**ITCS 3166**

**Midterm Project**

For this project, you’re going to create a client-server connection using Java code that we’ve practiced in the Activities this semester. This program is a very simple server-based program that has many uses in the modern age, such as online banking, online gaming, school registries, and many more. For this project, we’re going to create an application for online banking.

The idea here is simple. A user should be able to view their balance and add/withdraw money from their account through a server. More standard-use versions of this kind of application would typically have some SQL integration as well, but we’re going to simplify the concept. This kind of application is used by most modern banks today and helps control the number of users for each bank without having to do banking in person.

This is how it’s going to work. A user will log on to the online bank (which we will assume is done prior to the start of the application). For the purposes of this assignment, we’re going to use ‘localhost’ for the IP. You may use any PORT number you wish as long as it’s not reserved. When the application starts running, the user is logged in and given a few options. The user may then select from these options to do one of the following:

* View Current Balance
* Deposit Money
* Withdraw Money
* Exit

For the purposes of this assignment, we’re going to assume that the user has an unlimited amount of money for deposit purposes. So the options are as follows: the user can select to view their current balance; they can choose to deposit money – and choose how much to deposit; they can choose to withdraw money – and choose how much to withdraw; or they can choose to exit the program. **This program should loop though the options until the user selects the option to exit.** Upon doing so, the application should end and let the user know that they have successfully logged out.

There are a few guidelines that you must follow for this program. When you have finished creating the program, make sure to check back over these guidelines to make sure that everything has been accounted for.

The guidelines are as follows:

* The program must be a console-based application. No GUIs allowed.
* The program must loop through the options until the user selects to exit.
* You may get user input any way you choose – however, you must catch invalid entries. This means that if your options are 1-4, then a user entry ‘F’ should be caught and give an invalid entry feedback, then allow the user to reenter their choice.
* Your program must account for the amount of money in the user’s online balance. If a user chooses to withdraw $100 and their balance is $50, then an error message should show and the money should not be withdrawn.
* User-friendliness is important! Make sure that your program looks nice and easy to use.
* Send regular messages to the server’s console. At the very least, send a message when the server connects, when input is received from the client, and when the server disconnects.
* Make sure that in the natural runtime of your program when the server is disconnected that all sockets, scanners, and any other form of I/O is closed – leaving these open can cause a security leak that, while harmless on a localhost, can be dangerous on other IPs.
* Leave documentation and comments in your code to explain things step-by-step. This is just good practice. It doesn’t have to be a lot; just enough to explain your process in a simple way.
* Code written must be your own. You are not allowed to copy code from any source, including but not limited to other students, textbooks, websites, etc.

**Grading Rubric**

|  |  |  |
| --- | --- | --- |
| This assignment is worth 30 points. The point values are graded based on this rubric. | | |
| Connectivity | 10 points | The server opens a port on a localhost, and the client successfully connects. |
| User-Input | 5 points | The client program successfully obtains user-input, and data validation is done correctly to validate the input obtained. The user is asked for new input if invalid input is given. |
| User-Friendliness | 4 points | The look-and-feel of the program runs well; the client sends good user-friendly output; the server relays information to the console. |
| App Reliability | 8 points | The program correctly sends the user input to the server when necessary; the server correctly sends information to the user; all information sent and received is correct. |
| Good Practice | 3 points | No dead code; no unused imports or variables, IO and Sockets are correctly closed. Good documentation. |
| Total | 30 points |  |