CME 2201 Assignment 1

Due date: 18.10.2017 Wednesday 23:59. Late submissions are not allowed.

You must upload your all **'.java' files** as an archive file (.zip or .rar). Your archived file should be named as 'studentnumber_name_surname.rar/zip', e.g., 2016510031_Ali_Cuvitoglu.rar.

Control date: Control of the homework will be on 23rd, 24th and 25th of October. Research assistants will control your assignments in their office. Therefore, you should write your name to the schedule list on the door of R.A. (door no: 124). You will have 10 minutes to show your assignment.
Plagiarism Control: The submissions will be checked for code similarity. Copy assignments will be graded as zero, and they will be announced in the Classroom.

Please do not forget to bring your laptops while coming to the assignment control!

In this assignment, you are expected to implement a queuing model for banking system. The queuing rules will be implemented by using linked lists data structure **in Java**. You should develop your application by using the proper Object Oriented Programing (OOP) principles.

Queuing Rules:
-The bank has four type of customers, which are labelled as *0, 1, 2, 3.*
-Customers with label *0* has the highest priority, then *1, 2*, and, *3* have the lower priorities, respectively.
E.g., Assume that the queue contains *0-0-0-2-3*, and a new customer with label 1 joins to the queue, the updated queue will be *0-0-0-**1**-2-3*.
-*When we consider the example above, if a customer with label 0 comes continuously, the customer with 1, 2 and 3 labels will never be processed. To prevent this situation, you need to use the limitations.*

Limitations
-Each type of customer has limitations. A customer with label 0 can join the queue at most 5 times in a row, label 1 can join 3 times in a row, label 2 can join 2 times in a row, and label 3 can join at once in a row.
Ex: Assume the queue contains *0-0-0-0-0-1-1-1-2-3,* when a customer with label 0 wants to join, the new queue will be *0-0-0-0-0-1-1-1-2-3-**0.***
Ex**:* Assume the queue contains *0-0-0-0-0-1-1-1-2-3,* when a customer with label 1 wants to join, the new queue will be *0-0-0-0-0-1-1-1-2-3-**1.
Ex: Assume the queue contains *0-0-0-0-0-1-1-1-2-3,* when two customers with label 2 want to join the queue consecutively, the new queue will be *0-0-0-0-0-1-1-1-2-**2**-3-**2.***

Processing a Customer:
While new customers are joining to the queue, the bank staff can process a customer in the queue.
Ex: Assume the queue contains *0-0-0-0-0-1-1-1-2-3,* when the bank staff finish to process one customer, the new queue will be *0-0-0-0-1-1-1-2-3.*

Note*:* After processing a customer in the queue like the above example, if a new customer with label **0** joins, the new queue **must** look like *0-0-0-0-1-1-1-2-3-**0.***

Considerations:
Priorities should be considered.
Ex: Assume the queue contains *0-0-0-0-0-1-2-3,* when two customers with labels 0 and 1 joins consecutively, the content updates: *0-0-0-0-0-1-2-3 -> 0-0-0-0-0-1-2-3-**0** -> 0-0-0-0-0-1-2-3-**0-1.***
Although there is one available slot for 1 between 0 and 2 as indicated above, label 0 has a priority over label 1, so label 1 must be placed after label 0.
Note: The system must keep the **names** of the customers at the same time with their **priorities in the data structure.**

The standard output of your program should appear as follows:

```
------ Bank System --------
Current Queue: Empty
1- Add a new customer to the queue
2- Process a customer
2
There is no customer in the queue
Current Queue: Empty
1- Add a new customer to the queue
2- Process a customer
1
Please enter the type of customer: 0
Name of the customer: Ali
Current Queue: 0
1- Add a new customer to the queue
2- Process a customer
1
Please enter the type of customer: 1
Name of the customer: Mehmet
Current Queue: 0-1
1- Add a new customer to the queue
2- Process a customer
// After a while (Some Additions Done Here)
Current Queue: 0-0-0-0-1-2-2
1- Add a new customer to the queue
2- Process a customer
2
Ali is processed-0
Current Queue: 0-0-0-1-2-2
1- Add a new customer to the queue
2- Process a customer
// After a while (Only Deletions Done Here)
Current Queue: 1-2-2
1- Add a new customer to the queue
2- Process a customer
2
Mehmet is processed-1
Current Queue: 2-2
1- Add a new customer to the queue
2- Process a customer
1
Please enter the type of customer: 0
Name of the customer: Ahmet
```

```
Current Queue: 0-2-2
1- Add a new customer to the queue
2- Process a customer
1
Please enter the type of customer: 0
Name of the customer: Veli
Current Queue: 0-2-2-0
1- Add a new customer to the queue
2- Process a customer
```

Grading Policy

Item	**Percentage**
Implementing a Linked List Data Structure	30%
Implementing a Queue with Priorities	30%
Limitations and Considerations	20%
Application of OOP Principles	20%