Assignment Four – Approach to synchronization

**CoffeeShopManager**

I have built up a class named CoffeeShopManager which will have a record of all the customers being currently served in coffee shop. Along with the track of a cook serving an order with cook names and order number. This class will be a central and integral part of this application.

There are a few data structures inside this class namely:

1. HashMap<Customer, Boolean> orderStatus: indicating whether a customer’s order has been served or not.
2. ArrayList<Machines>: indicating currently serving active machines
3. ArrayList<Customer> currenltyServing: indicating a track of customers being served currently

All the classes such as cook, customer, machine and CookAnItem revolves around the above class. I have thought of two object locks predominantly for Customer side and another for Machines and Cook side.

Synchronization would be obtained on this lock. An important point to note here is there would be just one instance of CoffeeShopManager in all. This object will be initiated in Simulation.runSimulation() method with the parameters as passed to runSimulation like numTables, numCustomers, numCooks etc.

**Customer**

A customer thread will acquire lock on CoffeeShopManager’s instance variable lockOne and perform the run method. Duties of a customer include waiting for tables to be available, placing an order, receiving an order and wait for customer to eat up his/her order and leave the coffee shop. Another customer will do the same thing.

**Cook**

A cook thread will acquire lock on CoffeeShopmanager’s instance variable lockTwo and perform run method. Which will include following activities. Track whether cook is available to work or not. Then keep a track of Customer’s order. Delegate the order’s item individually to specific machine and wait for that machine to finish up. Updating the order status in CoffeeShopManager’s array list. Delegating or conveying this information to Customer thread indicating order has been completed.

**Machine**

A machine thread will check if machine is available to work and initiate a thread of CookAnItem as suggested in the pdf to cook. This thread will sleep for certain milliseconds to demonstrate a simulation.

During all of these executions, each of the class like Customer, Cook, machine etc will log the Simulation events for validator to validate.

This is the approach I have finalized for this application. There might be minor changes to what have been planned. I have created this application with a certain different approach just to verify whether this is working which we might call a hacky way of doing things. But in the coming week I will be improving this application to reflect above solution design. I am providing my code here as pseudo code in another text file.