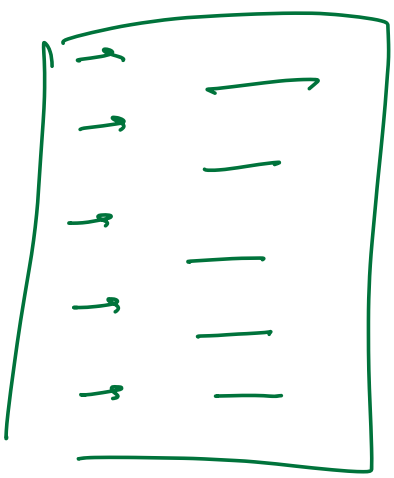
Splitwise-2 (Will start at 9:10 PM)

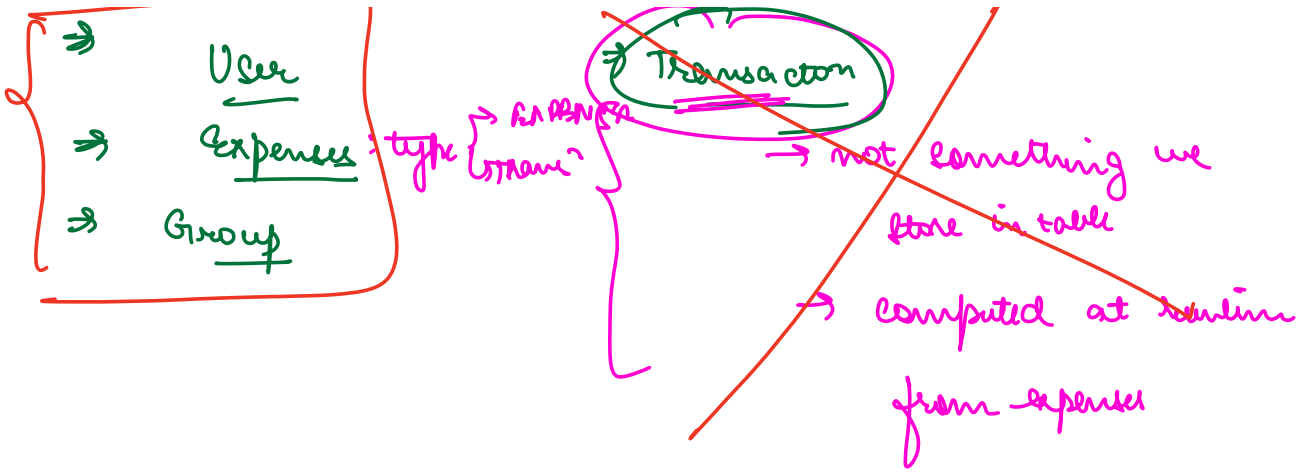
⇒ ① Class Diagram
② Schema Design
③ How to take input from command line

Class Diagram

→ Nouns
→ Visualization

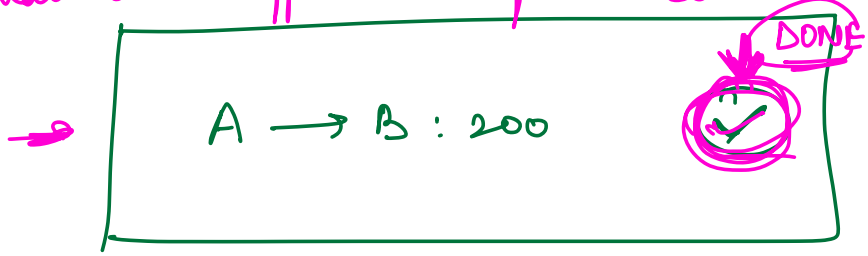
① go through every req
② find nouns in the req
③ we check if we have to store info about the noun.
(if a noun is there which represents an entity about which we are not storing any info in our system
⇒ it will not be a model)
④ We create a class for each of those





List <Transaction> settleUp(List<Expense> expenses) {

What will happen when person clicks on tick

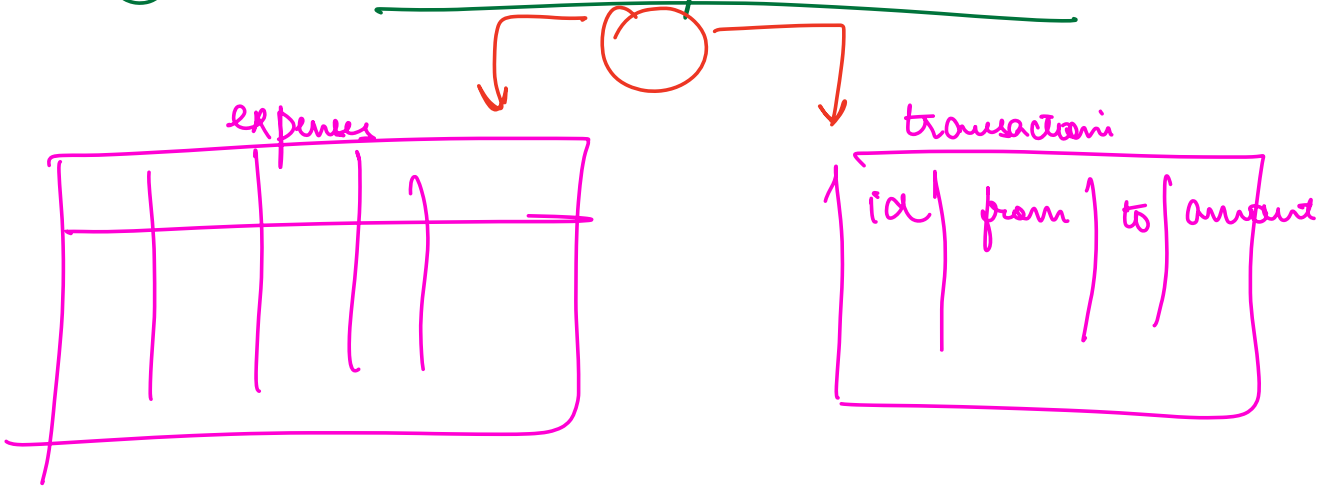


But I also have to store if a person has paid back the money

⇒ So that in future settle-up I can consider how much money has already been paid when suggesting transaction

Way 1

Store in a separate transactions

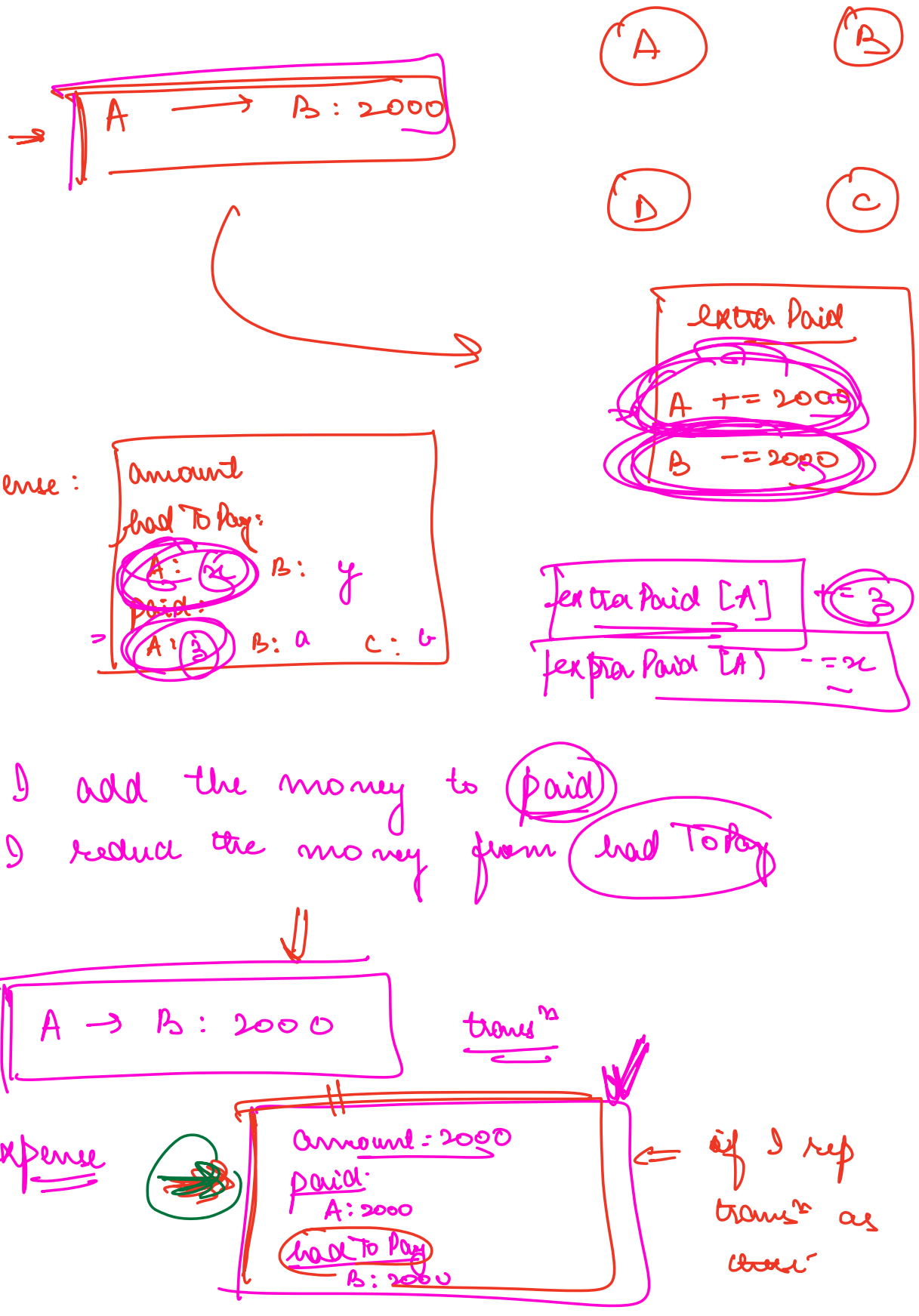


Issue: Next time when someone clicks on settle up you will have to go through 2 tables:

1. expenses
2. transactions

Way 2

Can I somehow use the expense class to store execution of a trans^n as well.


A
B
A → B : 2000
D
C
extra Paid
A += 2000
B -= 2000
expense :
amount
had To Pay:
A: x
B: y
paid:
A: 3
B: a
C: b
extra Paid [A] += 3
extra Paid [A] -= x
I add the money to paid
I reduce the money from had To Pay
A → B : 2000
trans^n
expense
amount : 2000
paid:
A: 2000
had To Pay
B: 2000
if I rep trans^n as above

Dummy expense

Q1 Will this dummy expense help me have the same outcome as having a trans^{n}

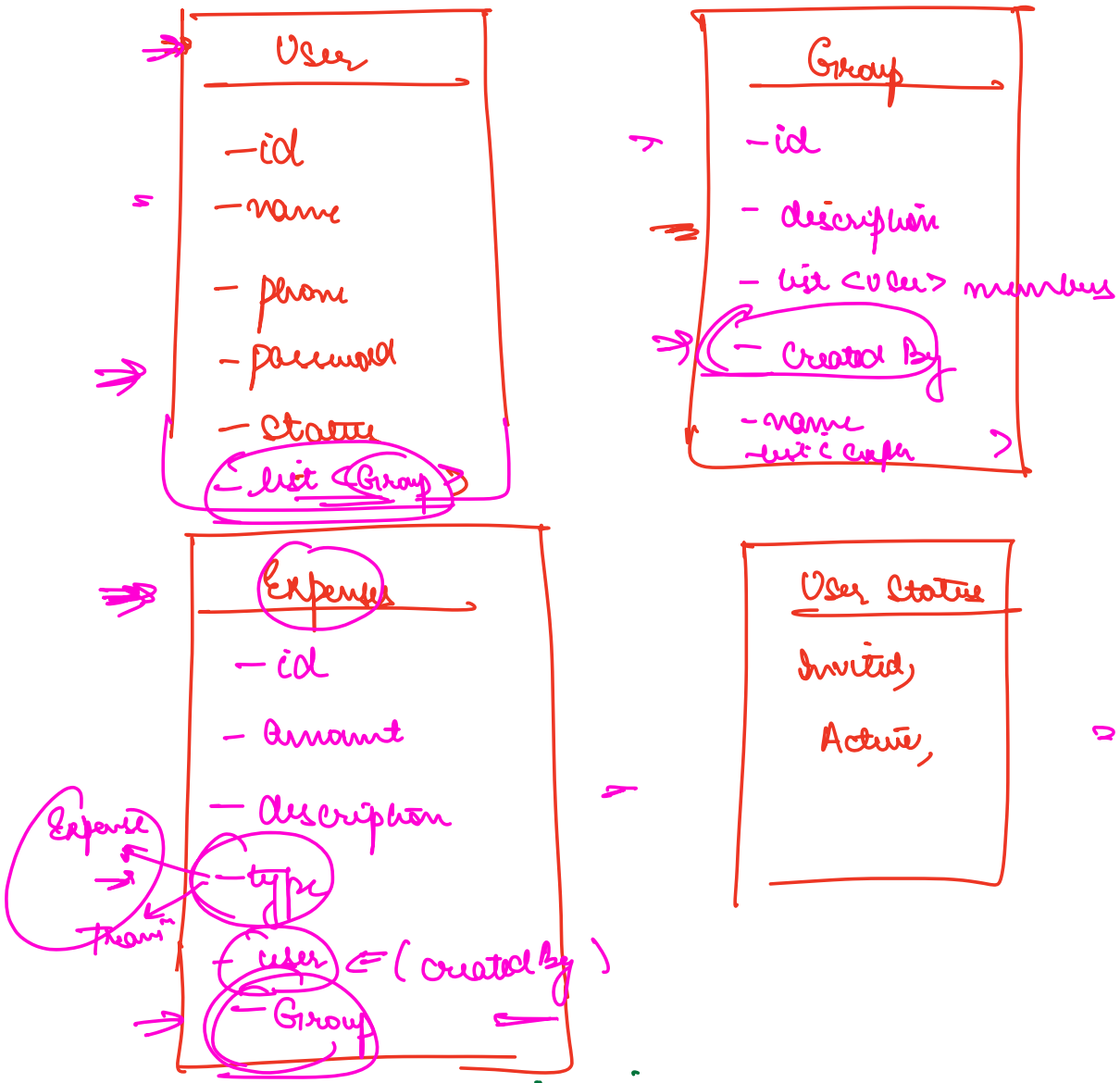
⇒ Yes

Q2 I will have to go through only 1 table to compute trans^{n} for next time

1 Find Balance money of everyone.

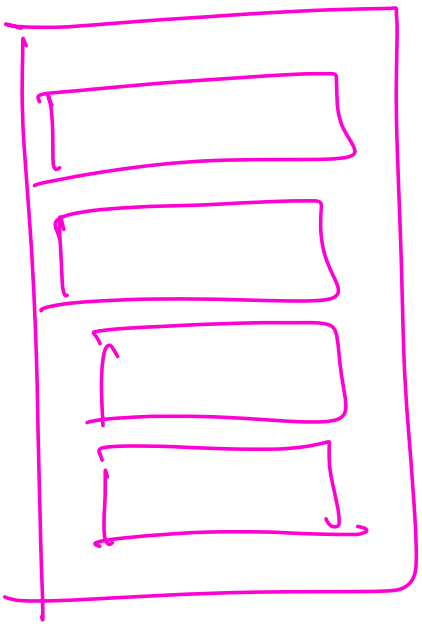
```
Select *
    from expenses
    where type = 'transfer'
    order by created_at
```

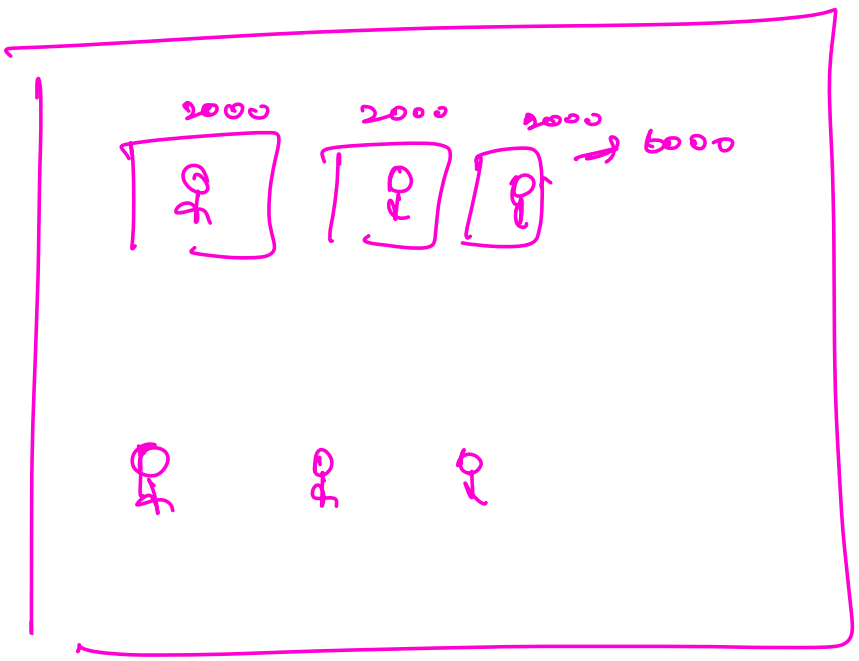
CLASS DIAGRAM



⇒ ideally prefer not having a list in models

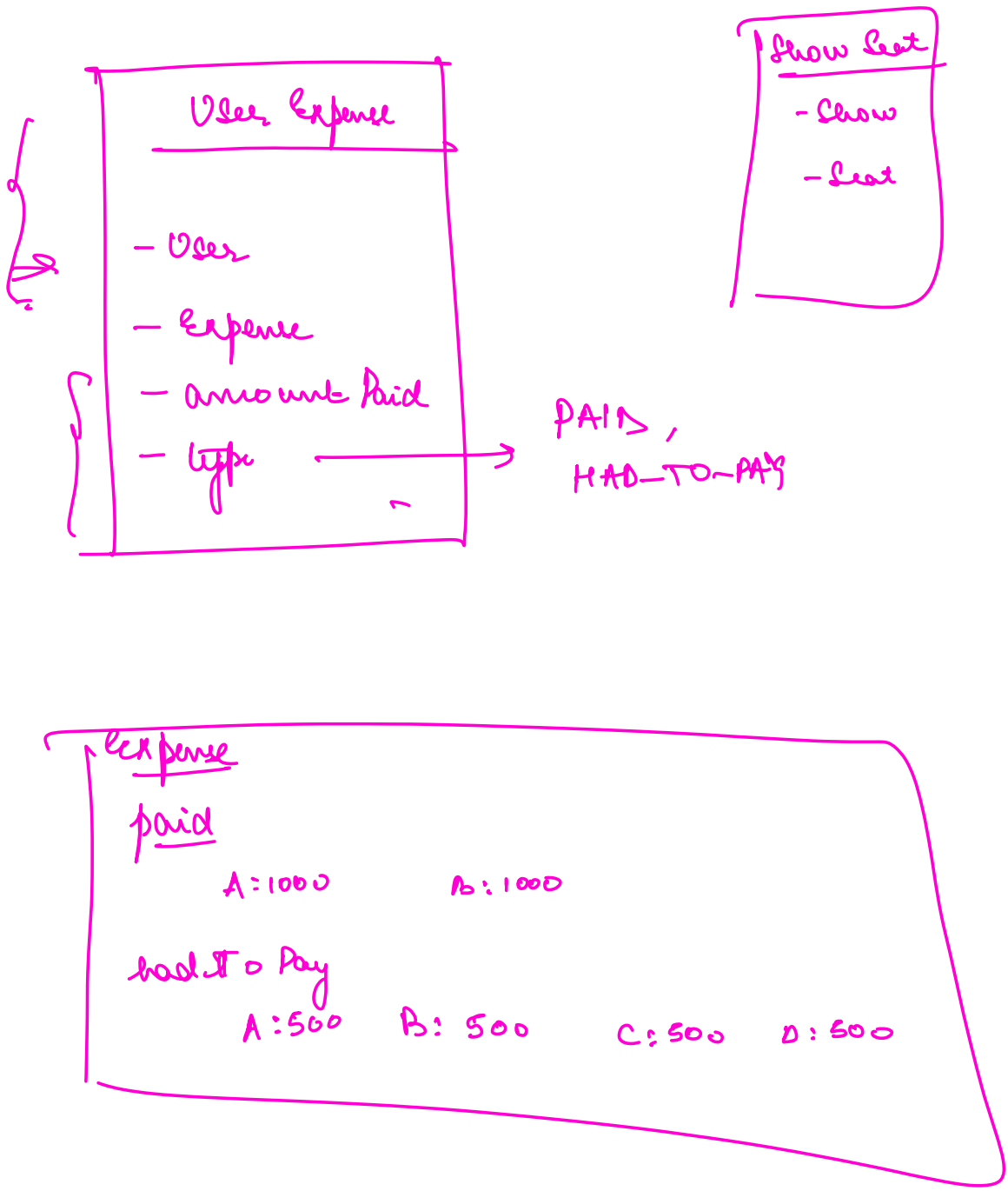
mapping class: iff rel^n has attrs



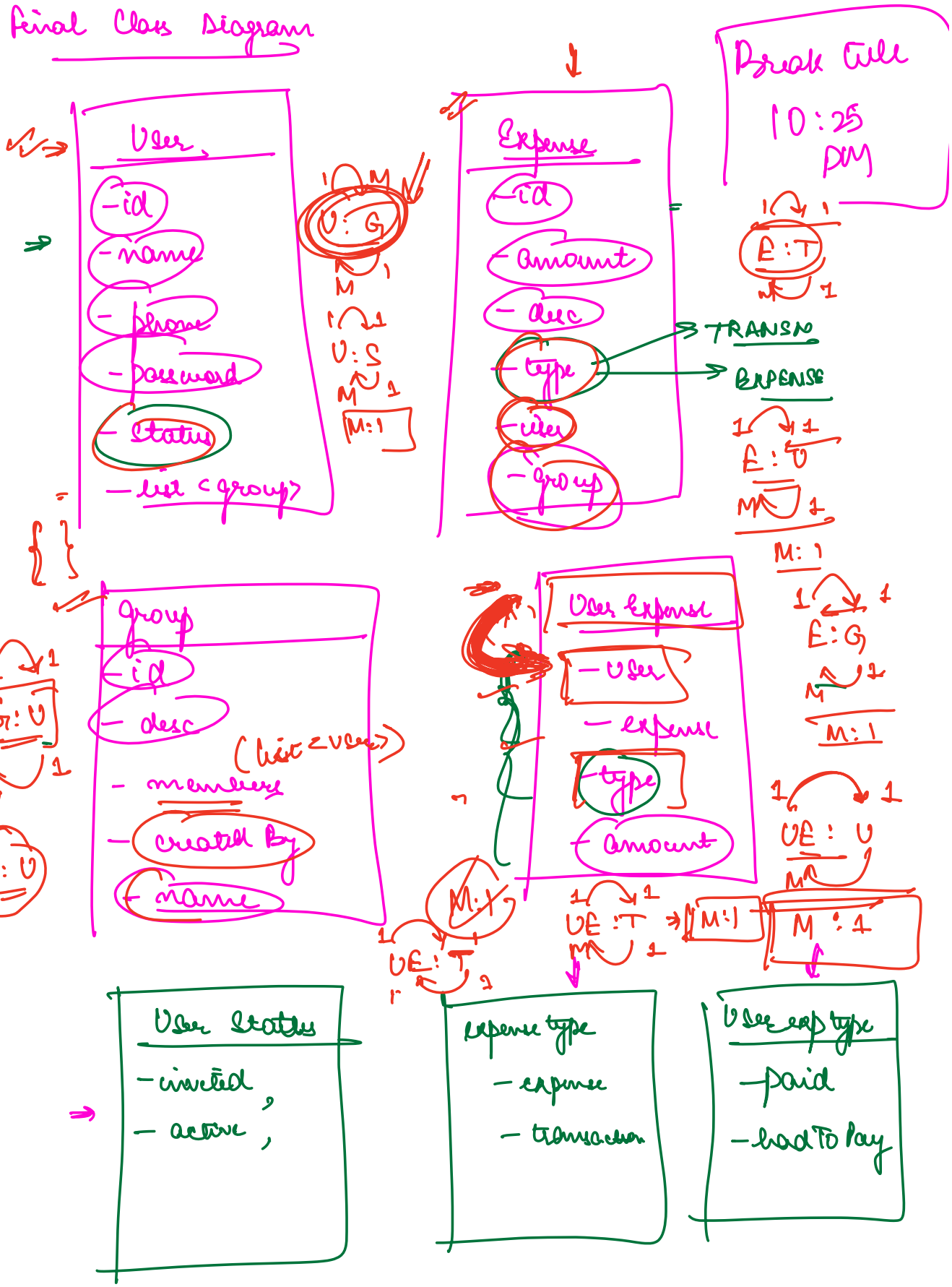


Who paid what ⇒ rel^n b/w users and expense

⇒ with attrs


User Expense
- User
- Expense
- amount Paid
- type
PAID,
HAD_TO_PAY
Show Cost
- Show
- Cost
Expense
paid
A: 1000
B: 1000
had to Pay
A: 500
B: 500
C: 500
D: 500

Final Class Diagram



Schema Design

1. for every class create a table
2. for primitive attr

User

id	name	phone	password	status_id

expense

id	amount	desc	type_id	created_by_id	group_id

group

id	desc	name	created_by_id

User_expense

amount	user_id	expense_id	type_id

User_status

id	value

exp_type

id	value

User_exp_type

| id | value |

User_groups

| user_id | group_id |

How to approach Design problem

How to take inputs from command line

1. User will give inputs to the command line 1 by 1
2. App will process those inputs and take relevant action

Splitwise App {

main() {

while (true) {

String inp = " ";

if inp. starts with ("Register") {

else if (inp. starts with ("Update Profile") {

Parse

}

}

}

→ else if (______) {

______

______

→ } else if (______) {

______

______

______

→ } else if (______) {

______

______

}

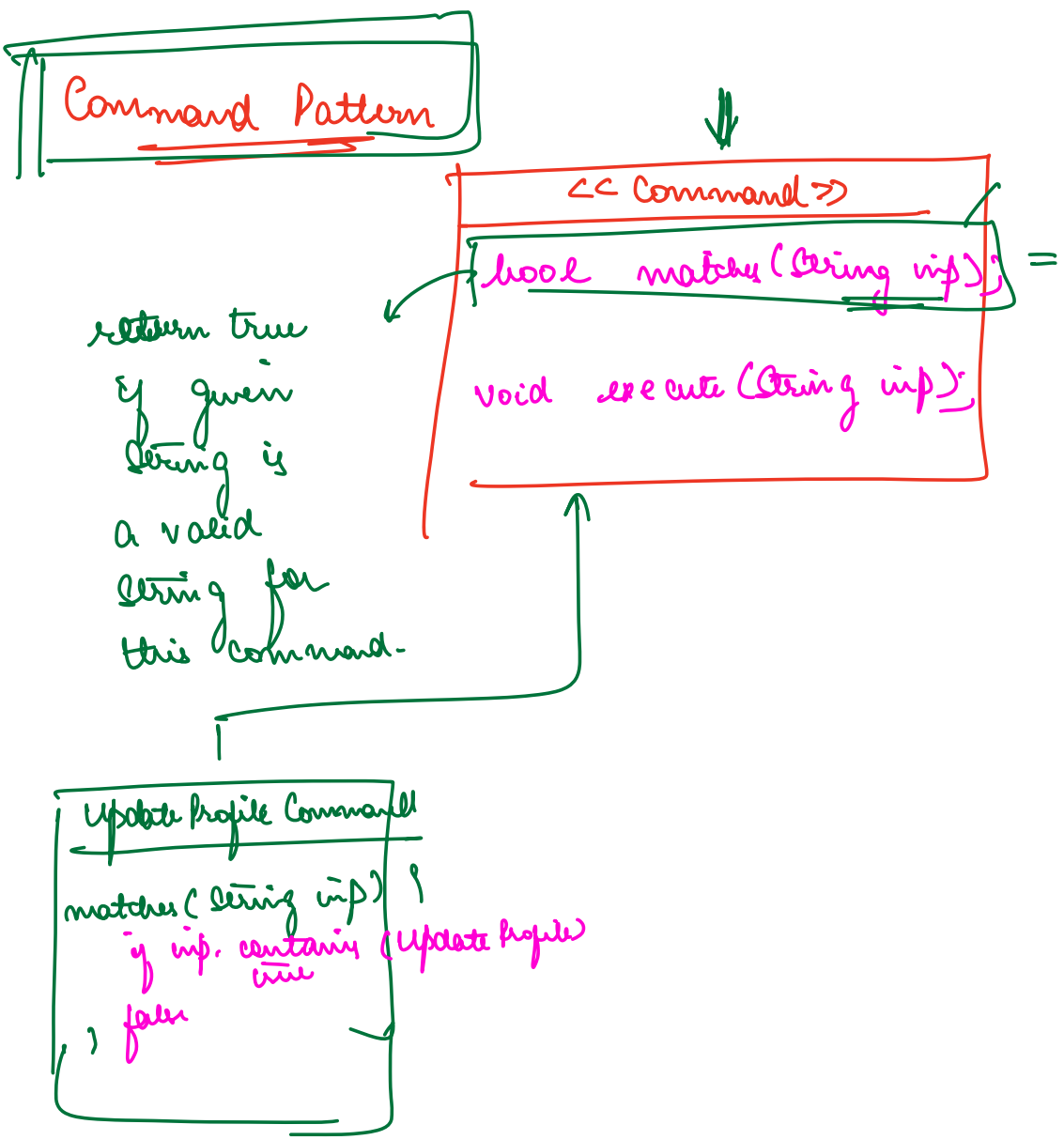
If we implement like above, we will violate SRP and OCP

→ String input = "______ xyz ______";

① Check which action does this input want me to take

② Parse input and execute action


```
username = inp.get(1)
phoneNo = inp.get(2)
UserController.updateProfile(username, phoneNo)
```



execute (inp) {

controller. update

profile (_____) ;

SplitwiseApp.java {

main () {

⇓

SRP ✓

OCP ✓

List < Command > { } ;

while (true) {

w String inp = ' ' ;

⇒ for (Command c : commands).

if c. matches (inp) {

c. execute (inp)

H/W

1. Create Splitwise SpringBoot App
2. Create models for each of the class
3. Ensure tables are created in DB by ORM (MySql)