Final Project

3220 Software Design

Erik Bowers

Nicholas Bouckaert

**Abstract:**

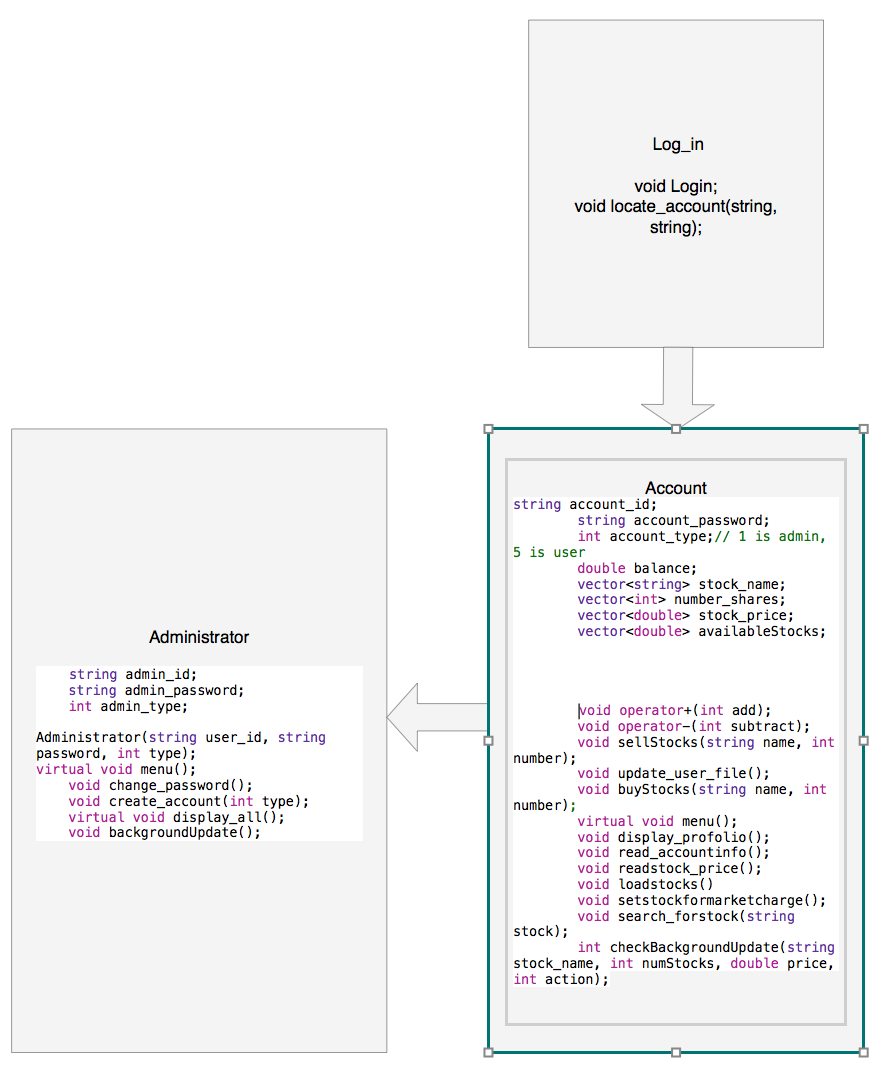
The overall goal for this final project was to implement a stock trading application. With the intention of replicating applications similar to what real financial firms utilize. Where users could login and buy or sell stocks at well from this program. While containing an administrator function that would have ability’s similar to the real world. Along with the prices of the stocks to change in a real-time fashion with the market.

**Introduction:**  The general idea of this project was to produce a stock-trading application similar to the ones used in industry. With two levels of users, one being an account holder or user. The other being an Administrator. In which the User would have access to just their own portfolio with the ability to buy and sell various stocks that they possess. While the admin could see all accounts on the system, change passwords, and push market updates to the system. While also signing the user into from there account and walking them through how to use the application from there end. Every user name, password, and level of account with the company is stored in an accounts file. From there every user that has a portfolio with the company with have a private.txt file that will be accessible once logged in. In which their balance and any stocks they own will be store. Separating all of them in folders was easier for us to code, however, being less efficient than a using data base. Since this class wasn’t databases we thought it was more important to spend our time implementing things we learn along our way in the course then going in and trying to learn databases. Also, that thought was our Professor’s too, which he shared with us after learn submitting our idea of the project.

**Background:**

The basic idea of this program came from a Scottrade, which is a brokerage firm with a branch in St. Louis. Of course, this program is built to nearly the same complexity or security concerns. However, in the generality of things it preforms similar operation to the one Scottrade. We never had the ability to see the administrator and user side of program. So, Eric and I took our best guess at a general idea to what they would exactly include from our perspective. From there we decided what features we wanted in program to have and implemented most of them that we came up with. Even though this is a basic idea of the application most brokerage firms use, it could be implemented across a variety of platforms including IOS, android, and Windows. The idea and goal of the program is shared with the rest of the industry as whole.

**Implementation:**



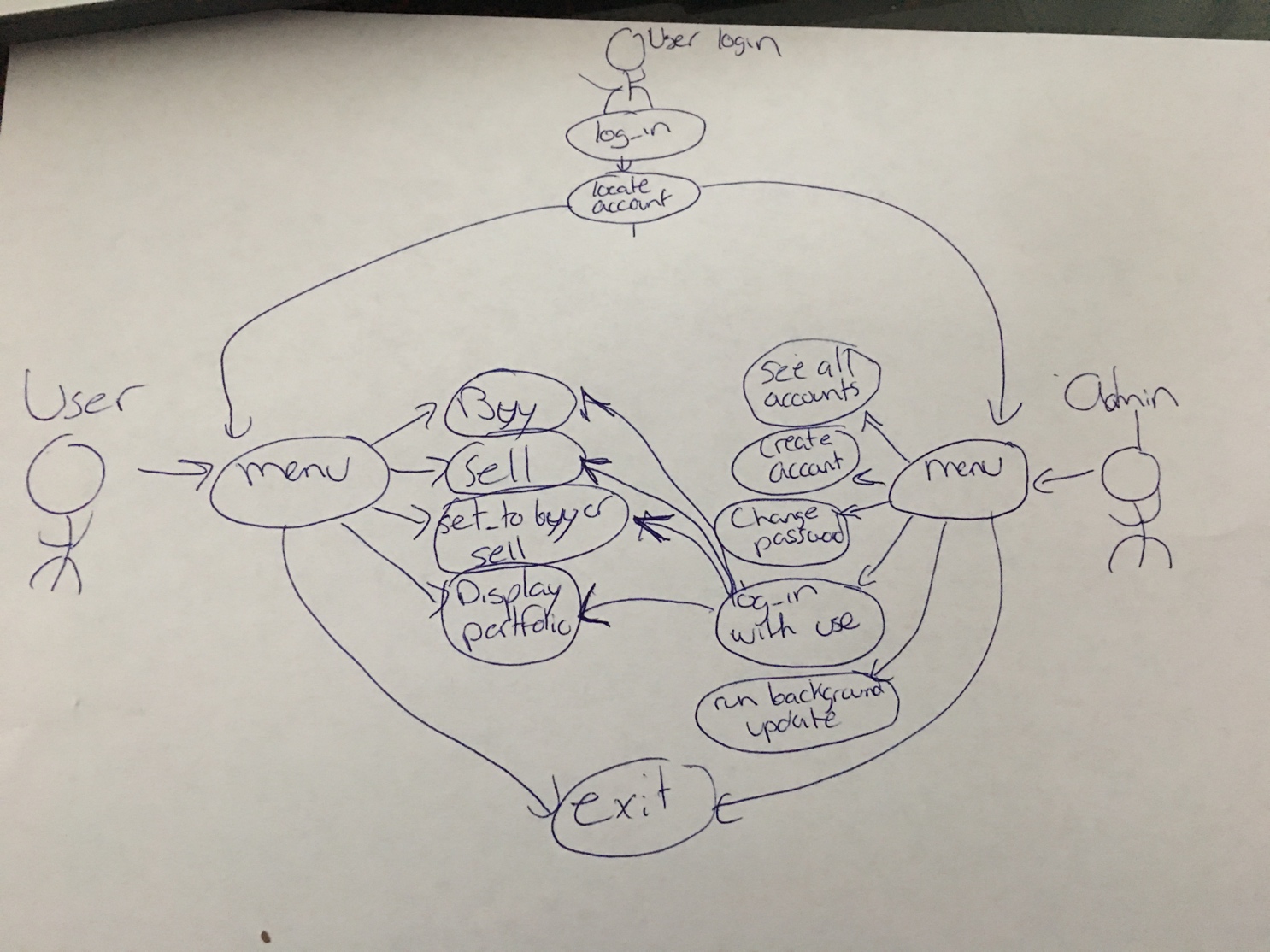
The objective of this project of this project was to create a functional stock-trading application that could be used within a financial firm. This consisted of two levels of user’s administrator and base user that had an account with the company. Administrator would have the ability to inheritance the complete power of the user, while user’s abilities would be limited purely to their individual profile. While, also including the ability to update the stock prices and show the users totally interest within every company in which they owned stocked in. Keep in mind that not every stock is available within this firms. Since the firm would choose a market such as New York Stock Exchange, in which some stocks are not trade in that market.

Upon run the program, the individual logging in gets prompted asking for them to enter a user name and password. Which would be used as a reference to see if that log in was listed in the account.txt file. If it was then the program would use the type (which determined if the user was administrator (1) or user (5)). At this point and object would be created. If the type was 5 then an object of the base class would be created. However, if the type was a 1 then an object of the derived class would be created. The user is never prompted to which account he wants to log in, it is all determined based on what is stored in the accounts file. This leaving the program in control preventing any possible way for the user to try and log into an account level in which they do not have access too.

Once logged in as a User, you get prompted with the options to buy, sell, set a stock for buy or sell, or display portfolio. In which the user can choose any of those options. Before the menu display the entire user, portfolio is load into the object include balance, stocks they own and number of shares in that company, along with the most current market value of those stocks. This the user sees in real time what each of their stocks are worth instead of having to check the market for themselves. Once prompted with the menu, if the user chooses to buy or sell they will be asked which stock there are wanting to unload. During this time, the name type will be compare with the names of stocks on the market or stocks in their account. If the stock exists at the chosen location the user will be prompted asking how many shares they would like to sell or buy. At this point once entered the program will automatically adjust user’s portfolio adding or subtracting from the balance and number of shares accordingly. The user can also set a stock to buy or sell at a certain point in the market preventing them to having to be on consistently. This writes to the set to buy or sell file. Whenever an update is passed the whole function will run. For the last User option, is display portfolio, this allows the user to see their whole portfolio at one time.

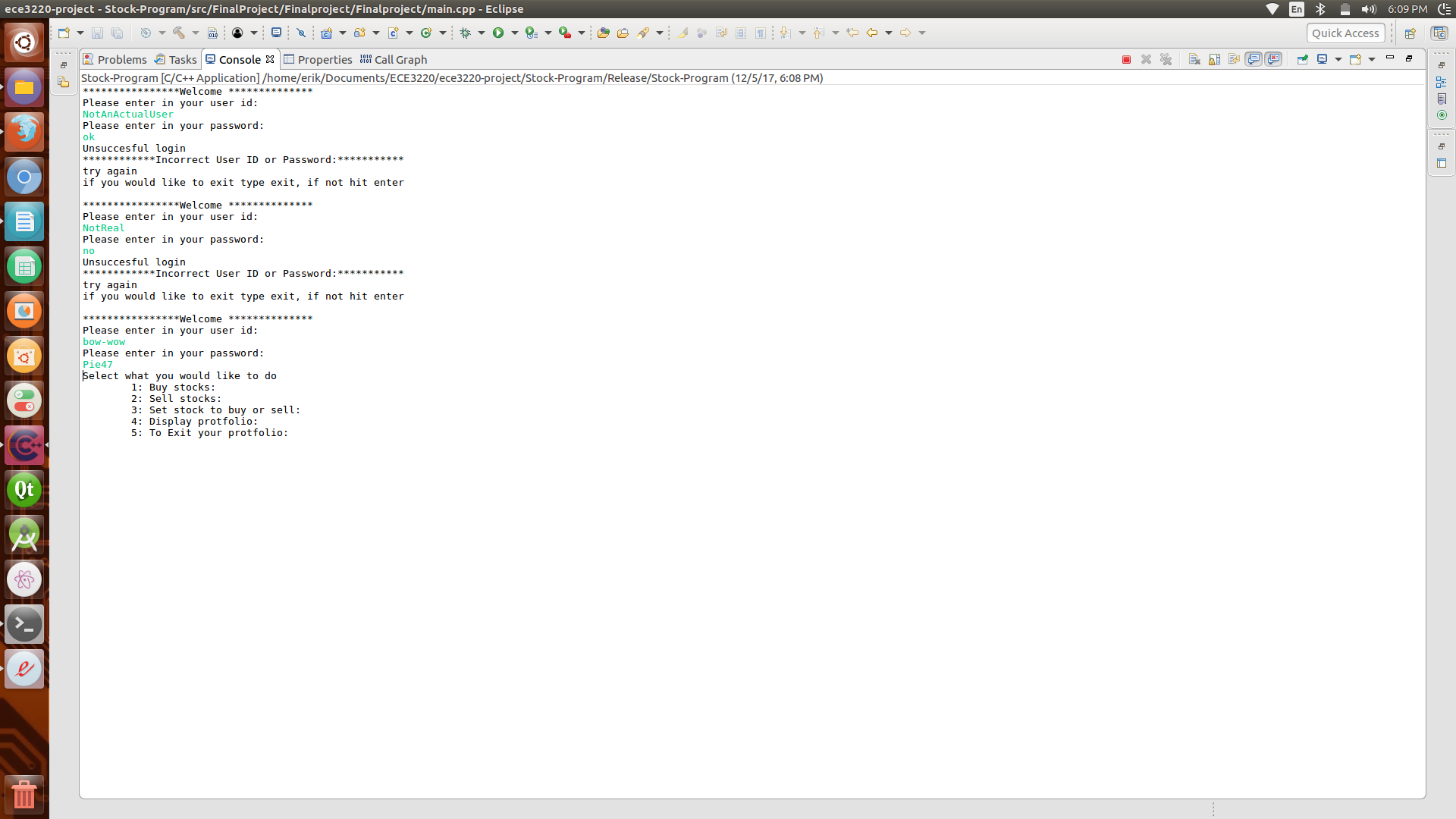
Admin will have inherited all of the functions from the user will having access to functions expanded in its derived class. In which they can see all the accounts the names listed in the accounts file and type of account they have. At this point the passwords are hidden from every administrator for security reason. No one person should ever be able to see the passwords of any users. The second option if for the admin to create and account or new portfolio in which the user will be add to the end of the file. Since, we know all everyone forget there password eventually so we added in the ability for the user to change any passwords they need to through option there. For the fourth option the admin has the ability to log in a user from there account. This still requires the user’s password and user name, and the program ask that the admin insures that the user is during the log in. However risky this it is just a demonstration derived class having access to the base class functions. Of course, this would be implemented differently in the real world. For the final function the admin has the option to ran an update in which the market prices could be changed and the stocks set to be brought or sold will occur if the requirements are meant.

The fifth option on both of the menu’s is for the user to exit their account in which the will be brought back to the log in portion of the program. This part of the program will never end unless someone main kills the program.

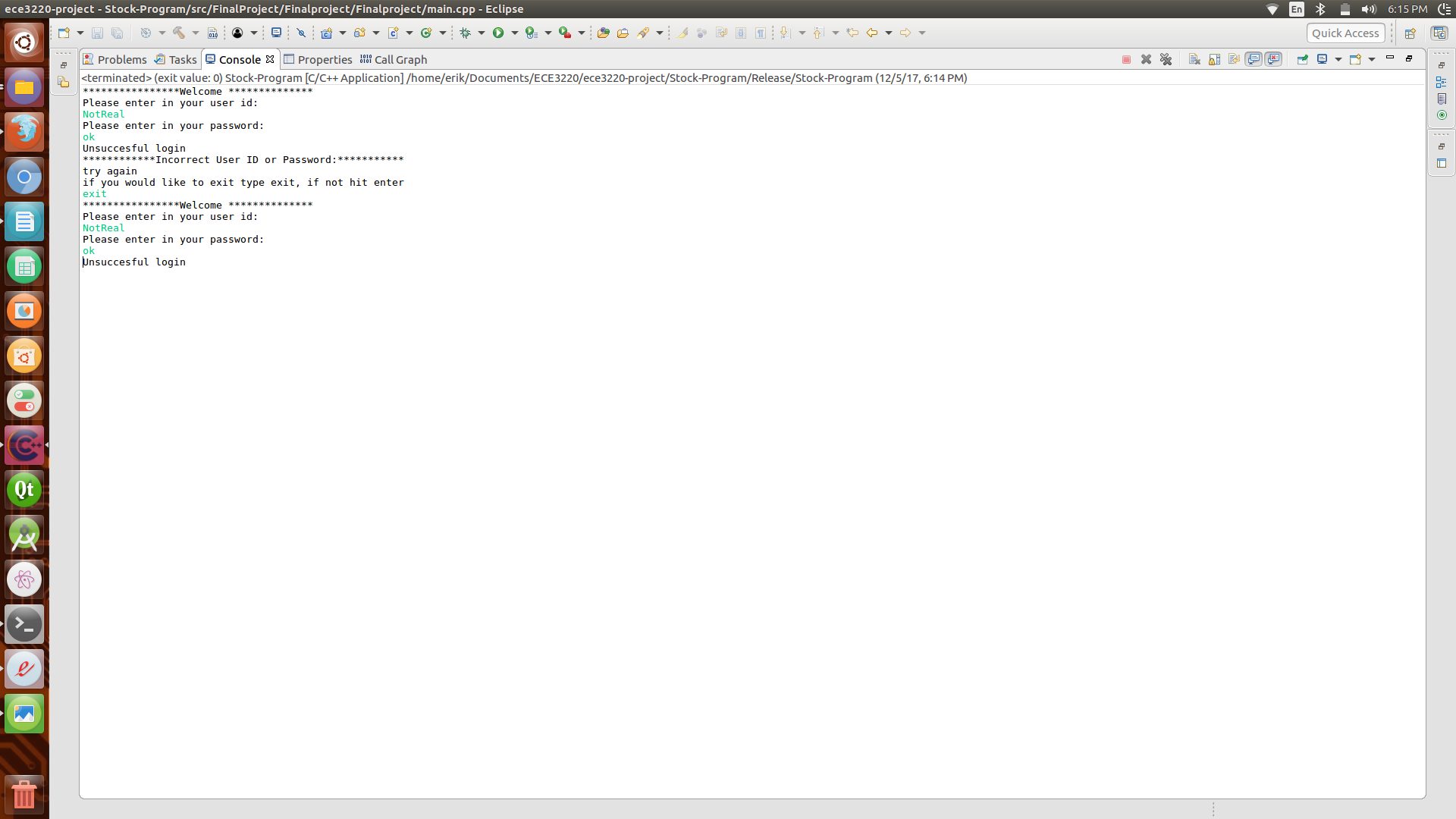


**Experiments and Results:**

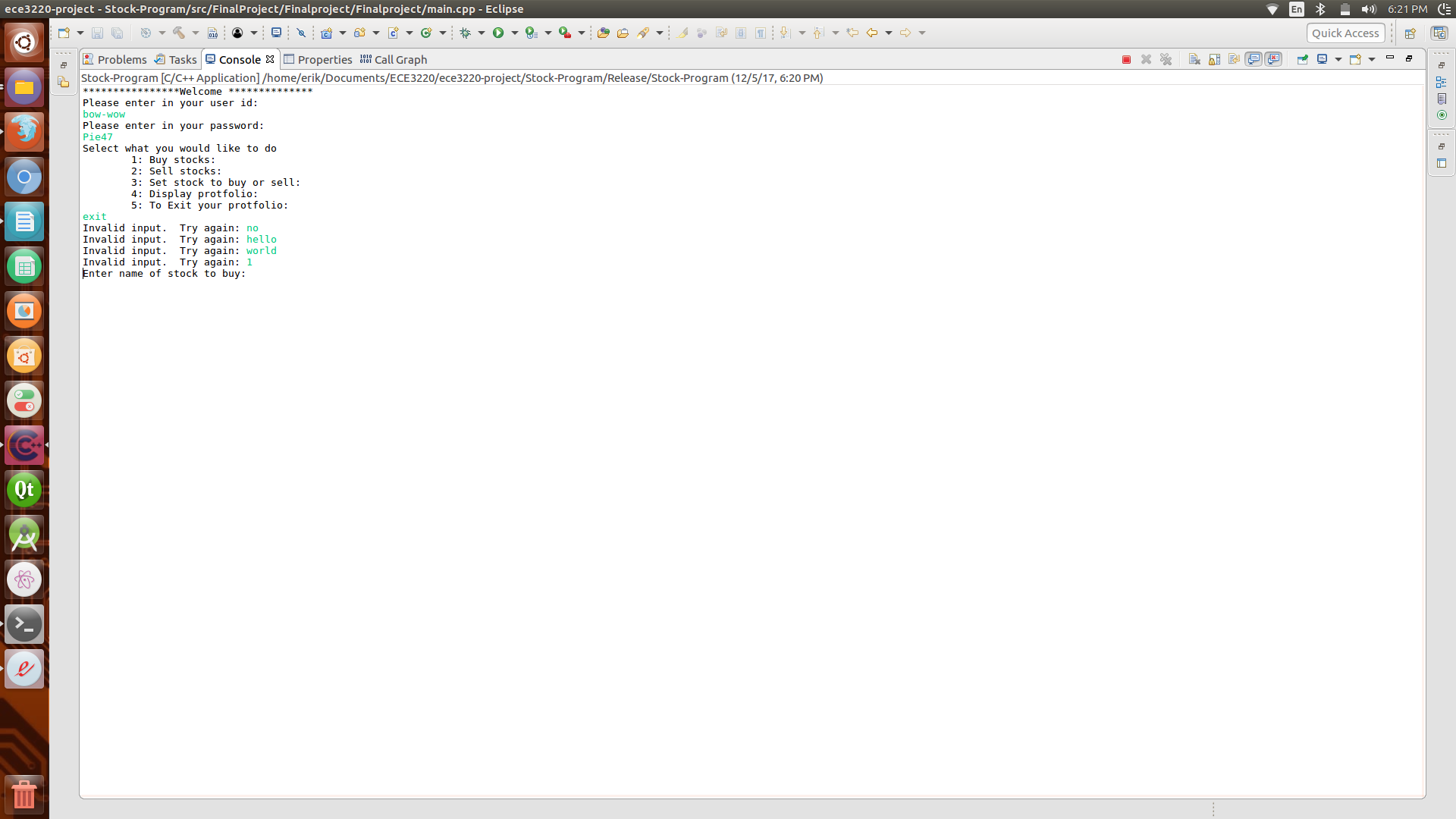
When the program is executed, you are initially asked for your username and password. When you enter a username and password combination that are stored, then you will be logged in. If you don’t, then you will be prompted again for your username and password. This is seen below in figure 1.

  
Figure 1

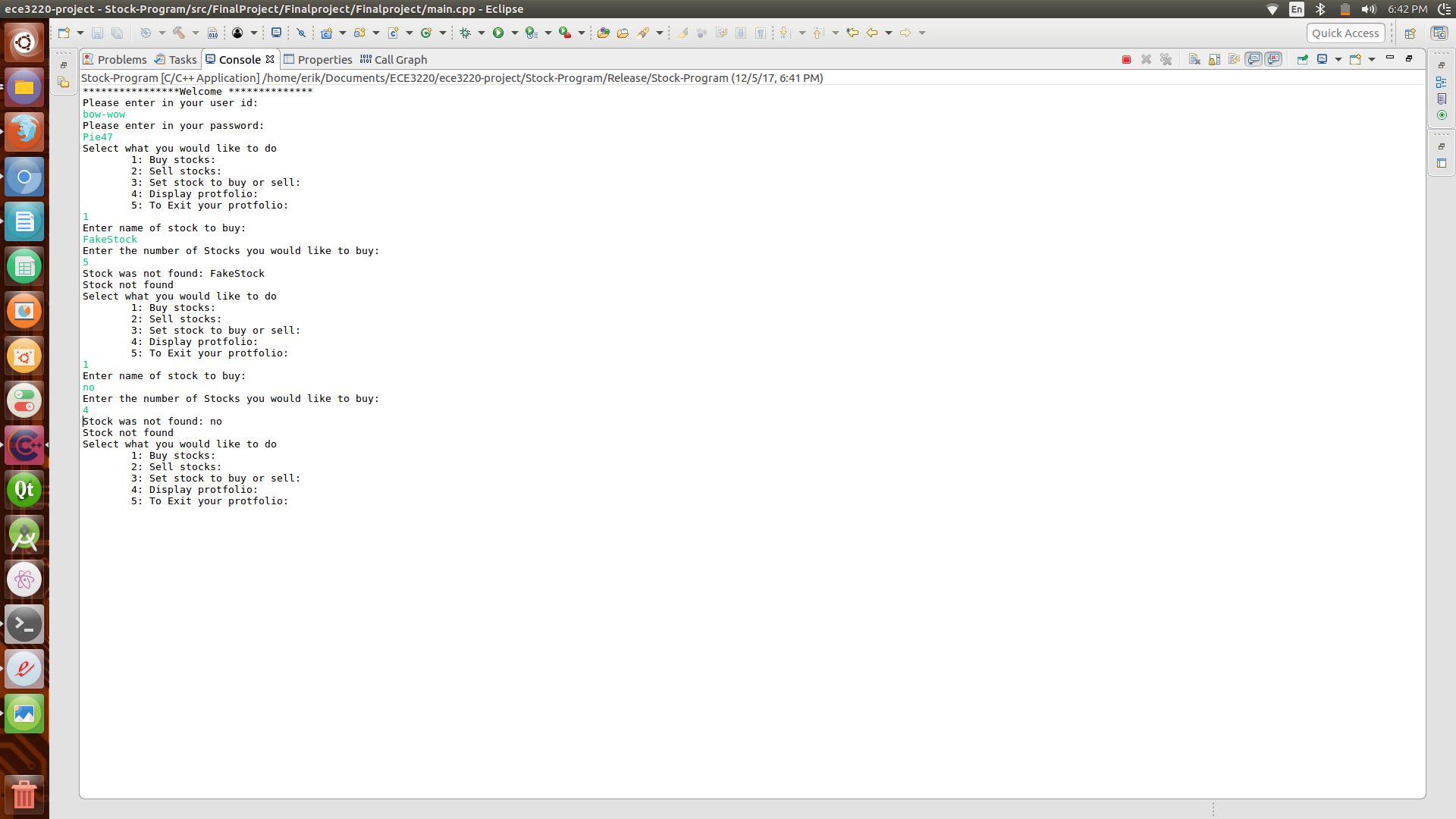
However, when you enter “exit” to exit the login screen, you don’t actually exit until after you enter an incorrect username/password combination one more time. This can be seen in figure 2. But the program does terminate.

  
Figure 2

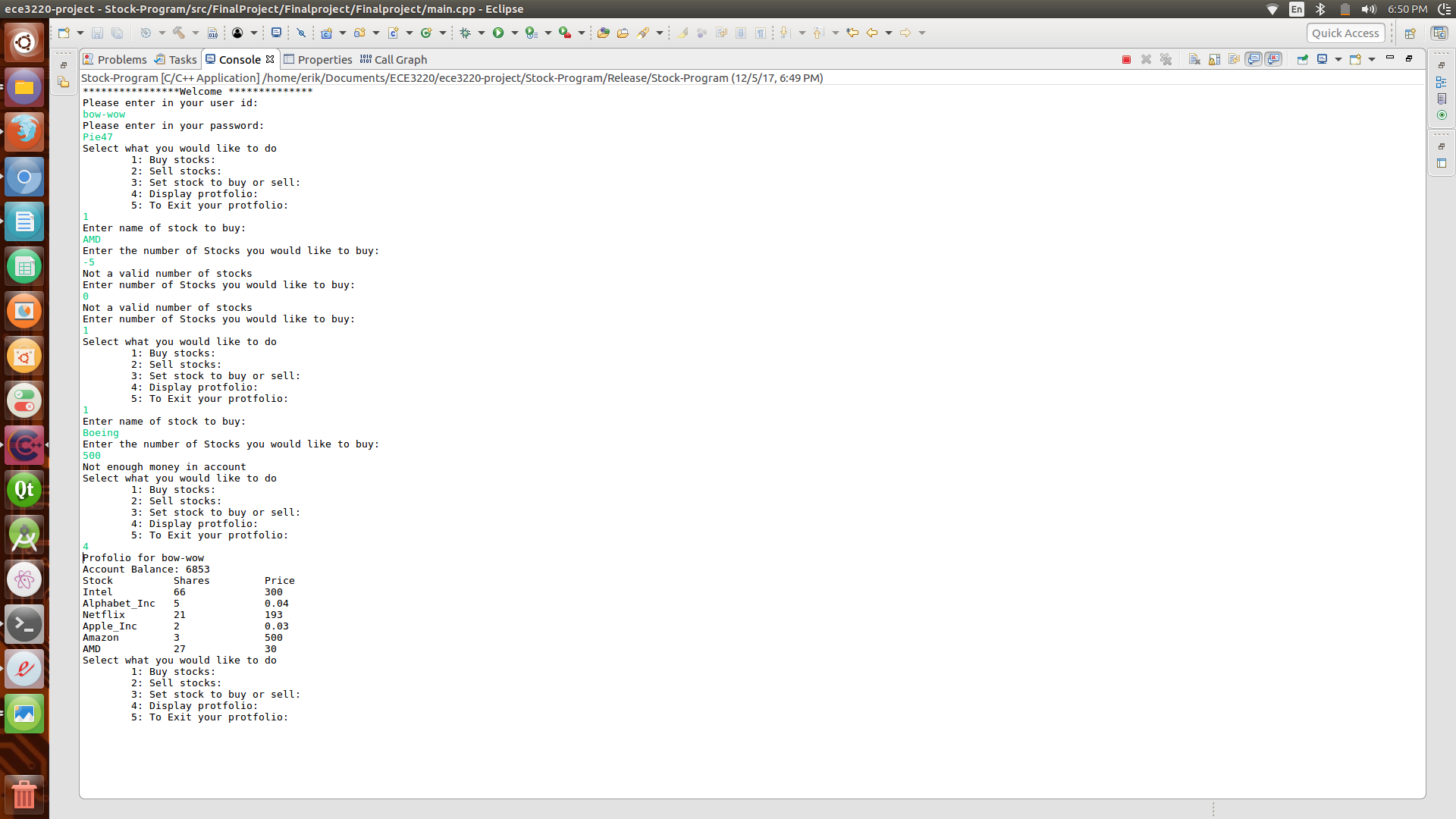
Once we login as a standard user we can select from a list of options what we would like to do. The user types in an integer to select an option from the list. One issue when using cin to take integers is when the user types in a letter or special character the program will enter into an output loop. This was fixed by checking if the input was valid and using cin.clear(). In figure 3 this error checking can be seen. This error checking was done for almost every input that requires and integer.

  
Figure 3

Option 1 gives us the ability to buy stocks. We realized we had to only add stocks that were stored in our database or file in our case. To do this, when the user enters a stock to buy we compare that stock to our file containing the available stocks. If it isn’t found then the user can’t buy that stock. This can be seen in figure 4.

  
Figure 4

We also can’t buy negative stocks. This would essentially sell stocks if we didn’t check for it, but there are some different checks that both functions require and the user shouldn’t be able to sell stocks when they selected the buy function. In addition to this check, we need to ensure the user has enough money to buy the stocks that they have selected. These checks can be seen in figure 5.

  
Figure 5