Michael Campos April 22, 2019

Prof. Chen CSE218-TR

Approaching a Historical Stock Data Analysis Program

Approaching a problem can take more thought than you might think. Finding the algorithm to solve the problem can be easy, but when they are requirements needed to start the program is another story, especially when it involves learning and thinking. When I tackled this project, I had to learn how to use an API (I don’t know what API means), designing the user interface to make it look nice and clean, and choosing the right data structure for storing millions of stock data. Without learning these elements, I wouldn’t have been able to start programming this project.

Learning an API (Application Programming Interface) I was unfamiliar with such as Alpha Vantage was like an adventure for me. First, I had to go on website, obtain a key for myself and start using the key to get the stock data for Apple and Amazon. The thing is, I didn’t know where to use the key. So, I had read the documentation. I had to read lines of English words instead of lines of code. Fortunately, I learned how to the API since it was based off using a search query function and syntax was simple. Also, in the documentation there were examples of using the search query. So, it was a fun little adventure while it lasted.

Designing the user interface for the historical stock data analysis program was troublesome. It was troublesome because every time I’m designing, I’ll make a lot of drafts in my head or on paper (usually the head which is a bad idea since humans are bad at visualizing). Where I want to put this text field, this label, and when a user clicks this button or this menu. How I want the controls to be laid out. How will the view change in front of the user when he or she clicks a buttons or changes something? I guess you can say that these are just preferences, but still. I want the user to be able to access these controls easily so the user doesn’t have to move his/her mouse everywhere around the screen to get somewhere. Eventually, I’ll be satisfied with my choices of where to put the controls, but now when it came it learning a new control such as the displaying a line graph from the Java Library was a fun challenge. I looked up tutorials on how to display a line graph and I would take a snippet of the code and paste it into my IDE (Integrated Development Environment) and played with the code to see when what happens when I deleted this line of code or added this line a code. Now, taking the knowledge from what I learned on displaying a line graph and applying it to displaying stock data would be child’s play since I’m just changing a few lines of code to display the date and the stock price on that date and adding lines of code to set the data that the program is going to graph. I could of used JavaFX SceneBuilder (visual layout tool) and drag and drop the controls and modifier some text fields to set the view of the window to my liking, but I prefer hard-coding the user interface.

When it came to choosing the right data structure, I had to sit down and think what kind of data I’m dealing with. For example, I had to deal with the millions of stock data for this project. The data structure I chose was a TreeMap that’s based off a Binary Tree. The operations: searching, deleting, and insertion is O(log(n)) and it’s a Red and Black Tree so it balances out the tree for me to always get O(log(n)) which is pretty good, but I want O(1) which is the most fastest algorithm. If I wanted, I could’ve used a HashMap which is based on an HashTable where searching, deleting, and inserting is O(1). Why didn’t I use the HashMap then? Since the HashMap is based off a HashTable, we’re using an array to store the stock data of a company and that’s not good because an array has a fixed size where the TreeMap does not. A good note is that the HashMap can be resized, however the way the data is stored is based off a hash function and the size of the HashMap is included in it, so resizing means changing the hash function, thus changing the way data is stored, thus the more data we insert, the more the HashMap gets resized and that’s bad. We don’t know how long the Company will exist until it dies, so we can only assume that the size of stock data is infinite. ­It makes sense to choose a TreeMap over a HashMap because size matters even though HashMap operations are O(1) and TreeMap operations are O(log(n)).

Hopefully, after reading this you might of have gotten the idea that solving the problem (or programming) is not the only thing you need to know. It will involve much more depending on how big the project is. I had to learn how to use an API to acquire the stock data, designing a user interface when designing is not my cup of tea because I’m majoring in Computer Science, not Art or UX (User Experience) Design! And then having to choose the right data structure to hold the stock data when there is no perfect data structure like figuring out the theme of a book, there is no right answer.