

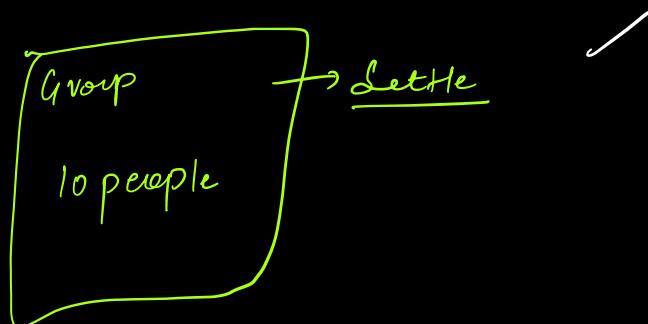
Today's Agenda :-

- 1) Algo to find min no of txn required for settle up.
- 2) Class Diagram
- 3) Schema Design
- 4) Taking Inputs via Command line

Command Design Pattern.

↳ Behavioural.

5 expense

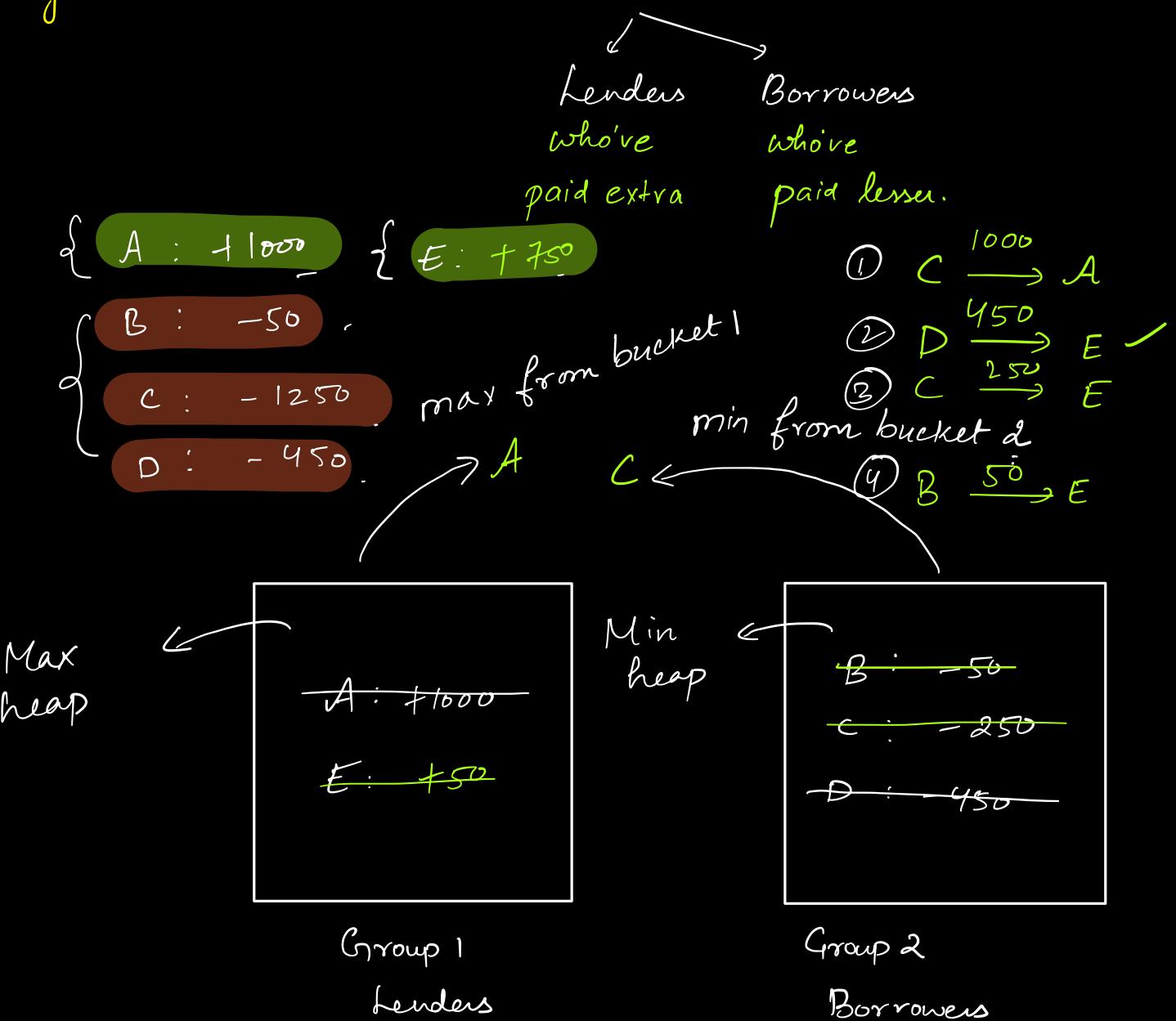


So in this below mtd max n transactions will happen and this is heap so time complexity will be $N \log N$

$$\# \text{lended} = \# \text{borrowed}$$

$$\frac{N \text{ no of people}}{N \log N}$$

Algo:- Divide the users into 2 buckets ✓



Borrowers should pay to the lenders.

I want to list down the txn required for the settle up

Entities :- Nouns about which you want to store
 ↓
 data in your database.

Go through
 req gathering

User ✓

Expense ✓

Group ✓

Transaction

User
<i>id</i>
<i>name</i>
<i>email</i>
<i>password</i>

who paid how much

who borrowed how much

£ 1000

A 200

A 250

B 800

B 250

C 250

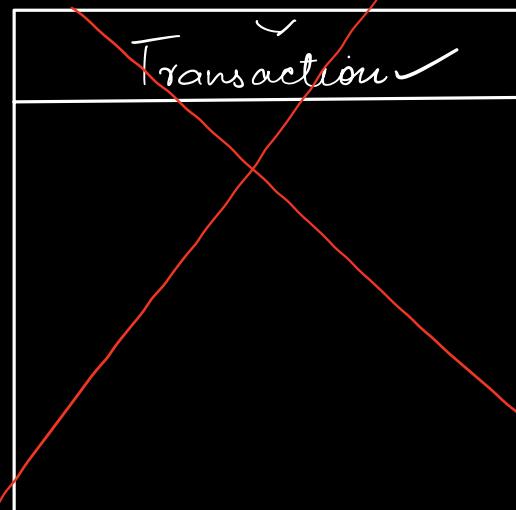
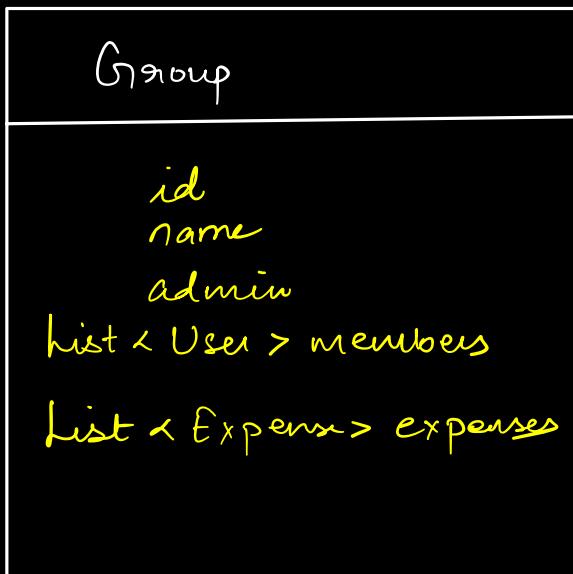
D 250

lenders :- { (A, 200), (B, 800) }

borrowers :- { (A, 250), (B, 250), (C, 250), (D, 250) }.

Expense
<i>id</i>
<i>description</i>
<i>amount</i>
<i>group</i>
Map<User, amount> lenders;
Map<User, amount> borrowers;
ExpenseType

ExpenseType
DUMMY,
ACTUAL

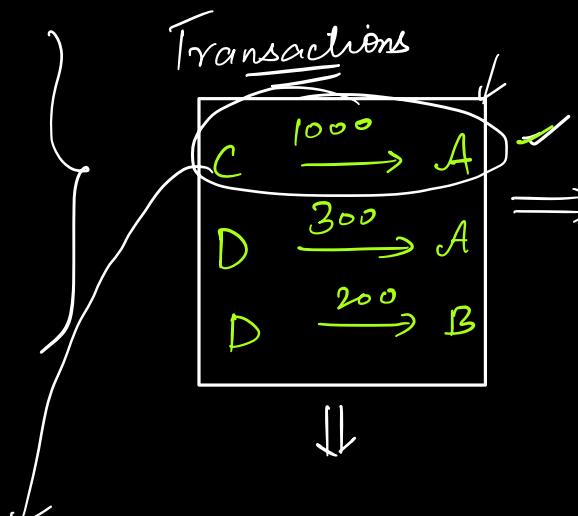


Example

{ Expense on a dinner for ₹ 3000
 lenders A : 2000 ✓ B : 1000 ✓
 borrowers A: 700 ✓ B: 800 ✓ C: 1000 D: 500

Settle up \Rightarrow `List < Txn >` required so that everyone's balance becomes zero.

$A : +300$
 $B : +200$
 $C : -1000$
 $D : -200$



Doesn't make sense to persist this data in the db.

When a user marks the transactions as done / completed, we will have to mark the effect of the txu on the amount.

→ Persist somehow the executed txu



Create a dummy expense.

$C \xrightarrow{1000} A$

amount : 1000

lender : C : 1000

borrower : A : 1000



$$\left\{ \begin{array}{l} A : +1750 - 450 = +1300 \\ B : -50 \\ C : -1250 \\ D : -450 + 450 = 0 \end{array} \right.$$

$\xleftarrow{\hspace{1cm}}$

$C \xrightarrow{1250} A$

$D \xrightarrow{450} A$

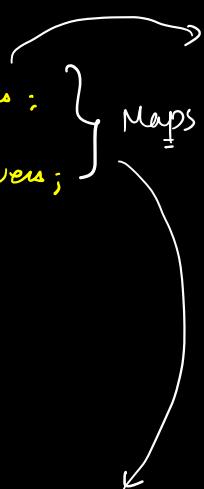
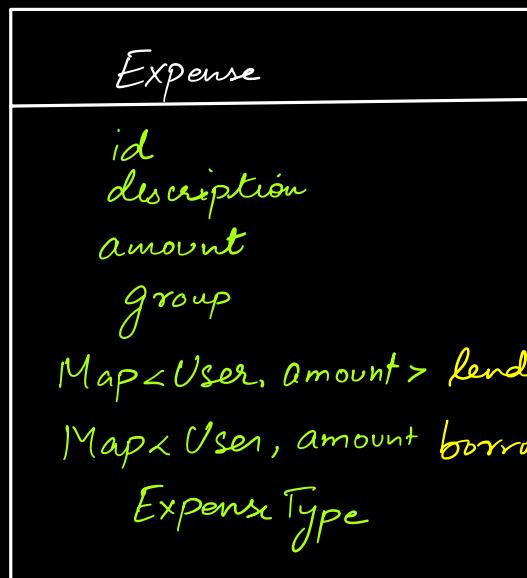
$B \xrightarrow{50} A$

dummy expense

Amount : 450 ✓
lender : D : 450
borrower : A : 450

Conclusion :- Instead of creating a separate class,

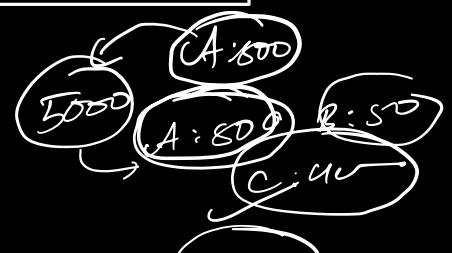
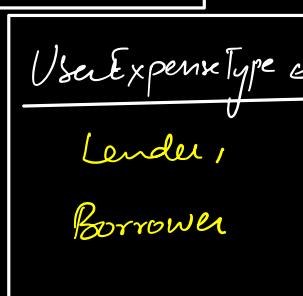
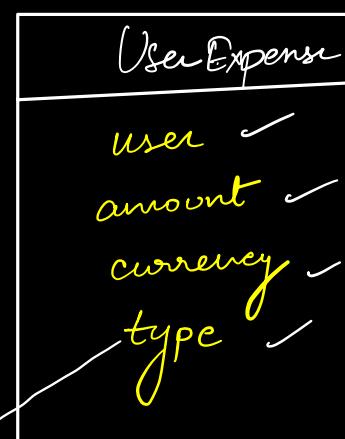
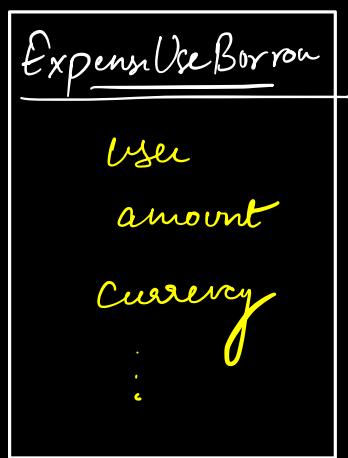
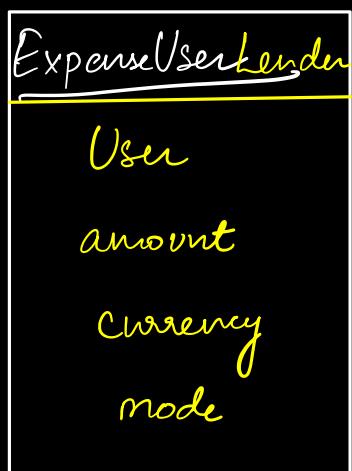
We'll create a dummy expense which have the same impact as of a executed / completed txu.



currency

$\text{£ } 5000$ ✓
 lend: $A : 2000$ ✓ $B : 3000$ ✓
 borrow: $A : 1000$ $B : 1000$
 $C : 1000$ $D : 1000$ $\text{€ } 1000$

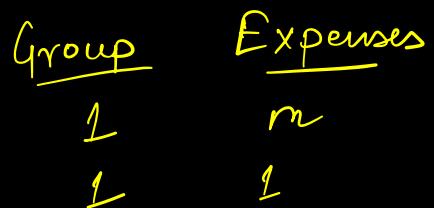
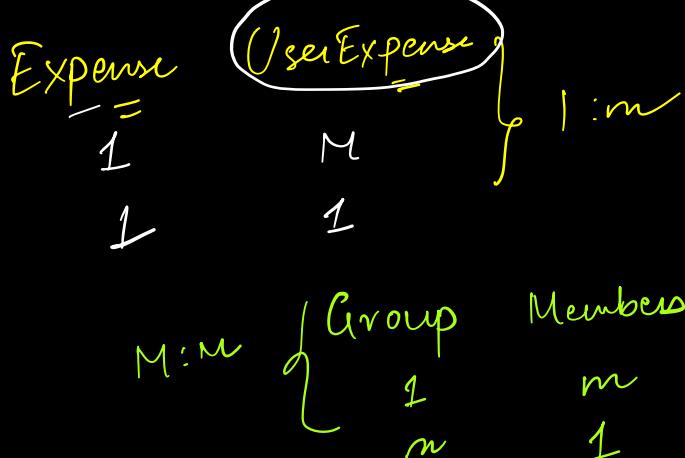
$100 \$$
 lend: $A : 80 \$$ ✓ $B : 20 \$$ ✓
 borrow: $(User), (amount, currency)$



Expense
<u>id</u>
<u>description</u>
<u>amount</u>
<u>group</u>
<u>list < UserExpense ></u>
<u>ExpenseType</u>

Lend 5000 E
 A : 1000 B : 4000
 borrow A : 2500 B : 2000 C : 500

(user : A, amount : 1000,
 currency : INR, type Lender,
 { — B, 4000,
 INR, Lender } ,



User
<u>id</u>
<u>name</u>
<u>email</u>
<u>password</u>

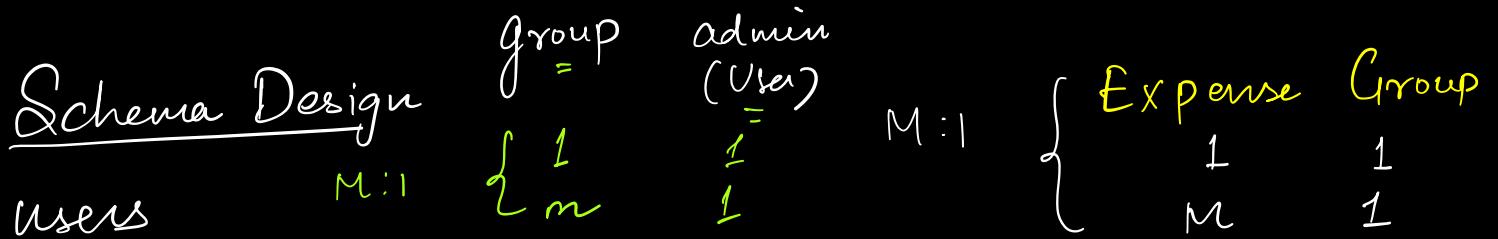
ExpenseType
DUMMY,
ACTUAL

Group
<u>id</u>
<u>name</u>
<u>admin</u>
<u>list < User > members</u>
<u>List < Expense > expenses</u>

Expense ✓
<u>id</u>
<u>description</u>
<u>amount</u>
<u>group</u>
<u>list < UserExpense ></u>
<u>ExpenseType ✓</u>
<u>AddedBy</u>

UserExpenseType
Lender,
Borrower

UserExpense
<u>user</u>
<u>amount</u>
<u>currency</u>
<u>type</u>



id	name	email	Password
----	------	-------	----------

expenses

id	descr	amount	group_id	expense-type_id
----	-------	--------	----------	-----------------

groups	user_id
id	name

user_expenses

id	user_id	amount	currency	expense_id	user-expense-type_id
----	---------	--------	----------	------------	----------------------

expense-type

id	value
----	-------

user-expense-type

id	value
----	-------

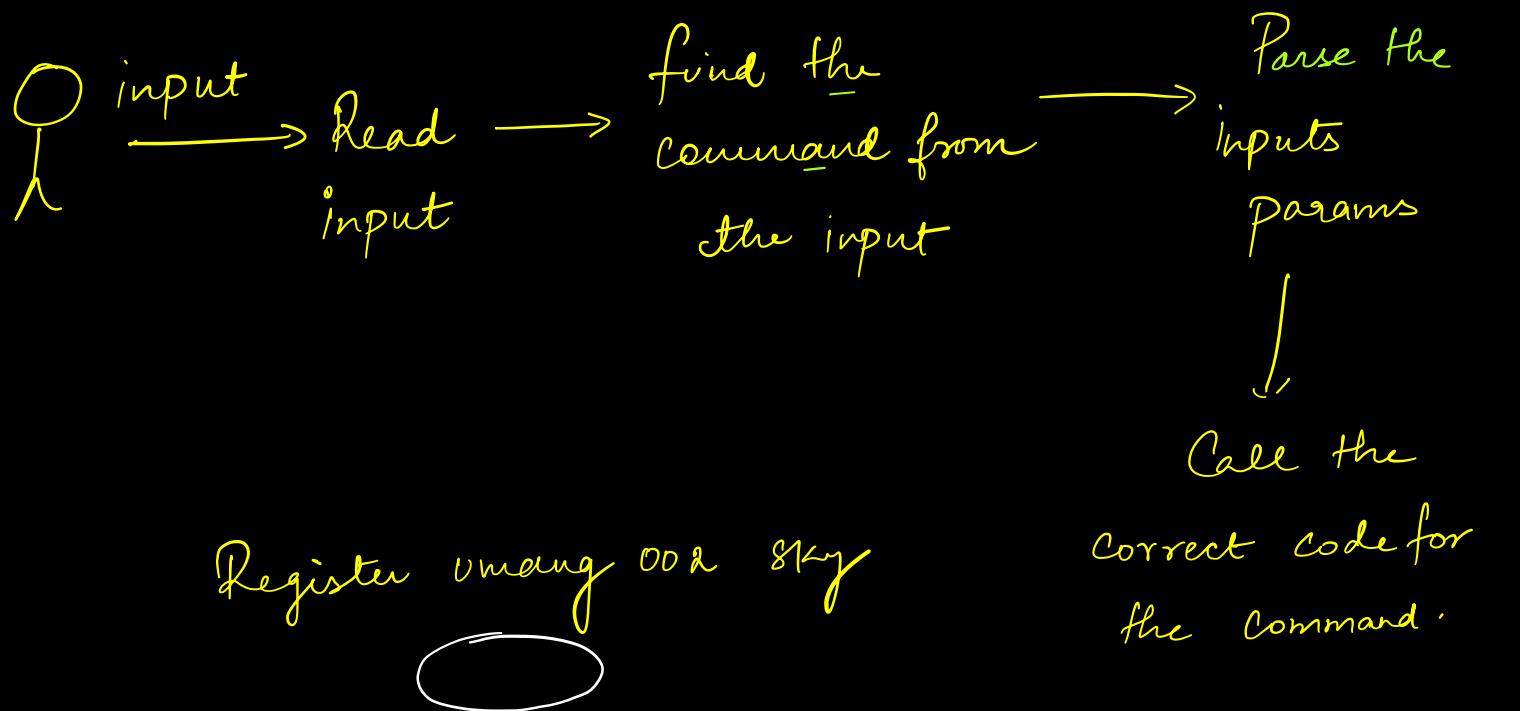
groups-users

group_id	user_id
----------	---------

Take inputs via Command line :-

↳ Behaviour

Register ---
— updateProfile —



```
String input = Scanner.next();
```

```
List<String> words = input.split(" ");
```

```
if (words.get(0).equals("Register")) {
```

```
    //
```

```
} else if (words.get(1).equals("UpdateProfile")) {
```

```
    //
```

```
}
```

```
else if (
```

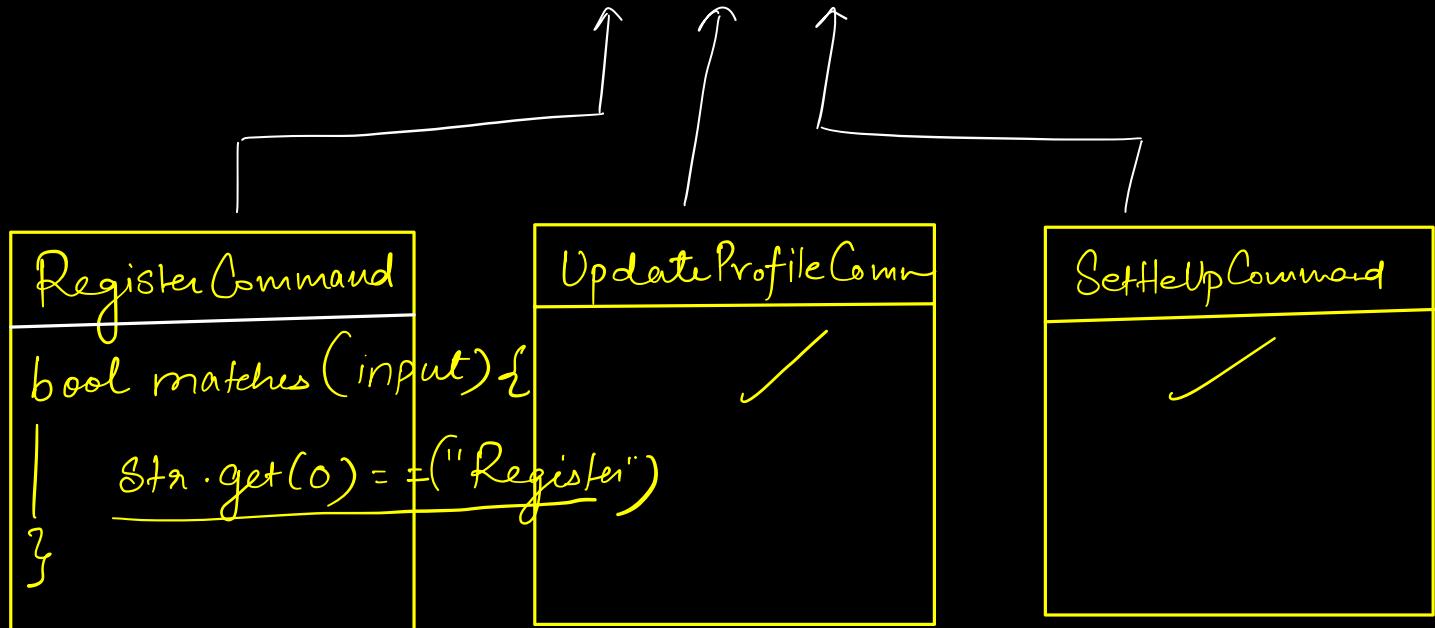
```
) .
```

Too many if else (SRP & OEP violation)

We need better
way to check if a
string is a command

We need to
execute the
command.

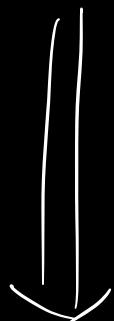
<< Command >>
void execute (input)
bool matches (input)



~~hist & Command > commands = { . . . }~~

input = Scanner.next();

for(Command cmd : commands) {
if (cmd.matches(input)) {
cmd.execute();
break;
}



input = Scanner.next();

CommandExecutor.execute(input)

CommandExecutor

hist & Command > commands = { . . . } //

void addCommand(Command cmd) {

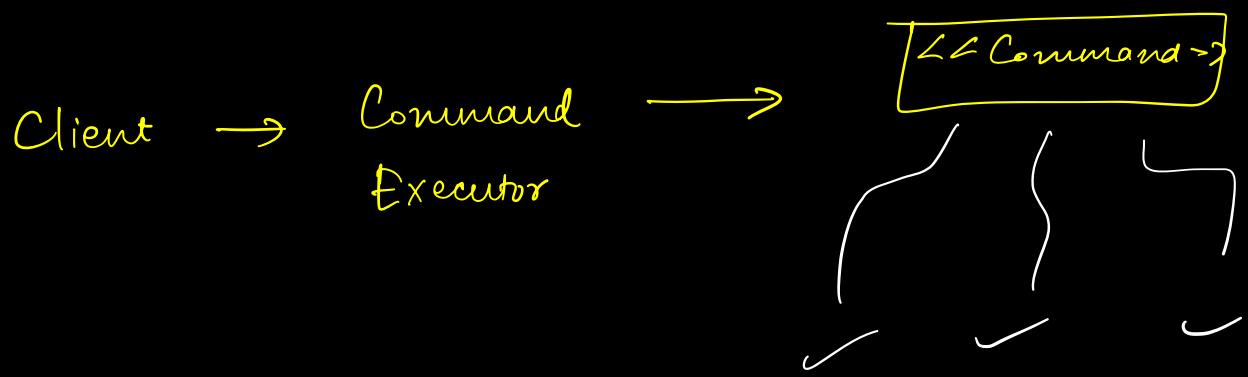
| commands.add(cmd)

| void execute(String input) {

| for(Command cmd : commands)
| if (cmd.matches(input)) {
| | cmd.execute();
| } break;

} }

static { }



Command Design Pattern

1

User Expense

Thank you