Project 01 Refelection Document

W200 – Spring 2020 – Saturday 8 AM
Professor Gerry Benoit
Samuel Gomez
March 14, 2020

This project intended to serve as a proof of concept warehouse management system. To accomplish this goal, first, the most fundamental objects necessary were initially created. As basic functionality between the objects took place, more and more sophisticated methods were created for added benefit. The main classes of this project, in order of use, include receiving line orders, totes, cold storages, and wash line orders.

Upon starting the program, the terminal window displays the main menu. The user should start by creating a receiving line order. The order requests that the user inputs arguments for the product variety, the weight of the load, and the destination of the product. Once the order is executed totes are created by calling the tote class from within the receiving line order class. The receiving order generated totes are placed into the cold storage that was defined by the user when creating the order. As a side note, receiving line orders are appended to a list called order history so that the user can reference every order at a late date.

Four cold storages (A, B, C, and D) are instantiated in this program at the onset. The cold storages are capable of holding 125 totes each. However, while testing my project, I created over 10 million cold storage locations and generated over 1.2 million bins in one order. The project was able to handle this quantity with no issues. The cold storage is defined by aisles, columns, and rows. The filling process follows that of an automatic storage and retrievable system such that it fills the first open location upon every tote storage cycle.

Once totes are placed into the storage, a wash line order must be created and executed to remove the totes from the cold storage. The wash line order requests product variety, product size, and the number of totes for arguments. Once the order is executed totes that match the criteria are removed from the cold storage on a first in first our basis. As the totes are removed from inventory a zero is placed into the location to indicate that it is now empty.

If time permitted, I would have liked to complete the wash line FIFO process for all cold storages. Currently, the FIFO process works for Cold Storage A only. I initially set up the program for one cold storage but took on more when I was satisfied with the progress of the program. The receiving line orders are able to store product into multiple cold storages but I didn't perfect the wash line orders pulling product from multiple cold storages. However, I could have accomplished this if I had more time. I was content with using one cold storage and added the others for more practice. Another interesting thing I would have like to do is put this code in an HMI/SCADA software I use at work to create a 21st century interactive user interface. I would very much like to pursue this at some point.

The most memorable challenge I overcame was creating the FIFO process. It was challenging to set up the logic of pulling the oldest bin that matched the variety and size criteria out of the storage. The process I was able to implement consisted of nested for loops to find the first bin that matched the criteria. Once the tote was located, I broke the nested for loops using else: continue, break statements. Next I iterated through every storage location that first had a tote, second matched the variety, and third was the same size defined by the user. I then compared the date and time to the tote I first

Project 01 Refelection Document

W200 – Spring 2020 – Saturday 8 AM
Professor Gerry Benoit
Samuel Gomez
March 14, 2020

located, if it was older I replaced the first tote as the FIFO tote and continued analyzing every other tote to the “new” FIFO tote.

The second biggest challenge for me was creating the storage. I first experimented with a single dimension list. Although it was sufficient, I could not easily determine the aisle, column, and row of the location. Though this would have worked, I wanted to create a location that could be referenced by the aisle, column, row format. I then experimented with a dictionary. The benefits of the dictionary allowed me to specify location ‘Aa0c0r0’ (Cold Storage A, aisle 0, column 0, row 0). I initially liked the format, but it was a pain to create other locations and to search through the dictionary. I ultimately decided to use a list within a list within a list. This three-dimensional approach enabled me to create the size of the storage with ease as well as refer to the location as a.location[0][0][0] (Cold Storage A, aisle 0, column 0, row 0). This format was more intuitive for me and worked well within the program. It also made error checking much easier when the user was prompted to search for a specific location within the cold storage.

The biggest takeaway from project 1 is that it is important to plan the classes, methods, and functions of the program before writing a single line of code. Also, accounting for what could be will make things much easier over the long-term. Creating a graphical flow/structure helped tremendously towards achieving a program that worked right out of the gate.

I am extremely satisfied with the outcome of this project. I believe that I could build upon this foundation to one day offer a software solution of my own. The features are self-explanatory and quite intuitive. Users familiar with warehouse management systems like Famous and ProducePro would be in familiar territory when interfacing with the program. I tested functionality with several colleagues in the office. The VP of Operations, CFO, IT and Engineering teams got a kick out of the program. The skills I have learned thus far this semester were put to the test. It was a great experience directly applying the material we cover in class to solve a problem in the real world.