Funds will be in an array that is part of each account, index 0-9 will correlate with a fund type, and int amounts at each index will represent amount in the account

Fund	10,000	500	350	0	9,000	0	0	100	0	0
	0	1	2	3	4	5	6	7	8	9

Each Fund will have a vector of transactions, this will be used to keep a history of all transactions for each fund. Use a vector so that it can grow dynamically

Vector<Transaction> [t1, t2, t3, t4, t5]

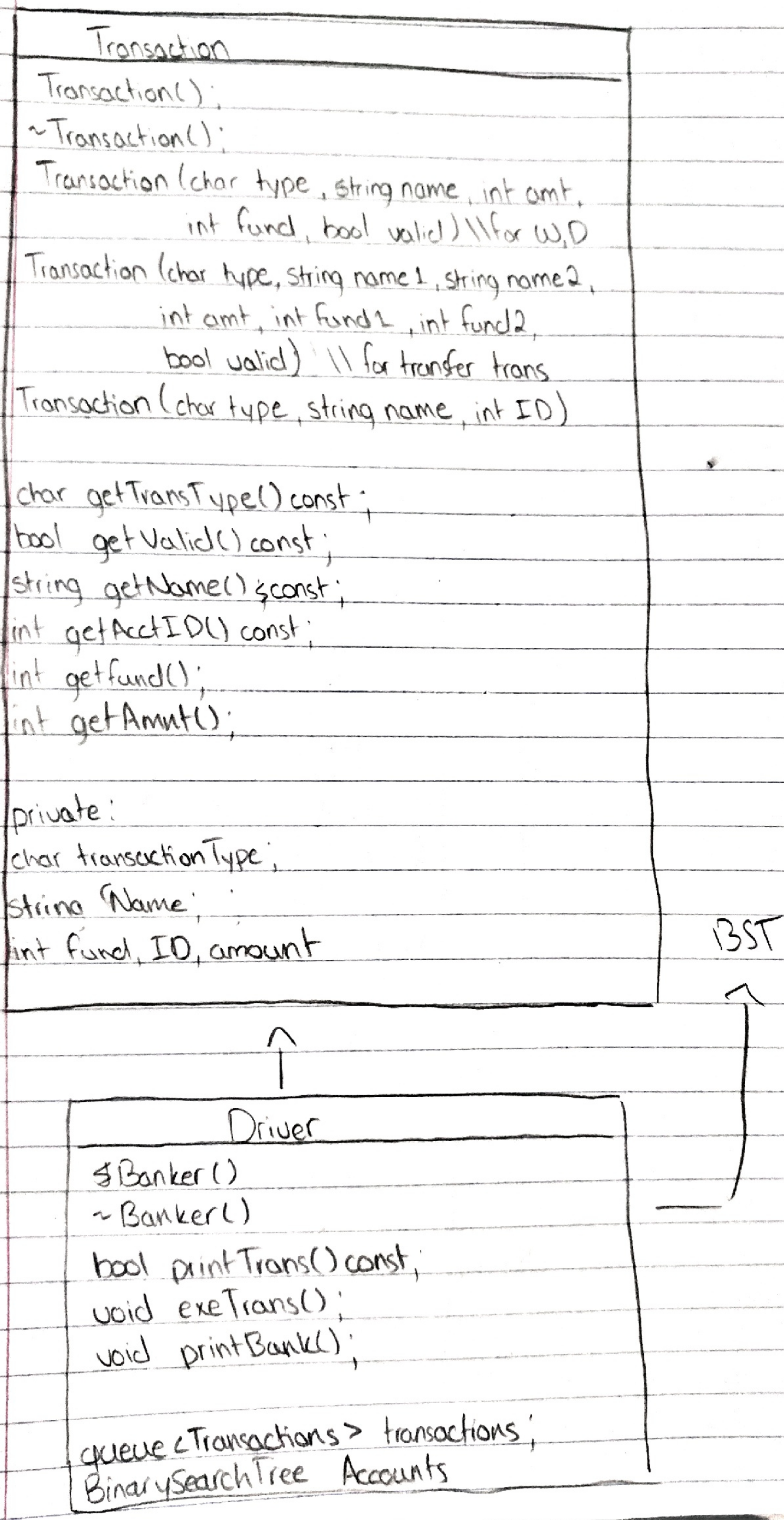
We can then have another vector that holds pointers to the different fund vectors, allowing us to access and print all transactions for the account (not 100% sure on this)

Program Flow

The driver function will read accounts from a file and insert into the BST. Then will read transactions from a file and fill the queue. The Bank class will then process the transactions, and once that is complete the updated bank will be displayed

Bank

The bank handles the transactions, and stores all of the accts with the BST. Each transaction will be processed, and the appropriate action will take place, printing errors when neccessary



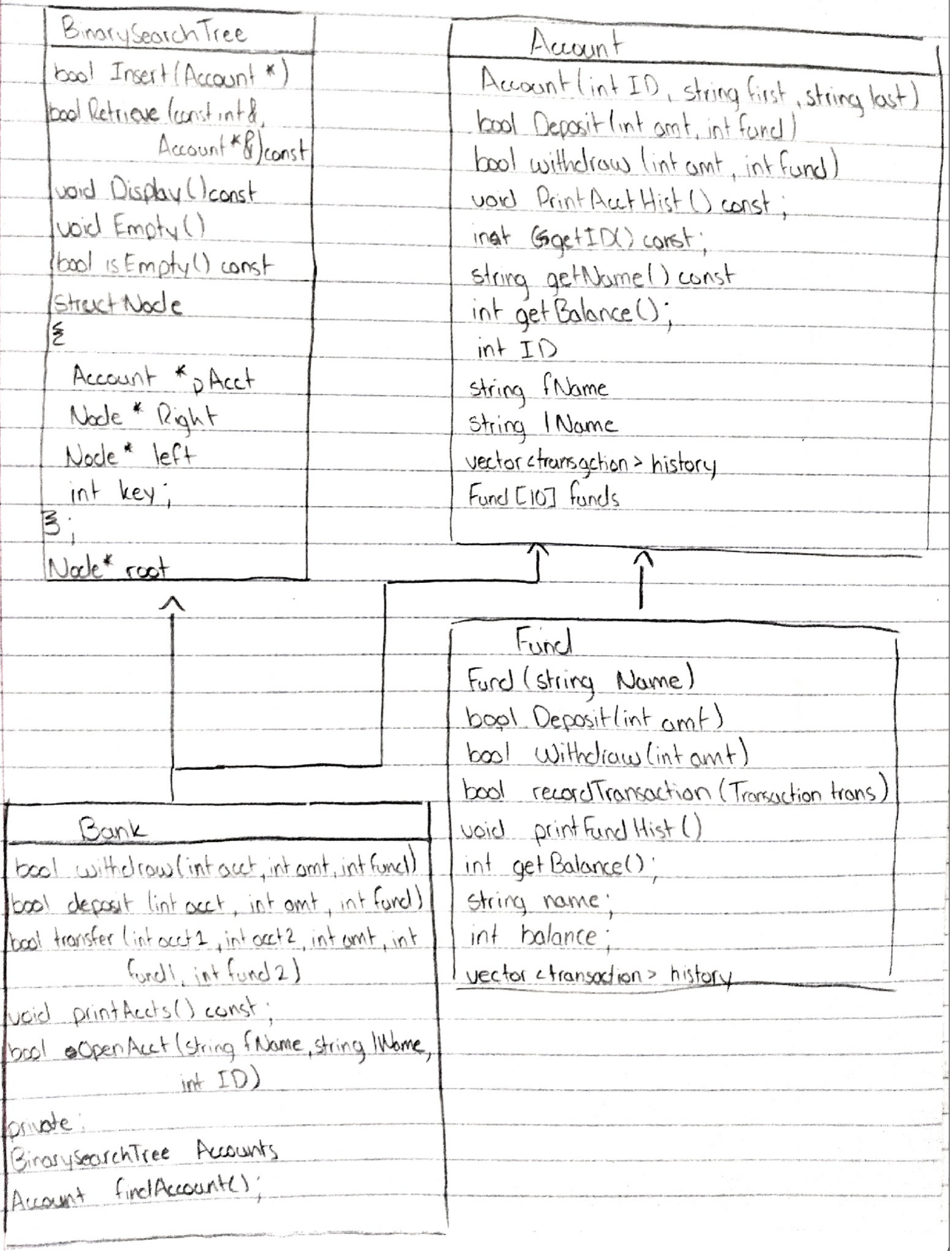
```
Transaction
Transaction();
~Transaction();
Transaction (char type, string name, int amt,
            int fund, bool valid) \\for W,D
Transaction (char type, string name1, string name2,
            int amt, int fund1, int fund2,
            bool valid) \\ for tranfer trans
Transaction (char type, string name, int ID)

char getTransType() const;
bool getValid() const;
string getName() ;const;
int getAcctID() const;
int getfund();
int getAmnt();

private:
char transactionType;
string Name;
int fund, ID, amount
```

```
Driver
$Banker()
~Banker()
bool printTrans() const;
void exeTrans();
void printBank();

queue <Transactions> transactions;
BinarySearchTree  Accounts
```


BinarySearchTree
bool Insert(Account *)
bool Retrieve (const int&, Account*&)const
void Display()const
void Empty()
bool isEmpty() const
Struct Node
{
Account *pAcct
Node * Right
Node* left
int key;
};
Node* root
Account
Account(int ID, string first, string last)
bool Deposit(int amt, int fund)
bool withdraw (int amt, int fund)
void PrintAcctHist() const;
int getID() const;
string getName() const
int getBalance();
int ID
string fName
String lName
vector<transaction> history
Fund [10] funds
Fund
Fund (string Name)
bool Deposit(int amt)
bool Withdraw(int amt)
bool recordTransaction (Transaction trans)
void printFundHist()
int getBalance();
string name;
int balance;
vector<transaction> history
Bank
bool withdraw(int acct, int amt, int fund)
bool deposit (int acct, int amt, int fund)
bool transfer (int acct1, int acct2, int amt, int fund1, int fund2)
void printAccts() const;
bool OpenAcct (string fName, string lName, int ID)
private:
BinarySearchTree Accounts
Account findAccount();