Object Oriented Systems – Assignment 1

# Task

The task is to create a banking system using Java. The banking system should allow one or more bank accounts to be created for the same customer. These bank accounts can be different types such as a current account or a savings account. The bank accounts should be uniquely identified using an account number that consists of eight digits 0-9 and should be formatted as “XXXXXXXX” and a sort code that consists of six digits 0-9 and should be formatted as “XX-XX-XX”.

## Functionality required

* The bank accounts should be able to be deposited into and withdrawn from.
* The user should be able to check the balances of the bank accounts.
* Overdrafts should be enabled for certain types of bank accounts.
* Interest rates should be set for certain types of bank accounts.

The bank system should be able to do the following account operations:

* Ability to browse bank accounts using a menu system
* Deposit and withdraw money from bank accounts
* Set overdraft limits on current accounts
* Apply a default bank charge when a current account exceeds the overdraft limit
* Apply an interest rate on a savings account
* Generate a statement of account transactions that can be saved to a file

## Overall Key Requirements

* The bank system should work on a client and server basis where the user will use their client to talk to the bank server to do any of the functionality in the [Functionality Required](#_Functionality_required) and any functionality stated elsewhere in this document.
* The bank system should not be limited to one server.
* The bank system should have the ability to connect to other banks that use this banking system so that the connector may perform transfers.

## Server/Client requirements

* Every client must run on its own thread regardless weather it is a server client or a connector client.

# Approach

Approach goes here…

# The Protocol

The server is run on the BTP(Bank Transmission Protocol) all communication between the client and the server use this protocol.

## Notes

All strings sent over the BTP are terminated using a new line character and a null terminator.

## Connecting to the account server

When a client connects to an account server it will send one MD5 hash of all the IP addresses of the account servers for its bank that it is aware of. The server will respond with a byte which contains a response code. If the response code is 0 then the client knows that it is up to date with the server IP addresses. If the response code is 1 then the client will receive a list of account server IP addresses from the server. The MD5 hash and all the IP addresses are sent in string form. The MD5 hash is created based on the same format as the IP addresses are sent.

## Authentication Process

The authentication process begins immediately after [section 3.2](#_Connecting_to_the). The client will first send 1 byte to the server this byte specifies the authentication type. Depending on the authentication type will depend on the actions that a client may do.

Below is a table of all the four possible authentication types.

|  |  |  |
| --- | --- | --- |
| **Value** | **Type** | **Access** |
| Decimal 0 | Customer Login | All bank accounts for the customer in question |
| Decimal 1 | Employee Login | All bank accounts in the entire database |
| Decimal 2 | Transfer Login | Transferring money into a bank account |
| Decimal 3 | Administrator Login | Access to all employee logs and has full access to employees and the account servers. |

The authentication type that is selected will determine how the client must authenticate its self.

If the client selected the customer login authentication type then the client will send a string which will represent the account number and it will then send another string that will represent the password. The server will then respond with a byte with the value 1 if the login failed or with the value 0 if the login was successful. If the login was successful the client can then query the server with the customer operation codes.

If the client selected the employee login authentication type then the client will send a string which will represent the username and it will then send another string that will represent the password. The server will then respond with a byte with the value 1 if the login failed or with the value 0 if the login was successful. If the login was successful the client can then query the server with the employee operation codes.

For both the customer login and employee login if the login fails then the client will connect to the next available account server that it is aware of and so on until the client has attempted to connect to all the account servers and failed to login to all of them or the client manages to login to a server successfully.

On a successful login to an account server a cache of the IP address and account number will be stored to speed up login attempts in the future.

If the client selected the transfer login authentication type then the client will be expected to send the sort code of its bank as a string and then the client is required to send its authentication key. The server will then respond with a byte with the value 1 if the login failed or with the value 0 if the login was successful. If the login was successful the client can then perform transfers and other operations in the transfer operation codes. This type of login is used by account servers connecting to other account servers to perform money transfers. See [how transfers work](#_How_transfers_work).

If the login authentication type is the administrator login type then the client will send a string which will represent the username and will then send a string which will represent the password. The server will then respond with a byte with the value 1 if the login failed or with the value 0 if the login was successful. If the login was successful the client can then query the server with the administrator operation codes.

## Operation Codes

The operation codes are commands that can be used in the command server.

### Customer Operation Codes

|  |  |  |
| --- | --- | --- |
| **Operation Code** | **Description** | **Detail** |
| 0 | Pings the server to tell it that it’s still here |  |
| 1 | Request transfer |  |
| 2 | Check balance |  |
| 3 | Get transactions |  |
| 4 | Get bank accounts |  |

### Employee Operation Codes

|  |  |  |
| --- | --- | --- |
| **Operation Code** | **Description** | **Detail** |
| 0 | Pings the server to tell it that it’s still here |  |
| 5 | Request transfer from bank account to bank account. | - |
| 6 | Check balance of a bank account | - |
| 7 | Get transactions of a bank account | - |
| 8 | Get bank accounts of a customer | - |
| 9 | Set an overdraft limit on a bank account | - |
| 10 | Apply an overdrawn charge on a bank account. That will charge them daily while they are over drawn. | - |
| 11 | Apply an interest rate on a savings account | - |
| 12 | Create a new customer | - |
| 13 | Create a new bank account for a customer | - |

### Transfer Operation Codes

|  |  |  |
| --- | --- | --- |
| **Operation Code** | **Description** | **Detail** |
| 0 | Pings the server to tell it that it’s still here |  |
| 1 | Transfers money into an account |  |

### Administrator Operation Codes

|  |  |  |
| --- | --- | --- |
| **Operation Code** | **Description** | **Detail** |
| 0 | Pings the server to tell it that it’s still here |  |
| 5 | Request transfer from bank account to bank account. |  |
| 6 | Check balance of a bank account |  |
| 7 | Get transactions of a bank account |  |
| 8 | Get bank accounts of a customer |  |
| 9 | Set an overdraft limit on a bank account |  |
| 10 | Apply an overdrawn charge on a bank account. That will charge them daily while they are over drawn. |  |
| 11 | Apply an interest rate on a savings account |  |
| 12 | Create a new customer |  |
| 13 | Create a new bank account for a customer |  |
| 14 | Get employee logs |  |
| 15 | Lock employee account |  |
| 16 | Modify employee account |  |
| 17 | Create employee account |  |

# How transfers work

## Customer Account money transfers

The client will first send opcode 1 as can be seen in the [customer operation codes table](#_Customer_Operation_Codes) this is to request a money transfer. The client will then send the sort code of the bank account that’s receiving the money. The client will then send the account number of the receiving bank account.

If the sort code received is that of its own bank then the server will check its own database for the account in question if it cannot find the account then it will search the other servers of its own bank to see if it can find the account in question. If it can find the bank account then the funds are removed from the customer’s account and deposited into the account of the receiver. This is all done on the servers of the same bank as the sort code of both the sender and receiver are the same meaning both the sender and receiver have their accounts in the same bank.

See [External Account Money Transfers](#_External_Account_Money) for information on transferring money to a bank account in a different bank.

## Employee and Administrator Money Transfers

The client will first send opcode 5 to initiate a bank account to bank account transfer. The client will send the sort code and account number of the bank account to take money from. These two entities are sent as strings separately. The client will then send the sort code and account number of the bank account to transfer the money into. These two entities are also sent as strings separately. Finally the client will send a float to the server specifying the amount to transfer. If the sort code of the bank account to take money from is not that of its own bank then the server will respond with a BANK\_MUST\_BE\_LOCAL response to signify that it’s illegal to transfer money from another bank that is not ours. If the sort code of the bank account to take money from is of our bank and the sort code of the bank account to send the money to is of our bank then the server will take the funds from the bank account sending the money and put it in the bank account of the receiver. This can be either in the local database of the server or another server in the bank or both as the bank account to take money from could be on server 1 and the bank account to deposit into could be on server 2. If the bank account to transfer money to does not have the sort code of our bank then it is an external bank so the server will check its trusted banks list and if it can find the IP address of the bank that matches the sort code it will then connect and authenticate as a transfer and then attempt to send the money across. Please see [External Account Money Transfers](#_External_Account_Money) for more information. If the transfer was successful regardless of how the transfer was made the server will return a TRANSFER\_COMPLETE response. If the transfer failed a TRANSFER\_FAILED response is returned. No extra information is given for security reasons.

## External Account Money Transfers

If the receivers sort code is not of the bank of the client then the server will check its list of trusted banks and if it finds a match for that sort code then the server will connect and authenticate with the other banks server using the [transfer login](#_Authentication_Process) authentication type. The server will then send opcode 1 to initiate and a transfer. The server will then send the account number in question and the amount to transfer. The other bank server will then respond with a TRANSFER\_COMPLETE response if it was successful or a TRANSFER\_FAILED response if the transfer failed. If the transfer to the bank was successful then the amount transferred is removed from the client’s bank account and a TRANSFER\_COMPLETE response is returned to the client. If the transfer failed, the sort code is not in the trusted bank list, or the connection failed then a TRANSFER\_FAILED response is returned to the client.

