CISC 327 Assignment 2: Design Document

Loaft Inc.

**note: the front end architecture diagram is also uploaded into the GitHub repository as a .png file for better quality.*

Below we describe the intention and functionality of each class:

`Frontend` is the main class that continues to accept transaction inputs from the user until logout has been called. It will ensure that the user cannot run any transactions until they have logged in. All transactions written to the CLI will be parsed and validated before executing its specified function such as the ones listed below.

`Terminal` houses the multiple variables of the users instance. One of them being the state of the instance which will be "`in`" or "`out`" which signifies whether the user is logged in or logged out. Mode is another variable that is set through login which will either be "Agent", "Machine" or "none". It also holds an ongoing list for all TSF entries as the user navigates through the system. Recurring functions such as validating the users account numbers, and their withdrawal/deposit limit can also be found here.

The `login` function is the first task that is executed. It will determine the users preference for agent mode or ATM mode. After the selection, the function will then validate each account number listed in `valid_accts.txt.` The terminal started for the instance will hold its mode as well as the list for all valid accounts in an array list.

`Createacct` will allow the user to create a new account (as long as they are in agent mode). After they input an account number and name, the program verifies that these values follow naming restrictions and that the account number is not already in use. Once the information is validated, the record is added to the Transaction Summary File.

`Deleteacct` is similar to createacct as the user inputs an account number and name. The account number will be checked to see that it already exists. Once the information is verified, the account deletion record will be added to the Transaction Summary File. This function also verifies that the user is in agent mode before being able to input anything.

The `deposit, withdraw`, and `transfer` classes use the valid accounts file as input as well as CLI input from the user to execute dollar transactions for amounts with accounts specified by them. Each of these classes each contain one function (`depositToAccount`, `withdrawFromAccount`, `transferBetweenAccounts`, respectively) that are quite self explanatory. Once the transaction is specified, it will be added to the Transaction Summary File.

`Logout` is the final class used to wrap up the users interaction with the system. It will export and validate the TSF file.

The architecture diagram below specifies the inputs and outputs of the front end, terminal, and each of the functions in our program explicitly.

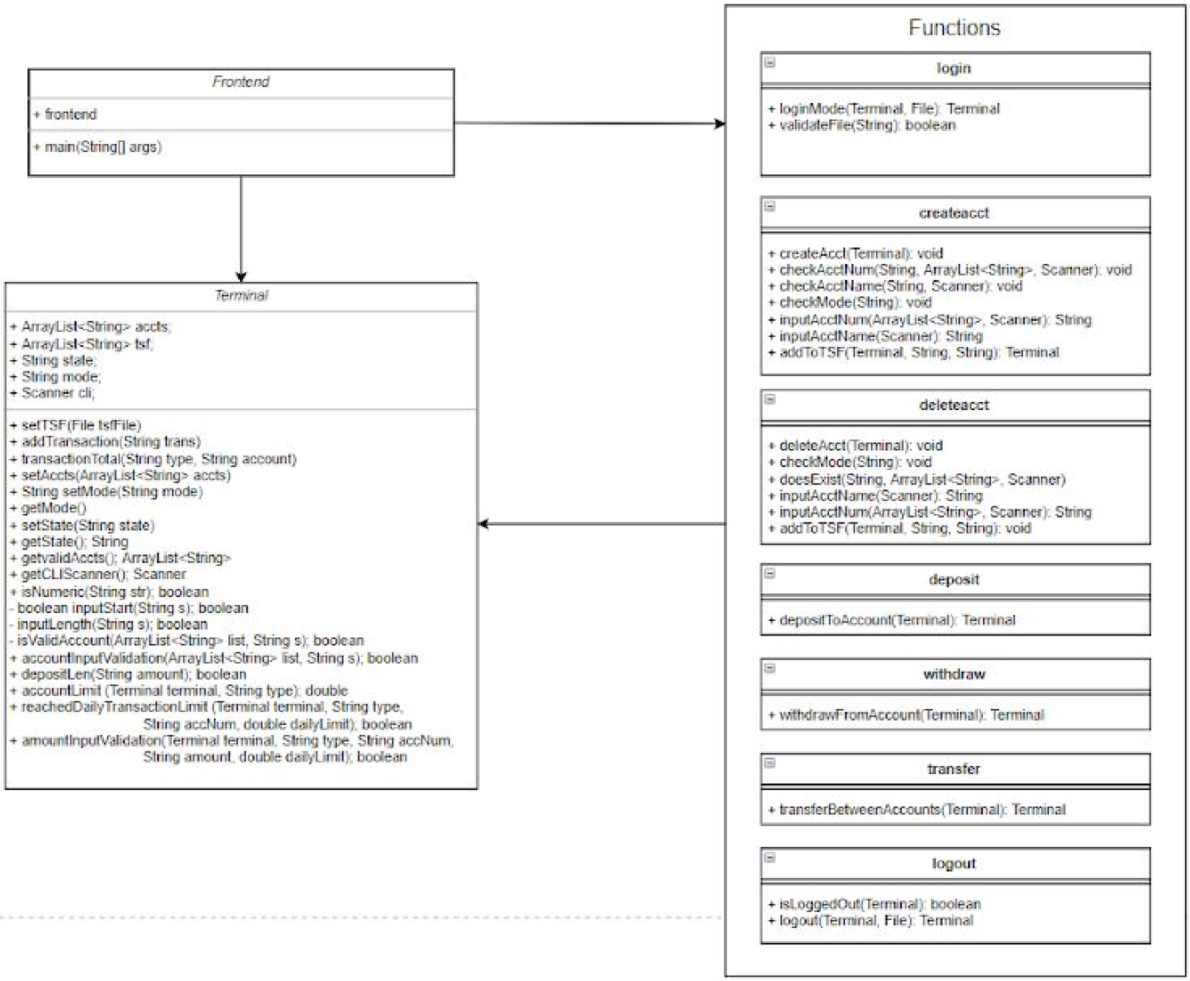
Frontend
+ frontend
+ main(String[] args)
Terminal
+ ArrayList<String> accts;
+ ArrayList<String> tsf;
+ String state;
+ String mode;
+ Scanner cli;
+ setTSF(File tsfFile)
+ addTransaction(String trans)
+ transactionTotal(String type, String account)
+ setAccts(ArrayList<String> accts)
+ String setMode(String mode)
+ getMode()
+ setState(String state)
+ getState(); String
+ getvalidAccts(); ArrayList<String>
+ getCLIScanner(); Scanner
+ isNumeric(String str); boolean
- boolean inputStart(String s); boolean
- inputLength(String s); boolean
- isValidAccount(ArrayList<String> list, String s); boolean
+ accountInputValidation(ArrayList<String> list, String s); boolean
+ depositLen(String amount); boolean
+ accountLimit (Terminal terminal, String type); double
+ reachedDailyTransactionLimit (Terminal terminal, String type, String accNum, double dailyLimit); boolean
+ amountInputValidation(Terminal terminal, String type, String accNum, String amount, double dailyLimit); boolean
Functions
login
+ loginMode(Terminal, File): Terminal
+ validateFile(String): boolean
createacct
+ createAcct(Terminal): void
+ checkAcctNum(String, ArrayList<String>, Scanner): void
+ checkAcctName(String, Scanner): void
+ checkMode(String): void
+ inputAcctNum(ArrayList<String>, Scanner): String
+ inputAcctName(Scanner): String
+ addToTSF(Terminal, String, String): Terminal
deleteacct
+ deleteAcct(Terminal): void
+ checkMode(String): void
+ doesExist(String, ArrayList<String>, Scanner)
+ inputAcctName(Scanner): String
+ inputAcctNum(ArrayList<String>, Scanner): String
+ addToTSF(Terminal, String, String): void
deposit
+ depositToAccount(Terminal): Terminal
withdraw
+ withdrawFromAccount(Terminal): Terminal
transfer
+ transferBetweenAccounts(Terminal): Terminal
logout
+ isLoggedOut(Terminal): boolean
+ logout(Terminal, File): Terminal

Figure 1: Architecture diagram for the front end program.