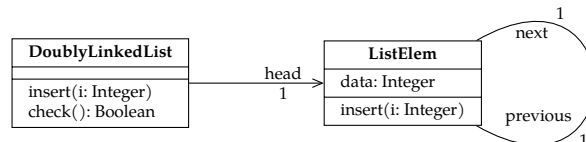


Aufgabe 14: Listen

Betrachten Sie folgendes Klassendiagramm, das doppelt-verkettete Listen spezifiziert. Die Assoziation `head` zeigt auf das erste Element der Liste. Die Assoziationen `previous` und `next` zeigen auf das vorherige bzw. folgende Element.



Implementieren Sie die doppelt-verketteten Listen in einer geeigneten objektorientierten Sprache (z. B. Java oder C++), das heißt:

- (a) Implementieren Sie die Klasse `ListElem`. Die Methode `insert` ordnet eine ganze Zahl `i` in eine aufsteigend geordnete doppelt-verkettete Liste `l` an die korrekte Stelle ein. Sei z. B. das Objekt `l` eine Repräsentation der Liste `[0, 2, 2, 6, 8]` dann liefert `l.insert(3)` eine Repräsentation der Liste `[0, 2, 2, 3, 6, 8]`.

```

3  public class ListElem {
4      private int data;
5      private ListElem previous;
6      private ListElem next;
7
8      public ListElem(int i) {
9          data = i;
10     }
11
12     public ListElem() {
13     }
14
15     public void insert(int i) {
16         ListElem newElement = new ListElem(i);
17         if (i <= data) {
18             if (previous != null) {
19                 newElement.next = this;
20                 newElement.previous = previous;
21                 previous.next = newElement;
22                 previous = newElement;
23             } else {
24                 newElement.next = this;
25                 previous = newElement;
26             }
27         } else {
28             if (next != null) {
29                 next.insert(i);
30             } else {
31                 newElement.previous = this;
32                 next = newElement;
33             }
34         }
35     }
36 }
  
```

```

35     }
36
37     public ListElem getPrevious() {
38         return previous;
39     }
40
41     public ListElem getNext() {
42         return next;
43     }
44
45     public int getData() {
46         return data;
47     }
48 }

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/ListElem.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/ListElem.java)

- (b) Implementieren Sie die Klasse `DoublyLinkedList`, wobei die Methode `insert` eine Zahl `i` in eine aufsteigend geordnete Liste einordnet. Die Methode `check` überprüft, ob eine Liste korrekt verkettet ist, d.h. für jedes `ListElem`-Objekt `o`, das über den `head` der Liste erreichbar ist, der Vorgänger des Nachfolgers von `o` gleich `o` ist.

```

3  public class DoublyLinkedList {
4      private ListElem head;
5
6      public DoublyLinkedList() {
7      }
8
9      public void insert(int i) {
10         if (head != null) {
11             // Immer einen neue Zahl einfügen, nicht nur wenn die Zahl kleiner ist
12             // ↳ als head.
13             head.insert(i);
14             // Es muss kleiner gleich heißen, sonst können mehrer gleiche Zahlen
15             // ↳ am Anfang
16             // nicht eingefügt werden.
17             if (i <= head.getData()) {
18                 head = head.getPrevious();
19             }
20         } else {
21             head = new ListElem(i);
22         }
23     }
24
25     public boolean check() {
26         ListElem current = head;
27         while (current.getNext() != null) {
28             if (current.getNext().getPrevious() != current) {
29                 return false;
30             } else {
31                 current = current.getNext();
32             }
33         }
34         return true;
35     }
36
37     public ListElem getHead() {
38         return head;
39     }
40 }

```

```
37     }
38
39     public static void main(String[] args) {
40         DoublyLinkedList list = new DoublyLinkedList();
41         // int[] numbers = new int[] { 1 };
42         // int[] numbers = new int[] { 1, 1, 1, 1, };
43         // int[] numbers = new int[] { 1, 1, 1, 2, };
44         // int[] numbers = new int[] { 2, 1, 1, 1, };
45         // int[] numbers = new int[] { 2, 1 };
46         int[] numbers = new int[] { 0, 2, 2, 6, 8, 4 };
47         for (int number : numbers) {
48             list.insert(number);
49         }
50         list.insert(3);
51
52         ListElem current = list.getHead();
53         while (current.getNext() != null) {
54             System.out.println(current.getData());
55             current = current.getNext();
56         }
57     }
58 }
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/DoublyLinkedList.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/DoublyLinkedList.java)