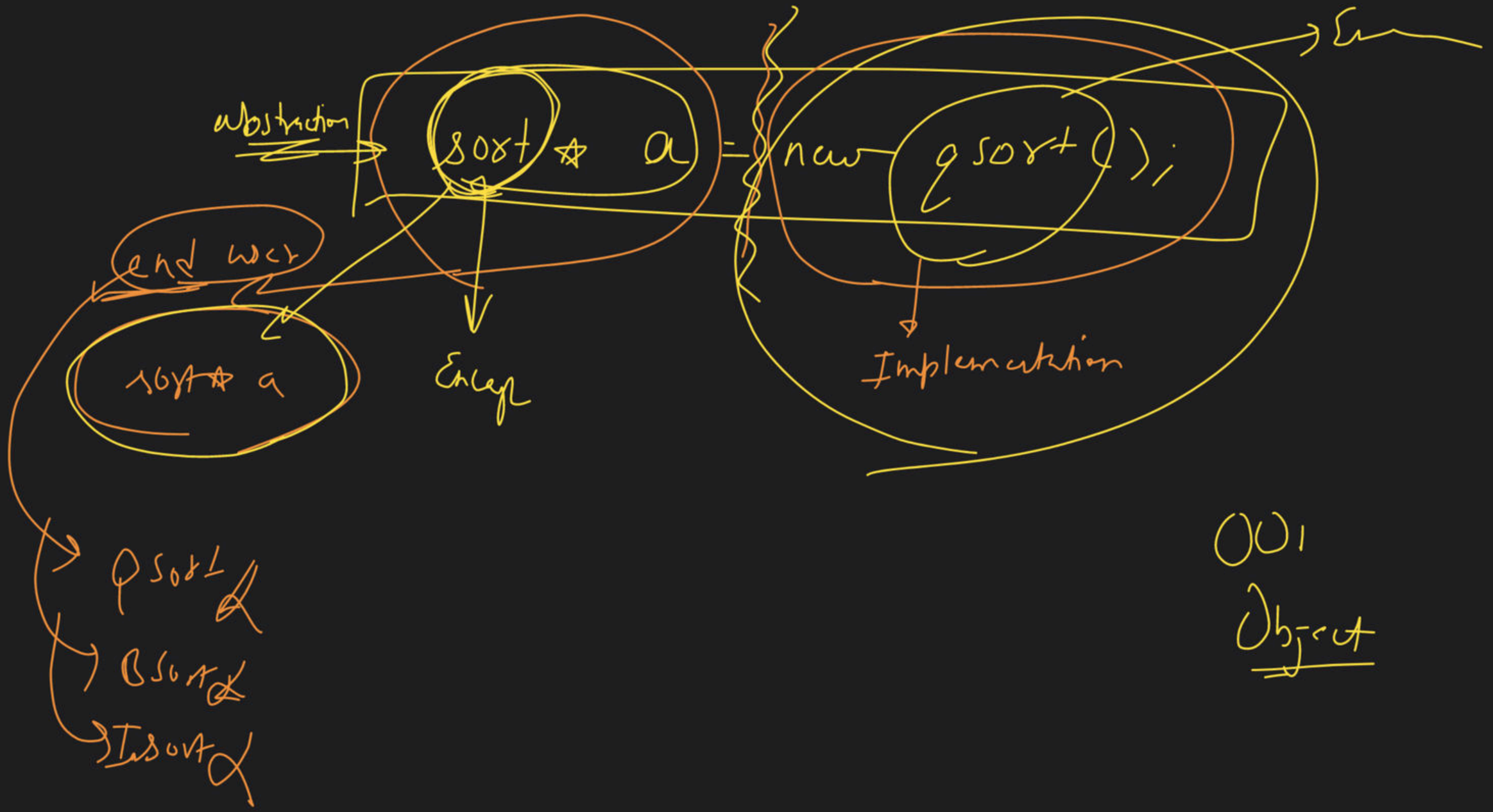


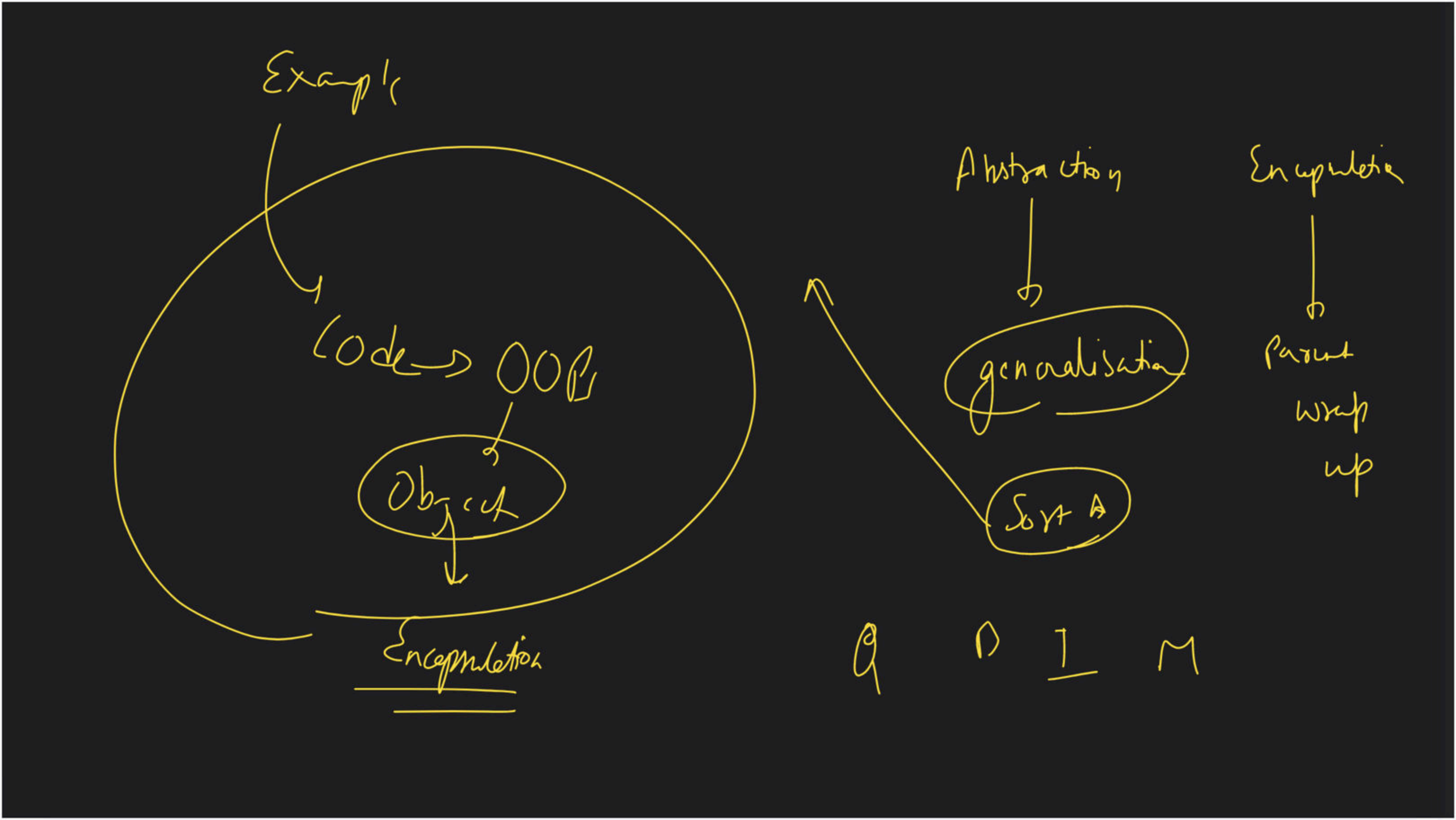
2 min

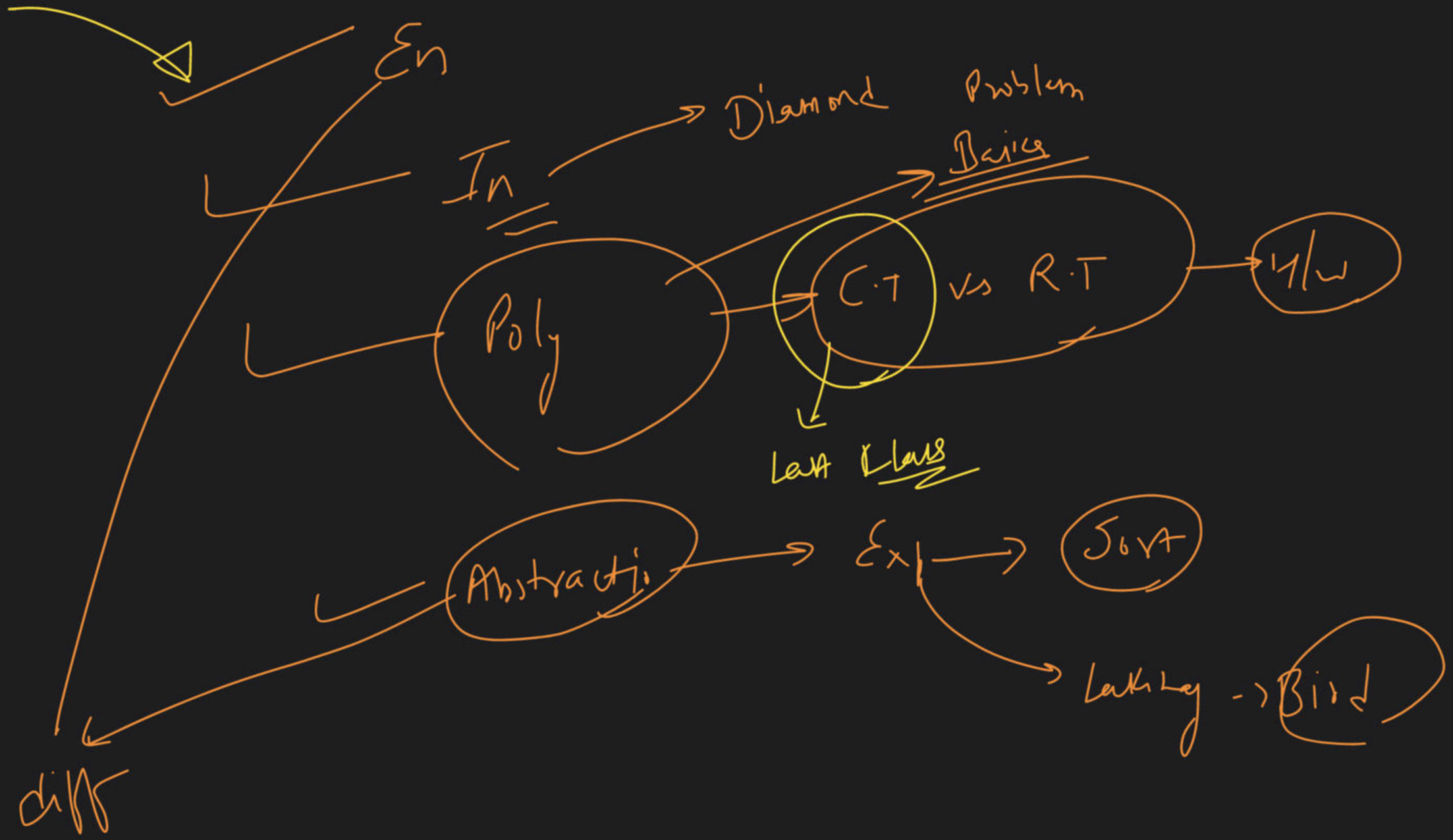

Abstraction
Encapsulation
is a subset
of
Abstraction
Encapsulation
class
wrap
(D.M / M.F)
Access
Modifier
private
essential info show
Ex
Key
car
start
Access modifier


a container containing multiple things
ABSTRACTION
a container containing a container
ABSTRACTION
ABSTRACT
ABSTRACT

→ a group of many things

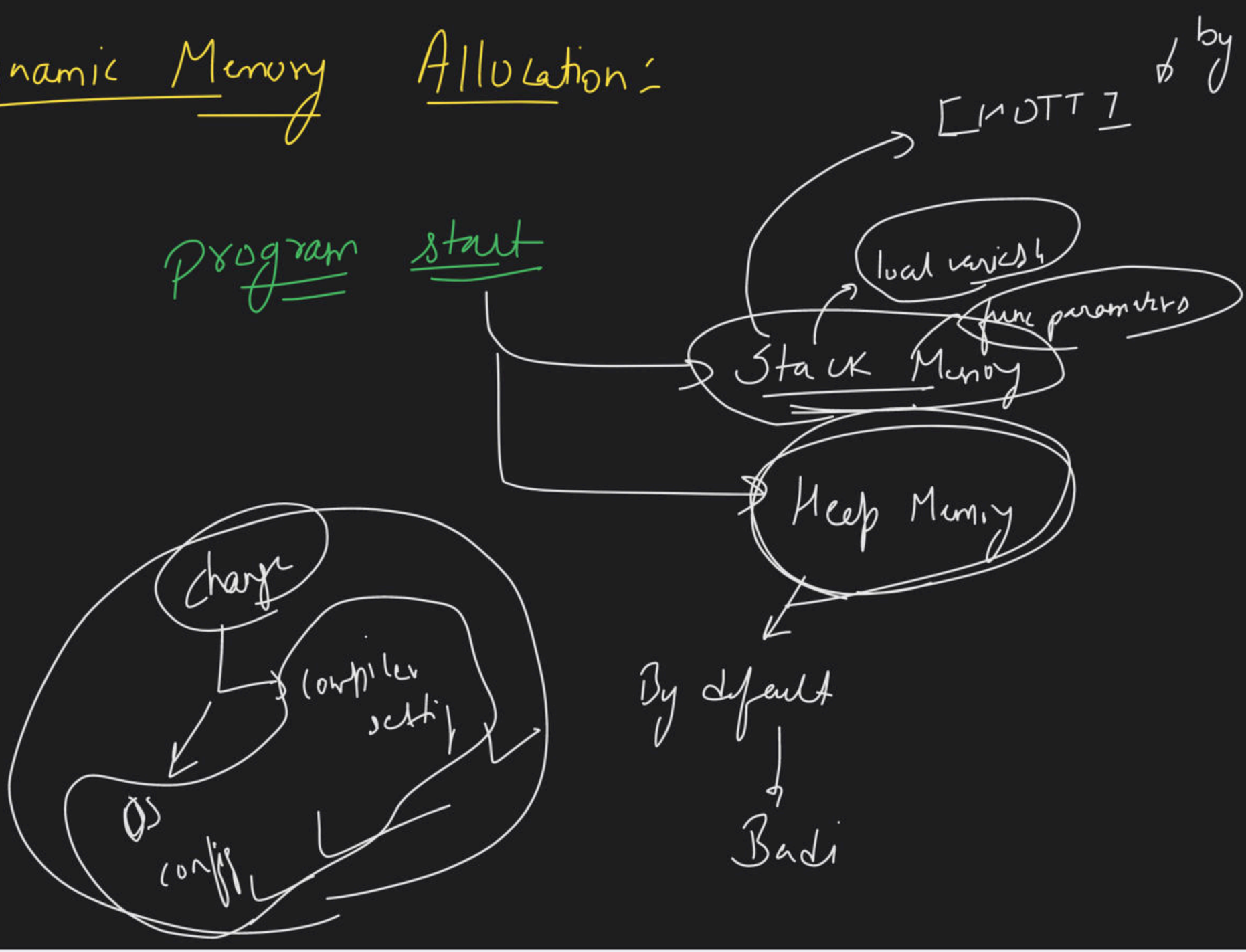

abstraction
qsort * a
=
new qsort();
end work
qsort * a
Encap
Implementation
qsort
Bsort
Isort
OOP
Object

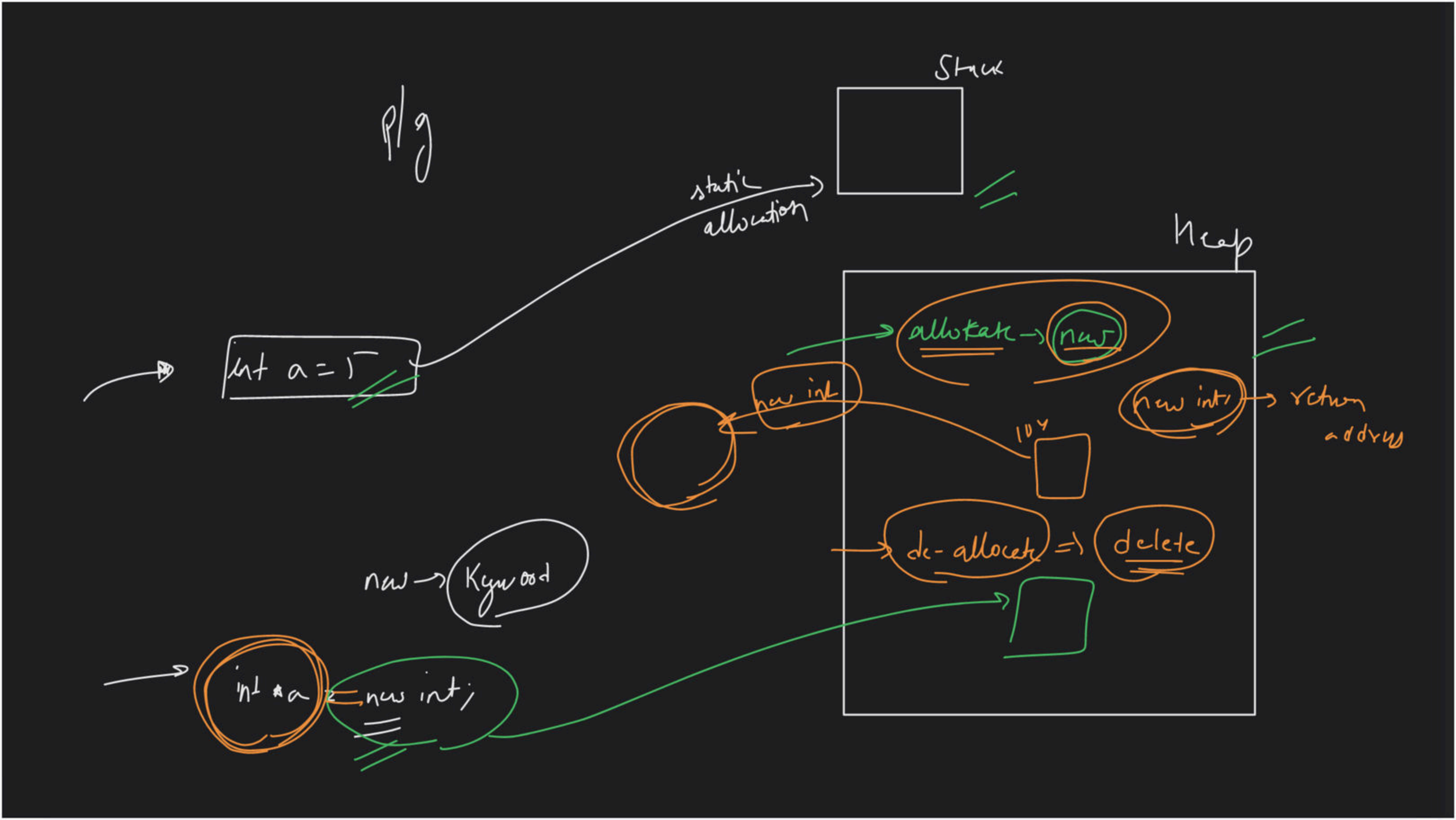

Example
Code → OOP
Object
Encapsulation
Abstraction
Encapsulation
generalisation
Parent
wrap
up
Syst A
OOPM

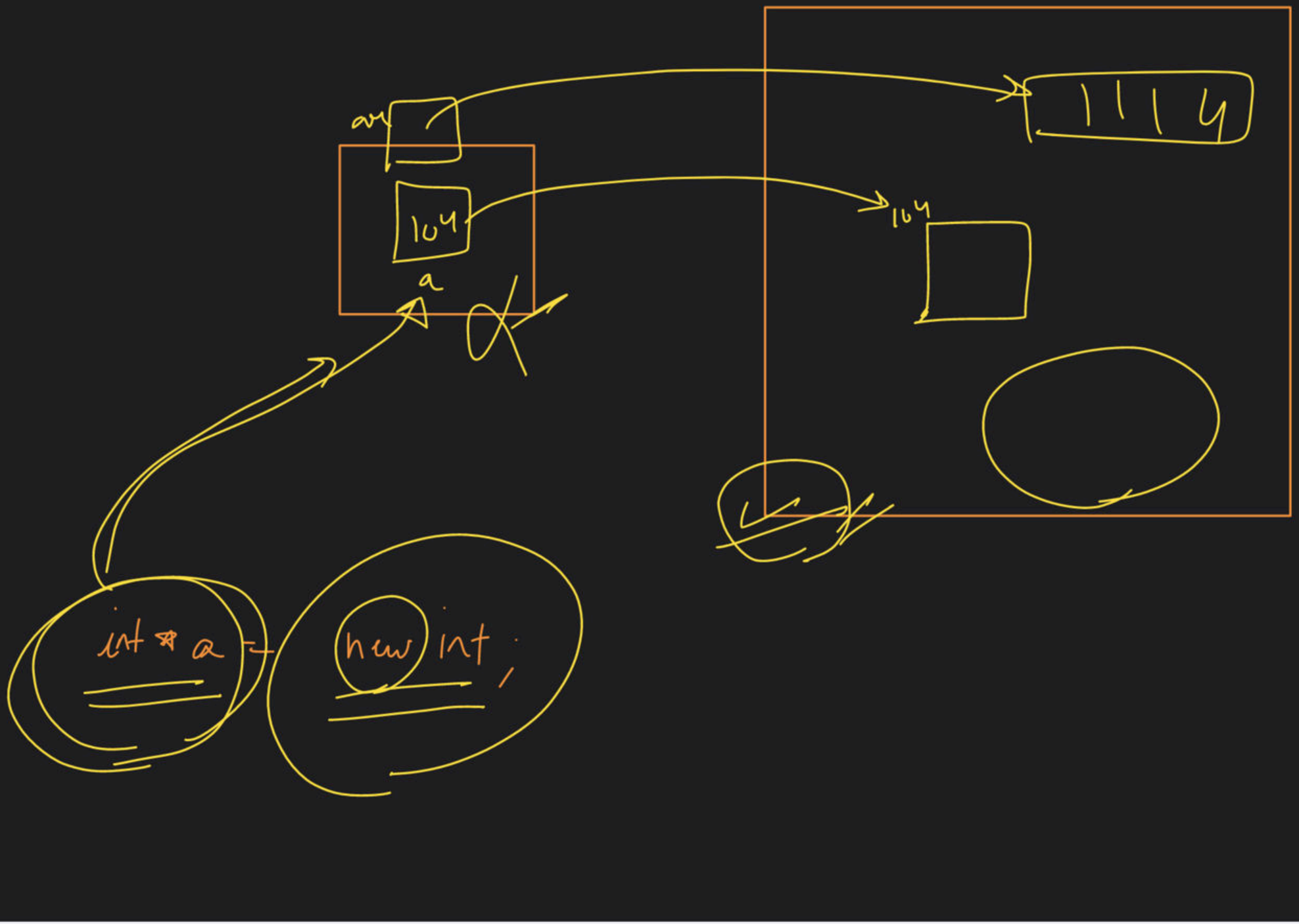

En
In
Diamond Problem
Basics
Poly
C.T vs R.T
H/W
Late Binding
Abstracti
Ex
JVM
Late Bind -> Bind
diff

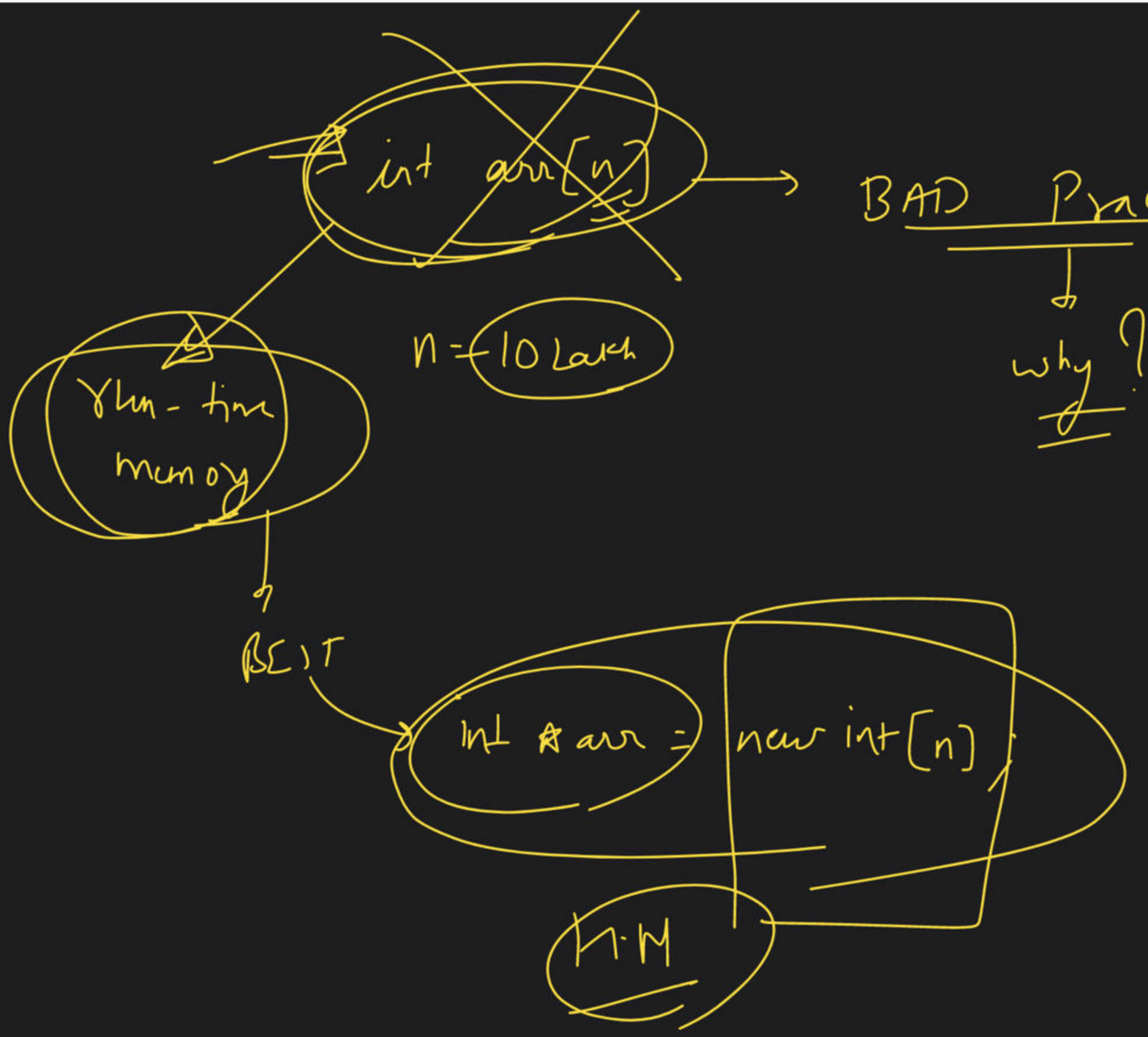
Dynamic Memory Allocation :-

Program start

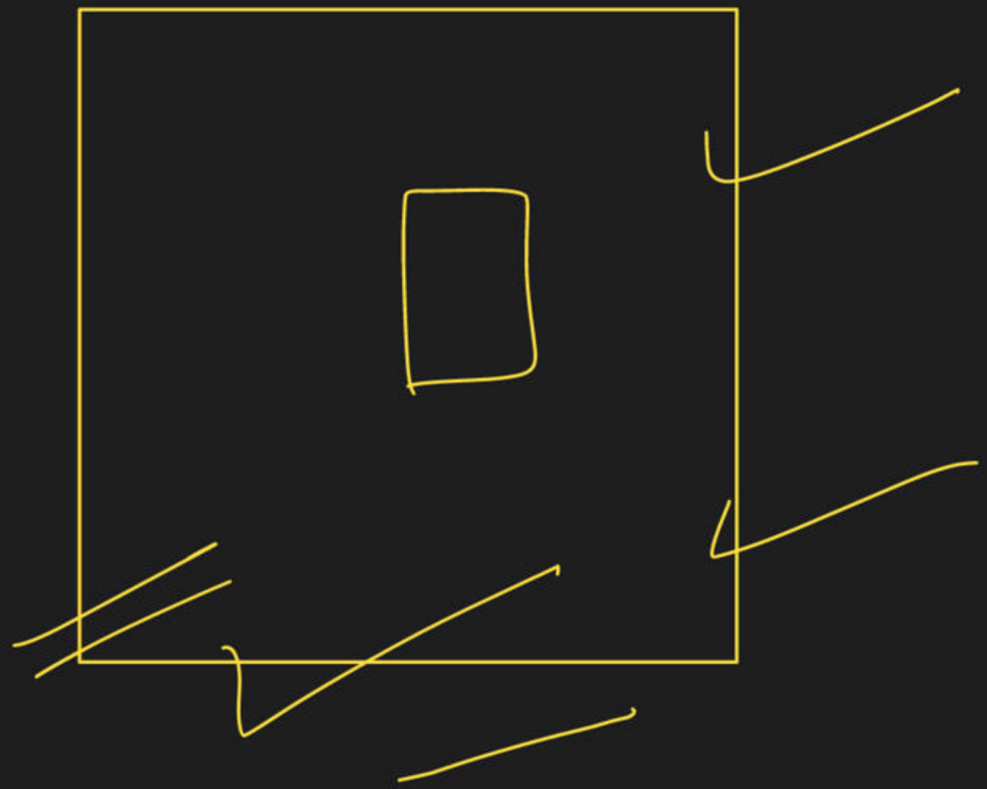


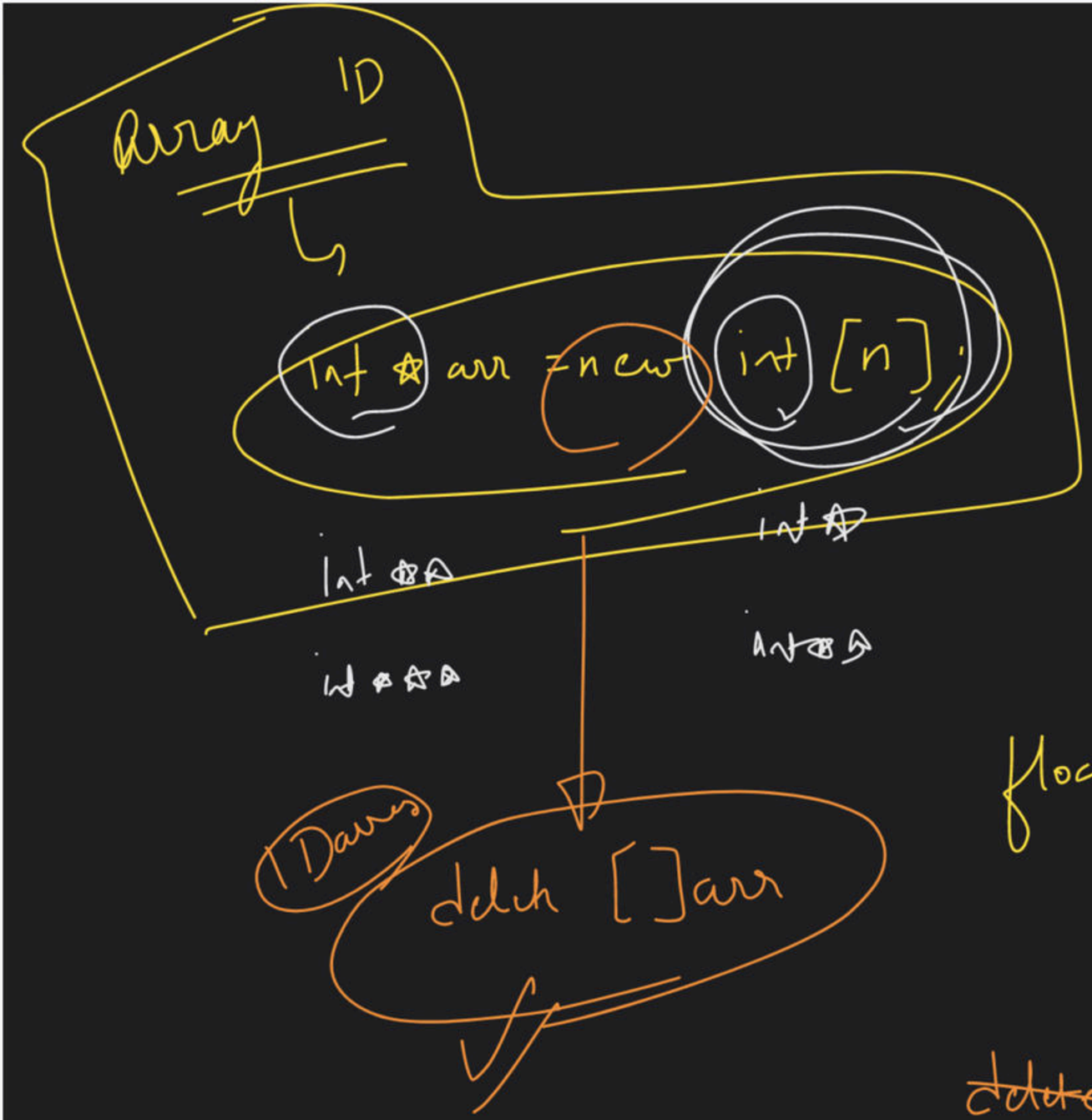

p/g
Stack
static allocation
Heap
int a = 1;
allocate -> new
new int
new int;
return address
10
new -> Keyword
int *a = new int;
de-allocate => delete


arr
104
a
104
int * a = new int;


int arr[n]
BAD Practice
why ?
n = 10 Lakh
Run-time memory
BEST
int * arr = new int[n];
M:M

int * arr = new int[n]





int

int * a = new int;

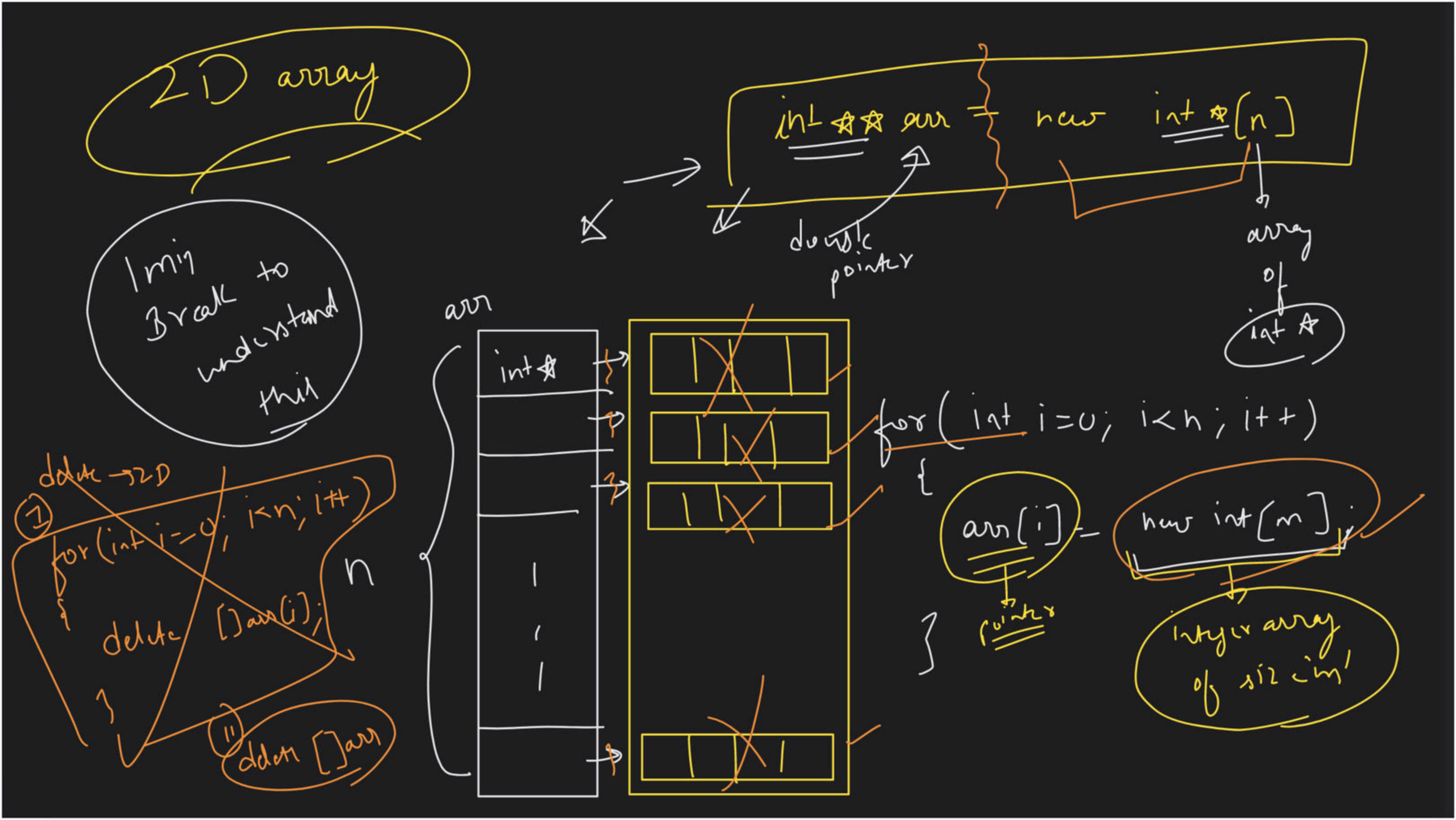
char

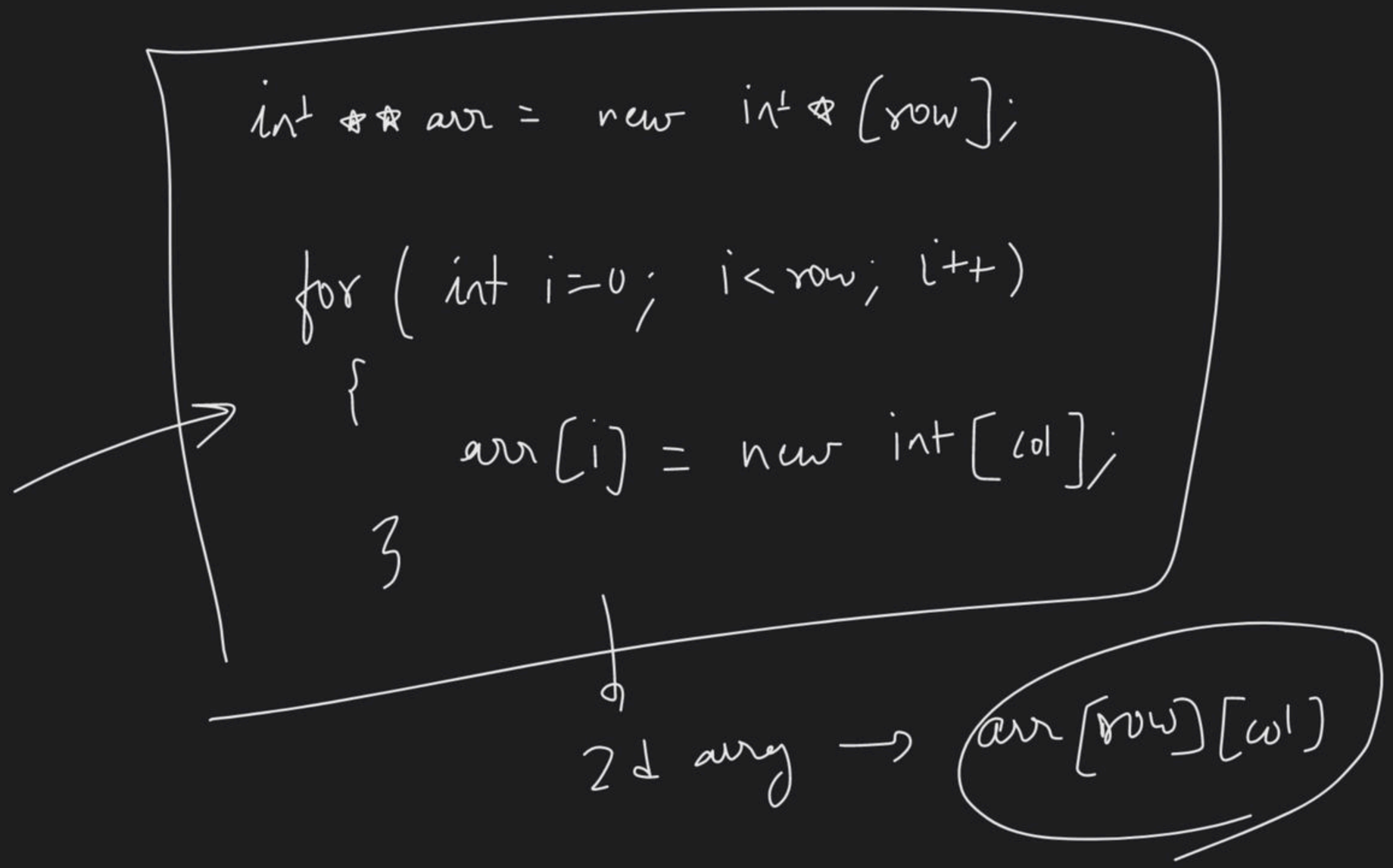
char * ab = new char;

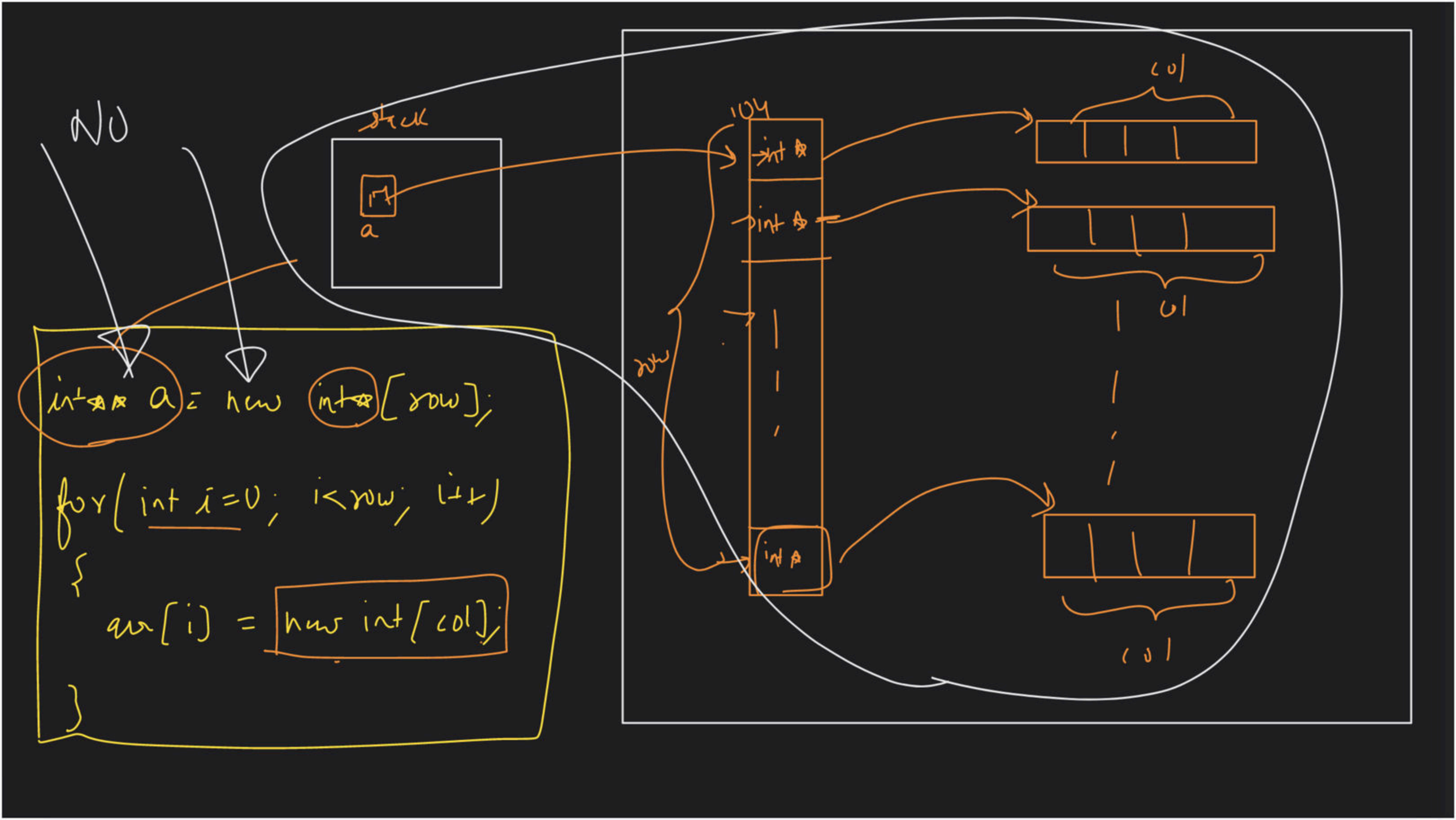
float

float * a = new float;

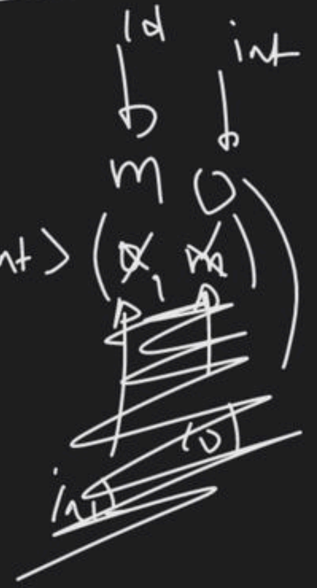
~~delete [] arr~~


2D array
Imp Break to understand this
int** arr = new int*[n]
double pointer
array of int*
arr
int*
for(int i=0; i<n; i++)
{
arr[i] = new int[m];
}
pointer
integer array of size m
delete 2D
for(int i=0; i<n; i++)
delete []arr[i];
n
delete []arr


int ** arr = new int*[row];
for (int i=0; i<row; i++)
{
arr[i] = new int[col];
}
2d arr → arr[row][col]


NU
stack
int
a
104
int *
int *
int *
row
col
col
col
int** a = new int*[row];
for(int i=0; i<row; i++)
{
a[i] = new int[col];
}

vector<vector<int>> arr(n, vector<int>(m, 0))

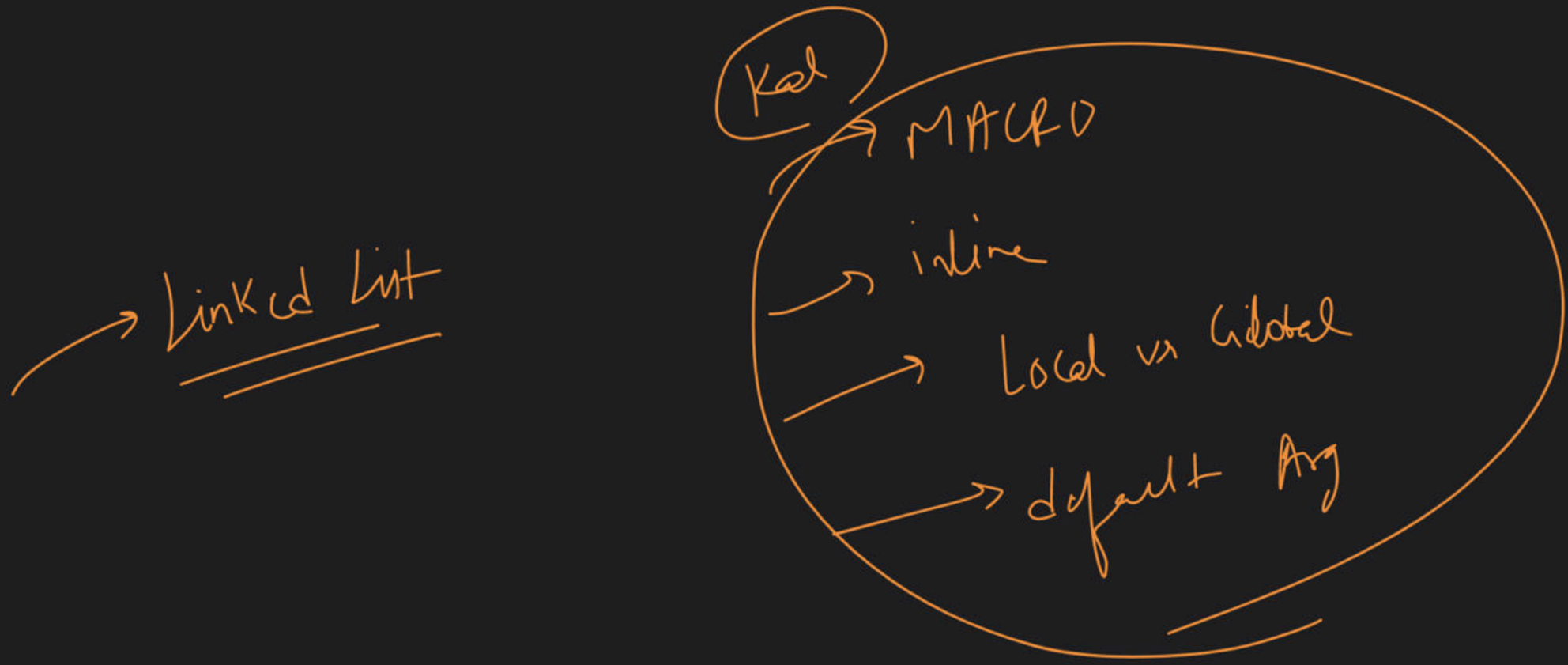


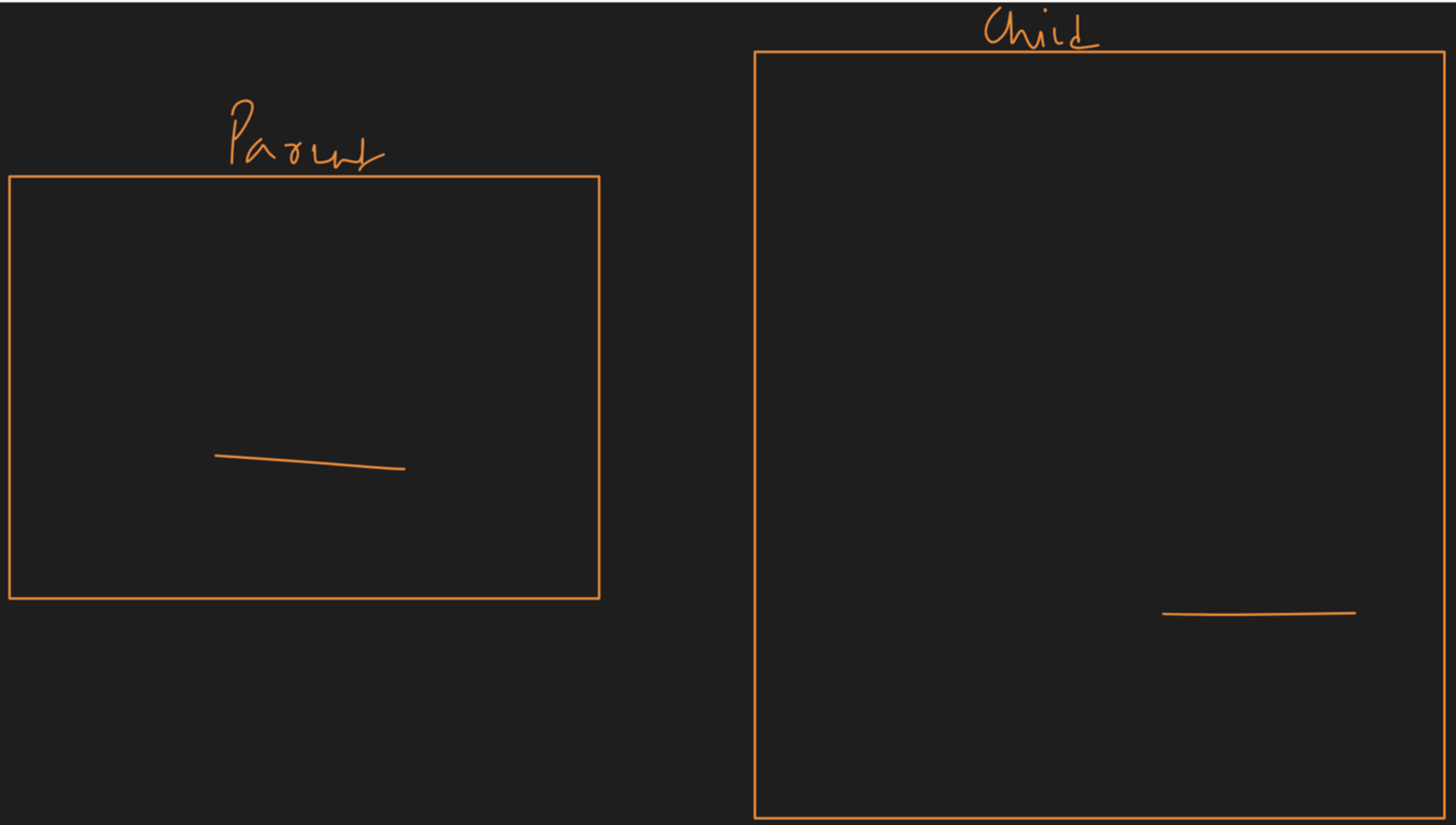
vector<vector<int>> arr(n, vector<int>(m, 0))

row

col

initialise value


Kal
MACRO
inline
Local vs Global
default Arg
Linked List


Child
Parent