Omer Baldo (osb5)

Deep Kotadia(dk734)

## Client

The client program begins by taking a single input in the command line that is the server machine IP address. It attempts to connect to the server IP address (via port 8080) and if it fails it exits. If the connection is successful, two threads are spawn. The first thread deals with input from the client’s command line. The next thread deals with output from the server and printing to the client’s console.

These two threads cannot read/write data at the same time. This would make the terminal really sloppy. So we implemented a lock so that only one thread at a time could read or write.

## Server

Lets first talk about the structures the server uses. The first structure is a linked list of open sockets that the server has with clients called **socketNode.** The next structure **client**, stores connection information such as socket number, and the struct sockaddr\_in which contains:

-IP Address

-Family:

-Port Number

The final structure is a size 20 array of account -nodes. Each account node contains information on:

-Whether or not it is in use

-Balance in the account

-Account Name

When the server program begins to run, it first **opens** a socket so that it can communicate with the outside world. Then it **binds** this socket to an open **port**. One thread is automatically created that deals with printing out each account from the account array structure. Next the server **listens** for incoming connections (allowing up to 10 clients wait in the queue). When an connection is found, the server **accepts** it. When accept is called, a client structure is filled with the information on the new connection. The new opened socket is recorder in the socket-linked list, and the client structure is sent to a method that deals with input:

* open <accountname>

Creates a new account and adds in to the account array.

Errors:

* + No space in bank,
  + Account with same name already exists
  + Account name length is 0 or greater than 100
* start <accountname>

Starts customer session for a specific account

Errors:

* No account exists
* credit <amount>

Adds money to an account

Errors:

* Not in a customer session
* Amount is negative/0
* debit <amount>

Takes money to an account

Errors:

* Not in a customer session
* Amount is negative/0
* There is not enough money to take out
* balance

Prints the account balance

Errors:

* Not in customer session
* finish

Leaves the account

Errors:

* Not in customer session
* exit

Closes connection by stopping socket.