**BTP400 Final Assessment (Winter 2020) [INDIVIDUAL] Due: 11:58 pm, April 12 (Sunday)**

**Khai Phan (100901164)**

1. **Plan of Migration and Design**

* Assignment1 comprises all of the functional method used for Bank and Account class (including GIC, Chequing and Taxable interface), and Lab4 about Thread and TCP socket-based of server/client application. For this Final Assessment, I will need to combine the code in the above work together. To handle multiple clients connect to one server, Thread must implement to have multiple clients connect and run concurrently. The server class is running all the time to continuously accepting a client. It is required to have one thread finish working on the task before having another thread updating it (Thread 1 update the balance before Thread 2 can continue to perform if Thread 1 executed before Thread 2). By using synchronized in the method used for Server (Client Handler) as well as other essential methods such as Deposit and Withdraw in Account object (including GIC and Chequing), I can help to control the data on both sides.
* Once the Thread established, the Server sends the pre-defined Bank account to Client Handler and starts receiving requests from Client sequentially. The Client and Server transfer data about the Account object and Bank object regularly once an operation completed to ensure the data stay update. Both Client and Server class use ObjectInputStream and ObjectOutputStream to send data and object to each other.

1. **Implementation**

* All classes have been placed in the appropriated Package.

**+** com.client contain Client.java

**+** com.seneca.accounts contains Account.java, Chequing.java, GIC.java, Taxable.java

**+** com.seneca.business contains Bank.java

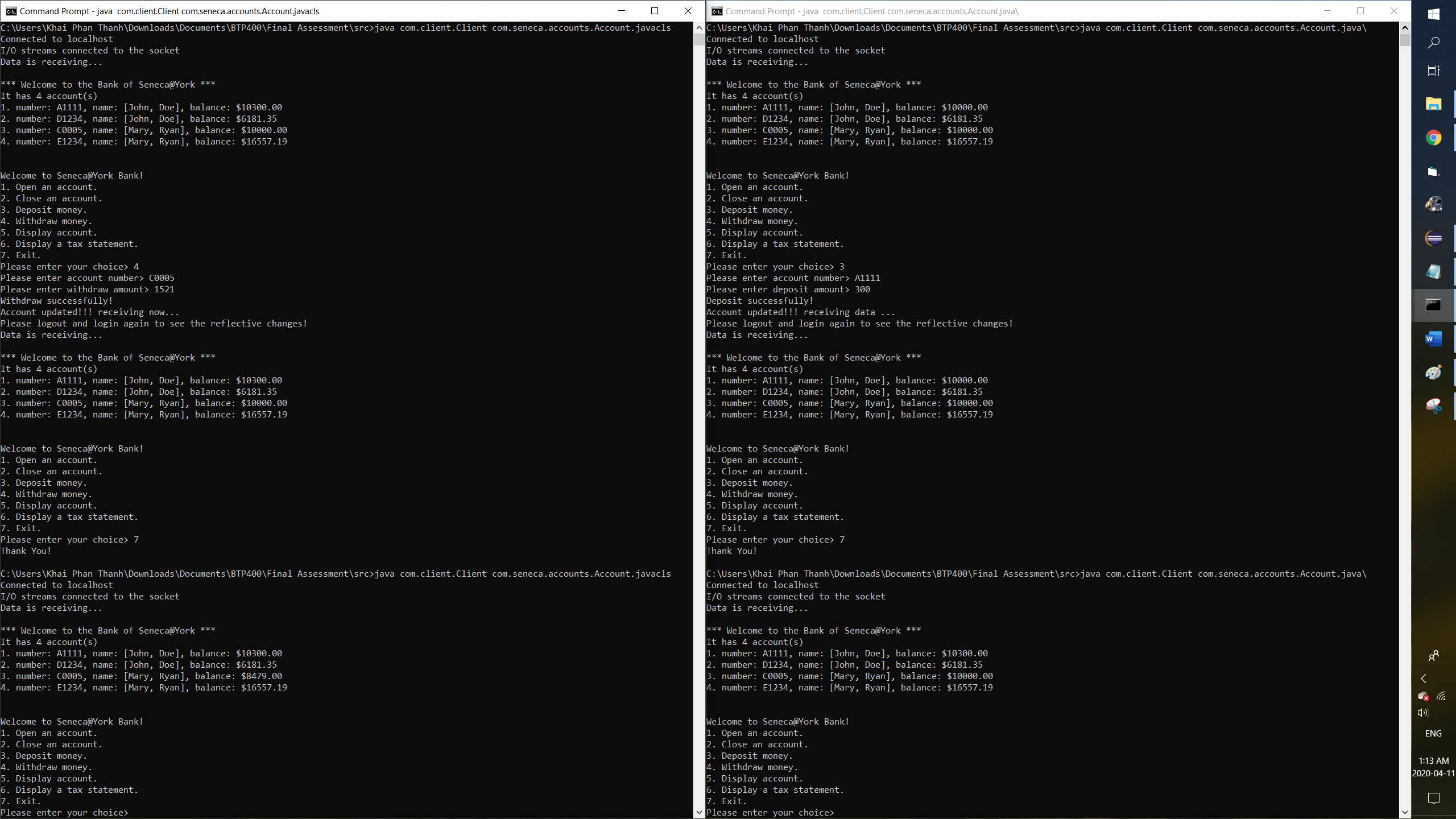
**+** com.server contain Server.java

1. **Testing**
2. Use port number 5678 and localhost for testing purpose. **(Completed)**
3. Use three command prompt windows to run two clients and the bank server. In the bank server window, there should be echo output whenever the server has processed a client request. **(Use Socket to display Socket information to distinguish which one is doing the operation)**
4. **Reason for design choice**

* In real world Client Server situation, almost most of the time, the Client will act as a front-end site where it takes all of the argument, instruction and anything that user is looking to perform and transfer that to Server, then the Server will do the operation and return what it has done to the Client for display.
* For my program Design Choice, all of the operation such as Create an Account, Remove an Account, Display account with information, Display Tax Information was generated from the Server side and send the object to Client side for display when it’s completed. This will ensure that Client-side will only receive data from user and user cannot interfere with the Server and the Server will only perform operation when Client-side send the data to. The reason I didn’t include the Deposit and Withdraw to the list will be explain on the next section.

1. **BUG**

* At the time I submit the document, I am experiencing a bug happen only at the Deposit and Withdraw method. For the rest of the program, the data has been transferred successfully and correctly at all times. With the Deposit and Withdraw, when the Client pass the amount required to deposit/withdraw, the Server will operate and then send back the entire object Bank to Client for display. For some reason, the Server was doing the process correctly, but when the Client receives the Bank object, nothing has been changed, and the changes will appear if the Client logout from the program and re-login to see it (noted as OPTION #2 in the program).
* To fix the issue, I have an alternative solution (noted as OPTION #1 in the program) is to have the Client side to receive the account number as well as the deposit amount, pass the parameter to the Server to check if the account is exist then the amount for validation. The Server will confirm back with a boolean and once Client receives it, it will do the Deposit/Withdraw operation (if boolean is true) and send back the entire Bank object back to server for update and result matching.
* Because of the high security issue with OPTION #1, I have decided to proceed with OPTION #2. It is, inconvenience but because of security issue, this has to be apply.



1. **Intructions for Screenshot**

* Since there might be a different order while I was doing the operation for both client (to see that it’s actually update on the server and has been updated on the other client before they can do any other operation), there is a Screenshot explaintion on the readme file