

Credit  
3110  
Assignment  
DoublyLinkedList

How has your program changed from planning to coding to now? Please explain?

```
{
    public static void main(String[] args)
    {
        DoublyLinkedList list = new DoublyLinkedList();

        list.addAtFront("Raj");
        list.addAtEnd("Jeni");
        list.addAtEnd("Yuri");

        System.out.println("display forward");
        list.displayList();

        System.out.println("display reverse");
        list.displayReverseList();

        System.out.println("remove jeni");
        list.remove("Jeni");

        System.out.println("display forward");
        list.displayList();

        System.out.println("display reverse");
        list.displayReverseList();
    }
}

13 public class DoublyLinkedList
14 {
15     private Node head;
16     private Node tail;
17
18     public DoublyLinkedList()
19     {
20         head = null;
21         tail = null;
22     }
23
24     // adds node to front
25     public void addAtFront(String value)
26     {
27         Node newNode = new Node(value);
28
29         if (head == null)
30         {
31             head = newNode;
32             tail = newNode;
33         }
34         else
35         {
36             newNode.next = head;
37             head.prev = newNode;
38             head = newNode;
39         }
40     }
41
42     // adds node to end
43     public void addAtEnd(String value)
44     {
45         Node newNode = new Node(value);
46
47         if (tail == null)
48         {
49             head = newNode;
50             tail = newNode;
51         }
52     }
53 }
```

```

46     Node newNode = new Node(value);
47
48     if (tail == null)
49     {
50         head = newNode;
51         tail = newNode;
52     }
53     else
54     {
55         tail.next = newNode;
56         newNode.prev = tail;
57         tail = newNode;
58     }
59 }
60
61 // removes first matching value
62 public void remove(String value)
63 {
64     Node current = head;
65
66     while (current != null)
67     {
68         if (current.data.equals(value))
69         {
70             if (current == head)
71             {
72                 head = current.next;
73             }
74             else
75             {
76                 current.prev.next = current.next;
77             }
78
79             if (current == tail)
80             {
81                 tail = current.prev;
82             }
83             else
84             {
85                 current.next.prev = current.prev;
86             }
87         }
88         current = current.next;
89     }
90 }
91
92 // displays list forward
93 public void displayList()
94 {
95     Node current = head;
96
97     while (current != null)
98     {
99         System.out.print(current.data + " ");
100         current = current.next;
101     }
102
103     System.out.println();
104 }
105
106 // displays list backward
107 public void displayReverseList()
108 {
109     Node current = tail;
110
111     while (current != null)
112     {
113         System.out.print(current.data + " ");
114         current = current.prev;
115     }
116
117     System.out.println();
118 }
119
120 // node class
121 private class Node
122 {
123     private String data;
124     private Node next;
125     private Node prev;

```

```
122 private class Node
123 {
124     private String data;
125     private Node next;
126     private Node prev;
127
128     public Node(String d)
129     {
130         data = d;
131         next = null;
132         prev = null;
133     }
134 }
135 }
136
```

---