**A BRIEF DESIGN DESCRIPTION**

The program composes of 5 classes: Transaction, Money, Account, BSTree, and JollyBanker. A driver\_main will put this all together to run the program. Transaction is a class that functions the transaction processes. Money is a class that provides operation (withdraw/add/check balance) on a specific type of funds. Account is a class that stimulates a client account. BSTree is a class that has functions to store/add/remove accounts. JollyBanker is a class that puts the 3 main phases of the program together. BSTree will store Account objects and a queue here will store the Transaction from the input file. Transaction history is made sure to be saved and printed out if needed.

**CLASSES INTERACTION**

* Transaction class performs the transaction processes.
  + Transaction class gathers all the information needed to perform a transaction: transaction type, user full name, account ID, fund number, receiving account ID, receiving fund number. There are setters and getters for each variable.
  + There are default constructors and other constructors that take in arguments depending on the transaction type. Those constructors serve for cases such as opening an account, depositing assets into a fund/withdrawing assets / transferring assets between funds (within one account), transferring assets between funds (to another account in different money type), displaying the transaction history, displaying the history of specific money type, failed transaction when withdrawing funds from the same account, and failed transactions when transferring money from one account to another.
  + There is one destructor.
  + Function bool IsFailed() to check for failed transactions.
* Money class operates on a specific type of fund. There are 8 funds in this program.
  + A string is needed to store the name of a fund, an integer to store the amount of balance, and a vector of Transactions to store the transaction history of a fund. There are getters and setters for fund name and balance.
  + Function AddMoney() adds a certain amount of money to a fund.
  + Function MinusMoney() withdraws a certain amount of a fund.
  + Function CheckMoney() checks the balance of a fund.
  + Function RecordTransactionHistory() saves the transaction history on a fund.
  + Function PrintSpecificFund() prints out the transaction history of a fund.
  + Function PrintTransactionHistory() prints out history of all transactions.
* Account class operates on a bank user’s account information.
  + The Account class gathers all information needed on a client account: ID (int), first name (string), last name (string), number of a fund (int), and an array of type Money to store 8 funds that each account has. There are getters and setters.
  + There is a default constructor, a constructor that has an ID, first and last name as arguments, and a destructor for Account objects.
  + The number of each fund is stored as global constant ints. There’s also a global constant vector of string to store all the names of the funds.
  + Function GetFundName(int fund\_id) const gets fund name from a fund ID.
  + Function GetFundBalance(int fund\_int) const gets balance of a specific fund.
  + Function AddMoney() adds an amount of money to a specific fund of the account and saves the record.
  + Function SubtractMoney() subtracts an amount of money from a specific fund.
  + Function Withdraw() withdraws or transfers an amount of money from a fund to another one in the same account and saves the record.
  + Function RecordTransaction() saves the record of transactions on an account.
  + Function CheckInfo() checks to see if there’s invalid activity on the account.
  + Function PrintSpecificFundHistory() prints out the transaction history of a specific fund.
  + Function PrintHistory() prints out the transaction history of all funds on the account.
* BSTree class implements a binary search tree to store the accounts.
  + We use a struct Node with a pointer to Account, Node pointer right and left, and a node pointer root.
  + Insert() function inserts a client account to the BSTree.
  + Retrieve() function retrieves an object of an account.
  + void Empty() function removes all the accounts on the BSTree.
  + isEmpty() function checks if the BSTree is empty.
  + Display() function prints out all the accounts that are in the BSTree.
* JollyBanker class implements the banking program in three phases:
  + There is a queue of Transaction objects and a BSTree to store accounts.
  + Function ReadInput() will read in a string of transactions from a file into an in-memory queue.
  + Function StoringAccounts() will next read from the queue and process the transactions in order. All the client accounts will be stored in the BSTree.
  + Function Display() will print out all open accounts and balances in those accounts after the queue is depleted.

**PSEUDOCODE**

The Transaction class implements all the functions that can stimulate the transaction process. The Money class inherits the Transaction class, operates on a specific fund, and saves each operation as a transaction in a vector of Transaction. The Account class inherits the Money class and builds each client account. The BSTree class implements a Binary Search Tree to store Account objects. The JollyBanker class puts the 3 main phases of the program together with a queue that stores Transaction objects and a BSTree that stores Account objects. The driver\_main will call functions in the JollyBanker class to put everything together and run the program.

**VISUAL OF KEY DATA STRUCTURES**

We have balance as integers. We also have a Binary Search Tree (BSTree) to store Account objects and a queue to store Transaction objects.

A picture containing diagram

Description automatically generated

Diagram

Description automatically generated