Stock Exchange

design, protocols, interfaces, modules, features

BAIG

2007/2008

Gengoux Pascal

Table of Contents

1 Introduction

1.1 Definition

The goal of this project is to design a simplified prototype of a stock exchange server, bank server and client application. The main focus would be on the network communications of these entities using TCP sockets.

1.2 Programming Language

We decided to program in C++ using “Codegear RAD C++ Builder 2007”. This a good language to start with GUI programming. It’s very close to the language Delphi (similar components) which is an advantage for some of us.

1.3 Operating System

The Applications; Stock Exchange, Bank Server and the Client will be able to run on an x86 Windows.
Unix/Linux is not supported.

1.4 Needs

The only thing needed to run and play the Stock Exchange games are in principle the exe’s of each Program (Stock Exchange Server, Bank Server and Client).There would be no additional programs/servers needed like BDE Client or Database Server.

2 Design

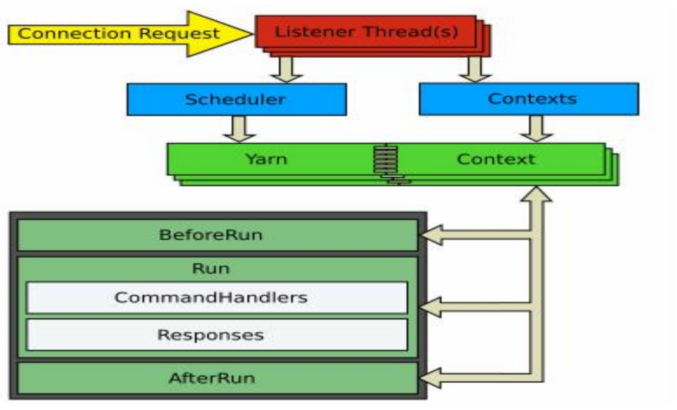
2.1 Components for connections

The components that we use for the connection between the different applications are:

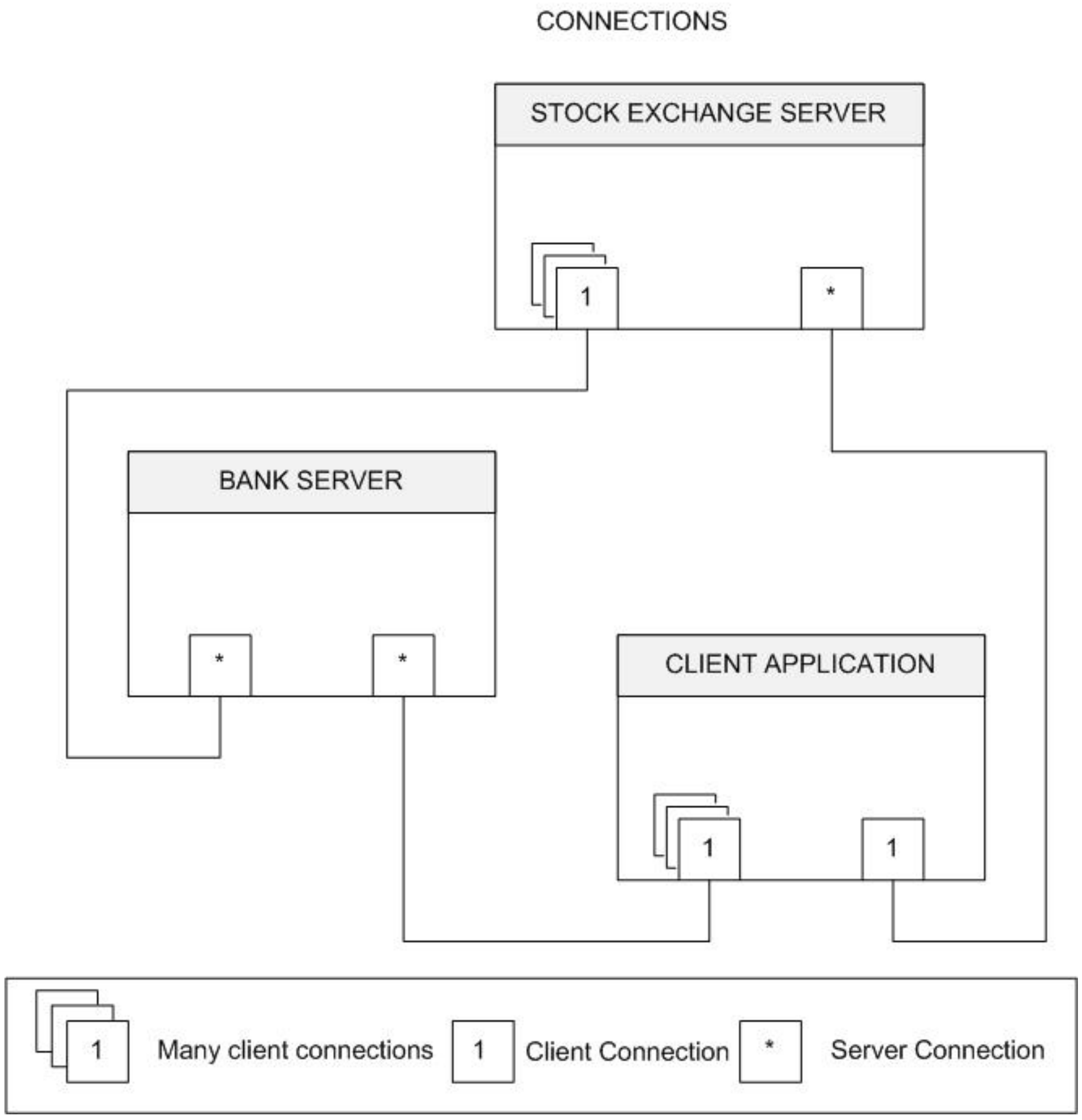
- IdCmdTCPClient
- idCmdTCPServer

These components are easy to use. For example they got an internal command handler which triggers the corresponding event depending which command was sent. Also the params could be retrieved by delimiting them by a character (our commands are separated by a space).

The Server component creates for each instance of a connection an own proper listener thread (view Picture below).



2.2 Global overview of the connections



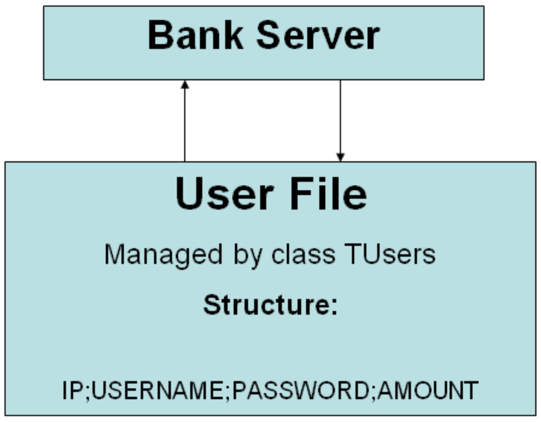
A Server connection in an application means that there is a design-time created idCmdTCPServer component.

A Client connection means that there is a design-time created idCmdTCPClient component.

Several “1” connections means they will be created at run-time depending on the need of client connections.

2.3 Modules

2.3.1 Bank Server

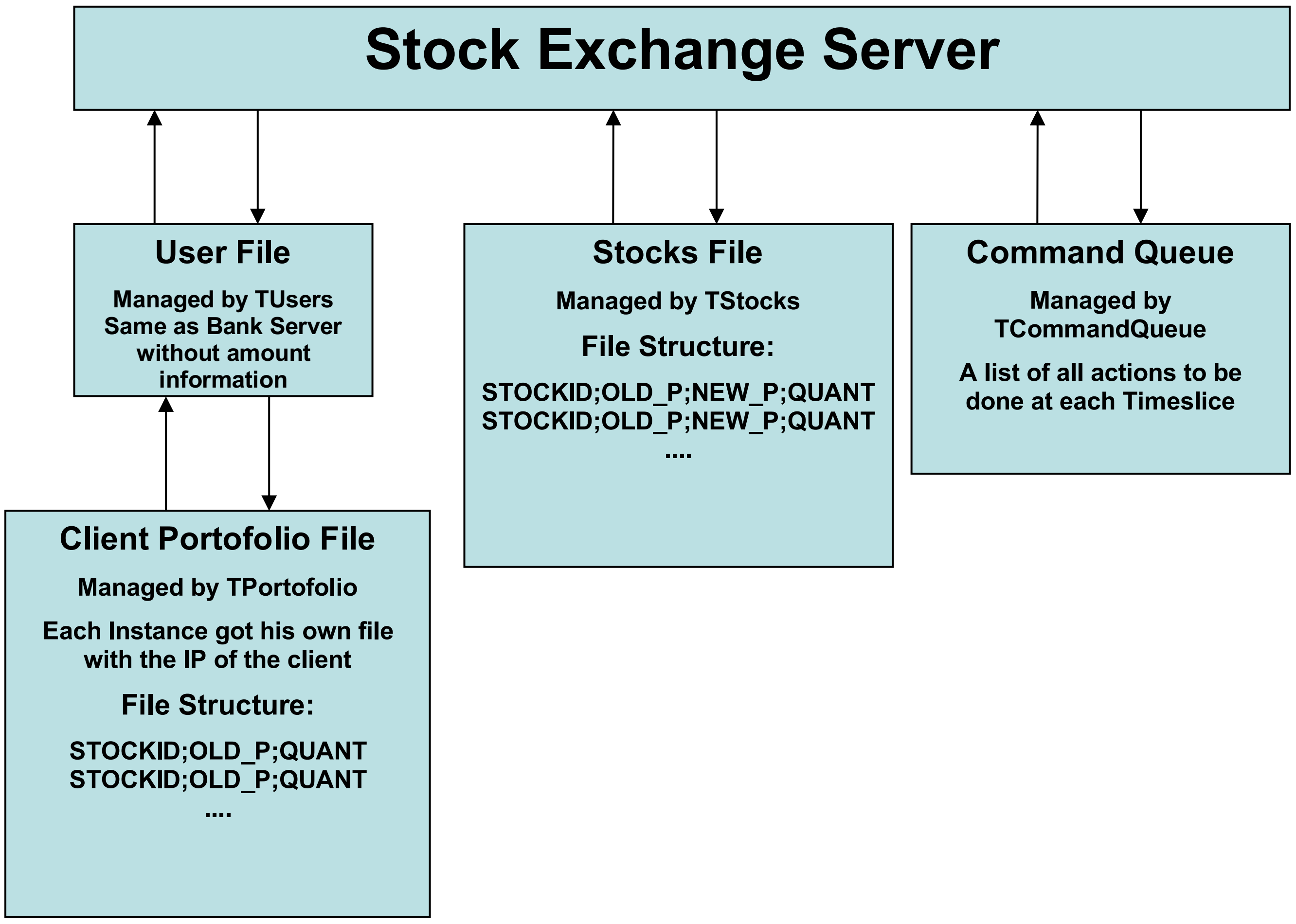


The Bank Server got an internal class TUsers which manages the Users. The Class manages the following:

1. Manage the user file
2. Update Amounts
3. Authentication of users
4. Delete / Add users

Depending on this class the server acts.

2.3.2 Stock Exchange Server



The stock Exchange Server is divided in 4 parts.

1. Users
   a. Manage the user file
   b. Update Amounts
   c. Authentication of users
   d. Delete / Add users
2. Client Portfolios
   a. Add, Delete Stock
   b. Change Quantity of a Stock
   c. List Stocks
3. Stocks
   a. Add, Delete Stock
   b. Change Quantity of a Stock
   c. List Stocks

4. Command Queue
    a. Collect the Buy & Sell Actions.
    b. Perform the Actions.
    c. Send reply to the source of the Buy & Sell Actions.

2.4 Justify the connections

2.4.1 Run-Time connections

A client should be able to handle at least two connections at the same time to several banks.

→ Run-time created client connections.

A Stock exchange server should be able to work with at least two banks at the same time.

→ Run-time created client connections.

2.4.2 Design-Time connections

A Bank server should handle more than 1 client connection at the same time. Fixed listening on a specific port.

→ Design-time created server connection

A Bank server should handle more than 1 stock exchange connection at the same time. Fixed listening on a specific port.

→ Design-time created server connection.

A Stock exchange server should handle more than 1 client connection at the same time. Fixed listening on a specific port.

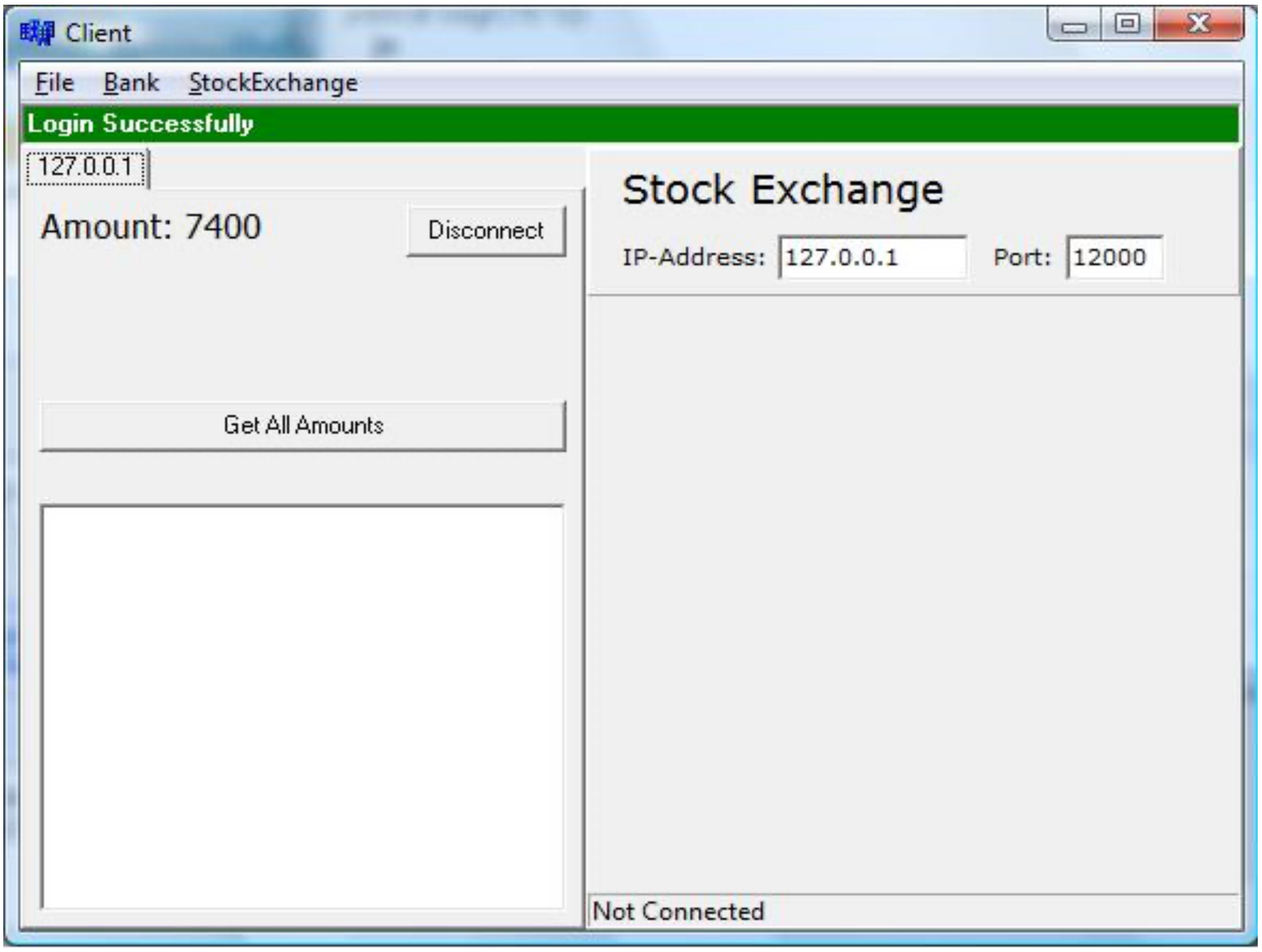
→ Design-time created server connection.

A Client should be able to handle one stock exchange connection.

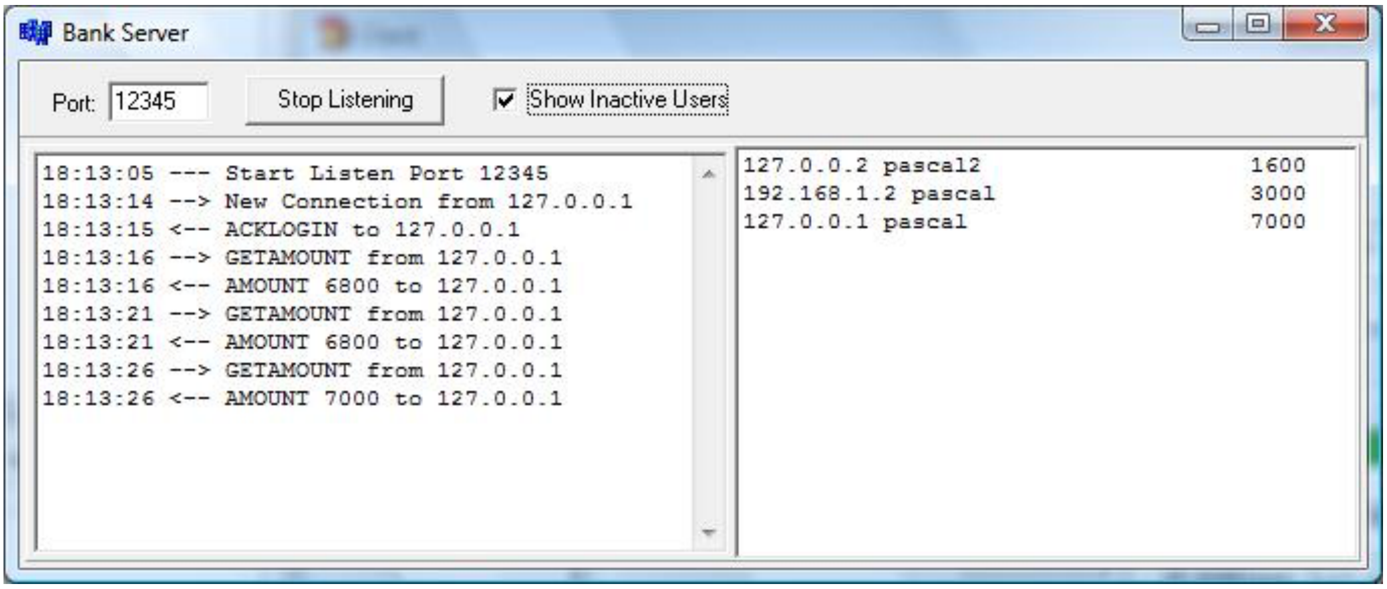
→ Design-time client connection

2.5 Maybe-Look-Likes

2.5.1 Client



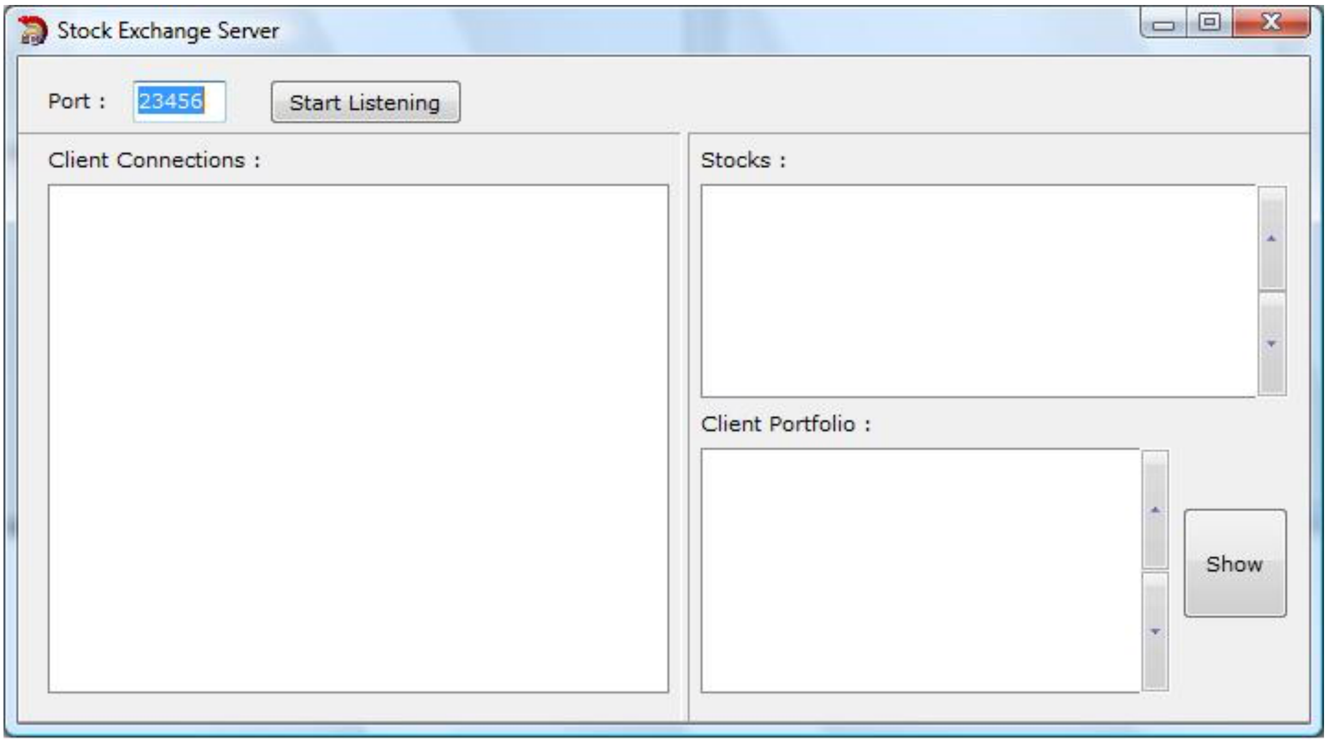
2.5.2 Bank Server



```
18:13:05 --- Start Listen Port 12345
18:13:14 --> New Connection from 127.0.0.1
18:13:15 <-- ACKLOGIN to 127.0.0.1
18:13:16 --> GETAMOUNT from 127.0.0.1
18:13:16 <-- AMOUNT 6800 to 127.0.0.1
18:13:21 --> GETAMOUNT from 127.0.0.1
18:13:21 <-- AMOUNT 6800 to 127.0.0.1
18:13:26 --> GETAMOUNT from 127.0.0.1
18:13:26 <-- AMOUNT 7000 to 127.0.0.1
```

```
127.0.0.2 pascal2                        1600
192.168.1.2 pascal                       3000
127.0.0.1 pascal                         7000
```

2.5.3 Stock Exchange Server



3 Protocols

3.1 Command List

Note that the Protocols that we use have the precondition that the program (server or client) can retrieve the IP of the client or server by the connection he made.

3.1.1 Client to Bank

- CREATE <Name> <Password>
- DELETE <Name> <Password>
- GETAMOUNT
- MOVE <Amount> <IP> ????
- LOGIN <Name> <Password
- GETALLAMOUNTS

3.1.2 Client to Stock Exchange

- CREATE <Name> <Password>
- DELETE <Name> <Password>
- LOGIN <Name> <Password>
- LIST <All / ME> (Me: just own Stocks>
- LISTCLIENTS
- BUY <StockID> <Quant> <BankIP>
- SELL <StockID> <Quant> <BankIP>

3.1.3 Stock Exchange to Client

- ACKCREATE <Name> <Password>
- NACKCREATE EXIST
- ACKDELETE <Name>
- NACKDELETE EXIST
- ACKBUY <StockID> <Quant> <BankIP>
- NACKBUY <StockID> <Quant> <BankIP> <Param>
    - BANK: Bank not reachable
    - DISP: Stocks not available for Quant
    - AMOUNT: Not enough Amount on Bank
- ACKSELL <StockID> <Quant> <Price>
- NACKSELL <StockID> <Quant> <Price>
    - BANK: Bank not reachable
    - DISP: Quant > available Quant.
- LISTME <StockID> <Quant> <OldPrice> <New Price> <StockID> …

  LISTALL START <Name> <StockID> <Quant> <OldPrice> <NewPrice> <StockID> … START <Name> <StockID> <Quant> <OldPrice> <NewPrice> <StockID> … …

3.1.4 Stock Exchange to Bank

- GET <Amount> <Client IP> → Get the money of a client from a bank
- SET <Amount> <ClientIP> → Give the money to a client from a bank

3.1.5 Bank to Client

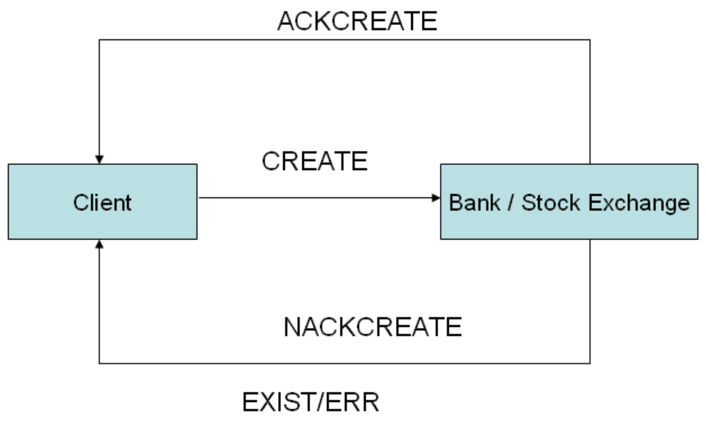
- ACKCREATE <Name> <Password>
- NACKCREATE
    - EXIST: Account exists
    - ERR: Other Error
- ACKLOGIN
- NACKLOGIN
    - EXIST: Client doesn’t exist
    - LOGIN: Wrong login / password
    - ERR: Other Error
- ACKDELETE
- NACKDELETE
    - EXIST: Client doesn’t exist
- ACKMOVE
- NACKMOVE
    - EXIST: Destination Account doesn’t exist
- AMOUNT <Amount> → Send the Amount information to the client

3.1.6 Bank to Exchange

- ACKGET <Amount> <ClientIP> <Param>
- ACKSET <Amount> <ClientIP> <Param>
- NACKGET <Amount> <ClientIP> <Param>
    - EXIST: Client doesn’t exist
    - DISP: Client Amount < Required Amount
    - ERR: Other Error
- NACKSET <Amount> <ClientIP> <Param>
    - EXIST: Client doesn’t exist
    - ERR: Other Error

3.2 Actions

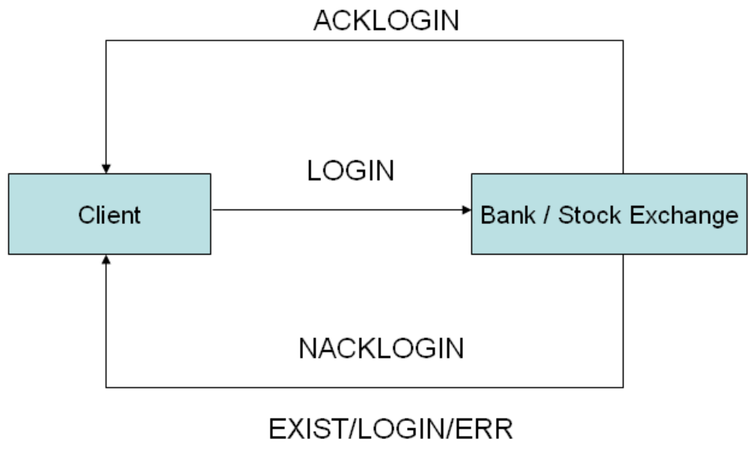
3.2.1 Create account on Stock Exchange or Bank



The client will send the Command CREATE with two param's (Username and Password) to the bank or Stock Exchange. Either the Account has been created (ACKCREATE) or not (NACKCREATE).

See Command List for Details.

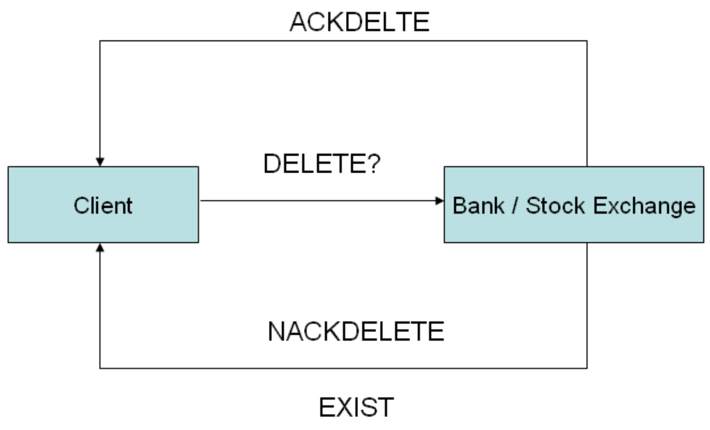
3.2.2 Login to a Bank or a Stock Exchange



A client will send a LOGIN request with two params (Username and Password).
Either the login will be successful (ACKLOGIN) or not (NACKLOGIN).

See Command List for Details.

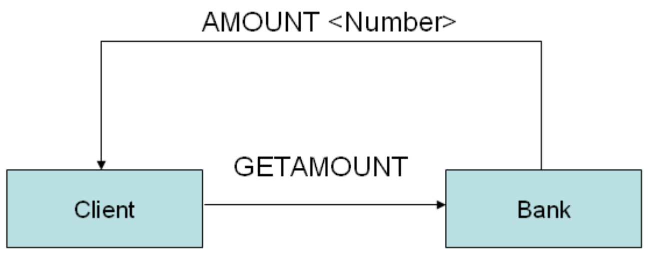
3.2.3 Delete Account on Stock Exchange or Bank



The Delete Command has two params (Username and password)
Either the Account will be deleted (ACKDELETE) or not (NACKDELETE)

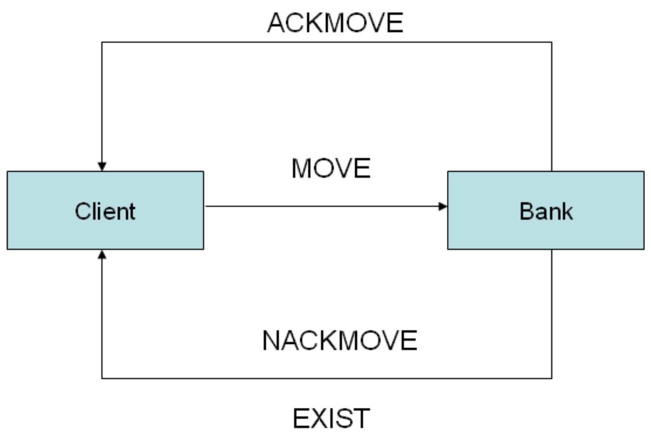
See Command List for Details.

3.2.4 Get Amount from a Bank



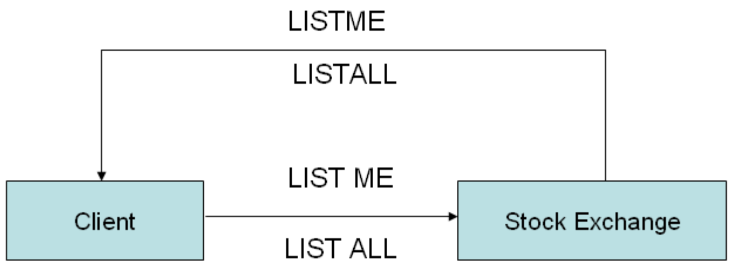
The client is able to get his current amount on a bank with the command GETAMOUNT. The answer AMOUNT will be followed with a param which contains the amount

3.2.5 Move Amount from one account to another



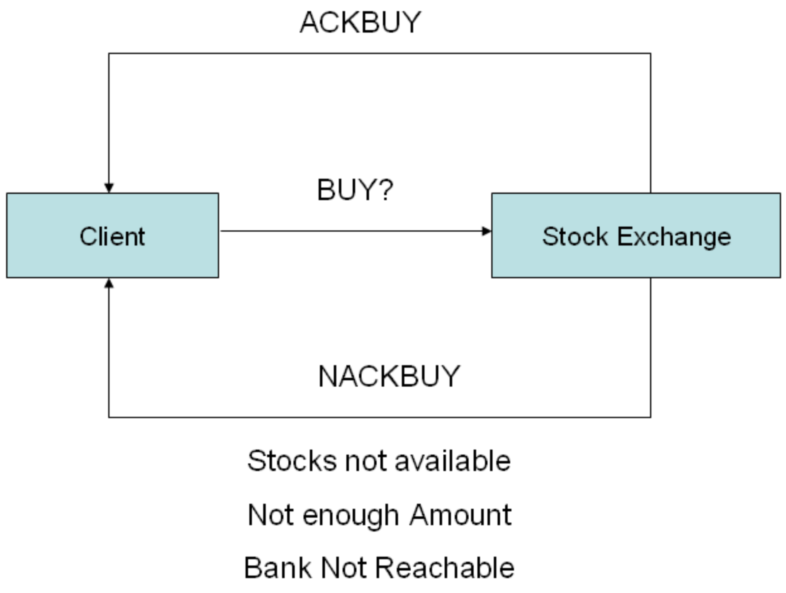
The Client sends the Command “Move” with the parameter AMOUNT and IP to the Bank
The Bank replies with ACKMOVE if it is ok or with NACKMOVE if it’s not EXIST

3.2.6 Get a list of own stocks (LISTME) or a list of all clients (LISTALL)



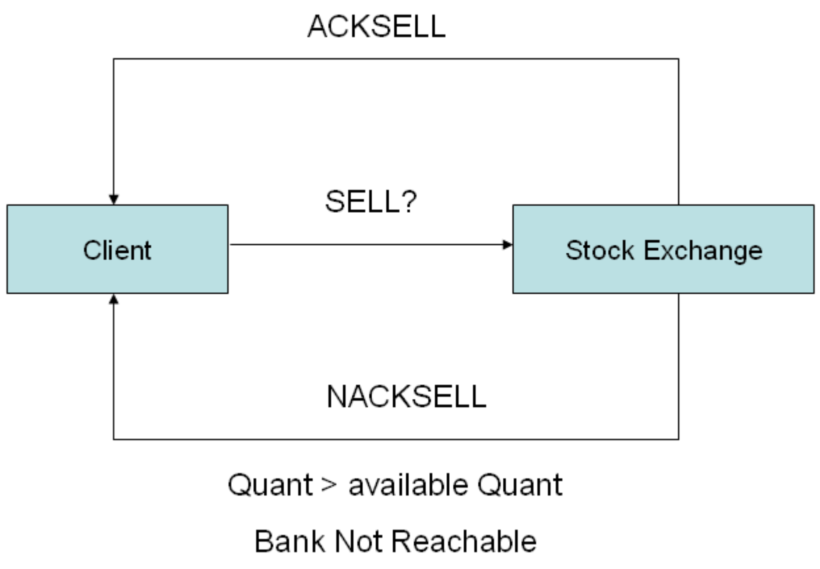
The Client sends the Command “LIST ME” to get his own portfolio from the stock exchange or “LIST ALL” to get the list of the portfolios of all the other clients

3.2.7 Buy Stocks on a Stock Exchange



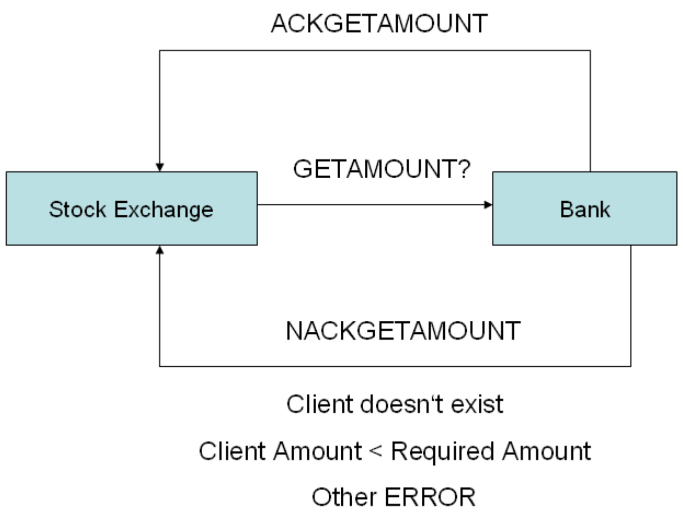
The Client sends the Command “BUY” with the arguments “STOCKID”, “QUANTITY” “BANKIP” to the Stock Exchange server if he wants to buy a stock. The Stock Exchange queries at the bank with the BANKIP if there is enough money. If there is enough money he validates the transaction buy sending back ACKBUY otherwise he sends NACKBUY. NACKBUY is also sent in the case that the stocks aren’t available or the bank isn’t reachable.

3.2.8 Sell Stocks on a Stock Exchange



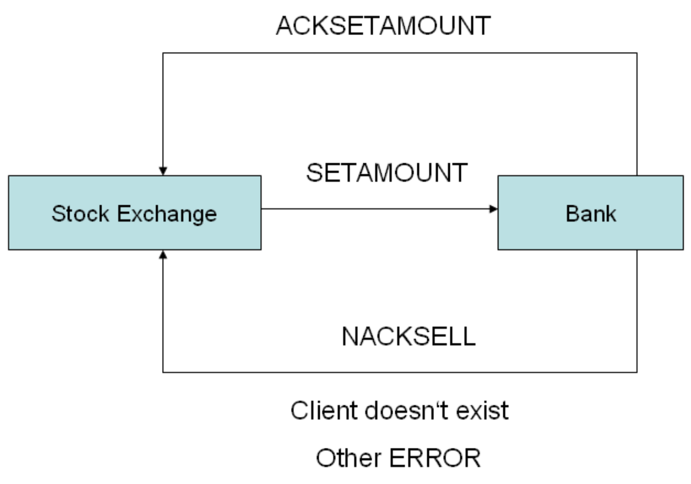
The Client sends the command “SELL” with the arguments “STOCKID”, “QUANTITY” and “BANKIP”. The Stock Exchange replies ACKSELL if it is alright. Otherwise he sends “NACKSELL”. This may be the case if the quantity the client wanted to sell is more then he actually possesses or if the Bank is not reachable.

3.2.9 Get Amount as result of Buy of a client (Stock Exchange ← → Bank)



The Stock Exchange Server sends “GETAMOUNT” to the bank to retrieve some money from the Bank account of one of the clients. This is the follow up to the Client – Stock Exchange’s Buy for example. The Bank replies ACKGETAMOUNT if it is alright or NACKGETAMOUNT if it is not. This may be the case if the Client does not exist, if the Client has not enough money on his account or for any other error that may occur.

3.2.10 Set Amount as result of Sell of a client (Stock Exchange ← → Bank)



The Stock Exchange Server sends "SETAMOUNT" to the bank to transfer some money to the Bank account of one of the clients. This is the follow up to the Client – Stock Exchange's Sell for example. The Bank replies ACKGETAMOUNT if it is alright or NACKGETAMOUNT if it is not. This may be the case if the Client does not exist or for any other error that may occure.

3.2.11 Other Commands

For the complete list of all the commands and their behaviour see the command list.

4 Features

4.1 Client

The Client is able to handle several connections to banks. For each connection a new Bank sheet is created left-handed (View Maybe-Look-Like).

On the right side of the program the stock exchange (only one) connection is managed; where the client is able to view his own stocks and the stocks of the other clients on that stock exchange. He is also able there to buy and sell stocks.

The client should be able to view his amount on a bank. That amount will be updated dynamically by a Timer with an adjustable Delay given by the client.

4.2 Bank Server

The Bank server can get many connections from the clients as well as from the Stock Exchange Server. He shows the incoming connections and protocols in a list box on the left side of the program. On the right side he shows the accounts of the clients of the bank that are offline.

The Bank Server starts by choosing the port and pushing the button "Start Listening". This button then changes to "Stop Listening" which would stop the Bank Server from running. The Checkbox labelled "Show inactive Users" if checked shows the list of the inactive clients on the right side of the program but doesn't refresh until unchecked and rechecked again.

In the Options dialog the Update Amounts timer's delay should be editable as well as the salary which will be added to each active account.

4.3 Stock Exchange Server

The Stock Exchange server is able to get many connections from clients and is able to create a connection for every buy/sell request that came from a client. He then creates an object which handles the requested operation and creates his own connection to the bank.

The graphical interface of the program shows us on the left side the incoming connections from the clients with the requests. And on the right side we see from top to bottom:

The list of the stocks with the quantity that is available

The list of the clients which by pushing the button show pop up to show the entire portfolio of this client

The Stock Exchange Server starts by choosing the port and pushing the Button “Start Listening”. This button then changes to “Stop Listening” which would stop the Stock Exchange Server from running.

4.4 Additional Features (Optional)

Found Transfer between accounts on a bank will be implemented.

The Found Transfer between accounts on different banks may be implemented depending on the time left.

5 Questions

1) The "User File" structure exists in two incarnations. Actually the Stock Exchange should not know the amount of money, but it should ask it from the bank. All the other fields of this structure should be synchronized in some way.

   The Userfile from the stockexchange is different to the userfile in the bank. I missed to write the difference in the documentation. Well no amount information is stored in the Userfile in the stockexchange. (Stock Exchange must connect to the corresponding bank to get the user's amount)

2) The fact that there might be several banks should be taken into account when you consider corersponding protocols: Client to Bank and so on. You have to add the parameter <Bank ID> or <Bank IP> to all the related functions.

   Client to Bank protocol and so on got the precondition that he is already connected to a bank. So the BankIP is already accessible by the Server. (Connections build at runtime Host Property)

3) If you distinguish clients by IP anywhere (in the Stock Exchange, Bank, etc.) then you should deny the parallel running of two clients on one host. Otherwise you should prevent all the client entities from cheating and so on.

   The client is able to run and connect with two apps on a machine but just on one account. So he didn't get more money than another client with only one application. His account can't be set to active twice. (Active or not active)

4) A user should not be able to login from two computers in parallel.

The fact that the user is authenticated by the IP deny's the login from two computers in parallel.