Front End Design:

Company: ***Scarlet FlashView***

Members:

Tongxu Ge (20054696)

Sunny Yang (20075550)

Jinting Zhang (20059360)

Jinghong Xiao (20035787)

Oct 18th, 2019

Structure of Frontend System

The overall front-end system is separated into four *python* classes. *Frontend* class is designed for opening user interface, allowing users to do each transaction including 10 methods related to transactions. The other three classes are utility classes to support *Frontend* class. *AccountCheck* class is used to check the inputs from users in terms of money amount, amount number and account name. *TransactionFile* class is used to write and modify transaction summary file and *ValidAccountsFile* is used to update valid accounts file when doing create or delete account. *Frontend* as a main class imports the other 3 classes.

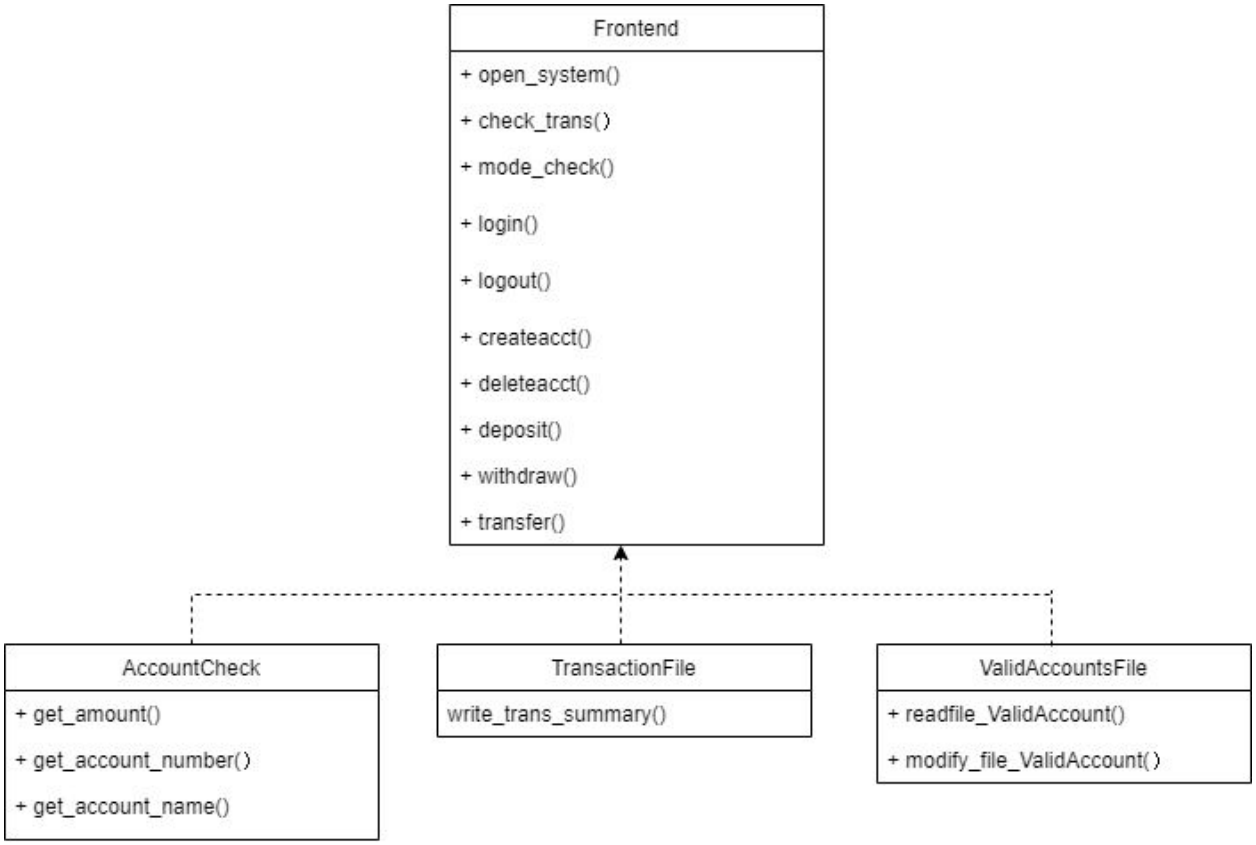

Diagram 1: Overall Structure of Frontend Architecture

The tables below show the intention of the four classes: Frontend, AccountCheck, TransactionFile, ValidAccountsFile and its methods inside each class.

Class: Frontend
overall bank interface to allow user to do transactions

Methods:	Intention:
open_system(self)	**ask user to choose the operation**
check_trans(self, trans, mode_name, transaction_list, valid_account_list, create_acct_list, use_daily_limit)	**call relative function based on the user selection of the 7 operations**
mode_check(self, transaction_list,valid_account_list)	**ask user to choose the mode from ATM or Agent**
login(self, transaction_list, valid_account_list)	**allow user to login and issue error prompt**
logout(self, transaction_list, valid_account_list)	**allow user to logout and issue error prompt**
createacct(self, transaction_list, mode, valid_account_list, create_acct_list)	**allow user to create account in agent mode and issue error prompt**
deleteacct(self, transaction_list, mode, valid_account_list, create_acct_list)	**allow user to delete account in agent mode and issue error prompt**
deposit(self, transaction_list, mode, valid_account_list, create_acct_list, use_daily_limit)	**allow user to deposit within limit in agent or ATM mode and issue error prompt**
withdraw(self, transaction_list, mode, valid_account_list, create_acct_list, use_daily_limit)	**allow user to withdraw within limit in agent or ATM mode and issue error prompt**
transfer(self, transaction_list, mode, valid_account_list, create_acct_list, use_daily_limit)	**allow user to transfer within limit in agent or ATM mode and issue error prompt**

Table 1: Intention of Methods in Frontend Class

Class: AccountCheck
get and check the valid account information including money amount, account number and name

Methods:	Intention:
get_amount(self)	**get the input of money amount from user and issue invalid amount**
get_account_number(self)	**get the input of account number from user and issue invalid number**
get_account_name(self)	**get the input of account name from user and issue invalid name**

Table 2: Intention of Methods in AccountCheck Class

Class: TransactionFile
manage and write in transaction summary file for ending bank session

Method:	Intention:
write_trans_summry(self, transaction_list)	**open and write in transaction summary file for the finished operations**

Table 3: Intention of Methods in TransactionFile Class

Class: ValidAccountsFile read valid accounts list file and write in new accounts	
Methods:	**Intention:**
readfile_ValidAccount(self)	**open valid accounts list file**
modify_file_ValidAccount(self, valid_account_list)	**add new accounts into the valid accounts list file**

Table 4: Intention of Methods in ValidAccountsFile Class

The tables below show the exact input and output of each method in the four classes.

Class: Frontend

overall bank interface to allow user to do transactions

Method:	Input	Output
mode_check(self, transaction_list, valid_account_list, create_acct_list, use_daily_limit)	self, transaction_list, valid_account_list, create_acct_list, use_daily_limit	"\nThere have two types of mode you can login:\n") 'Enter [m]: ----> for ATM mode' 'Enter [a]: ----> for Agent or privileged (teller) mode\n' not 'm' or 'a': "Please enter a valid mode!"
check_trans(self, trans, mode_name, transaction_list, valid_account_list, create_acct_list, use_daily_limit))	self, trans, mode_name, transaction_list, valid_account_list, create_acct_list, use_daily_limit)	None
login(self, transaction_list, valid_account_list, create_acct_list, use_daily_limit)	self, transaction_list, valid_account_list, create_acct_list, use_daily_limit	more than one account: " Error prompt for multiple login"
logout(self, transaction_list, valid_account_list)	self, transaction_list, valid_account_list	no logins: "Error prompt for login failed" one login: "\nLog out successfully!"
createacct(self, transaction_list, mode, valid_account_list, create_acct_list, use_daily_limit)	self, transaction_list, mode, valid_account_list, create_acct_list, use_daily_limit	no logins: "Error prompt for login failed" agent mode but same number: "Should not be same account number (account already exist)" agent mode with new number: "\nCreate an account successfully! Go back to main menu!\n" ATM mode: "Error prompt for ATM create account or delete account! Please enter other operations!\n"
deleteacct(self, transaction_list, mode, valid_account_list, create_acct_list, use_daily_limit)	self, transaction_list, mode, valid_account_list, create_acct_list, use_daily_limit	no logins: "Error prompt for login failed" agent mode but not exist number: "Account number not exist" agent mode with exist number: "\nDelete an account successfully! Go back to main menu!\n" ATM account: ("Error prompt for ATM create account or delete account! Please enter other operations!\n"
deposit(self, transaction_list, mode, valid_account_list, create_acct_list, use_daily_limit)	self, transaction_list, mode, valid_account_list, create_acct_list, use_daily_limit	no logins: "Error prompt for login failed"

		not exist: "Account not exist! Enter a exist account to deposit!" just created: "Error! Account just create, cannot do anything!" ATM more than 2000: "Over deposit limit, enter a valid amount!" agent more than 999999.99: "Over deposit limit, enter a valid amount!" within range: ("\nDeposit successfully! Go back to main menu!\n"
withdraw(self, transaction_list, mode, valid_account_list, create_acct_list, use_daily_limit)	self, transaction_list, mode, valid_account_list, create_acct_list, use_daily_limit	no logins: "Error prompt for login failed" not exist: "Account not exist! Enter a exist account to deposit!" just created: "Error! Account just create, cannot do anything!" ATM more than 1000: "Over withdraw limit, enter a valid amount!" agent more than 999999.99: "Over withdraw limit, enter a valid amount!" within limit: "\nWithdraw successfully! Go back to main menu!\n
transfer(self, transaction_list, mode, valid_account_list, create_acct_list, use_daily_limit)	self, transaction_list, mode, valid_account_list, create_acct_list, use_daily_limit	no logins: "Error prompt for login failed" "Transfer from ------>" "Transfer to ------>" just created: "Error! Account just create, cannot do anything!" ATM mode more than 10000: "Over transfer limit, enter a valid amount!" agent more than 999999.99: "Over transfer limit, enter a valid amount!" more than 3 times: "Error! Over daily transfer limit!" otherwise: "\nTransfer successfully! Go back to main menu!\n"

Table 5: Input and output in Frontend class

Class: AccountCheck

get and check the valid account information including money amount, account number and name

Method:	Input	Output
get_amount(self)	self	not valid number: "Enter a valid amount!"
get_account_number(self)	self	not digit: "Please enter a valid digit number!" not 7: "Please enter a valid account number! (Only 7 digits)" start with 0: "Account number first digit cannot be zero!"
get_account_name(self)	self	length out of range: "Enter a valid account name!" begin or end with space: "Enter a valid account name(no space for beginning or ending)!"

Table 6: Input and output in AccountCheck class

Class: TransactionFile

manage and write in transaction summary file for ending bank session

Method:	Input	Output
write_trans_summry(self, transaction_list)	self, transaction_list	None

Table 7: Input and output in TransactionFile class

Class: ValidAccountsFile

read valid accounts list file and write in new accounts

Method:	Input	Output
readfile_ValidAccount(self)	self	None
modify_file_ValidAccount(self, valid_account_list)	self, valid_account_list	None

Table 8: Input and output in ValidAccountsFile class