Fall 2022, Quiz 5

You may have open and refer to any of the add protocol examples that we looked at in class, or any of your code that you have written for the project. You should not refer to any other resource as you complete the quiz. I will not have good ways to know if you only use these resources, I am trusting that you will follow these instructions.

This quiz is due at midnight on Friday, November 11. YOU MAY NOT TURN THIS QUIZ IN LATE. TURN IN WHATEVER YOU HAVE. TALK TO ME BEFORE HAND IF YOU HAVE A REASON THAT YOU NEED AN EXTENSION.

Posted on Blackboard are 2 NetBeans projects to be used with this quiz. One project is a quiz server. The server manages a set of bank accounts and listens for ATM connections to deposit, withdraw, and check the balance of an account. You should not need to make any changes to this code, however you will want to look at the code to see what accounts are already created including each accounts account number and pin. You will only need to run the server so that you can test your client code. The second project is a quiz client. This client builds a GUI that has fields for inputting an account number, a pin and an amount and buttons for withdrawing, depositing and checking the balance of an account. The GUI project also has a CommInfo class that contains information needed to connect to the server (ip address and port number). An instance of CommInfo is created in the constructor method of the GUI and this object has methods to retrieve the ip address and port number of the server.

Your task in this quiz is to implement the event handler for the WITHDRAW button. The GUI builder was used to create the GUI and the event handler method has already been created and is named withdrawHandler. Your code for that method needs to open a socket connection and follow the protocol given below. **Submit the Java file on Blackboard when you are finished (just the Java file that has the method you wrote – not the entire project).**

PROTOCOL

GUI sends: WITHDRAW or DEPOSIT or BALANCE (in all caps, server responses will be in all caps)
SERVER responds: WHO or BAD COMMAND (The server will close the socket and stop sending an receiving if it sends the message BAD COMMAND)
GUI sends: num (num is an account number in the textfield of the GUI)
SERVER responds: VALID or INVALID (if the account does not exist it responds with invalid and stops the protocol)
GUI sends: num (num is the pin number for the account in the textfield of the GUI)
SERVER reponds: VALID or INVALID (if the pin does not match what is stored for that account it responds with invalid and stops the protocol)
GUI sends: amount (amount comes from the textfield in the GUI)
SERVER responds: SUFFICIENT or INSUFFICIENT
If the server responds with sufficient, it immediately sends another line which is the new balance of the account.
If the server responds with insufficient because there were not enough funds to cover the withdrawal it stops the protocol (A withdraw command is the only time the server would send insufficient)
GUI sends: COMPLETE (only if the previous message was SUFFICIENT with a new balance)
SERVER responds: SAYANORA

The GUI should display an appropriate message with a JOptionPane message box giving the result of the transaction and should update the Balance text field if the transaction was successfully completed. Remember that you are only writing the client portion of this protocol.

For full credit your code must recognize when it gets BAD COMMAND or INVALID and give the user an appropriate message and stop the protocol.