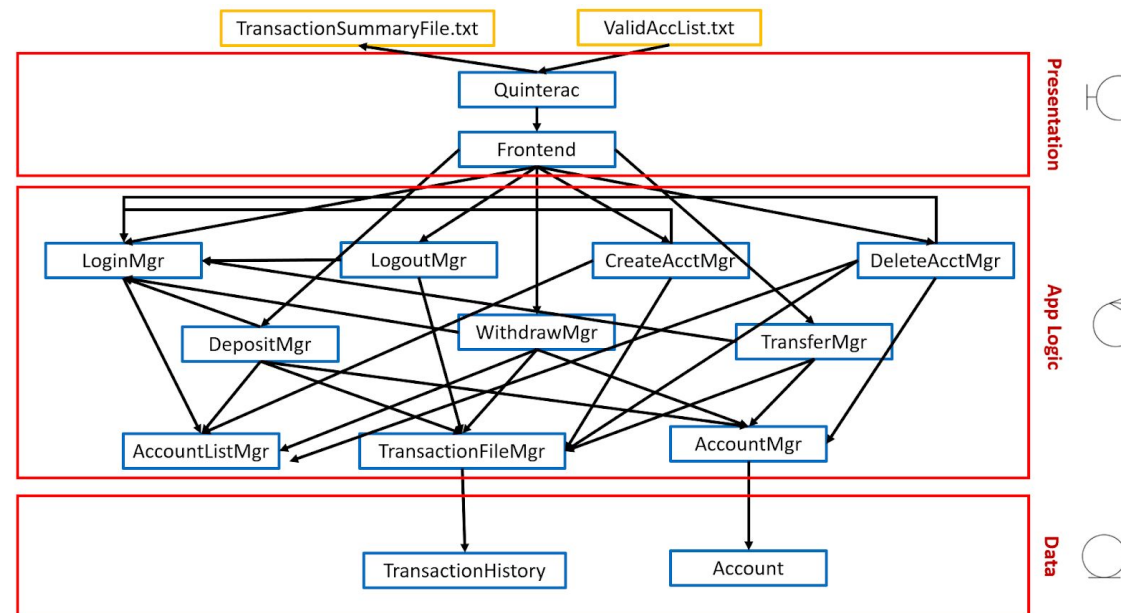


## Assignment 2 - Design Document

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.



	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
<b>Control</b>	LoginMgr	Manage login operations and exceptions	loggedIn mode	login - method to perform login operation isLoggedIn - checks if user is logged in checkMode - checks the current mode (machine or agent) 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 checkAccNum - checks validity of account number 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 atmCheckDepositValid - method to check if deposit amount is valid in machine mode 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 atmCheckWithdrawValid - method to check if withdraw amount is valid in machine mode 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 atmCheckTransferValid - method to check if transfer amount is valid in machine mode 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 checkAccNumtExists - checks if the account number exists in the file addAccNum - adds account number from hashMap 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 addDepTransaction - adds deposit transaction into transaction summary file addWdrTransaction - adds withdraw

				transaction into transaction summary file addXfrTransaction - adds transfer transaction into transaction summary file addNewTransaction - adds account creation transaction into transaction summary file addDelTransaction - adds account deletion transaction into transaction summary file 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 checkAccExistsInMap - checks if the account exists in dailyAccountMap addAccToMap - adds an account to dailyAccountMap checkDailyDepositLimit - checks dailyDepositLimit for a particular account performDailyDeposit - increase dailyDeposit amount for a particular account checkDailyWithdrawLimit - checks dailyWithdrawLimit for a particular account performDailyWithdraw - increase dailyWithdraw amount for a particular account checkDailyTransferLimit - checks dailyTransferLimit for a particular account performDailyTransfer - increase dailyTransfer amount for a particular account

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