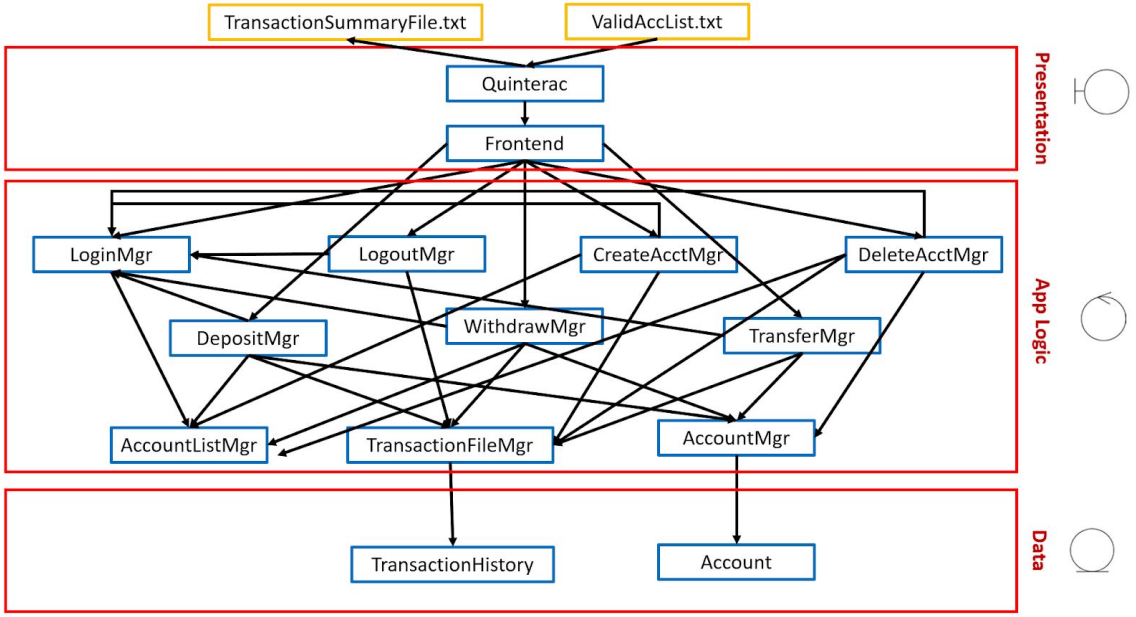
**Assignment 2 - Design Document**

The architecture we are adopting is the entity-control-boundary class layered architecture. An entity class stores persistent data in the data layer, a control class controls the logic of the program in the app logic layer while a boundary class interacts with the users through the presentation layer. Input and output files to the program are passed in and out of the program through the boundary class in the presentation layer.

The diagram below illustrates the classes that belong to the different layers. Each layer only interacts with the layer above and below it, hence reducing dependencies. For the Class Diagram (shows classes and methods), please see “Assignment 2 - Class Diagram”.




| | Class | Intention | Attributes | Methods |
|---|---|---|---|---|
| **Boundary** | Quinterac | Launch Quinterac program; handle | - | main - method to start Quinterac program |

| | | | | |
|---|---|---|---|---|
| | | interfacing between front-end and back-end; reads in input files of ValidAccList.txt and blank TransactionSummaryFile.txt | | |
| | Frontend | Launch frontend program | - | mainFrontend - method to start frontend program |
| **Control** | LoginMgr | Manage login operations and exceptions | loggedIn<br>mode | login - method to perform login operation<br>isLoggedIn - checks if user is logged in<br>checkMode - checks the current mode (machine or agent)<br>resetLogin - sets loggedIn to false and mode to null |
| | LogoutMgr | Manage logout operations and exceptions | - | logout - method to perform logout operation |
| | CreateAccMgr | Manage create account operations and exceptions | - | createacc - method to perform create account operations<br>checkAccNum - checks validity of account number<br>checkAccName - checks validity of account name |
| | DeleteAccMgr | Manage delete account operations and exceptions | - | deleteAcc - method to perform delete account operations |
| | DepositMgr | Manage all deposit operations and exceptions | - | deposit - method to perform deposit operation<br>atmCheckDepositValid - method to check if deposit amount is valid in machine mode<br>agentCheckDepositValid - method to check |

| | | | | |
|---|---|---|---|---|
| | | | | if deposit amount is valid in agent mode |
| | WithdrawMgr | Manage all withdraw operations and exceptions | - | withdraw - method to perform withdraw operation<br>atmCheckWithdrawValid - method to check if withdraw amount is valid in machine mode<br>agentCheckWithdrawValid - method to check if withdraw amount is valid in agent mode |
| | TransferMgr | Manage all transfer operations and exceptions | - | transfer - method to perform transfer operation<br>atmCheckTransferValid - method to check if transfer amount is valid in machine mode<br>agentCheckTransferValid - method to check if transfer amount is valid in agent mode |
| | ValidAccListMgr | Handle the reading in of the valid accounts list file; checks if an account number is valid | validAccList (HashMap) | readValidAccList - reads in the valid account list file<br>checkAccNumtExists - checks if the account number exists in the file<br>removeAccNum - removes account number from hashMap |
| | TransationFileMgr | Manage all the transactions performed in the day | transactionsList (arrayList of transactionHistory objects) | resetTransactionList - clears arraylist of transactions<br>addDepTransaction - adds deposit transaction into transaction summary file<br>addWdrTransaction - adds withdraw transaction into transaction summary file<br>addXfrTransaction - adds transfer |

| | | | | |
|---|---|---|---|---|
| | | | | transaction into transaction summary file<br>addNewTransaction - adds account creation transaction into transaction summary file<br>addDelTransaction - adds account deletion transaction into transaction summary file<br>writeToTransactionFile - writes transactionList to transaction summary file |
| | AccMgr | Manage the hashMap of accounts which perform transactions in each day | dailyAccMap (HashMap) | resetDailyAccMap - clears hashmap of accounts<br>checkAccExistsInMap - checks if the account exists in dailyAccountMap<br>addAccToMap - adds an account to dailyAccountMap<br>checkDailyDepositLimit - checks dailyDepositLimit for a particular account<br>performDailyDeposit - increase dailyDeposit amount for a particular account<br>checkDailyWithdrawLimit - checks dailyWithdrawLimit for a particular account<br>performDailyWithdraw - increase dailyWithdraw amount for a particular account<br>checkDailyTransferLimit - checks dailyTransferLimit for a particular account<br>performDailyTransfer - increase dailyTransfer amount for a particular account |

| | | | | |
|---|---|---|---|---|
| **Entity** | TransactionHistory | Store the information of each transaction, which is subsequently written to the transaction summary file | transactionCode<br>firstAccNumber<br>amount<br>secondAccNumber<br>accName | TransactionHistory - constructor for each transaction history<br>getTransactionCode - gets transaction code<br>setTransactionCode - sets transaction code<br>getFirstAccNum - gets first account number<br>setFirstAccNum - sets first account number<br>getAmount - gets amount<br>setAmount - sets amount<br>getSecondAccNum - gets second account number<br>setSecondAccNum - sets second account number<br>getAccName - gets account name<br>setAccName - sets account name |
| | Account | Keep track of the daily deposit, withdraw and transfer limits of each account | accNum<br>dailyDeposit<br>dailyWithdraw<br>dailyTransfer | Account - constructor for each account<br>getAccNum - gets account number<br>getDailyDeposit - gets daily deposit amount<br>increaseDailyDeposit - increases the daily deposit amount<br>getDailyWithdraw - gets daily withdraw amount<br>increaseDailyWithdraw - increases the daily withdraw amount<br>getDailyTransfer - gets daily transfer amount<br>increaseDailyTransfer - increases the daily transfer amount |