**Implementation details**

I separated the server and client architecture into two parts.

Client:

-Customer.java:

Listen customer's input

try to new client use the input of customer and the client will connect Barista

if connect successfully, print the "connect successfully"

always listen customer’s input and use call\_function to handle customer's order

Runtime.*getRuntime*().addShutdownHook(new Thread() {

Add a thread which can handle sigterm elegantly.

-Client.java:

after creating new client

use localhost and port 8888 to connect Barista

send Barista user's name to put information into HashMap

The call\_Function function can execute different commands based on the content of input.

Thread(print\_thread)：

It is a thread which can listen the socket.getInputStream and print the information from Server(Barista).

To receive the information from barista, it has a loop which always try to receive the input stream using reader.nextLine(), and the received will be handled and print.

Server:

-Barista.java:

start the two threads in OrderHandle.java

run Runserver method in Barista

when Runserver starts, Barista will create new Server Socket using port 8888

and Barista will always listen if there is a new client connection, a new thread ClientHandle will be created using the new socket and start.

Thread (ClientHandler(socket) ))

ClientHandler.java:

When a new clienthandler is created, it will listen user's input and use the input of first time as user's name. If this user is a new user, initialize it and push the information into HashMap

Method: run

it will create two thread (writer\_Barista and completed\_Order)

And then clienthandler will keep listening the socket.getInputStream(),which is the input of client. When customer print "order status", it will return user.status which include the count of tea and coffee in three different areas.

if the input contains "coffee" or " tea ", string will be replaced and use OrderTea/OrderCoffee to modify user's status.

ClientHandler will return order information to customer based on the count and user's name.

if received unknown line, it will also return unknow commend

Thread(writer\_Barista):

it is a thread to listen the System's input from Barista and write to customer.

It is created when ClientHandler starts, writer\_Barista will listen input by using in.nextline, and send to customer.

Thread(completed\_Order):

It is a thread which can distinguish if user's order is finished,

if the total count of teas and coffees in waiting area and brewing area ==0, it will return customer information of order and set the count of tea and coffees in the tray to 0. If a customer would like to order something after the last finished order, it can also work.

OrderHandler.java

It has a HashMap here which records user's information cause HashMap can store and read these information quickly and we don’t need to delete user's information ,so it is fit for the requirement.

method(checkerUser)

traverse all the HashMap, if user exist return user, if not , return null

method(initialUser)push user's information into HashMap

method(orderTea)increases user's count of tea in waiting area

method(orderCoffee)increases user's count of coffee in waiting area

Thread(tea\_maker):

traverse HashMap, if user need tea, make tea

Thread(coffee\_maker):

traverse HashMap, if user need coffee, make coffee

run ():

when initializing Barista , these four thread will start run and prepare tea/coffee

start tea\_maker1

start tea\_maker2

start coffee\_maker1

start coffee\_maker2

User

method to increase and decrease the count of tea and coffee

method(completed\_Information())

**Bonus feature:**

1. can receive messages from the server at any time. Not just as a response to a command sent by the client.
2. Can transfer coffee/tea from tray when client is disconnected.
3. Add Thread to handle SIGEMNT CTRL-C

**Functionality was not implemented**

The only function I did not realize is I cannot run my program on the terminal although it can run in idea and can be complied by javac -cp “.” Barista.java. After I complied and try to run it, the thread doesn’t work.

I guess java -cp xxx.jar Customer/Barista will be useful.

Although I did not know the reason for this problem, it made me deeply understand that my reliance on IDEA was incorrect.

**Project review and personal reflection**

I'm actually very happy with the results because I did almost everything I was asked to do, and I put a lot of time and effort into the program.

But perhaps because my programming skill is not excellent enough. I did not finish all of the Bonus feature.

I think the most challenging part is the part to handle data, cause It's easy for me to make mistakes here.

The features that I am particularly proud is I used the HashMap and finish the part which can transfer tea/coffee from tray.

I think HashMap is a way to greatly increase the speed of store data and fits well for this assessment.

If next time I only have fewer time constraints, the first thing I should do is have a general idea of the overall structure and architecture of the program. I think this can greatly reduce the programming time. And have a good comment habit can also reduce the developing time.

I think I have a steady pace, which encourage me to finish the function step by step.

I still have a long way to go to become a good programmer.

This assignment is very helpful for me. I have learned how to arrange and plan a project and how to reasonably plan the structure of the project. Meanwhile, I may try to understand some more basic knowledge in the future, such as the advantages and disadvantages of different data structures and their implementation principles, and java compilation on the command line.

At the same time, I will deepen my understanding of locks, because I think I did not use it in the assessment very well.

That’s all

Regards