Coding Challenges > Backend > Expenses Management

Expenses Management

Problem statement Sample I/O Badges and score Credits

Build instructions Help Center

You work at a fintech startup that helps housemates calculate and manage their expenses. Your task is to build this expense calculator.

The housemates stay in a 3 bedroom house where a maximum of 3 people can stay. The expenses calculator can be used only when there are at least 2 people in the house. Once there are at least 2 people in the house, then the expenses are tracked in the software you build. Once their expenses are tracked, a member of the house can only move out of the house if they have cleared the dues.

Input Commands

MOVE_IN <name-of-the-member>

This command allows a member to be moved into the house. It outputs SUCCESS if the addition is successful. If the house is already full with the maximum number of members, then it should print an error message HOUSEFUL.

Example-

```
MOVE_IN ANDY
MOVE_IN WOODY
MOVE_IN BO
MOVE_IN REX
```

Output

```
SUCCESS
SUCCESS
SUCCESS
HOUSEFUL
```

Next step:
You have attempted this problem in java and scored 70. Read our help docs and resubmit code to achieve a higher score.

Starter Kit View Portfolio

Example-

SPEND 3000 ANDY WOODY BO

SPEND 300 WOODY BO

SPEND 300 WOODY REX

Output:

SUCCESS

SUCCESS

MEMBER_NOT_FOUND

In the above example after the first command, WOODY owes 1000 to ANDY and BO owes 1000 to ANDY. After the next command, BO owes 150 to WOODY. This 150 gets adjusted in the dues so that WOODY now owes 850 to ANDY and BO owes 1150 to ANDY and 0 to WOODY.

DUES <member-who-owes>

This command prints all the dues of a member against all the other members in the format - <member-who-lent> <amount> Dues against all the members should be printed and each member should be printed in one line in the descending order of the pending amount. If the amount due is same, then those should be sorted on the ascending order of the member's name. If the member is not yet added then an error message MEMBER_NOT_FOUND should be printed.

Example-

DUES BO

DUES WOODY

Output:

ANDY 1150

WOODY 0

ANDY 850

BO 0

CLEAR_DUE <member-who-owes> <member-who-lent> <amount>

Any member can clear the due of the other member who lent using this command. The amount can be partial or full. It cannot be more than the amount owed as per the DUES command. If they pay more, then an error message INCORRECT_PAYMENT should be printed and none of the payments should go through. This command should print the remaining dues of the member against whom they paid.

Example-

CLEAR_DUE BO ANDY 500

CLEAR_DUE BO ANDY 2500

Output :

650

INCORRECT_PAYMENT

MOVE_OUT <name-of-existing-member>

Next step:

You have attempted this problem in java and scored 70. Read our help docs and resubmit code to achieve a higher score.

Starter Kit View Portfolio

```
CLEAR_DUE BO ANDY 650
MOVE_OUT BO
MOVE_OUT REX
```

Output:

```
FAILURE
FAILURE
FAILURE
0
SUCCESS
MEMBER_NOT_FOUND
```

Important Notes

1. Maximum number of members in a house is 3.
2. A member can move out of the house only if he/she has paid their dues to all other members.
3. Expenses can be added only if there are at least 2 members in the house.
4. 3rd member can be added any time and they share the expenses incurred after they move in
5. A member can move out only if they have paid all the dues and no other members are owing any money to them.
6. All calculations should be rounded off to the nearest integer. No decimal values are used.
7. A member can only pay less than or equal to the amount they owe the other members. If they pay more, then an error should be shown and none of the payments go through.
8. For the DUES command, dues against all the members should be printed and each member should be printed in one line in the descending order of the pending amount. If the amount due is same, then those should be sorted on the ascending order of the member's name.
9. This problem doesn't require use of databases. All the calculations are to be handled in memory.

Next step:
You have attempted this problem in java and scored 70. Read our help docs and resubmit code to achieve a higher score.

Starter Kit

View Portfolio