**Description of the Intention of Each Method and Class**

**Akira Aida - 100526064**

**Kathryn McKay - 100524201**

**Alexander Wheadon - 100514985**

|  |  |
| --- | --- |
| **Method or Class** | **Description** |
| class Account | Stores information about a customer's account's balance, etc. |
| Account() | Constructor |
| std::map<std::string, std::vector<Account\*> > AccountParser::Parse(const char\* fpath) | Convert from file to accounts map. |
| int AccountStatus::QueryAccountStatus(Account\* account) | Returns one of AccountStatus: Active, disabled, etc. Where a deleted account state takes precedence. |
| std::string AccountStatus::GetErrorMessage(int status) | Returns the message corresponding to an account status. That means error messages except for when status == kActiveAccount, in which case it's just a  statement that the account is active. |
| class Commands | Bank System storing, tracking, and querying the accounts. The commands the user and admin use during runtime. |
| Commands::Commands() | Constructor |
| void Commands::SetAccounts(std::map<std::string, std::vector<Account\*> >&& accounts) | Gives map created from the Parse method to this class. |
| void Commands::login() | Corresponds to 'login' command. |
| void Commands::withdrawal() | Corresponds to 'withdrawal' command. |
| Void Commands::transfer() | Corresponds to 'transfer' command. |
| void Commands::paybill() | Corresponds to 'paybill' command. |
| void Commands::deposit() | Corresponds to 'deposit' command. |
| void Commands::create() | Corresponds to 'create' command. Admin only. |
| void Commands::delete\_account() | Corresponds to 'delete' command. Admin only. |
| void Commands::disable() | Corresponds to 'disable' command. Admin only. |
| void Commands::changeplan() | Corresponds to 'changeplan' command. Admin only. |
| void Commands::logout() | Corresponds to 'logout' command. Causes output to the transaction file. |
| void Commands::enable() | Corresponds to 'enable' command. Admin only. |
| bool Commands::UserExists(std::string name) | Check if user exists |
| bool Commands::CheckUnit(double amount) | Checks if 'amount' is valid currency. |
| std::string Commands::DetermineSession() | (private function) Determines if it's a standard or admin session |
| Account\* Commands::GetAccount(std::string name, int account) | (private function) Returns nullptr if the name/account pair is not found in system. |
| std::string Commands::GetAccountOwner(int account) | (private function) Find customer name corresponding to account number. Returns empty string if account was not found |
| void Commands::PushTransactionRecord(int code, std::string name, int account\_number, double money, std::string misc); | (private function) Pushes transaction record with that info onto stack so that transactions file can later be created on logout. Name, account\_number, money, and misc have default values. |
| std::string Commands::FitStringToSpace(std::string string, size\_t size, char fluff, bool align\_right) | (private function) Formats string to match certain number of characters, using fluff to fill space. align\_right has a default value of true. |
| std::string Commands::PromptForAccountHolder() | (private function) Retrieves the account holder's name. |
| bool Commands::CheckLogin(bool admins\_only) | (private function) Prints an error if the user is not logged in. Prints an error if the user is not admin, if admins\_only is set to true. admins\_only has a default value of “false”. |
| double Commands::GetTransactionCharge(std::string name, int account\_number) | (private function) Retrieves Transaction Charge for a particular account. Behaviour is undefined if the account doesn't exist. |
| void TransactionIO::PrintToTransactionFile(std::deque<std::string>\* transactions, std::string fpath); | Consumes deque; appends to transactions file. fpath defaults to “transactions.txt” |
| namespace AccountStatus | Defines the assessment of an account's validity e.g. is it available for use. Also provides error messages corresponding to those states. |
| AccountStatus enum | Representation of various ways an account can be invalid; or valid. |
| namespace AccountParser | Contains utility to parse in bank customer information from file. |
| namespace TransactionIO | Functions enabling the use of the transactions file. |