When: Wednesday Feb 15, 2017 at 8-10pm

Where: AEC 402

Who: Zainab Hussein and Cameron Zurmuhl

Discussed

The plan for the project is to have 3 classes: Customer, Operations and Main class to answer the question of the optimum number of cashiers for this new café.

- Customer class implements interface Queue
  - Constructor – create two lists:
    A general customers list for when a new customer is created/ enters the café is stored in a queue of LinkedList. The other list for served customers stored in ArrayList data structure for direct access of a given customer. Instantiations of these are done in the constructor setting both to null in the beginning. They are all declared as global variables within this class.

  - Methods

    newCustomer() for creating new customers entering the café. Returns a string of customers.

    isServed() keeping track of when a customer is served. Expect a return value for number of customers served. When a customer is served, pull from Queue of LinkedList and add to ArrayList.

- Operation class calls Customer class
  - Constructor – declaration of various variables used in the calculations, such are:
    double lamda – average # customers arriving per minute
    double u – random # drawn uniformly (0,1] using math.random java operation
    int s - # cashiers
    double c – amount of money per cashier per day
    double p – profit/customer
    long arrival_time per customer = -ln(u)/λ using math.log java operation
    double r – average # customers served by cashier per min
    long service_time

  - Methods:

overallCustomerTime() tracks the time spent by the customer in the café by the equation:

$$Depart_{time} = Arrive_{time} + Wait_{time} + Service_{time}$$

$$Arrive_{time} = -\frac{lnu}{\lambda}$$

$$Service_{time} = -\frac{lnu}{r}$$

And wait time is a constant that is different for each customer depending on their position in the queue. Since served customers are stored in ArrayList,

$$Wait_{time} = k * (indexOf(customer_{instance}) + 1)$$

Where k is a constant long value.
This method returns a long value for Depart_time – Arrive_time

Overflow() returns number of customers turned away on two occasions: during the day when the # customers is greater or equal to 8 times the cashiers as illustrated:

```
If( no_customers >= 8s ) { …turn away…}
```

And at the day when the café is closing down as illustrated:

```
If( time == 960 && !isEmpty(Queue<LinkedList> instance
                    ) { …turn away…}
```

turnAwayCustomer() tracks if customers turned away at the end of the day almost at closing time using Boolean type return illustrated: `if( (960-Serve_time) >= Arrive_time ) { return true }`

- Main class
  - Arguments – s, p, c, r, λ as specified in manual
  - Priority queue – ArrayList of customers served stored in priority queue for efficiency in access ordered parameters from the Operations class. Need to discuss this further.