CS 2316

Homework 9b – GT Brokers

Due: Wednesday July 23rd, Before 11:55pm
Out of 100 points

Files to submit:

1. **HW9b.py**

For Help:
- TA Helpdesk – Schedule posted on class website.
- Email TA's or use T-Square Forums

Notes:

·**Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus).**

· ***Do not wait until the last minute*** *to do this assignment in case you run into* problems.

· **Read the entire specifications document before starting this assignment.**

Premise

This homework is part two of the previous GT Brokers assignment. You will add on to the login functionality that you built to create two more GUI pages that use a database to have the functionality of a simple online brokerage application.

Although this program does not require knowing much new information, it does test SQL query and insertion abilities as well as coding practices to create two more GUI pages that can be opened, closed, or updated at the click of a button. **This new information may take some understanding, so please do not wait until the last minute to begin this assignment.**

Database Format Information

The database for this part of the assignment only concerns a different table of transactions, called BrokerageTransaction:

```
CREATE TABLE BrokerageTransaction
        (TransactionID INTEGER NOT NULL AUTO_INCREMENT,
        User            VARCHAR(20)     NOT NULL,
        Type            TEXT            NOT NULL,
        Stock           TEXT            NOT NULL,
        Quantity        INTEGER         NOT NULL,
        PRIMARY KEY(TransactionID),
        FOREIGN KEY(User) REFERENCES BrokerageUsers(USERNAME),
        CHECK(Type = "Buy" OR Type = "Sell"),
        CHECK(Stock = "GT" OR Stock = "UGA"))
```

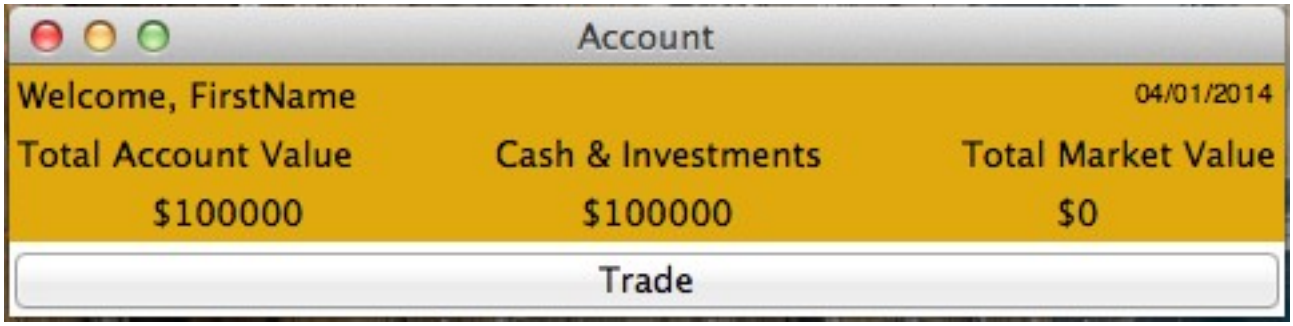
The SQL statement above was used to create the table. As can be seen, USER and the transaction details are all NOT NULL, meaning you have to insert a value for each of the fields. Also the User field must be identical to the person's Username, so the two tables we are using can be easily related to one another.

To be clear we'll take a second to break it down: what is a transaction?
A transaction in GT Brokers is either selling or buying a stock.
The stock choices are: "UGA" or "GT", and the prices will fluctuate with time (just like in the real world!). You will have to scrape these prices from a website every time you run your code, and that website is posted under this assignment on the class webpage. Once you have this price, you can use SQL to record transactions using the User, Type ("Buy" or "Sell"), Stock ("UGA" or "GT"), and Quantity (How many?).

Further Development

You will be writing on top of your already written Homework 9a. **If your old program is not perfect feel free to update it as needed!**

Your new program will create an account summary page for GT Brokerage users, that will be opened right after they click the "Login" button (Instead of closing the program as in the last assignment). An example of this GUI is shown below, and will be described in more detail later:



Method Name: GetPrices

The first method you should focus on in this assignment, before the GUIs, is the one that scrapes the prices from the online GT Brokers stock page. You will be scrapping the price of both GT and UGA from the website linked under this assignment on the class webpage.

Method Name: ComputeStats

The next logical step is to compute all the statistics we need to make the creation of our home page and order page as easy as possible. This will require getting the stock price from the previous method, and using several SQL queries to get information about the user from the database. The base equation we are using is simple, and is as follows:

Total Market Value + Cash = Account Value

It is obvious now how you will calculate account value, but as far as the other two go:

- Cash is simply the dollar amount for the current user in the BrokerageUsers table, in the Dollars column. If the user has never made a transaction then we expect this number to be = $100,000. As the user spends or gains money, **YOU WILL NEED TO UPDATE THIS VALUE.**
- Total Market Value is a little trickier, but is essentially the number of stocks that user has times their **current** market price. That means you will have to find all of their past transactions and use them to calculate their total number of stocks. If the user has, for example, 10 UGA stocks at $10 each, and 5 GT stocks at $20 each, they would have a TMV of (10*10) + (5*20) = $200

At the end of this function you should have these three values computed, and ready to input into your two new GUIs!

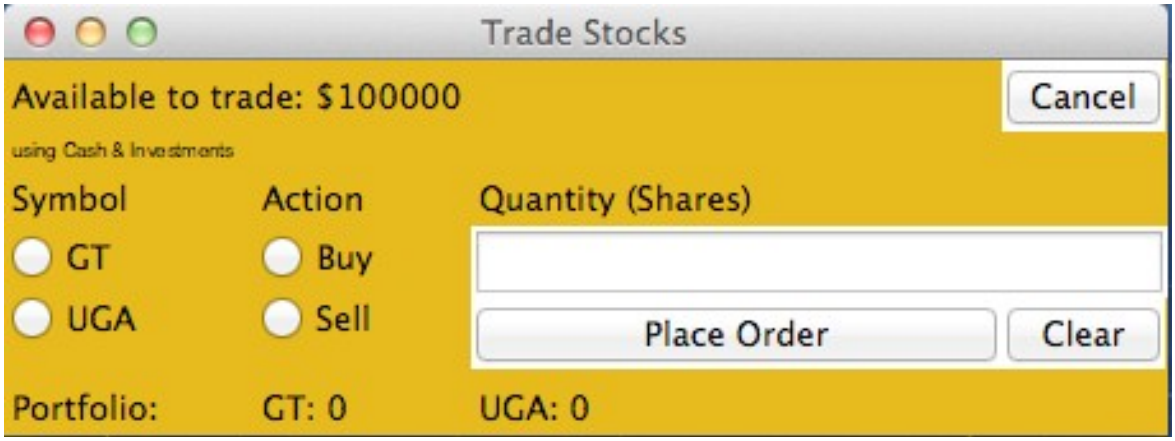
Method Name: AccountPage

This method is called to create a new account summary window. It should be displayed when you successfully login from the previous homework's login page. This method is responsible for arranging and initializing your GUI. You may generate this window however you wish, but it is highly suggested that you use the grid layout manager. Things to note about the GUI you see:

- The Homepage window has the title: "Account"
- The top row welcomes the user by their stored First Name
- The top row also displays the date, using the time module ("import time")
- The information you computed in the computeStats method will be displayed in the next two rows of the GUI, and should be **updated every time this GUI is opened.**
- The trade button is at the bottom, spans the entire window, and will close the Account Page GUI and open an Order Page GUI, described in the next method:

Method Name: OrderPage

This method will create a new window that the user can use to complete the transactions described earlier in the assignment. Before we get into too much detail, here is what the GUI will look like:



Some notes about the GUI:

- The number presented at the top of the page is simply the number you already computed for the users' Cash. Your PlaceOrder method will check to ensure the user doesn't spend more cash than they have on hand!
- As described earlier, there are a total of four options for Gt Broker users, and that is Buying GT Stock, Buying UGA Stock, Selling GT Stock, or Selling UGA Stock. This information is presented in the form of two sets of radio buttons that the user will select.
- The Quantity entry box has a NORMAL state and can be edited by the user, but you will have to ensure that the user entered an integer (You can't buy fractional stocks.)

- The “Clear” button will reset the GUI entirely (including radio buttons). We suggest you make a helper ClearOrder method to implement this.
- The “Place Order” button will call the final method, which will use SQL to update the transactions table, then return the user to the Home Page
- The Portfolio portion at the bottom of the window shows the users' current holdings in GT/UGA stock.
- The cancel button should **close the current window and open the Account window.**

Method Name: AccountToOrder

This method should be called whenever you need to switch from your Account page to your Order page. After switching, the visible page must be displaying the correct information.

Consider creating your windows from your __init__ method, this way you are not creating new toplevel windows every time you switch windows.

Method Name: OrderToAccount

This method should be called whenever you need to switch from your OrderPage to your AccountPage. After switching, the visible page must be displaying the correct information.

Method Name: PlaceOrder

In this final method, called by “Place Order” button on the order screen, you will execute all the SQL necessary to complete the user’s transaction. The first portion of this method should be dedicated to error checking, the main points of which are:

- The user must select a radio button for each of the two categories, if not a message box should appear telling them to please select a button!
- The user must input an integer value into the entry box before they click the “Place Order” button. If the entry box is empty, or contains a non-integer value then a message box should appear telling them so.
- The user **cannot** spend more than they already have, so you’ll have to check the quantity of stock they are buying and it’s price, against the Cash amount they can spend on the stock. If they try to buy too much, politely let them know that they don’t have the capital (via message box).
- The user cannot sell more stock than he/she owns. If they try to then use message box to let them know they can’t.

Now that the error checking is done we can assume that this is a valid transaction, which means we will need to update BOTH of our SQL tables. Firstly, we will input the transaction into BrokerageTransactions. The table is described in

detail earlier in this pdf, but remember that Type is either “Buy” or “Sell” and Stock is either “GT” or “UGA”.

Once the transaction is input into BrokerageTransactions all we have left to do is update the dollar amount in the Dollar part of BrokerageUsers. The exact change will be based on the stock price, whether the user bought or sold, and the quantity of the exchange.

Lastly, upon completing an order the user will be returned to their account summary page with freshly updated variables. From here they could choose to trade again, or simply exit the program.

Grading:

You will earn points as follows for each piece of functionality that works correctly according to the specifications.

Account GUI	**15**	
	GUI has all components with proper characteristics	5
	Trade button works as described	5
	Inputs correct first name	3
	Contains todays date in correct format	2
OrderPage GUI	**20**	
	GUI identical to the one displayed	10
	Place order works as described	5
	Cancel button works appropriately.	5
AccountToOrder	**2**	
	Properly switches windows.	2
OrderToAccount	**2**	
	Properly switches windows.	2
GetPrices()	**6**	
	Correctly finds the prices of each stock	6
ComputeStats()	**25**	
	Correct Cash & Investments	15
	Correct Market Value	10
PlaceOrder()	**30**	
	Correctly identifies if a user has enough cash to buy stock	10
	Correctly identifies if user has enough stock to sell	10
	Accounts for all possible errors	5
	Reopens account summary page after order is placed	5