**HOMEWORK 4 – COFFEEE SHOP**

**Simulation.Java**

* Create a List of Customers to store a orderList
* Create a List of capacity to store the currentCapacity
* Create a static object for burger,fryer,coffeemaker // Declare the object
* Define the Machine object to startup the machine for the three static object defined // Start up Machine
* Declare the Threads for Cooks

**Thread[] cooks = new Thread[numCooks];** and start the thread

Code for Customer order is already given

* Wait till customer thread join **customers[i].join();** // Till customer finish
* Stop all the machine threads and log the end of the simulation

**Machine.java**

* Create a List to maintain listOfFoodInMachine to know the what type of food the cook is feeding inside the machine.
* Add the food item to listOfFoodInMachine
* Start a thread for cooking and make the thread sleep for given cookTimeMs depending on the food item.
* **lock the listOfFoodInMachine**
* Once the food is done, remove the item from listOfFoodInMachine and **notify all** items waiting for listOfFoodInMachine lock.
* **Release the lock**
* **lock on the current cook finishedFoodOrder list // list created in cook class**
* add the item in the finishedFoodOrder
* **notify all** other item waiting for finishedFoodOrder object.
* **Release the lock**

**Customer.java**

* **lock on currentCapacity -**  number of tables that are currently occupied by the customer // check the entrypoint
* check whether current Capacity is less than the total number of tables
* if table is available, add the customers to table and notify the other customers else wait till there is a space
* **Release the lock**
* **Lock on orderList** - queue to keep track of order in which the customer has to be served // place an order
* Add the customer to the orderList
* Once added notify all the items waiting to obtain the lock to orderList.
* **Return lock**
* Have a map **completedOrderGiven** with customer as key and Boolean that returns true or false as value
* **Lock the map**
* Wait till the order is complete that is till map.get(customer object) returns true
* **Notify all once complete**
* **Release the lock**

**Cook.java**

* **Lock on the orderList.**
* Wait if the orderList is empty. Get the customer based on priority (Gold or Regular)
* **Release the lock**
* Simulate a call to each machine with the order
* **Lock on each foodList**
* check for the capacity of the machine with the given foodList
* if its more, wait till it is less than the capacity else proceed.
* then call makeFood in Machine
* **Release the lock**
* **Lock on the finishedFoodOrder –** created to know whether the given food is complted or not. Used in Machine class
* Check whether the **finishedFoodOrder** is equal the total dishes ordered by the customer
* Then notify all items waiting for **finishedFoodOrder** and also the customer
* Change the **completedOrderGiven** map to true to indicate the customer that his order is complete.
* **Release the lock**