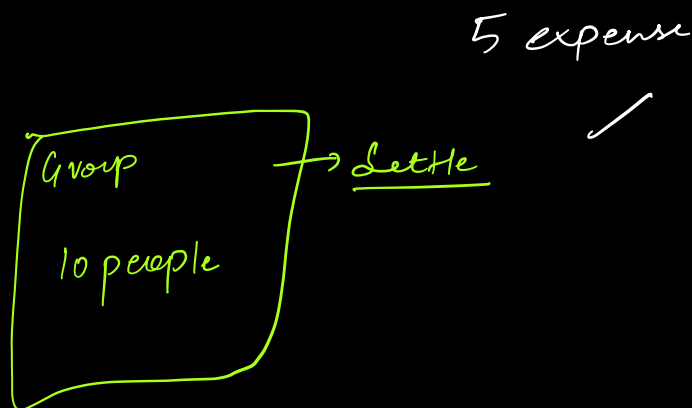


Today's Agenda :-

- 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.



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

#lended = # borrowed

N no of people

$N \log N$

Algo:- Divide the users into 2 buckets ✓

Lenders
who've
paid extra

Borrowers
who've
paid lesser.

{ A : +1000
B : -50
C : -1250
D : -450 }

{ E : +750 }

max from bucket 1

min from bucket 2

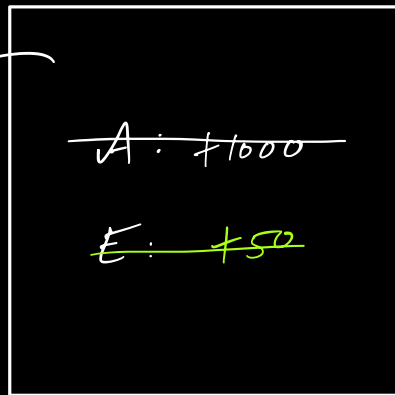
① C $\xrightarrow{1000}$ A

② D $\xrightarrow{450}$ E ✓

③ C $\xrightarrow{250}$ E

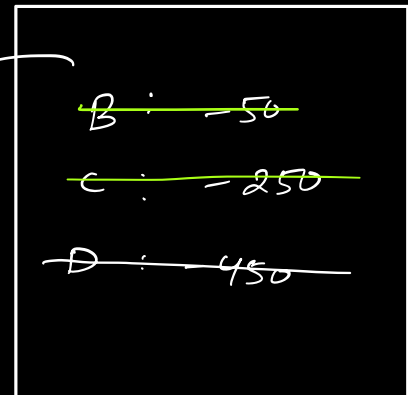
④ B $\xrightarrow{50}$ E

Max
heap



Group 1
Lenders

Min
heap



Group 2
Borrowers

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
id
name
email
password

who paid how much

who borrowed how much

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

£ 1000

A	200	A	250
B	800	B	250
		C	250
		D	250

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

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

ExpenseType
DUMMY,
ACTUAL

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

Transaction ✓

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

Transactions

C	1000	→	A
D	300	→	A
D	200	→	B

⇓

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 txn on the amount.

↳ Persist somehow the executed txn

⇓

Create a dummy expense.

$C \xrightarrow{1000} A$

amount : 1000

lender : C : 1000

borrower : A : 1000

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

$C \xrightarrow{1250} A$

$D \xrightarrow{450} A$

$B \xrightarrow{50} A$

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 txn.

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

lend
borrow

currency
£ 5000 ✓
A: 2000 B: 3000
A: 1000 B: 1000
C: 1000 D: 1000 E: 1000
100 \$
Lend: A: 80 \$ B: 20 \$
(Users), (amount, currency)

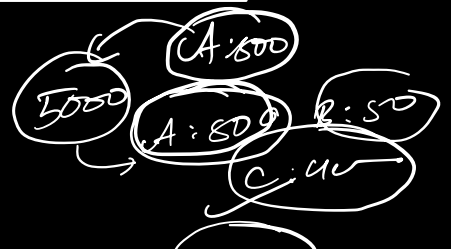
ExpenseUserLender
User amount currency mode

ExpenseUserBorrow
User amount currency :

→

UserExpense
user ✓ amount ✓ currency ✓ type ✓

UserExpenseType
Lender, Borrower



Expense
id
description
amount
group
hist < UserExpense >
ExpenseType

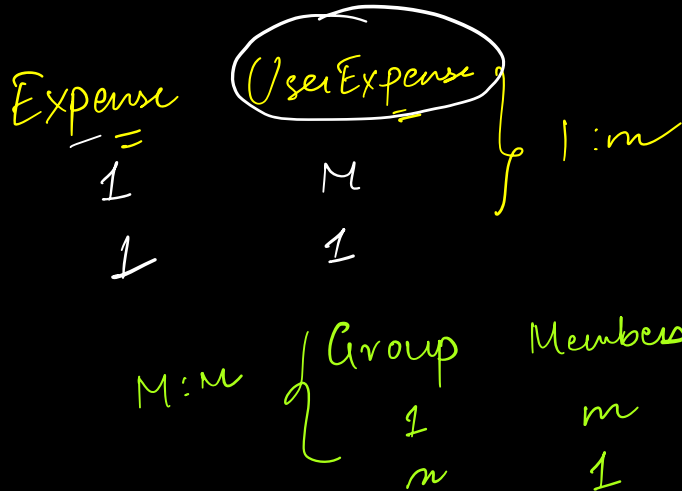
5000 ₹

Lend A: 1000 B: 4000

borrow A: 2500 B: 2000 C: 500

{ user: A, amount: 1000, currency: INR, type: Lender }

2 — B, 4000, INR, Lender }



Group	Expenses
1	m
1	1

User
id
name
email
password

ExpenseType
DUMMY,
ACTUAL

Group
id
name
admin
hist < User > members
hist < Expense > expenses

Expense
id
description
amount
group
hist < UserExpense >
ExpenseType
AddedBy

UserExpenseType
Lender,
Borrower

UserExpense
user
amount
currency
type

Schema Design

users M:1 group = admin (User) M:1 Expense Group

1 1

M 1

id	name	email	password
----	------	-------	----------

expenses

id	descr	amount	group_id	expense_type_id
----	-------	--------	----------	-----------------

groups

id	name	admin_id
----	------	----------

user_id

user_expenses

id	user_id	amount	currency	expense_id	user_expense_type_id
----	---------	--------	----------	------------	----------------------

expense_type

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

user_expense_type

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

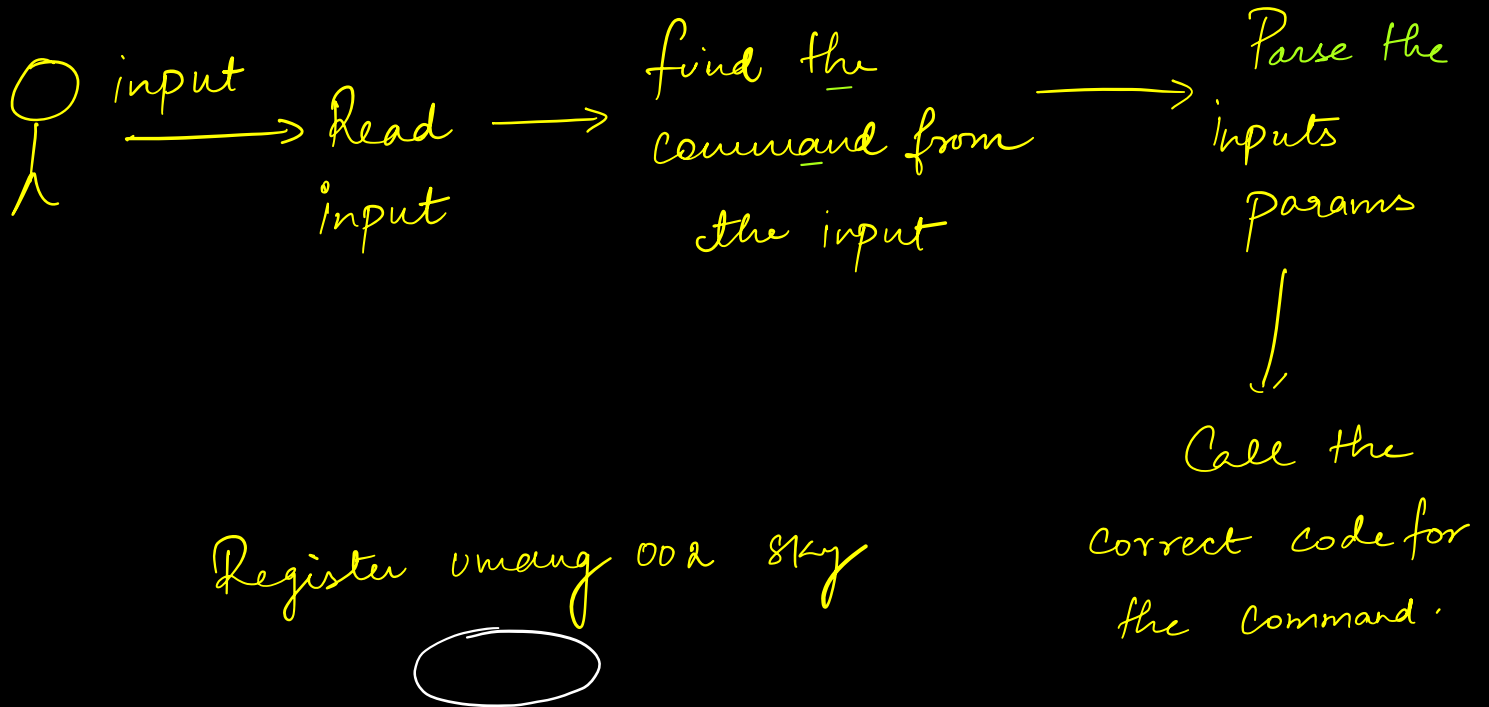
groups-users

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")) {
```

```
    //
```

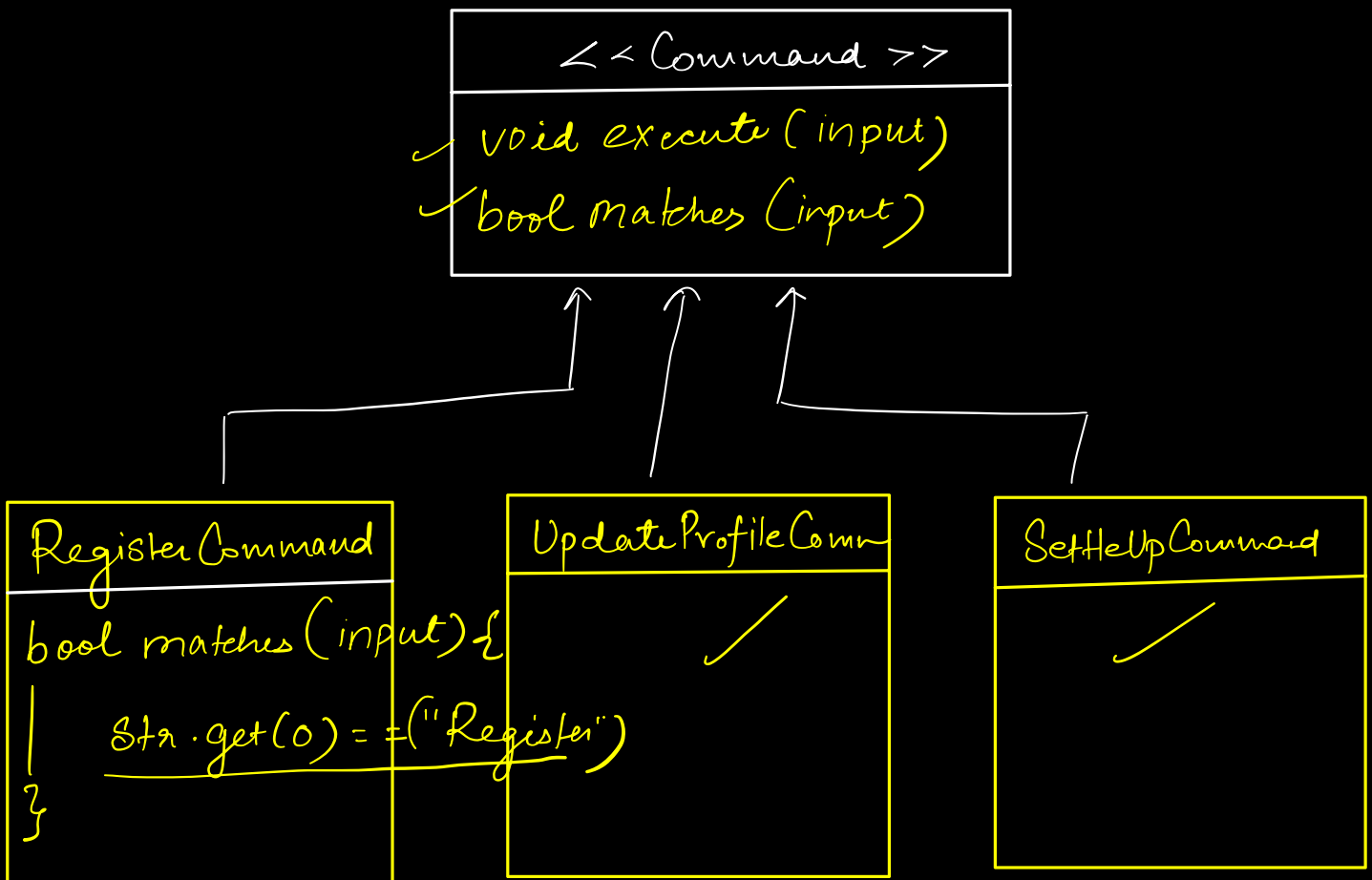
```
} else if (
```

```
)
```

Too many if else (SRP & ocp violation)

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

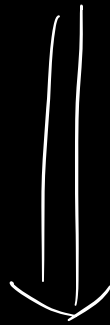
We need to execute the command.



✓ $\text{List} \langle \text{Command} \rangle \text{ commands} = \{ \dots \}$

input = Scanner.next();

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



input = Scanner.next();

CommandExecutor.execute(input)

Command Executor

✓ $\text{List} \langle \text{Command} \rangle \text{ commands} = \{ \dots \}$ ✓✓

void addCommand (Command cmd) {

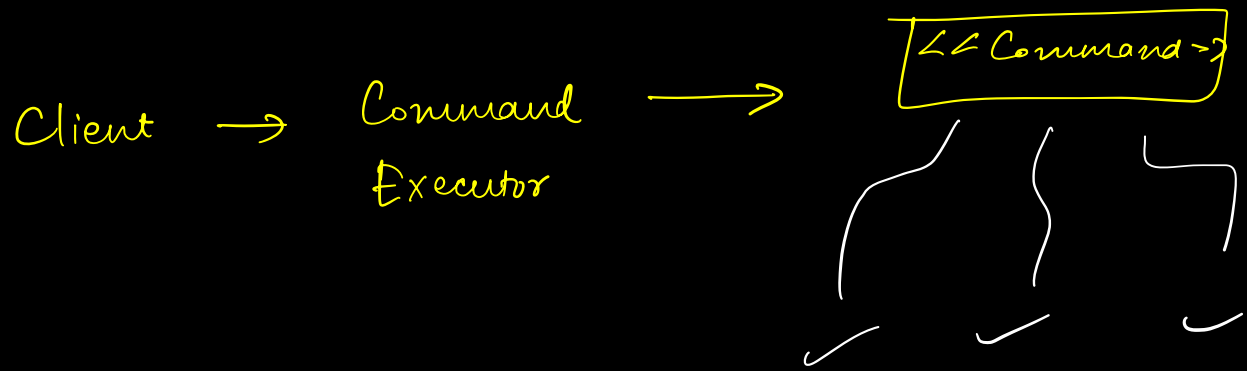
 commands.add (command)
}

void execute (String input) {

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

}

Static



Command Design Pattern =

1

User Expense

Thank You