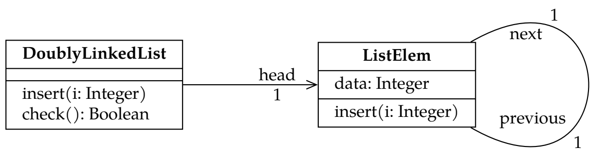
Aufgabe 14: Listen

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]`.

```
public class ListElem {
  private int data;
  private ListElem previous;
  private ListElem next;

  public ListElem(int i) {
    data = i;
  }

  public ListElem() {
  }

  public void insert(int i) {
    ListElem newElement = new ListElem(i);
    if (i <= data) {
      if (previous != null) {
        newElement.next = this;
        newElement.previous = previous;
        previous.next = newElement;
        previous = newElement;
      } else {
        newElement.next = this;
        previous = newElement;
      }
    } else {
      if (next != null) {
        next.insert(i);
      } else {
        newElement.previous = this;
        next = newElement;
      }
    }
```

```java
  }

  public ListElem getPrevious() {
    return previous;
  }

  public ListElem getNext() {
    return next;
  }

  public int getData() {
    return data;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/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. ob für jedes ListElem-Objekt o, das über den head der Liste erreichbar ist, der Vorgänger des Nachfolgers von o gleich o ist.

```java
public class DoublyLinkedList {
  private ListElem head;

  public DoublyLinkedList() {
  }

  public void insert(int i) {
    if (head != null) {
      // Immer einen neue Zahl einfügen, nicht nur wenn die Zahl kleiner ist
      ↪  als head.
      head.insert(i);
      // Es muss kleiner gleich heißen, sonst können mehrer gleiche Zahlen
      ↪  am Anfang
      // nicht eingefügt werden.
      if (i <= head.getData()) {
        head = head.getPrevious();
      }
    } else {
      head = new ListElem(i);
    }
  }

  public boolean check() {
    ListElem current = head;
    while (current.getNext() != null) {
      if (current.getNext().getPrevious() != current) {
        return false;
      } else {
        current = current.getNext();
      }
    }
    return true;
  }

  public ListElem getHead() {
    return head;
  }

  public static void main(String[] args) {
    DoublyLinkedList list = new DoublyLinkedList();
    // int[] numbers = new int[] { 1 };
    // int[] numbers = new int[] { 1, 1, 1, 1, };
    // int[] numbers = new int[] { 1, 1, 1, 2, };
    // int[] numbers = new int[] { 2, 1, 1, 1, };
    // int[] numbers = new int[] { 2, 1 };
    int[] numbers = new int[] { 0, 2, 2, 6, 8, 4 };
    for (int number : numbers) {
      list.insert(number);
    }
    list.insert(3);

    ListElem current = list.getHead();
    while (current.getNext() != null) {
      System.out.println(current.getData());
      current = current.getNext();
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/DoublyLinkedList.java`