**Diary of Shop Project (April)**

**17/04 – First Look at Specifications -** I read through the specifications of this project and found that file handling/persistence and abstract classes would be needed, this required me to do a bit of research to become more comfortable with manipulating them.

**17/04 – Mind-map Design -** I designed my mind-map with this in mind, I added a task for taking from & writing to a file. These are only in the mind-map to show where the data for my shop is stored**.**

**18/04 – Previous Shop project -** I am unsure how this project differs from the previous shop project I designed, other than the use of persistence and inheritance.

**18/04 – Menu Structure -** I created the menu structure, based off of the menu within the specifications, this will have additional options to accommodate the personal tasks.

**19/04 – Abstract Item Class –** The Item class needed to be abstract as I wasn’t planning on making any item objects but instead to use it as a shell for my food and drink items, which are subclasses.

**19/04 – CSV -** I made CSV files for my food and drink classes to use as a file for persistence, I faced some minor difficulties when spaces were between the variables and the commas, this wasn’t difficult to fix once I found what was causing the problems.

**20/04 – File Handling –** I created a class where I am putting all the code associated with handling files. Read and Write methods were necessary for saving/loading the contents of the files.

**20/04 – ManageItems Class –** This class is used to handle set operations against the subordinate classes (Food & Drink).

**21/04 – Personal Task -** I have decided not to put as much time into my personal task for this project as I have a lot of assignments for other modules which I must allocate more time to.

**21/04 – Personal Task (Shopping Cart) –** This will keep a running total of money spent as well as a list of items with their quantities attached. This will be persisted into a file that will keep track of pervious transactions, how many of each products are sold, while those that have sold at least one item.

**22/04 – Cancel Button –** Using JOptionPane in the past, I neglected the cancel button feature of the dialog boxes. I have made it so that the cancel button or hitting ok without inputting any information will result in the program assuming the last option of the menu is input, which is the exit menu function.

**22/04 – Drop-down menu –** I added a JOptionPane drop-down menu, to choose which items the user would like to delete/change quantity of within the cart.

**23/04 – List to HashMap –** The use of List for cartItems and items became very cumbersome – every method needed to use a locate method to fin an object by name. Since our product names were unique anyway, I used a HashMap, with the item name as the key. This took a couple days to change everything across.

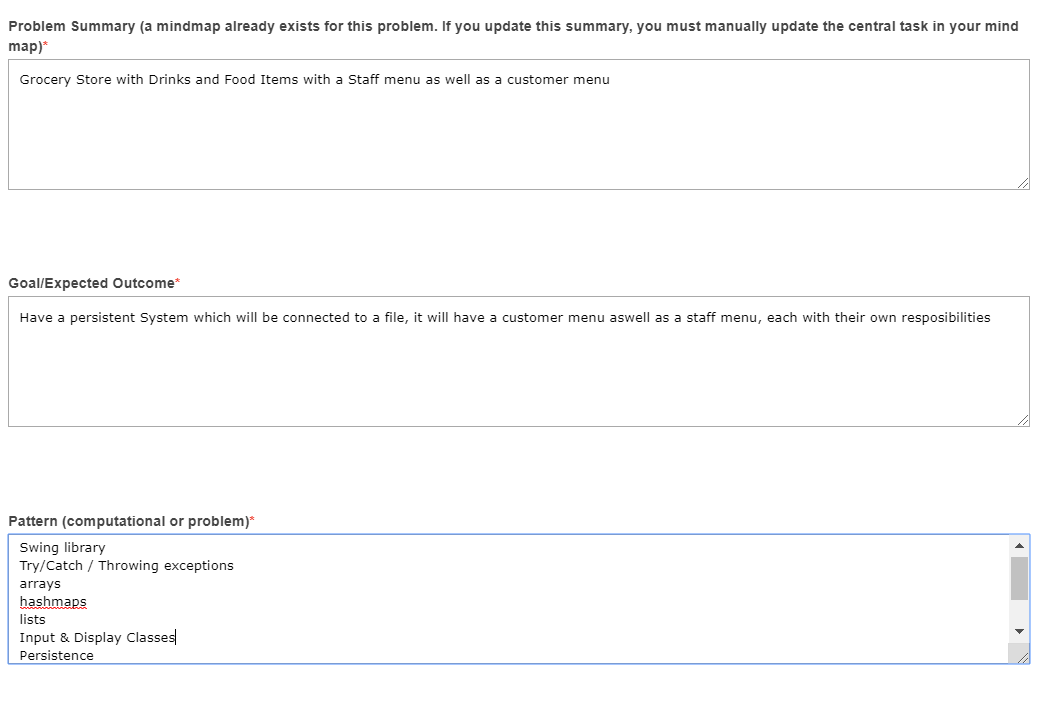
**23/04 – JOptionPane Buttons –** Along with adding function to the ok & cancel buttons in each usage of the dialog box, I made a dialog with custom buttons for the check-out screen, these buttons show “Check-Out” & “Cancel” and will perform those actions when interacted with.

**24/04 – Formatting –** I used String.format to format the list displays in JOptionPane.

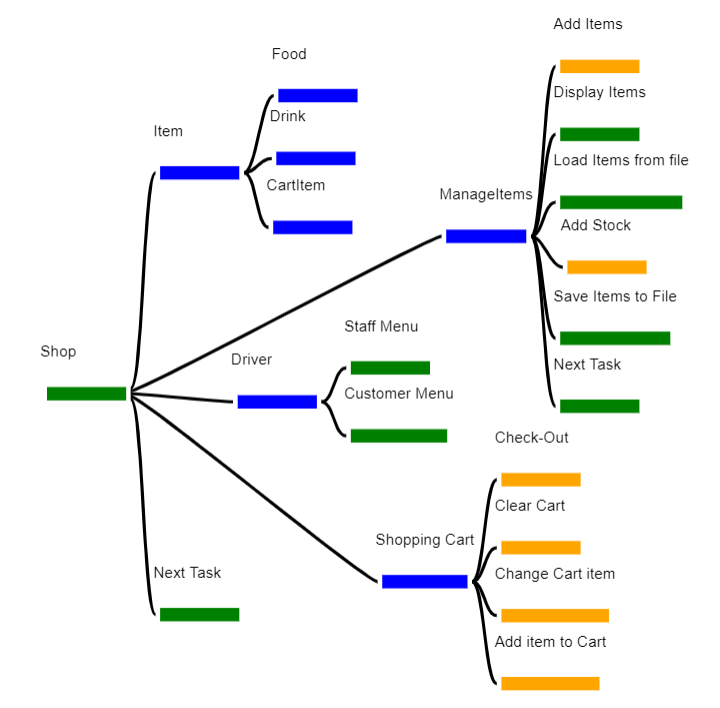
**25/04 – Debugging –** Had to re-do some of the code as I wasn’t checking the list when adding a new product. This along with an error message displaying where it shouldn’t have while inputting a value for updating stock were the final changes I needed to make to the project.

**Understanding Of Project**

My understanding of this project is that I am required to use file handling to allow my program to be persistent, This is necessary for keeping track of the new items made within the program and also updating the stock levels of each of the items. There is a need for a staff and customer menu which have different functionality as expected. Though I added in a feature within the customer menu; this will be a shopping cart menu, this will provide functionality to view and change your cart how you please. I was able to use some of the classes I’ve used in previous projects, though the file handling class is new, which meant I would need to make that class so that I can reuse it In future also.

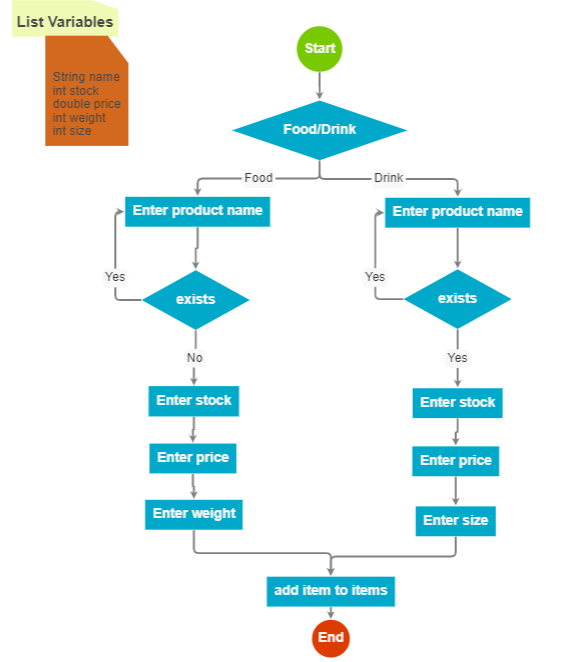


**Mind-Map Solution for Project**

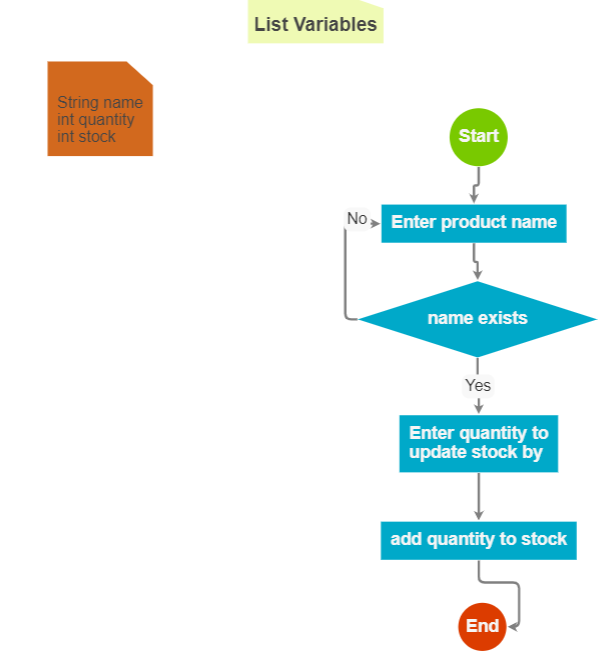


**Design Artefacts of Tasks (DATs)**

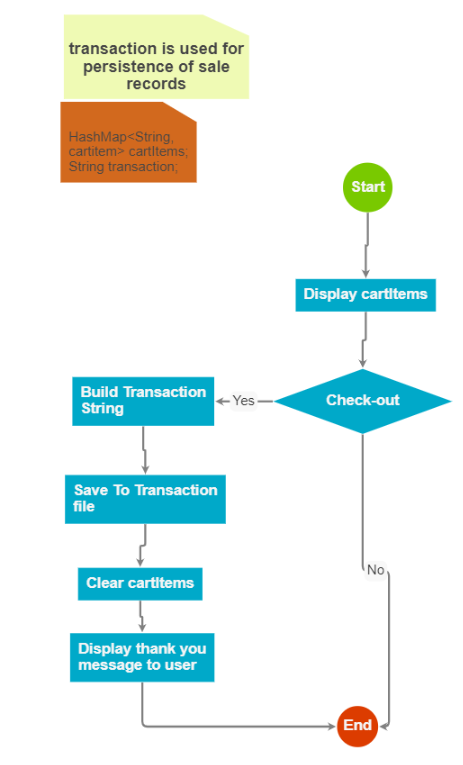
Add Items



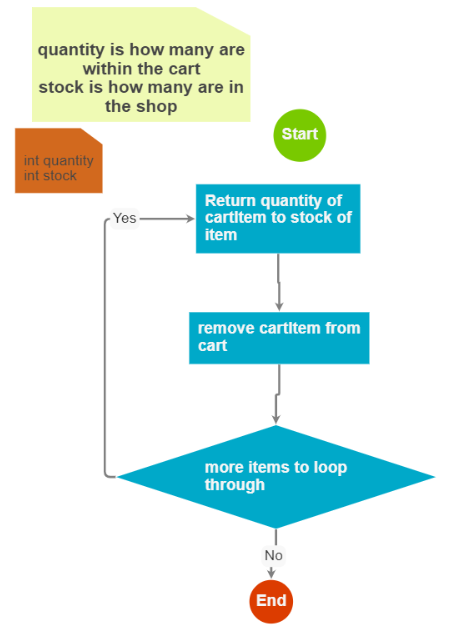
Add Stock



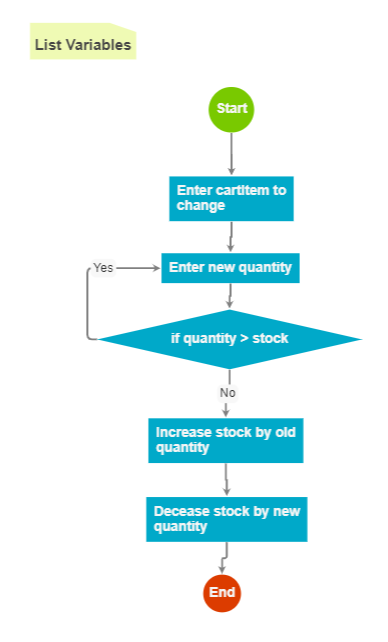
Check-Out



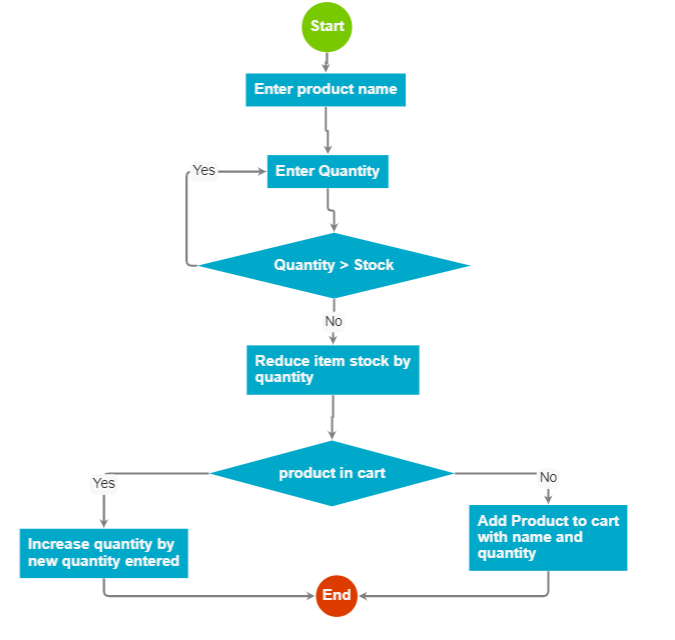
Clear Cart



Change Cart Menu

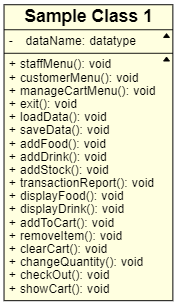


Add item To Cart

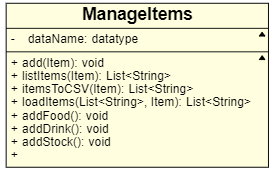


**UML Class Diagrams**

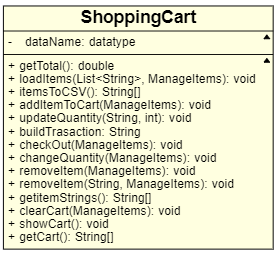
UML Driver



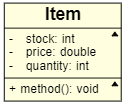
UML Manage Items



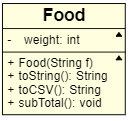
UML Shopping Cart



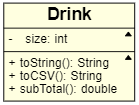
UML Item



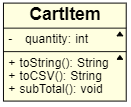
UML Food



UML Drink



UML CartItem



**Reflection of Solution**

* **Creating problem summary –** It was easy to translate what was within the spec into the problem summary
* **Articulating goal -** - I found this rather easy as it was very similar to a previous program with only a few adjustments
* **Applying pattern recognition and reusing DAT’s within the solution –** I reused my input, menu and display classes, which I have used in the past. For the project I was required to make a file Handling class which I will likely need to reuse in future projects.
* **Creating Mindmap –** This required some thought as I needed to think about how my system was going to work and where would require new
* **Creating DAT(s) for modules/functions –** Each DAT I made was for modules/functions and a lot of them are for modules, with extra modules performing tasks within them modules.
* **Creating DAT(s) for regular tasks -–** I made a DAT for each of the cart actions as they had more to them than the rest of the tasks
* **Using Data correctly and efficiently –** I only write and read from the external files once in the course of the program making it more efficiently accessed, the transactions file updates each time someone checks-out.
* **Translating design to code –** This wasn’t too difficult, though I added a few new things to my JOptionPane usages. This includes ok & cancel button operations as well as designing a new button.
* **Reconciling design and code –** There were a few small tasks that were overlooked within the design, but most of it was logical and well thought-out.