

```
1  import java.util.*;
2
3  // implementing a Queue as a Doubly Linked List
4
5
6  class Node{
7
8      private int data;
9      private Node nextNodePtr;
10     private Node prevNodePtr;
11
12     public Node () {}
13
14     public void setData(int d){
15         data = d;
16     }
17
18     public int getData(){
19         return data;
20     }
21
22     public void setNextNodePtr(Node nodePtr){
23         nextNodePtr = nodePtr;
24     }
25
26     public Node getNextNodePtr(){
27         return nextNodePtr;
28     }
29
30     public void setPrevNodePtr(Node nodePtr){
31         prevNodePtr = nodePtr;
32     }
33
34     public Node getPrevNodePtr(){
35         return prevNodePtr;
36     }
37 }
38
39
40 class Queue{
41
42     private Node headPtr;
43     private Node tailPtr;
44
45     public Queue(){
46         headPtr = new Node();
47         tailPtr = new Node();
48         headPtr.setNextNodePtr(null);
49         tailPtr.setPrevNodePtr(null);
50     }
51
52     public Node getHeadPtr(){
53         return headPtr;
54     }
55
56     public Node getTailPtr(){
57         return tailPtr;
58     }
59
60     public boolean isEmpty(){
61
62         if (headPtr.getNextNodePtr() == null)
63             return true;
64
```

```

65     return false;
66 }
67
68
69 public void enqueue(int data){
70
71     Node newNodePtr = new Node();
72     newNodePtr.setData(data);
73     newNodePtr.setNextNodePtr(null);
74
75     Node lastNodePtr = tailPtr.getPrevNodePtr();
76
77     if (lastNodePtr == null){
78
79         headPtr.setNextNodePtr(newNodePtr);
80         newNodePtr.setPrevNodePtr(null);
81
82     }
83     else{
84
85         lastNodePtr.setNextNodePtr(newNodePtr);
86         newNodePtr.setPrevNodePtr(lastNodePtr);
87
88     }
89
90     tailPtr.setPrevNodePtr(newNodePtr);
91
92 }
93
94
95 public int dequeue(){
96
97     Node firstNodePtr = headPtr.getNextNodePtr();
98     Node nextNodePtr = null;
99
100    int poppedData = -100000; //empty queue
101
102    if (firstNodePtr != null){
103        nextNodePtr = firstNodePtr.getNextNodePtr();
104        poppedData = firstNodePtr.getData();
105    }
106    else
107        return poppedData;
108
109    if (nextNodePtr != null){
110        nextNodePtr.setPrevNodePtr(null);
111        headPtr.setNextNodePtr(nextNodePtr);
112    }
113    else{
114        headPtr.setNextNodePtr(null);
115        tailPtr.setPrevNodePtr(null);
116    }
117
118    return poppedData;
119
120 }
121
122
123 public int peek(){
124
125     Node firstNodePtr = headPtr.getNextNodePtr();
126
127     if (firstNodePtr != null)
128         return firstNodePtr.getData();

```

```

129         else
130             return -100000; //empty queue
131
132     }
133
134
135 };
136
137 class DoublyLinkedList{
138
139     public static void main(String[] args){
140
141         Queue queue = new Queue();
142
143         Scanner input = new Scanner(System.in);
144
145         int queueSize;
146
147         System.out.print("Enter the number of elements you want to enqueue: ");
148         queueSize = input.nextInt();
149
150         Random randGen = new Random(System.currentTimeMillis());
151
152         int maxValue;
153
154         System.out.print("Enter the maximum value for an element: ");
155         maxValue = input.nextInt();
156
157         System.out.print("Elements enqueued: ");
158         for (int i = 0; i < queueSize; i++){
159
160             int value = randGen.nextInt(maxValue);
161             queue.enqueue(value);
162             System.out.print(value + " ");
163         }
164
165         System.out.println();
166
167         System.out.print("Elements Dequeued: ");
168         while (!queue.isEmpty()){
169
170             System.out.print(queue.dequeue() + " ");
171         }
172
173         System.out.println();
174
175     }
176
177 }

```

```

Enter the number of elements you want to enqueue: 10
Enter the maximum value for an element: 50
Elements enqueued: 44 16 10 0 13 11 3 7 18 46
Elements Dequeued: 44 16 10 0 13 11 3 7 18 46

```