Jessica Runandy
November 26, 2019
Program 5 Design

Objects:

- **BSTree(.h/.cpp)**
  //Binary Search Tree: retrieve, insert, display, empty, isEmpty, remove (do remove last)
  Public:

```
BSTree();
~BSTree();

bool Insert(Account *);
bool Retrieve(const int &, Account * &) const;
void Display() const;
void Empty();
void Remove();
bool isEmpty() const;
```

  Private:

```
struct Node {
        Account *pAcct;
        Node *right;
        Node *left;
};
Node *root;
};
```

- **Transactions (.h/.cpp)**
  Public:

```
Transactions();
Transactions(vector<string> history, char transactionType[]);
bool OpenAccount(Account &client, const int amount);
bool Withdraw(Account& client);
bool Deposit (Account& client, const int amount);
bool Transfer (Account& client1, const int amount, Account& client2);
void DisplayHistory(Account& client) const;  // All transactions for client account
```

  Private:

```
vector<string> clientHistory;
char transactionType[5];        // O, D, W, T, H
```

- **Account (.h/.cpp)** // Each account has 10 funds by default
  Public:

```
Account();
Account(string firstName, string LastName, int ID, int balance, int fundType,
vector<Fund> funds);
```

```
string getFirstName() const;
string getLastName() const;
int getID() const;
int getBalance() const;

void setFirstName(string &firstName);
void setLastName(string &lastName);
void setID(int &ID);
void setBalance(int &balance);

Private:
string firstName;
string lastName;
int ID;    // 1000 <= ID <= 9999
int balance;
vector<Fund> funds; // all accounts have 10 funds/ fund types
```

- **Fund (.h/.cpp)**

```
Public:
Fund();
Fund(int balance, string fundName, vector<string> fundHistory);
~Fund();

int getBalance() const;
string getFundName() const;

void setBalance(int &balance);
void setFundName(string &fundName);

void displayFundHistory();
Private:
int balance;
string fundName;
vector<string> fundHistory;
```

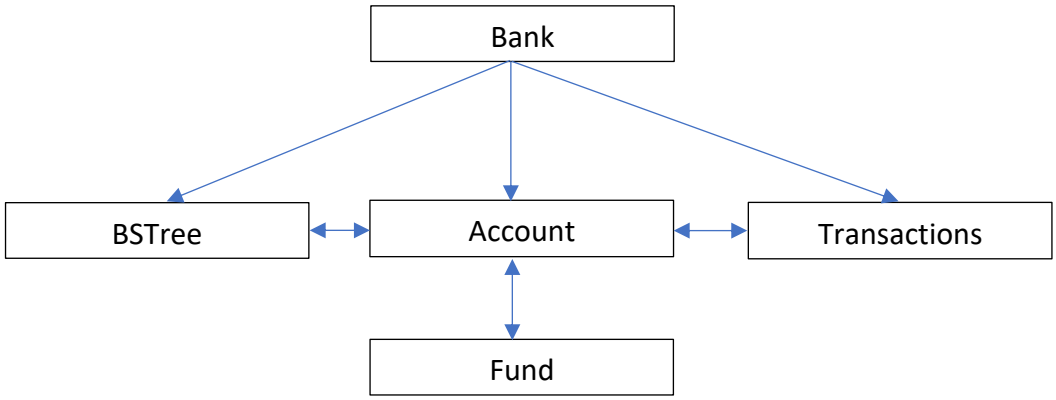
- **Bank (.h/.cpp)**

```
Public:
Bank();
~Bank();
void BuildQueue(const string fileName);
void DisplayHistory();

Private:
queue<Transactions> transactionQueue;
BSTree binarySearchTree;
```

How classes interact:

- Bank builds queue containing the accounts (account class) and puts the account information into the queue
- BSTree uses queue to find which account is next on the queue (BSTree class, account class)
- Determines the transaction type using transaction class to open, deposit, withdraw, transfer, and display history.
- Transaction calls on Account class to change the balance depending on the transaction type and fund type (account class, fund class)



Program flow:

1. Bank class: read string of transactions from text file and put it in a queue
2. Bank class: Store client accounts in binary search tree (BSTree – retrieve, insert, empty)
3. Pull the item from the queue (FIFO) and search it in the binary search tree
4. Read the first letter (transaction type)
5. Determine the type of transaction (open, deposit, withdraw, transfer, display history).
6. Go through with the transaction using the client ID by depositing/withdrawing/transferring/displaying history based on the fund type (Money Market, Prime Money Market, etc.).
    - Fund 0 and 1, 2 and 3 are linked.
    - If account number already used, throw exception
    - If amount they want to withdraw is too much, throw exception
    - A transaction that would cause negative balance is error unless it can take amount from linked account (only for 0 and 1, 2 and 3)
7. Add the transaction information into the client account history and add transaction information into the fund history.
    - History: 100 was transferred from Account ID 1234 to Account ID 2345
8. Once queue is empty, print out all history from the history vector.

Queue:

Transaction1: D 12341 100	Transaction2: W 12340 500	Transaction3: T 12340 1000 12341	Transaction4: T 12340 1000 56780	Transaction5: H 1234

Binary Search Tree: