CIS 152 Final Project Alex Rooney

**I Proposal**

* A local music venue is hosting a pre-show VIP concert and only has select tickets available for the event. Subscribers to the venue email list were sent out a 6 digit access code that will enable users to purchase the VIP tickets when they go on sale. The venue is looking to keep a purchase log of all the people who bought tickets for the VIP event and store that in an organized way, sorted by access code, so they connect with these people for future pre-sales and concerts.
* To solve the VIP issue, I decided on using a stack data structure since each purchase could easily be added to the top of the stack. I also decided on using a stack because it would be easy to export the purchase data into my bubble sort algorithm. I decided on using a bubble sort because in this scenario, there are only 100 VIP tickets available and the sorting population will never be large enough where speed will be an issue. The music venue wants to keep track of purchaser first and last name, street address, city, state, zip code, credit card information to complete the purchase, and the number of tickets (max of 4) that they a person with an access code can buy.

**II Time/Change Logs**

* I didn’t get daily change logs for this final, but I did have weekly progress. I started the project trying to implement a queue data structure at first and wrote my Account and Queue classes. After I created those 2 basic classes I tried to implement a timer function and tried to see how that would work with a queue and re-ordering the queue based on the timer. I quickly came to realize that was a larger ask than I was prepared to finish and so I decided halfway through the project timeline to instead change to a stack data structure as that made more sense for purchases (once you remove the timer aspect). In the last 2 weeks I created the Stack class and my Frame class (which also houses the main method of the program).

**III Lessons Learned**

* My project idea changed once I tried to implement a timer countdown and have that placed within the GUI. A lot of the examples I was checking into were a bit more complicated than I thought they would be, and after thinking through the end goal of the problem (which is to print out a list of purchases) it made a lot more sense to use a stack and a sorting algorithm. After I changed my structure to a stack I didn’t have near as many issues implementing the overall program. I did have to refresh my knowledge on GUI/Swing interfaces and used dropdown boxes within my purchase form which I hadn’t done before.

**IV CODE**

****

**V User Manual**

* Run the program from ‘theFrame.java’ (that is where the main method is located). After the program starts, a purchase window will open up. The user will need to input their information, starting with the 6 digit VIP access code found in their email. If the user doesn’t fill in all of the text boxes with their purchase information the form will not submit, and a prompt will let the user know there is an issue on the form. To export the sorted results either hit the ‘CANCEL’ button or ‘X’ out of the frame in the top right corner. This will print the sorted results.

**VI Summary**

* Modularity is when you write code so it’s re-usable. This means writing classes that can be used throughout various parts of the program. Efficiency means that you’re using the best time and memory when running your program. Having robustness means that your program shouldn’t break, but where you know it might, there needs to be error checking to handle exceptions that may occur. Usability means that the program should be intuitive where it can be and easy to use. Having a GUI helps with this. Your code should be readable and elegant, meaning that the literal source code needs to be organized in a visually pleasing way and should include comments describing certain parts of the program to help the end user.
* My ticket selling program can be used to track user purchasing information and will sort all inputs by access code in ascending order. The program checks for a complete form before being able to submit to the stack and uses a bubble sort to display the stack results after all the users have input their information.
* I would like to add a few enhancements in the future. I want to figure out how to display a timer countdown on the frame when the program starts, and to restart that timer every time a purchase is pushed to the stack.