COMP-10205 – Data Structure and Algorithms

Assignment#5
Queues

Optimizing the Line-up
5% of course grade

Submission Requirements

Complete the following exercise and submit electronically in the assignments folder on eLearn as an IntelliJ Project – Zip the entire folder not just the source files in MyCanvas. Please refer the course Calendar for the exact date and time of the submission. This assignment is to be completed individually.

Background

You are to write a program that will attempt to optimize the time it takes to process customers in check-out lines inside a grocery store. You will use the Queue data structure for this assignment.
The general idea is to always place customers in the best checkout line possible. This will be based on the total number of customers and the number of items in each cart being processed. The time to serve a single customer is calculated by the following formula:

*t = 45 + 5*ni;*
where: t, time in seconds that it takes to serve a customer,
ni, the number of items in the customers cart.

A store consists of 1 or more check-out lines defined by:

- *n* = number of normal check-out lines
- *f* = number of express check-out lines (less than or equal to x items)

A store could have up to 4 fast check-out lines.

Customers at the store may have many items in their cart or only a few. Each new customer will be added to the check-out line with the cumulative least amount of time in front of it (the shortest line). Any customer may optionally enter one of *f express lines* if they have less than or equal *x* items in the cart. It should be noted that they do not need to enter an express line if another check-out line would result in a shorter service time. Customers with more than *x* items must enter one of the other *n* check-out lines in the store. The total number of check-out lines will be equal to *n + f*;

Example Case

Example Data: *x = 10, n = 2, f = 1* and 8 customers and carts come to the checkout in the following order: 6,7,45,10,11,21,2,4 (the number is the number of items in the cart). You can assume they all arrive at the check-out at the same time for this lab. When placed into the optimum check-out line the following check-out lines (queues) will result.

1(express)	2 (normal)	3 (normal)
6(1 – 75s)	7(2 - 80s)	45(3 – 270s)
10(4 – 95s)	11(5 - 100s)	

2(7 – 55s)	21(6 – 150s)	
4(8 – 65s)		

Part A

You need to output your optimum starting case for all of the check-out lines in the store and then calculate the amount of time it will take for the store to be empty of all customers. Your output must show each Queue in order. An example output is shown below:

```
PART A - Checkout lines and time estimates for each line

CheckOut(Express) # 1 (Est Time = 290 s) = [[6(75 s)],[10(95 s)],[2(55 s)],[4(65 s)]]
CheckOut(Normal ) # 2 (Est Time = 330 s) = [[7(80 s)],[11(100 s)],[21(150 s)]]
CheckOut(Normal ) # 3 (Est Time = 270 s) = [[45(270 s)]]
Time to clear store of all customers = 330 s
```

Part B

Simulate the removal of customers from each check-out by servicing the customers. It is suggested you set up a loop to calculate the state of the Checkout lines after each second, but only display lines every 30 or 60 seconds simulated. Once the amount of time has passed required to service the customer at the start of the Queue, the customer must be removed and processing will continue with the next customer. For the above case we would start at time = 0 and continue until time = 330 s at which point all customers would have been serviced. It is suggested that you not show the output after every second, but instead after every 30 or 60 seconds. The example below shows every 60 seconds for the same data that was used in part A. In your solution you must use a simulation step of 30 seconds.

```
PART B - Number of customers in line after each minute (60s)

t(s)  Line 1  Line 2  Line 3
 60       4       3       1
120       3       2       1
180       2       1       1
240       1       1       1
300       0       1       0
330       0       0       0
```

Example data file for sample cases

1 2 10	f (express lines) n(normal lines) x(when ni<=x, customer can go to fast lane)
8	number of Customers
6	ni for Customer#1 cart
7	ni for Customer#2 cart
45	ni for Customer#3 cart
10	ni for Customer#4 cart
11	ni for Customer#5 cart
21	ni for Customer#6 cart
2	ni for Customer#7 cart
4	ni for Customer#8 cart

Suggested Steps:

1. Create a new project in IntelliJ.
2. Add the example data (Customer_Example.txt) and evaluation data set (Customer.txt) to the project.

3. Add a `toString` method to the LinkedQueue.java that will print all the items in the Queue on a single line.
4. Add a new class to the project to hold each `Customer`. Add a property for the number of items in the cart and a method to calculate how long it will take to serve a customer and any other properties and methods you feel are appropriate.
5. Override the `toString` method to display the contents of the object neatly (in my example the Customer will print the number of items and the time it will take to serve a customer).
6. Inside the main class you will need to create a `LinkedQueue` for each of the checkout lines. You will also need to track the current wait time for each queue.
7. Read the check-out line and customer information from the data file and add them to the correct Queue.
8. Output your results for Part A and verify the test case and the Customer data case.
9. Solve Part B by using a loop to process all the lines simultaneously one second at a time. Remove the customer when the loop counter equals the time to process for that check-out line.

LinkedQueue.java class must be used for the lab

```
package comp10205_lab5;

/**
 * Linked Queue class to be used for Lab#5
 * Used code from COMP-10152 - Data Structures and Algorithms (pg.417-419)
 * Modified by Mark Yendt to be Generic
 */
public class LinkedQueue<E> {

    /**
     * Node class to be used by the LinkedQueue class
     *
     * @param <E>
     */
    private class Node<E> {

        E value;
        Node<E> next;

        Node(E value, Node<E> next) {
            this.value = value;
            this.next = next;
        }
    }
    private int count;
    private Node<E> front;
    private Node<E> rear;

    /**
     * Constructor for a LinkedQueue
     */
    public LinkedQueue() {
        front = rear = null;
        count = 0;
    }

    /**
     * Add an item to the Queue
     *
     * @param value item to be added to the Queue
     */
    public void enqueue(E value) {
        if (rear != null) {
            rear.next = new Node(value, null);
            rear = rear.next;
        } else {
            rear = new Node(value, null);
            front = rear;
        }
        count++;
    }

    /**
```

```
     * Remove an item from the Queue - throws exception if queue is empty
     *
     * @return the item at the from of the Queue
     */
    public E dequeue() {
        if (isEmpty()) {
            throw new IllegalStateException("Cannot dequeue - Queue is empty");
        }

        E value = front.value;
        front = front.next;
        count--;

        if (front == null) {
            rear = null;
        }

        return value;
    }

    /**
     * Check is queue is empty
     *
     * @return true if empty, false if not
     */
    public boolean isEmpty() {
        return front == null;
    }

    /**
     * Shows front of Queue
     * @return the value of the first item in the Queue
     */
    public E peek() {
        return front.value;
    }

    /**
     * Obtain the number of elements in the Queue
     * @return
     */

    public int size() {
        return count;
    }

    /** TODO : Add a ToString method to the class
}
```

Hints

- Use the `isEmpty()` method of the `LinkedQueue` to decide if queue has customers.
- Use the `size()` method of `LinkedQueue` to indicate the number of customers in any checkout line.

Marking Scheme

Program structure – Comments, follows best programming practices – 20%
LinkedQueue class modified with toString() – 5%
Customer class built as specified – 25%
Part A (Correct Results) – 25%
Part B (Correct Results) – 25%