```java
//Alan Stoloff
//Dr Benjamin
//csi220
//This is an implementation of a priority Queue
public class PriorityQueue <T extends Comparable<T>> implements PriorityQueueInterface<
T>{
    //initialize global variable
    private Node header;

    //constructor
    public PriorityQueue(){
        Node front=new Node();
        Node rear=new Node();

        front.prev=null;
        front.next=rear;
        rear.prev=front;
        rear.next=null;

        header=front;

    }
    //creates node class
    private class Node{
        public T data;
        public Node prev;
        public Node next;
    }
   //Adds data to the list based on priority
    public void enqueue(T item){
        Node ptr=header.next;
        while(ptr.next!=null){
            if(ptr.data.compareTo(item)<0){
                Node add=new Node();
                add.data=item;
                add.prev=ptr.prev;
                add.next=ptr;
                ptr.prev.next=add;
                ptr.prev=add;
                return;
            }
        ptr=ptr.next;
        }
        Node temp=new Node();
        temp.data=item;
        temp.prev=ptr.prev;
        temp.next=ptr;
        ptr.prev.next=temp;
        ptr.prev=temp;
        return;
    }
    //removes the first thing in the list
    public T dequeue(){
        Node ptr=header.next;
        if(ptr==null){
            return null;
        }
        header.next=ptr.next;
        return ptr.data;
    }
    //returns the first thing in the list
    public T front(){
        Node ptr=header.next;
        if(ptr==null){
            return null;
        }
        return ptr.data;
    }
    //determines if the list is empty
    public boolean isEmpty(){
        Node ptr=header.next;
        while(ptr.next!=null){
            ptr=ptr.next;
        }
        if(header.next.equals(ptr)){
            return true;
        }
        return false;
    }
    //determines if the list is full
    public boolean isFull(){
        return false;
    }
    //String representation of the whole list and its data
    public String toString(){
        String last="Front\n";
        Node ptr=header.next;
        while(ptr.next!=null){
            String temp=String.valueOf(ptr.data);
            last=last+" "+temp;
            ptr=ptr.next;
        }
        last=last+"\nrear";
        return last;
    }
}
```