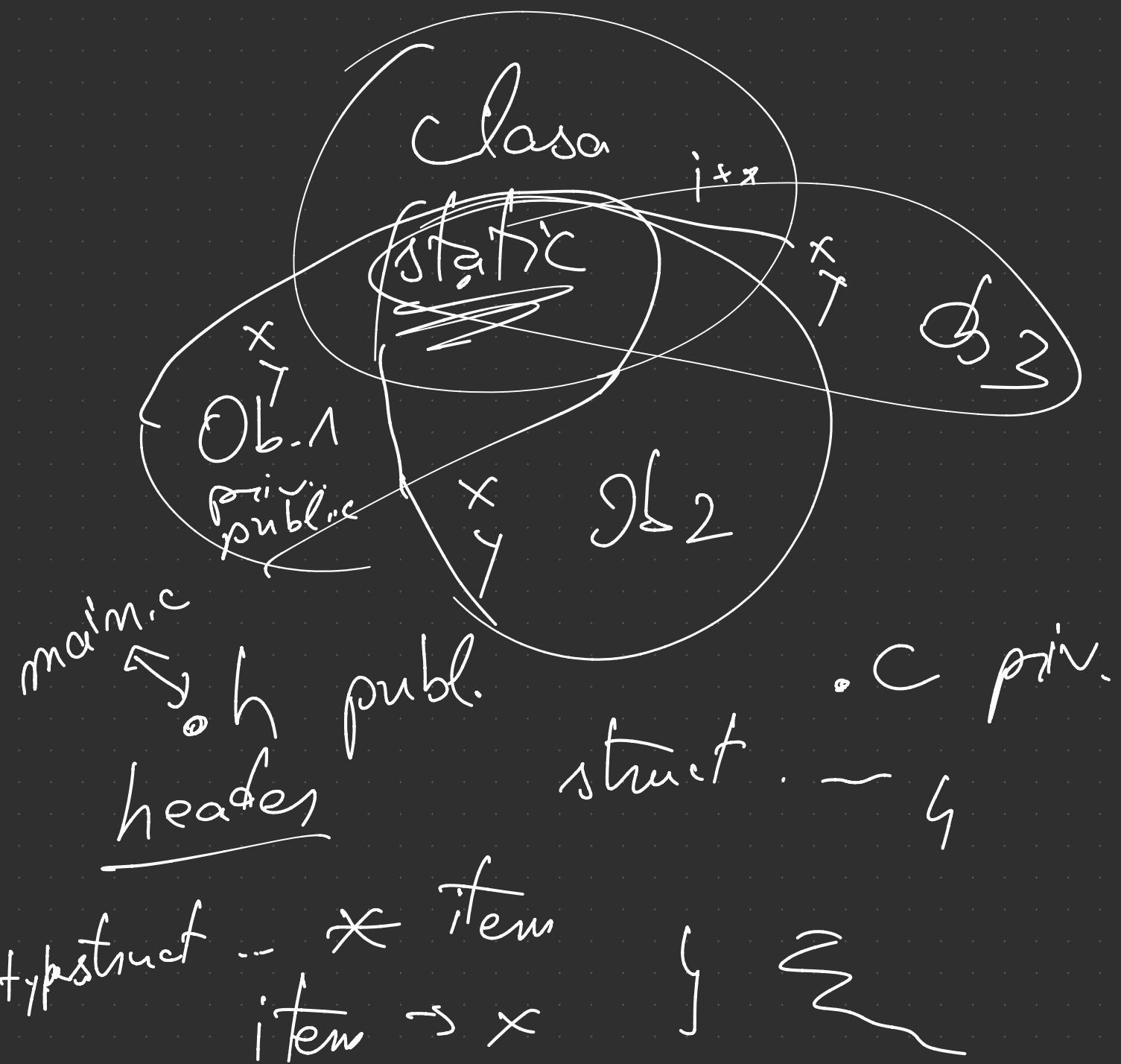


→ All data should be private within a class

Membri Statici



void ---(a)

{
 a=2

5

main

~~5~~

void. (l a)

static a = 2 → 2

int a = 0 → 0

→ Err. Compilare

These are sens pf. static

Utilizzare: Name(Classe.metodo)

Met. statice NU se execute
per un object al clasei

Static → proprietati ale CLASEI

Variabile FINAL

→ atribuție O SINGURĂ ATĂ

→ trebuie instantiat neapărat

- II - const

public final static int NOTA_MAX=10;

Unified Modelling Language

→ Diagrama de Clase

NUME CLASA	CLOCK
MEMBRI - hour : int - minute : int - seconds : int - count : int	mem. statici se sublimiază + public - privat
METODE + Clock(h: int, m: int, s: int) → param. + setTime() : void + print() : void + getCount() : int	= return value

Negruini Teoretice

Identitatea obiectului \rightarrow prop. unii
obiect de a îl face unic
(CNP, Adresa de mem.)

Referinta / adresa unde este alocat

Nume de variabila \neq identitate
(Aliasing)

Sferea obiectului - prop. obiectului
plus valoile curente pt. acele prop.

Comportamentul obiectului:

Interfata

Încapsularea

Compunerea

Suprincarcarea metodelor

(Overloading)

→ semnatura metodei

→ număr m. de parametri
→ sau tipul lor

```
print( int: )  
print( float: )
```

$f(a, b)$ $f(a)$

Clock()

{
 thus (12, 0, 0) -> prima instr.

}

Clock(h, m, s)

{
 setTime(h, m, s);

}



Constructor: multiplici

Transmitere parametri

→ prin valare

int a = 5;

Clock b = new Clock();

b.setTime(..);

obj. called(a, b);

a → stack (5) word called (int c,
clock d)
b → heap (0x45..)
adresa

c → 5

d → 0x45.. (adresa)

c nu afectează a

d afectează b

(d are ca alias pe b)

Valoarea b → adresa

~~→~~ stare obiectului

Nu \exists transmisie prin referință
în Java!

Class obj = null;

Var. locale NU se initializează implicit.

obj. stuff

Compilează ✓ Executie X

Clase și metode de bibliotecă

Orice obiect are metodele :

pub. boolean equals (Obj o)

pub. String toString

protected void finalize

public int hashCode()

→ definite în clasa Object
(default)