CSC 530/730 – Programming and Data Structure
Homework 3
(100 points)

Problems

In this project, we will implement priority queues ***using linked lists***, and then use a priority queue to help sort a given group of integer numbers into ***non-increasing*** order.

1) Implement class `PriorityQueueLinkedList` as follows:

```
Class PriorityQueueLinkedList
{
    private Node front;          // reference to the front

    public PriorityQueueLinkedList()
    {
        // your code comes here
    }

    public boolean isEmpty()
    {
        // your code comes here
    }

    public void enqueue(int k)
    {
        // your code comes here
    }

    public int dequeue()
    {
        // your code comes here
    }

    public int peekFront()
    {
        // your code comes here
    }
}
```

The class should only include five functions (<u>Do NOT implement any additional functions</u> in the class):

- `PriorityQueueLinkedList():` Creates an empty priority queue.
- `isEmpty()`: Returns true if the priority queue is empty.
- `enqueue(int k)`: Inserts a key `k` into the priority queue.
- `dequeue()`: Removes the front item from the priority queue and return its key; If the priority queue is empty and has no item to remove, returns -1.
- `peekFront()`: Reads the key of the front item in the priority queue and return it; Returns -1 if the priority queue is empty.

Let's make the following assumptions on the priority queues:

- All keys in the priority queue are non-negative.
- Duplicated keys are allowed in a priority queue.

Hint:

- When implementing class `PriorityQueueLinkedList`, you can borrow ideas from class `SortedLinkedList` in example project `Example08SortedLinkedList`

2) Write necessary statements in `main()` function to either allow users to enter keys or let the program randomly generate keys, and then sort them in non-increasing order by using a priority . Here are some examples when running the Homework 3 project.

```
Select from:
1. Read keys
2. Generate keys
3. Sort
0. Quit
1
Enter keys (negative to stop): 5 2 8 1 4 3 5 2 9 -1

Select from:
1. Read keys
2. Generate keys
3. Sort
0. Quit
3
9 8 5 5 4 3 2 2 1

Select from:
1. Read keys
2. Generate keys
3. Sort
0. Quit
2
Enter the number of keys to generate: 10
The following keys have been generated: 0 6 6 5 7 2 4 2 0 8

Select from:
1. Read keys
2. Generate keys
3. Sort
0. Quit
3
8 7 6 6 5 4 2 2 0 0

Select from:
1. Read keys
2. Generate keys
3. Sort
0. Quit
0
Thanks for using my program.
```

Note:

- For options 1 and 2 in the menu, the keys entered by the user or generated by the program should be stored into a `PriorityQueueLinkedList` object.
- When option 2 in the menu is chosen, the user needs to further specify the number of keys, *n*. Then the program will randomly generate *n* integer numbers as keys in range 0~(n-1) by using function call `nextInt(n)` of a `Random` object.
- For option 3 in the menu, your program should NOT directly traverse the entire sorted linked list and display the keys one by one. Instead, the sorting functionality should be implemented by calling functions `isEmpty()`, `dequeue()` of the `PriorityQueueLinkedList` object that stores the keys.

Submission

Compress the JAVA project folder into a *.zip* file and submit it on Blackboard.