

Oct 29, 20 14:34

PriorityQueue.java

Page 1/2

```

1 //Alan Stoloff
2 //Dr Benjamin
3 //csi220
4 //This is an implementation of a priority Queue
5 public class PriorityQueue <T extends Comparable<T>> implements PriorityQueueInterface<
   T>{
6     //initialize global variable
7     private Node header;
8
9     //constructor
10    public PriorityQueue(){
11        Node front=new Node();
12        Node rear=new Node();
13
14        front.prev=null;
15        front.next=rear;
16        rear.prev=front;
17        rear.next=null;
18
19        header=front;
20
21    }
22    //creates node class
23    private class Node{
24        public T data;
25        public Node prev;
26        public Node next;
27    }
28    //Adds data to the list based on priority
29    public void enqueue(T item){
30        Node ptr=header.next;
31        while(ptr.next!=null){
32            if(ptr.data.compareTo(item)<0){
33                Node add=new Node();
34                add.data=item;
35                add.prev=ptr.prev;
36                add.next=ptr;
37                ptr.prev.next=add;
38                ptr.prev=add;
39                return;
40            }
41            ptr=ptr.next;
42        }
43        Node temp=new Node();
44        temp.data=item;
45        temp.prev=ptr.prev;
46        temp.next=ptr;
47        ptr.prev.next=temp;
48        ptr.prev=temp;
49        return;
50    }
51    //removes the first thing in the list
52    public T dequeue(){
53        Node ptr=header.next;
54        if(ptr==null){
55            return null;
56        }
57        header.next=ptr.next;
58        return ptr.data;
59    }
60    //returns the first thing in the list
61    public T front(){
62        Node ptr=header.next;
63        if(ptr==null){
64            return null;
65        }
66        return ptr.data;
67    }
68    //determines if the list is empty

```

Oct 29, 20 14:34

PriorityQueue.java

Page 2/2

```
69     public boolean isEmpty(){
70         Node ptr=header.next;
71         while(ptr.next!=null){
72             ptr=ptr.next;
73         }
74         if(header.next.equals(ptr)){
75             return true;
76         }
77         return false;
78     }
79     //determines if the list is full
80     public boolean isFull(){
81         return false;
82     }
83     //String representation of the whole list and its data
84     public String toString(){
85         String last="Front\n";
86         Node ptr=header.next;
87         while(ptr.next!=null){
88             String temp=String.valueOf(ptr.data);
89             last=last+" "+temp;
90             ptr=ptr.next;
91         }
92         last=last+"\nrear";
93         return last;
94     }
95 }
```