Simon Mai

Dr. Kent Jones

CS 172

12 December 2018

Individual Summary

This project was both fun and frustrating at the same time. I saw it as a chance to demonstrate the many things we have learned this semester, although that didn’t really pan out in the end. Largely, Ivan and I ended up using classes, vectors, and object-oriented programming a lot for our final project. Although vectors are really just arrays with dynamic memory allocation allowing them to store as many things as a computer RAM will allow, we found them really useful in establishing the menus for all of the food court venues and for storing the orders to be later outputted on the “receipts” or text files. Given the limited amount of ordering and the limited amount of food and drink items, an array might have been better for faster performance but if we were to expand the program more, vectors would have been a more surefire way to ensure that we didn’t need to keep constantly expanding the array when a vector could just do it by itself. As for classes and object-oriented programming, these two things were truly the core of what we learned all this semester in CS 172. Fittingly (and due to the core requirements of this project anyways), we made these two concepts a core part of our project. There are five venue classes bound together by one store class. When the classes and their functions are called, they are done so through objects.

What surprised me the most about this project was the fact that it could have been enormously more complex then it is in its current state. In its current state, the program can only manage a few orders, one at a time, and the menus for both food and drink are quite limited. It would have been nice to take a graphical approach to this program instead of making it text-based, making the program look and function more like an app you would find on a phone. The text based program as it currently exists looks very odd and unwieldy. Instead of using numbers to select from the venues and the menu items, I would have preferred to type in the actual string, as if I was verbally ordering the food myself. I would have also tried to figure out how to use inheritance to reduce the clutter of member variables and functions that the venue classes depend upon.

This program as it exists now could be built on much further. It could use inheritance from a single base class (as was the original intent, although that was cut out due to a time constraint) for the venue places. Polymorphism might have been a viable choice as well, in the case that each individual class having its own unique menu items. For example, the chinese class could have accommodated entrees and appetizers. The selectChinese() class used to implement the menus could have then been tweaked to allow for appetizers. Similarly, the taco class could have included a customization deal where users could have picked a base taco and have a choice of ingredients to create their own perfect taco. Each class could have benefited as well from having varied menu lengths, instead of just being fixed to 6 food items and 5 drink items. Using vectors in this case (or even pointers) could have allowed for this possibility, with a theoretical burger class having 20 combination of different types of burgers. While the same function would have used, there would have been different implementations. As mentioned before, a graphical approach could have been utilized for greater usability and especially since towards the end of this project, when Ivan and I saw that this could very easily be turned into an app for a local mall where people could order food and drink in case the munchies hit.

All in all, the program worked for what it is. It could have benefitted enormously from more advanced concepts, yet time constraints did not allow that to happen. In order to implement those other concepts we learned in class though, significant time and examples would have been needed outside of class for smooth integration into the program.