Splitwise

4 of your friends went for Goa trip.

Dinner	→ 5000	Everybody has made some contribution. We need to settle them up!
cabbing	→ 3000	
Hotel	→ 8000	
Bikes rent	→ 4000	

Requirements are given to you! - You've to create complete working code.

Requirements

- Users can register and update their profiles.
- A user's profile should contain at least their name, phone number and password
- Users can participate in expenses with other users
- Users can participate in groups.
- To add an expense, a user must specify either the group, or the other users involved in the expense, along with who paid what and who owes what. They must also specify a description for the expense.
- A user can see their total owed amount
- A user can see a history of the expenses they're involved in
- A user can see a history of the expenses made in a group that they're participating in
- Users shouldn't be able to query about groups they are not a member of
- Only the user who has created a group can add/remove members to the group
- Users can request a settle-up. The application should show a list of transactions, **which when executed** will ensure that the user no longer owns or receives money from any other user. Note that this need not settle-up with any other users.
- Users can request a settle-up for any group they're participating in. The application should show a list of transactions, which if executed, will ensure that everyone participating in the group is settled up (owes a net of 0 Rs). Note that it will only deal with the expenses made inside that group. Expenses outside the group need not be settled.
- When settling a group, we should try to minimize the number of transactions that the group members should make to settle up.

Note: All tests will be performed in one go. The application doesn't need to persist data between runs.

What is an expense?...

(i) actual share of each person

(ii) who has paid the amount.

ex: whoPaid → Map<Person, Amount>

(Keerthi → 5000) (Abhi → 6000)

WhoHadToPay -> Map<Person, Amount>

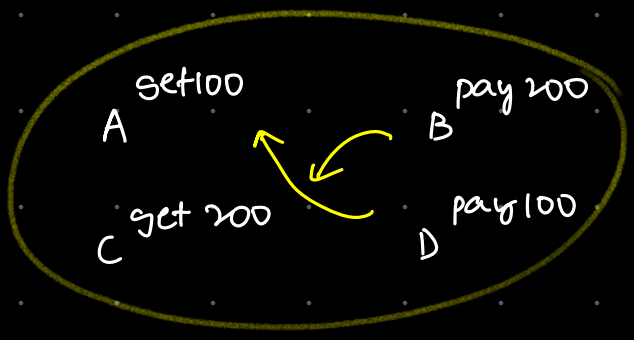
5 of us went for goa trip.

trekking ->

(Keerthi -> 500) (Abhiram -> 100)

minimum no. of transactions

A, B, C, D.



One option

D -> C [100]

B -> A [100]

B -> C [100]

x

Another option

B -> C [200]

D -> A [100]

✓

How to settle up?

1. N no. of expenses
2. someone clicks on settle up.

Display a set of transactions to settle

How to calculate the amount to pay /recieve.

(From who // To whom)

Minimum transactions.

1. How to calculate amount to pay / recieve

Consider the below expenses of 4 people.

Exp 1 dinner	whoPaid :	A: 1000	B: 1000		
	whoHadTo :	A: 500	B: 500	C: 500	D: 500
Exp 2 carting	whoPaid :	A: 3000			
	whoHadTo :	A: 1000,	B: 200,	C: 800,	D: 1000
Exp 3 hotel	whoPaid :	C: 500, D: 800			
	whoHadTo :	A: 500,	B: 100,	C: 200,	D: 500
Exp 4 Bike rent	whoPaid :	D: 1000			
	whoHadTo :	A: 250,	B: 250,	C: 250,	D: 250

Calculate the extra amount that A should recieve or pay

```
extra-amount = 0
for (expense in expenses)
    extra-amount = extra-amount + whoPaid.get(A)
    extra-amount = extra-amount - whoHadToPay.get(A)
```

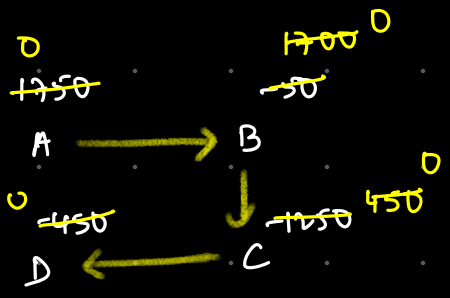
	exp1	exp2	exp 3	exp 4	
A	-500+1000	-1000+3000	-500+0	0-250	= 1750
B					= -50
C					= -1250
D					= -450

2. From whom to get/give.

(minimum no. of transactions ⇒ NP hard problem).

NP -> Non-deterministic polynomial. [There's no guaranteed optimal soln].

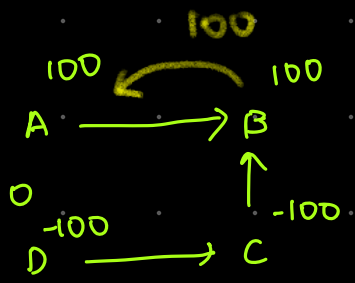
Soln:



B → A (1750)

C → B (1700)

D → C (450)



It's a working soln but it's not used in splitwise.

Because, it can create anxiety among people

N-1 transactions.

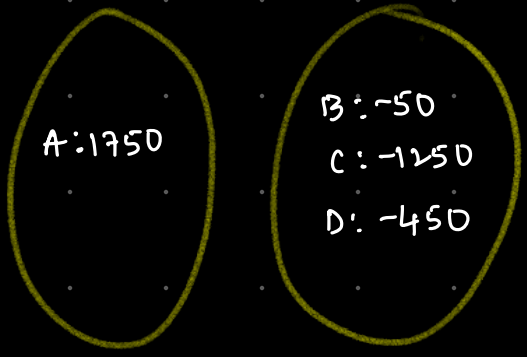
Soln 2: Divide people into 2 groups.

A = 1750

B = -50

C = -1250

D = -450



Recievers givers

Who should give first, who should get first.

The one who's to pay max should give first

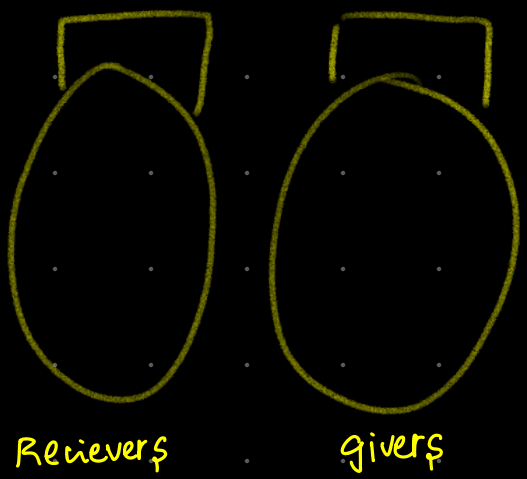
The one who's to recieve max should get first

C → A

D → A

B → A.

* Total no. of transactions ⇒ N-1 [everytime 1 person is settled]
* The containers are max heaps.



C: -1250

A: ~~1750~~ 500

A is not settled yet.

Push A back to heap.

D: -450

A: 500 -50

A is not settled yet.

Push A back to heap.

B: -50

A: -50

Heaps are empty, stop here.

B : -1200
A : ~~1000~~ 0.

TC: NlogN.

Total transactions are (N-1).

HW.

1. identifying nouns
2. Create class dia / schema dia
3. Create models
4. Write the code for settleup strategy.

(Heap settleup strategy).