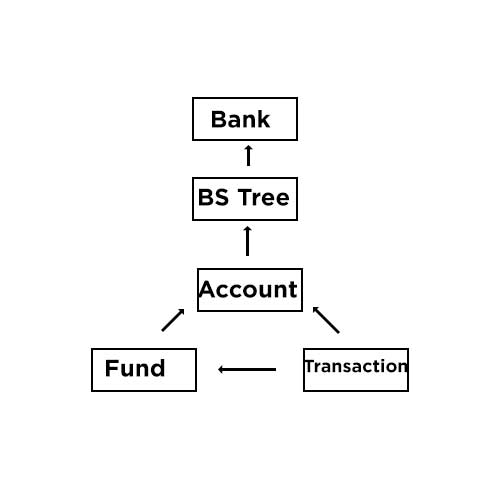
Juliano Nguyen

11/30/16

Project 5 Design

The design will have a bank class, binary search tree class, account class, fund class, and transaction class. The bank class will read in the transactions from the text file and process them. The binary search tree will store all of the accounts. The account class will contain information such as their name, a 4-digit id, and a fund id. The account can deposit money to a fund and withdraw money from a fund. The fund class will control the balance of each fund for an account. The fund also checks if there is enough money when withdrawing more than the current balance. The transaction class will take in a letter to determine the type of transaction for an account, like deposit or withdraw.

The bank will read in the transactions and store the accounts in a binary tree. Each account will have all the different funds. Each account can make a transaction with itself or with another account. Making a transaction will change a fund’s balance.



**Class**: Bank

**Functions**

public:

bool ReadTransactions(string filename);

void ProcessTransactions();

private:

queue<Transaction> list;

**Class**:Binary Search Tree

**Functions**

public:

BSTree();

~BSTree();

bool Insert(Account \*);

bool Retrieve(const int &, Account \*&) const;

void Display() const;

void Empty();

bool isEmpty() const;

private:

struct Node

{

Account \*pAcct;

Node \*right;

Node \*left;

};

Node \*root;

**Class**:Client Account

**Functions**

public:

ClientAccount();

ClientAccount(Transaction client);

~ClientAccount();

string getFirstName() const;

string getLastName() const;

int getId() const;

Fund getFund(int fundId) const;

void History();

bool Deposit(int amount, int fundId);

bool Withdraw(int amount, int fundId);

bool operator==(const ClientAccount &one) const;

bool operator<(const ClientAccount &one) const;

private:

string firstName;

string lastName;

int id;

vector<Fund> funds;

**Class**: Fund

**Functions**

public:

Fund();

~Fund();

int getBalance() const;

void changeBalance(int amount);

bool isEnough(int amount);

private:

int balance;

**Class**:Transaction

**Functions**

public:

Transaction(string letter, int id, int fundId, int amount);

Transaction(string letter, int id, int fundId, int amount, int transferId, int transferFundId);

Transaction(string letter, int id, int fundId);

Transaction(string letter, string last, string first, int id);

string getFirstName() const;

string getLastName() const;

int getId() const;

int getFundId() const;

int getAmount() const;

int getTranferId() const;

int getTransferFundId() const;

private:

string firstName;

string lastName;

int id;

int fundId;

int amount;

int transferId;

int transferFundId;

bool Insert(Account \*)

{

Loop to go through the tree, starting from the left

Check if inserting account is less/greater than left account

If less, than continue in left branch of that account; otherwise the right

Insert account when no more accounts to compare

}