hourly_rate
Submit time carde()

Submit time stamp

last time paid

~~method = submit~~

select method of payment()
~~Submit sales receipt()~~

monthly_salary
commission rate
Submit sales receipts
last time paid

employee union

weekly dues
Service charges

employees → all employes get salary
→ all have name, id

hour — month

each of it has own implementation of get_salary.

interface
↳ ~~service cha~~ ~~submit~~

union → class/interface
submit service charges()
~~weeklydues()~~

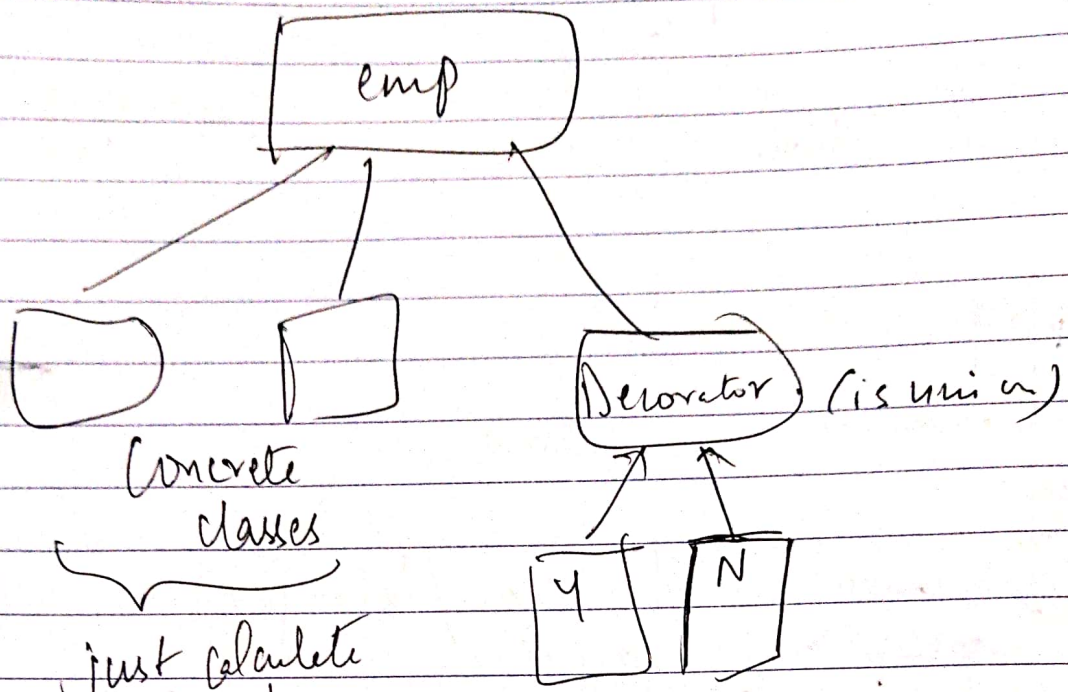
union — not union
is in union
getSalary() { deduct ~~fees~~ dues
~~emp~~
in_union

employee
~~union u~~ → interface
~~get salary - w~~
union → interfaces
no_union
get ~~other~~ final (salary after deduction
(~~union~~ u . getsalary)

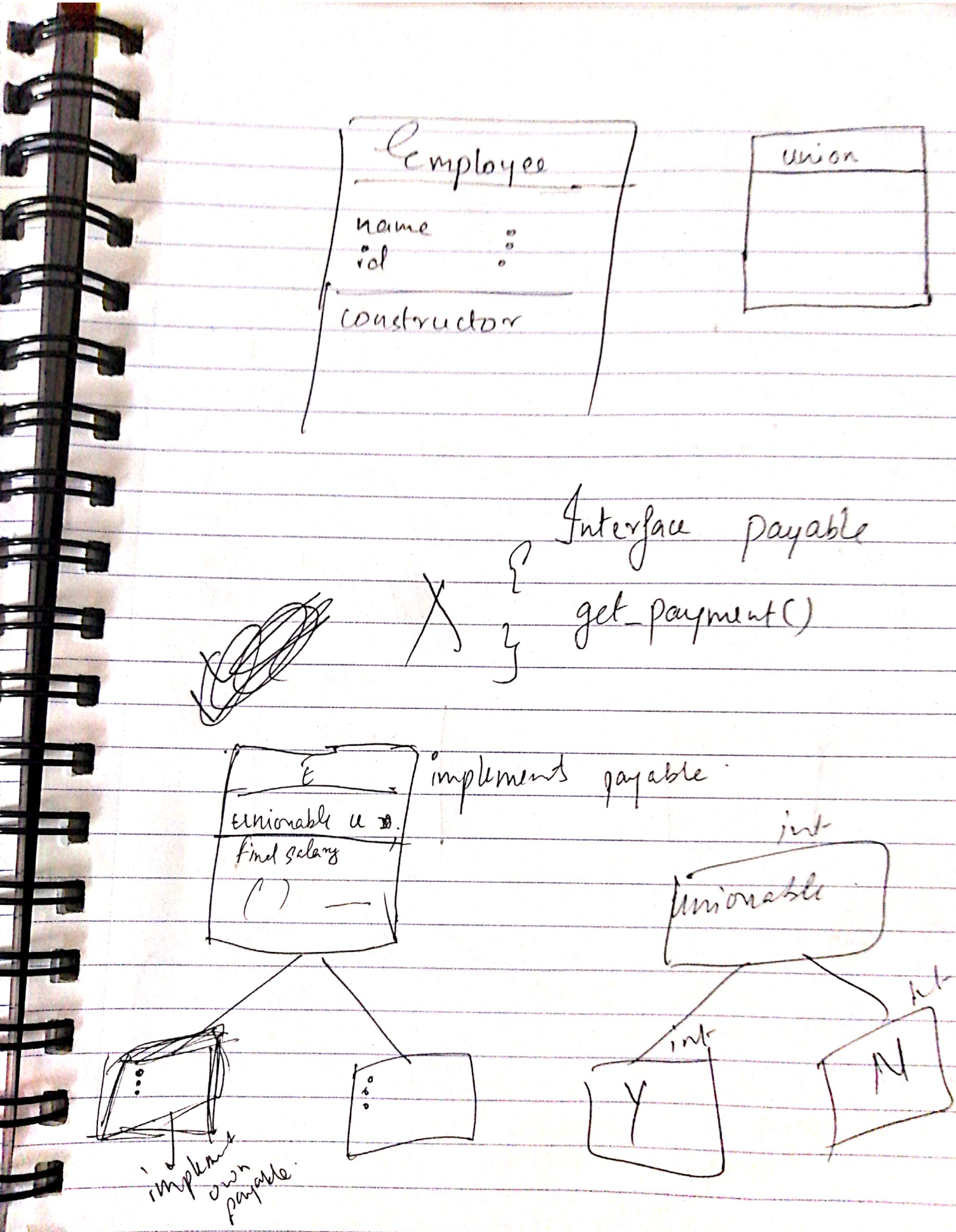
M

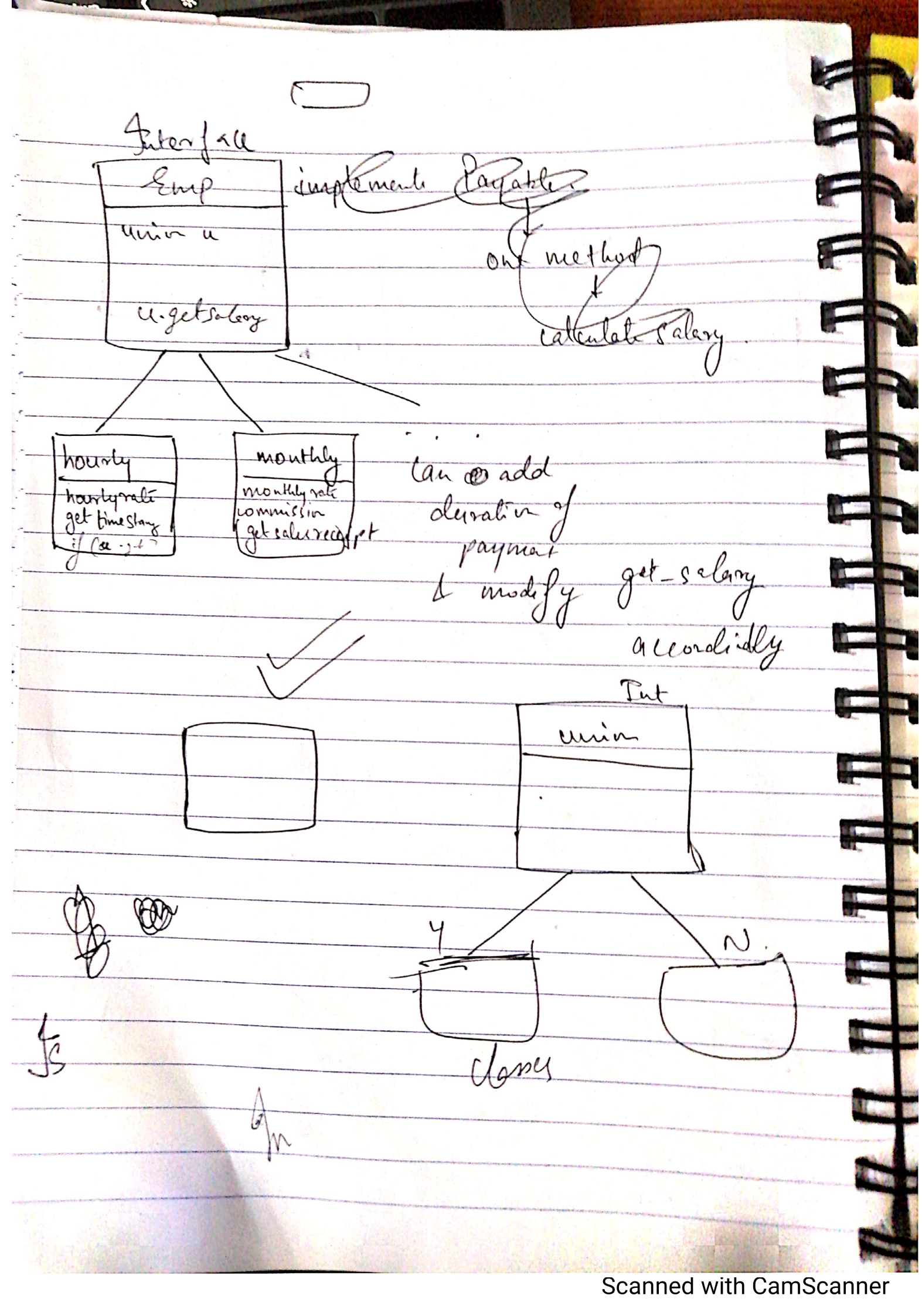
H
~~sal~~ get_salary()
final salary()

Scanned with CamScanner



- Employee e (type)
{
if type = h
new h_e()
" " = m
" m_e()





Friday.

- need to pay weekly employees
- need to pay commissions to some.

Database : employee payroll

table emp

get all employee ids with isWeekly = true

array

for (id in array) :

{

~~get temp.~~ get method of payment

create object Employee Method

↓

is obj ke call function pay()

↓

"temp Inst to ... VIA ...)

See connection

main →

Output : Select an option : 1-7.

if 1 →

add_employee();

Take all inputs, create object.

" insert into emp values ---------- ;

if 2 →

get employee id.

remove from table emp where id = input

3 → Submit time card

output : aaj ka date

I/P : no. of hours, emp id

call get_time card()

→ update temp

" UPDATE TABLE, SET temp = ~ where id = ~ .

Feature 01 - → initialised too

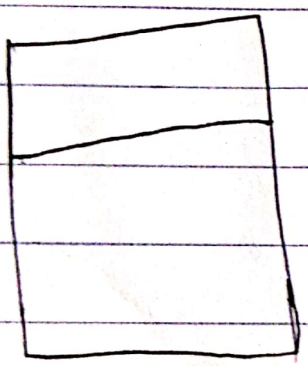
added constructors payroll() changed database management

hard

@ arraylist

1st commit → dev classes made.

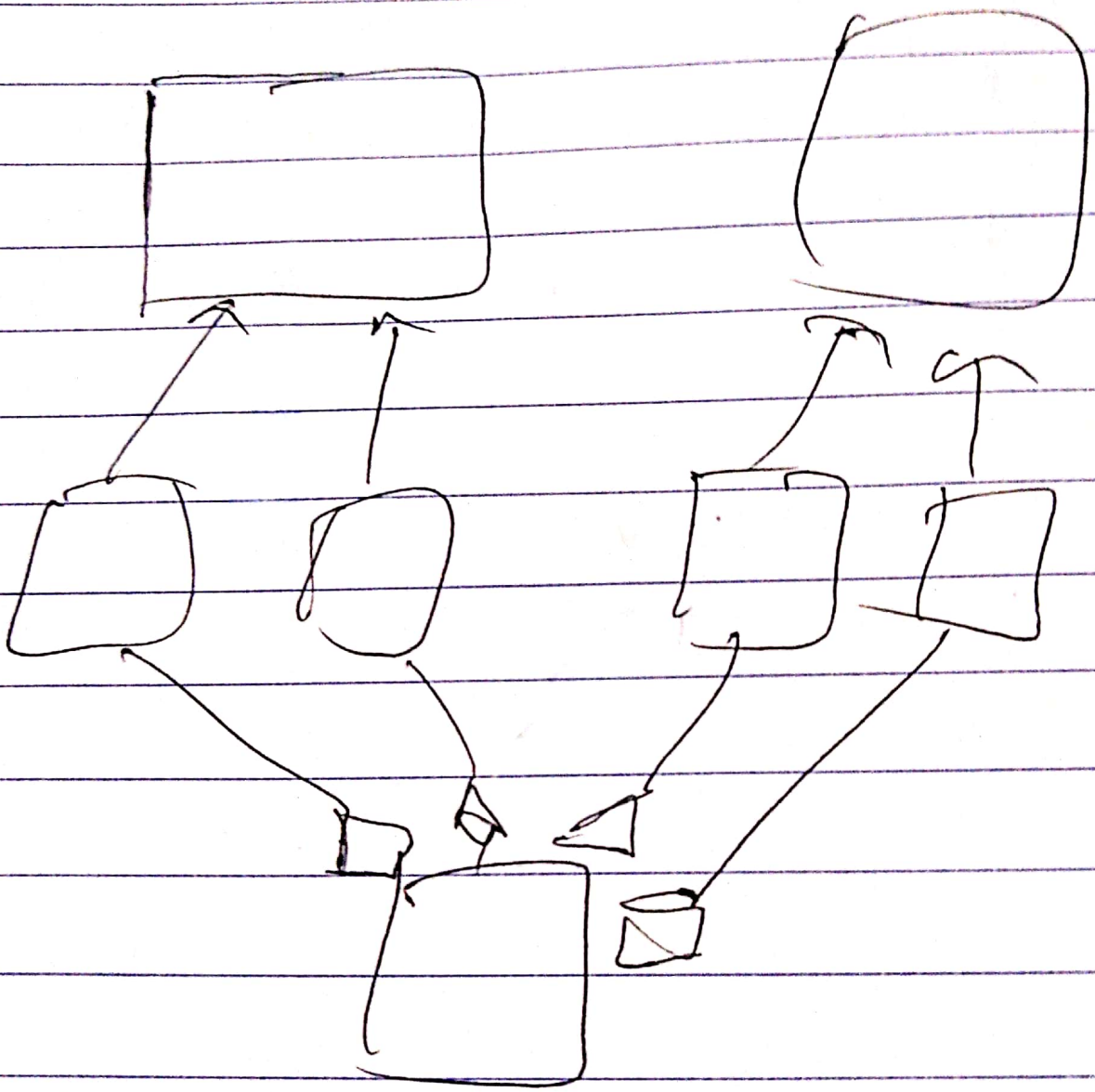
0 ⟍ features



ASSUMPTIONS

- Joing of 1st of month.
- Bank details?
- Only today hai time card

Common methods → pay()



Design towards change.

How to submit time cards?

Employee

name
empid
currentDues
calculateSalary(); // weekly or monthly
calculateDues(); // weekly for all

2 types of employees

WEEKLY
hourly rate
overtime rate

Flat salary.
commission

Single responsibility : pass methods.

Single salary generator
↳ diff methods for 2 groups.

Design for :

- Single Responsibility (Lambda ?)
- Loose coupling (then inh fine ?) (no composition in java)
- Changeability (HOW ??)

Design for invariance

- Generics ??

* Func. is not func. type in Java.

Type of Instance of ~~abstraction~~ functional interface

- Design towards immutability. (final)
- Wherever abstract class → use interface.

In Java → func. are obj
→ Building blocks are classes

Design by convention rather than config.
↓ ↓
Interface Concrete class

Composite better than inheritance, but can we use in Java?

Passing behavior to behavior.

Encapsulation is spl case of localisation