Lab 9

CSE 274

There are two different types of Priority Queues: priority queue with fast-deQueuing and priority queue with fast-enQueuing. The priority Queue data structure that we learned in Lab 8 was of fast-deQueuing type. In a fast-deQueuing priority queue, packets are placed in the queue based on their priorities. Accordingly, the deQueue operation runs fast, as there is no need for searching for the packet with the highest priority. For a fast-deQueuing priority queue the `enQueue` and `deQueue` operations have time complexities of O(n) and O(1), respectively.
In this Lab, we implement a fast-enQueueing priority queue using linked lists. In such a priority queue the `enQueue` method adds the new packets at the end of the linked list. Accordingly, the enQueue operation runs fast with a time complexity of O(1). In contrast, the time complexity of the `deQueue` method is O(n) as the `deQueue` method should search for the packet with the highest priority in the queue to `return` back the `Data` inside the packet. More precisely, the `deQueue` method searches for the packet with the highest priority, starting at the first link of the linked list up to the last link of the link list. Once the packet with the highest priority is found, the `deQueue` method deletes the packet from the linked list and `return` back the `Data` inside the packet.
Import the `Application.java` file in Eclipse, run the application and make sure that the application runs without any error.

I. The PriorityQueueFastEnQueue Class

Copy the following block of code into the body of the `main` method, run the application and make sure that you see the expected output:

```
PriorityQueueFastEnQueue myQueue = new
PriorityQueueFastEnQueue();
myQueue.enQueue("A", 1);
myQueue.enQueue("B", 3);
myQueue.enQueue("C", 2);
myQueue.enQueue("D", 5);
myQueue.enQueue("E",4);
myQueue.displayQueue();
```

The expected output is printed below:

```
Data: A B C D E

Next: B C D E null

Previous: null A B C D
```

The `PriorityQueueFastEnQueue` class creates a priority queue data structure using a linked list. The linked list stores objects of class Packet.

II. The deQueue method

Develop a `deQueue` method for the `PriorityQueueFastEnQueue` class using the following logic:

```
Packet highPriorityPacket=first;

Packet current=first

while(current.next!=null)

      current=current.next

      if (highPriorityPacket.Priority < current.Priority)
            highPriorityPacket=current

String Data= highPriorityPacket.Data

if (highPriorityPacket==first)

      Delete the first link from the linked list
      return Data

if (highPriorityPacket==last)

      Delete the last link from the linked list
      return Data

Delete the highPriorityPacket from the linked list
return Data
```

Use the following block of code to test the developed method:

```
PriorityQueueFastEnQueue myQueue = new PriorityQueueFastEnQueue();

myQueue.enQueue("A", 1);
myQueue.enQueue("B", 3);
myQueue.enQueue("C", 2);
myQueue.enQueue("D", 5);
myQueue.enQueue("E",4);

while(!myQueue.isEmpty())
      System.out.println(myQueue.deQueue());
```

The expected output is printed below:

```
D
E
B
C
A
```

III. Submitting the Assignment

When submitting your response to this assignment, keep the block of code of Section II in the body of the `main` method.