Front End Design

To run our first implementation of the front end on your bash terminal navigate to the src folder and type the command:

./frontend.sh valid_accounts_file.txt transaction_summary.txt

Our solution uses an object oriented approach with specialized objects that contain groupings of useful methods. The are four objects used by the frontend Python script. The SessionHandler object tracks the state of the session and handles the users input and the flow of the frontend. The InputValidation object holds methods that check if given account numbers, names and monetary amounts are in a valid format. The TransactionSummary object contains a reference to the transaction summary text file and methods that write transactions in the correct format to the file. The ValidAccounts object takes in a valid account file and stores the accounts in the file in a dictionary that it adds and deletes accounts from.

frontend.sh

Method	Desc	Parameters	Output
NA	Makes frontend launchable from bash terminal	transaction_summary.txt valid_accounts_file.txt	NA

front_end.py

Method	Desc	Parameters	Output
frontend()	Acts as beginning of program and instantiates the session handler Initializes while loop that asks for commands	trans_summary_file = an empty .txt file acount_list _file = a .txt file containing valid accounts	NA

session_handler.py

Class: SessionTypes(Enum): Contains ienumerable constants for checking and setting session type.

Class: SessionHandler

Method	Desc	Parameters	Output
__init__()	Creates attributes of the class that will store and track the state of the session	Valid_account_file = reference to a .txt file that is formatted according to specifications and holds valid accounts Trans_summary_file = reference to a .txt file to hold the transaction summary	NA
get_command()	Asks for command from user and passes the command to the handle_command function	None	None
handle_command()	Takes the users command and starts the corresponding method making sure the user is in a valid session type	Command = string indicating which command the user is asking for	True if the program should continue after this command, False is the program should end
login()	Handles the login of the session and asks the user for which type of session is to be started	None	Sets the session type of the session handler Sets loggedIn to true
logout()	Handles the logging out of the session and resets all the attributes of the session handler	None	Closes the transaction summary file, resets the daily limit dictionaries, resets the session type and sets loggedIn to false
createacct()	Brings the user through the process of creating an account and validates their input	None	If successful sends the transaction to the transaction summary file to be written
deleteacct()	Brings the user through the process of deleting an account and validates their input	None	If successful sends the transaction to the transaction summary file to be written
withdraw()	Brings the user through the process of withdrawing funds and validates their input	None	If successful sends the transaction to the transaction summary file to be written
deposit()	Brings the user through the process of depositing funds and validates their input	None	If successful sends the transaction to the transaction summary file to be written
transfer()	Brings the user through the process of transferring funds and validates their input	None	If successful sends the transaction to the transaction summary file to be written

input_validation.py

Class: SessionTypes(Enum): Contains enumerable constants for checking and setting session type

Class: InputValidation

Method / Class	Desc	Parameters	Output
__init__()	Takes current session type and sets it for checking permissions against	Session_type = session type from enum class	NA
valid_account_num()	Verifies that account number is not larger or smaller than 7 digits.	accountNum = account # getting its validity checked.	True if account # is valid, False if account # is not valid.
valid_account_name()	Verifies that account name does not begin or end with a ' ' and that its length is no shorter than 3 and no longer than 30.	accountName = account name getting its validity checked.	True if the account name is valid, False if account name is not valid.
valid_check_limits ()	Verifies that amount is of type integer and that it is within permissions for session type	Atm_limit = Atm limit for transaction type. Amount = amount of money being used in transaction	True if amount is within permissions for session type and is in the correct format, False otherwise.

transaction_summary.py

Class: TransactionSummary

Method	Desc	Parameters	Output
__init__()	Opens transaction summary file and saves it to TransactionSummary object.	File path to transacttion_summary.txt	NA
add_transaction()	Formats and writes transaction to transaction summary file. Utilizes default parameter values for unsupplied fields.	transaction_type = Type of transaction (WDR, DEP..) to_account = first account number amount = Amount of money for transaction. from_account = second account number name = account name	Formatted string written to Transaction Summary file. String formatting: *"transaction_type , to_account, amount, amount, from_account, name"*
transaction_close()	Closes and saves transaction summary file.	NA	No output but transaction summary file is available where it was initially accessed from

valid_accounts.py

Class: ValidAccounts

Method	Desc	Parameters	Output
__init__()	Reads through valid accounts file and saves each account in a dictionary.	file_path = a .txt file formatted according to specifications holding valid accounts	Creates dictionary of valid accounts stored in ValidAccount Object
add_valid_account()	Adds a new account to the valid accounts dictionary.	account_num = account number of new account to be added to the dictionary,	Updated Dictionary
delete_account	Deletes account from valid account dictionary	Accoun_num = number of the account which is going to be deleted	Updated Dictionary
contains_account()	Checks for account in the dictionary to see if it is valid.	account_num = number of account which is having its validity checked.	True if the account is valid, False if it is not.