Aufgabe 2

Gegeben sei die folgende Java-Implementierung einer doppelt-verketteten Liste.

```
3  class DoubleLinkedList {
4    private Item head;
5
6    public DoubleLinkedList() {
7      head = null;
8    }
9
10   public Item append(Object val) {
11     if (head == null) {
12       head = new Item(val, null, null);
13       head.prev = head;
14       head.next = head;
15     } else {
16       Item item = new Item(val, head.prev, head);
17       head.prev.next = item;
18       head.prev = item;
19     }
20     return head.prev;
21   }
22
23   public Item search(Object val) {
24
25     Item aktuell = search(val);
26     if (aktuell != null) {
27       if (head.next.equals(head)) {
28         head = null;
29       } else {
30         if (aktuell.equals(head)) {
31           head = aktuell.next;
32         }
33         aktuell.prev.next = aktuell.next;
34         aktuell.next.prev = aktuell.prev;
35       }
36     }
37
38     Item item = null;
39     if (head != null) {
40       item = head;
41       while (!item.next.equals(head)) {
42         if (item.val == val) {
43           return item;
44         }
45         item = item.next;
46       }
47     }
48     return item;
49   }
50
51   public void delete(Object val) {
52     Item item = search(val);
53     if (item != null) {
54       if (head.next.val == head.val) {
55         head = null;
56       } else {
57         if (item.val == head.val) {
58           head = item.next;
59         }
```

```
60            item.prev.next = item.next;
61            item.next.prev = item.prev;
62          }
63        }
64      }
65
66      class Item {
67        private Object val;
68        private Item prev;
69        private Item next;
70
71        public Item(Object val, Item prev, Item next) {
72          this.val = val;
73          this.prev = prev;
74          this.next = next;
75        }
76      }
77
78      public void searchAndPrint(Object val) {
79        Item item = search(val);
80        System.out.println(item.val);
81      }
82
83      public void printAll() {
84        System.out.print("All items: ");
85        if (head != null) {
86          System.out.print(head.val + " ");
87          Item item = head;
88
89          while (!item.next.equals(head)) {
90            System.out.print(item.next.val + " ");
91            item = item.next;
92          }
93        }
94        System.out.println();
95      }
96
97      public static void main(String[] args) {
98        DoubleLinkedList list = new DoubleLinkedList();
99        list.append("a");
100       list.append("b");
101       list.append("c");
102       list.printAll();
103
104       System.out.println("Test search");
105
106       list.searchAndPrint("a");
107       list.searchAndPrint("b");
108       list.searchAndPrint("c");
109
110       list.delete("a");
111       list.printAll();
112     }
113   }
```

(a) Skizzieren Sie den Zustand der Datenstruktur nach Aufruf der folgenden Befehlssequenz. Um Variablen mit Zeigern auf Objekte darzustellen, können Sie mit dem Variablennamen beschriftete Pfeile verwenden.

```
1  DoubleLinkedList list = new DoubleLinkedList();
2  list.append("a");
3  list.append("b");
```

```
4  list.append("c");
```

(b) Implementieren Sie in der Klasse `DoubleLinkedList` die Methode `search`, die zu einem gegebenen Wert das Item der Liste mit dem entsprechenden Wert, oder `null` falls der Wert nicht in der Liste enthalten ist, zurückgibt.

(c) Implementieren Sie in der Klasse `DoubleLinkedList` die Methode `delete`, die das erste Vorkommen eines Wertes aus der Liste entfernt. Ist der Wert nicht in der Liste enthalten, terminiert die Methode „stillschweigend“, d. h. ohne Änderung der Liste und ohne Fehler- meldung. Sie dürfen die Methode `search` aus Teilaufgabe b) verwenden, auch wenn Sie sie nicht implementiert haben.

(d) Beschreiben Sie die notwendigen Änderungen an der Datenstruktur und an den bisherigen Implementierungen, um eine Methode `size`, die die Anzahl der enthaltenen Items zurück gibt, mit Laufzeit $\mathcal{O}(1)$ zu realisieren.