

BackEnd

- oldMasterAccountsFile: String
- mergedTransactionSummaryFile: String
- accounts: HashMap<String, Account>

- + main(String[]): void
- overDailyLimit(int, String, String, ArrayList<String>): boolean
- readOldMasterAccountsFile(String): HashMap<String, Account>
- processMergedTransactions(String): void
- deleteAcct(String, String): void
- transfer(String, int, String): void
- withdraw(String, int, String): void
- deposit(String, int, String): void
- createAcct(String, String): void
- inputFileValidity(String, String): void
- isValValid(String): boolean
- isTsfValid(String): boolean
- newMasterAcctFile(): void
- newValidAccList(): void

Account

- accountID: String
- balance: int
- accountName: String

- + Account(String, int, String)
- + getAccountID(): String
- + getBalance(): int
- + getAccountName(): String
- + setBalance(int): void

Validation

- + accountNumberValid(String): boolean
- + isAllDigits(String): boolean
- + accountNameValid(String): boolean
- + accountNumberExists(HashSet<String>, String): boolean
- + validMonetaryAmount(String): boolean

BackEnd.java

Main

This method takes in both the oldMasterAccountListFile location and mergedTransactionSummaryFile location via parameters, and calls other functions to begin the calculations and processing.

overDailyLimit

This method performs the necessary calculations regarding daily transaction limits, and returns whether a transaction is valid or not (based on whether it has surpassed the limit or not).

readOldMasterAccountsFile

This method reads the old master accounts file using parameters given from the main class, and every line into the accounts hash map.

processMergedTransactions

This method reads every line in the merged transaction file and calls on corresponding methods to apply the requested action (e.g. calling deleteAcct if a request was made to delete an account).

deleteAcct

This method takes in two parameters (the account number and name), and then deletes that account from the accounts hashmap.

transfer

This method takes in three parameters (two account numbers and an amount), and then transfers the monetary amount from the first account to the second.

withdraw

This method takes in three parameters (an account number, name, and amount), and then deducts the amount from the total account balance.

deposit

This method takes in three parameters (an account number, name, and amount), and then adds the amount to the total account balance.

createAcct

This method takes in corresponding values needed to create an account, and then adds it to a new Account object, which is then added to the accounts hash map.

inputFileValidity

This method calls on corresponding methods that check the validity of the input files, ending the system if either of the called methods return false.

isValValid

This method runs through every line within the valid accounts list file, and checks the data to see if it's formatted correctly or not. If anything is considered invalid, the method will return false.

isTsfValid

This method runs through every line within the transaction summary file, and checks the data to see if it's formatted correctly or not. If anything is considered invalid, the method will return a false.

newMasterAcctFile

This method reads all Account objects within the accounts hash map, and adds their values to a new file named NewMasterAccountsFile.txt

newValidAcclList

This method reads all Account objects within the accounts hash map, and adds all account numbers to a new file named newValidAcclList.txt

Account.Java

Account

This method assigns corresponding variables to their respective values, as outlined by the requirements (A constructor). There are two version of this constructor, with the first version assigning the variables their default values, and the second one assigning using user-defined values to their corresponding variables.

getAccountID

this, when called, returns the ID of the specified Account object.

getBalance

this, when called, returns the balance of the specified Account object.

getAccountName

this, when called, returns the name of the specified Account object.

setBalance

this, then called, assigns the "balance" variable a new value depending on the parameter.

Validation.java

isAllDigits

This method is a common helper function that is called by multiple larger methods that verifies that the account number entered consists only of decimal digits.

accountNumberExists

This method is a common helper function that is called by multiple larger methods. This takes in a string as a parameter and verifies that the string is currently in the valid accounts list file. This is done by checking through the hashset that was created in the login method, ensuring O(1) efficiency.

accountNumberValid

This method is a common helper function that is called by multiple larger methods that verifies the account number is exactly seven decimal digits not beginning with 0.

accountNameValid

This method is a common helper function that is called by multiple larger methods that verifies the string follows all rules and constraints for account names such as no leading or trailing spaces.

validMonetaryAmount

This method is a small helper function that ensures no account has a balance of less than 0, or greater than 99999999. If an account has a balance out of those bounds, the method will return false.

The Validation class was created to support both frontend and backend classes. This class contains all validation methods, with which both frontend and backend can call in order to validate any data. Many of these methods were formerly in the frontEnd.java class.