**Problem Description**

In the context of a simplified blockchain-like system, our task is to design an interface class for a Transaction Manager. This Transaction Manager is responsible for handling transactions between wallets, enabling the transfer of funds from one wallet to another. The system's primary data structure is an array of wallets, each identified by a unique wallet ID represented as a string.

**The foundation of this blockchain** is that you will start with one wallet (wallet0). Wallet0 will have unlimited money. the wallet0 can give money to any wallet. if the receiving wallet are not existing it will create new wallet. The new wallet can send money to other wallet or create new wallet (all other wallets can’t pop money from air unlike wallet0)

**What to do:**

Create interface class TransactionManager with 3 method

* boolean transferFunds() with 3 parameters string, string and double
* double getBalance()with 1 parameter string
* bollean isValidWallet()with 1 parameter string

**Create class SimpleTransactionManager that implement from TransactionManager.**

For starting the SimpleTransactionManager should first consist of 4 base variables.

  private String[] wallets  
  private String[] transactions  
  private int walletCount;  
  private int transactionCount;

**Your “real” job is to implement TransactionManager**

**By the following rules**

create array of string wallets (1000 member) that will contain all walletid

create array of string Transactions (10000 member) that will contain all transaction

The constructor of SimpleTransactionManager will first accept array of string then it will register every member of accepting array to wallets array(for this problem only, the accepting wallets array will only have one member “wallet0” that will be at the position[0] of the String[] wallets)

Then it’s time to implement function!! 😊

However, Master Sukree has tricked us! the order is reverse! ☹.

First isValidWallet(String walletid)

This method should check if a wallet with the given Wallet ID exists in the String[] wallets.

Second the getBalance()

Just copy Master sukree lol.

 public double getBalance(String walletId) {  
      if (!isValidWallet(walletId)) {  
          throw new IllegalArgumentException("Invalid wallet ID.");  
      }  
  
      double balance = 0.0;  
      for (int i = 0; i < transactionCount; i++) {  
          String[] parts = transactions[i].split("\\|");  
          if (parts[0].equals(walletId)) {  
              balance -= Double.parseDouble(parts[2]);  
          }  
          if (parts[1].equals(walletId)) {  
              balance += Double.parseDouble(parts[2]);  
          }  
      }  
  
      return balance;  
  }

The concept is tie with String[] transactions

all member of String[] transactions will follow this pattern.

"source wallet id|receive wallet id|amount"

It will read all the transaction that happen and return the remaining balance in the wallets like get 400 give 200 the remaining will be 200

the last method public boolean transferFunds(String senderWalletId, String receiverWalletId, double amount)

Transfer money of one wallet to other. If the wallet does not exist create one. Then save the transaction to String[] transactions

check for 2 conditions(use the 2 methods that we already create to help):

if senderWalletId already exist in String[]wallets? if not throw exeption InvalidTransactionException("Sender wallet does not exist.");

if senderWalletID have enough money(compare to amount) to send to the other?

If not throw new IllegalArgumentException("Not enough balance in the source wallet.")

Hint: its associate with adjusting position of array member.

  private int walletCount;  
  private int transactionCount;

will help keep track of the position in array.

Be aware!!!

array of 1000 member all member will first be “null” not a string until you change it manually to the string. So, the String.equals() will be error.