Implementation details

For the server, I was told to use 3 different data structures which would act as the waiting area, brewing area, and the tray area respectively.

- The waiting area is represented by the stack
    - Reason why I chose stack: It's quick to insert and delete data from. It has a multitude of that can manipulate the extremes of the stack particularly, which I now realize it might not have been the smartest choice.

- The brewing area is represented by the queue
    - Reason why I chose queue – (specifically ConcurrentLinkedQueue): It's quick to insert and delete data from. It has a multitude of methods that can manipulate the data structure in with methods that would have been inaccessible due to being of a different type. I can use methods associated with concurrency, linked list, and queue in one data structure.

- The tray area is represented by the array list
    - Reason why I chose array list: It's quick to insert and delete data from. It has a multitude of that can manipulate the entire data structure. I am comfortable using this data structure since I have been using it for a while in a million different cases.

The threads that are running in the server are for the transfer of drinks between orders, coupled with the use of a synchronized lock.

For the client, I have used the Map family, specifically the LinkedHashMap and HashMap. I did this due to make sure that the OrderID in the order method stays unique. Subsequently, I created an ArrayList<String> that stored the drinks and from there, the user chose a number from 1 to 10, the numbers being the index of the said drinks.

Unfinished implementations

I couldn't complete the connection between the server and the client, but I have tested the server, and it's able to receive connections.

I couldn't complete the order status method since I didn't know how structure it concretely.

Project review and personal reflection

I found the project tedious, but enjoyable. My time management skills were not on par with the assignment, so the final product is not the vision I had imagined.

It went well at the beginning, but from there it became more difficult.

I think the most challenging part for me was not being able to have a good plan from the beginning to stick to, and the server-side implementation. In addition to this, I believe that I spent a lot of time being unsure producing code. Furthermore, I think the main issue that I had was to underestimate the time constraints and overestimated my abilities to finish the assignment, which resulted in me not being able to complete it by my own standards.

My original plan was to take inspiration from the code that showed a simulation of bank client and server system. In addition to this, my original plan was to add a Key Event object, which would have been for the SIGTERM and exit strategies. The reason why I haven't been able to stick to my original plan was due to me not knowing the full understanding of how to implement it correctly, and the same thing applies with using JSON, even though we learned through the lecture and applied it in the lab, I was sceptical to use it, since it could cause malfunctions in IntelliJ, the IDE I was using to code. While I was devising a plan based on that, I realized that I was not able to differ the varied methods and functions completely and precisely

There are a few features that I am proud to have made and that is the login section in the Customer.java file and I also created a method using ArrayList<String> that shows the drinks menu of the virtual café. I have also added a regex to check if the username followed the format.

If I had a chance to do this piece of coursework again, I would make sure that I have made a through plan before beginning the coursework, and have mini-individual targets, which would aide me in staying motivated, concentrated, and stimulated along with asking for help without feeling insecure or scared. Overall, I think my assignment shows that I used the required function and Objects, while adding a unique take on it.