Universität der Bundeswehr München
Elektrotechnik und Technische Informatik
Oswald / Baumann

Wintertrimester 2021
Praktikum 06
Abgabe: 22.02.-23.02.2021

Grundlagen der Programmierung

Programmieren mit Listen

Aufgabe 1: Einfach verkettete Liste

Gegeben ist Ihnen die Klasse `SLLInt`, die eine Sequenz als einfach verkettete Liste darstellt.

```
public class SLLInt {     // SLLInt: Single Linked List
    public int element;   // aktuelles Element
    public SLLInt next;   // Referenz auf den Rest der Liste

    public SLLInt(int e, SLLInt n) {
        element = e;
        next = n;
    }
    public SLLInt(int e) {
        element = e;
        next = null;
    }
}
```

`SLLInt` ist ein Listenelement mit Elementen vom Datentyp `int`, die im Objektattribut `element` gespeichert werden. Im Objektattribut `next` ist der Verweis auf den Nachfolger in der Liste gespeichert. Die Abbildung 1 zeigt eine schematische Darstellung einer Liste.

Verzichten Sie in den folgenden Aufgaben auf die in der Vorlesung besprochenen Klassenmethoden `empty`, `isEmpty`, `make`, `first` und `rest`. Verwenden Sie statt dessen den direkten Zugriff auf die Attribute `element` und `next` oder den Konstruktor.

Schreiben Sie die folgenden Klassenmethoden in einer Klasse `SLLIntOperation`. Testen Sie jede dieser Klassenmethoden mit einer leeren, einer ein-elementigen und einer mindestens zwei-elementigen Sequenz. Falls die Klassenmethode mehr als eine Sequenz als Parameter besitzt, dann testen Sie die verschiedenen Kombinationen aus einer leeren, ein-elementigen und mindestens zwei-elementigen Sequenz.

a) Schreiben Sie eine rekursive Klassenmethode
`public static String getString (SLLInt seq)`,
die die Sequenz als String in der Form e_1`:`e_2`:...:`e_n zurückgibt. Für eine Sequenz mit den Elementen 1, 2, 13 und 5 wird also der folgende `String` erzeugt: `1:2:13:5`.

b) Schreiben Sie eine nicht-rekursive Klassenmethode
`public static int getLength (SLLInt seq)`,
die die Länge der Sequenz `seq` zurück gibt.

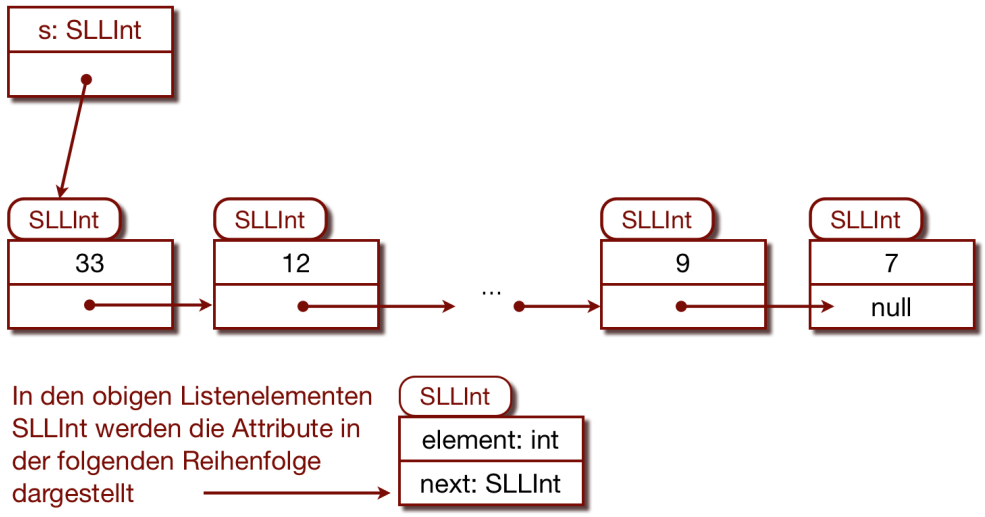

Abbildung 1: Einfach verkettete Liste

c) Schreiben Sie eine nicht-rekursive Klassenmethode
`public static boolean isSorted (SLLInt seq)`,
die zurückgibt, ob die eingegeben Sequenz aufsteigend sortiert ist.

d) Schreiben Sie eine nicht-rekursive Klassenmethode
`public static SLLInt insert (int value, SLLInt seq)`,
die ein Listenelement mit dem Wert `value` erzeugt und das Element dann in eine aufsteigend sortierte Eingabesequenz `seq` einsortiert. Der resultierende Sequenz wird zurückgegeben.,

Testen Sie Ihre Klassenmethode auch mit den Fällen, dass der Wert `value` kleiner oder größer als alle Werte der Eingabesequenz ist. Natürlich sollten Sie auch den Fall testen, dass der Wert `value` zwischen dem ersten und letzten Element einsortiert werden muss.

e) Schreiben Sie eine rekursive Klassenmethode
`public static boolean isEqualRekursiv(SLLInt seq1, SLLInt seq2)`
Die Klassenmethode soll `true` zurückgeben, wenn die beiden einfach verketten Listen `seq1` und `seq2` gleich sind. Ansonsten soll die Methode `false` liefern. Die beiden Listen dürfen in der Klassenmethode nicht verändert werden.

f) Schreiben Sie eine nicht-rekursive Klassenmethode
`public static boolean isEqualIterativ(SLLInt seq1, SLLInt seq2)`
Die Klassenmethode soll `true` zurückgeben, wenn die beiden einfach verketten Listen `seq1` und `seq2` gleich sind. Ansonsten soll die Methode `false` liefern. Die beiden Listen dürfen in der Klassenmethode nicht verändert werden.

g) Schreiben Sie eine nicht-rekursive Klassenmethode
`public static SLLInt concat (SLLInt seq1, SLLInt seq2)`,
die die Sequenz `seq2` an die Sequenz `seq1` angehängt zurückgibt. Die Klassenmethode soll keine neuen Listenelemente erzeugen.

Aufgabe 2: Doppelt verkettete Liste mit Listenkopf

Gegeben sind Ihnen die beiden Klassen `DLLInt` und `SeqByDLL`, die eine Sequenz als doppelt verkettete Liste darstellt.

```
public class DLLInt {     // DLLInt: Double Linked List
    public DLLInt prev;   // Referenz auf das vorherige Element
    public int element;   // aktuelles Element
    public DLLInt next;   // Referenz auf den Rest der Liste

    public DLLInt(DLLInt p, int e, DLLInt n) {
        prev = p;
        element = e;
        next = n;
    }
}

public class SeqByDLL {
    public DLLInt head = null;   // Anfang der Liste
    public DLLInt tail = null;   // Ende der Liste

    public SeqByDLL(DLLInt h, DLLInt t) {
        head = h;
        tail = t;
    }
}
```

`DLLInt` ist ein Listenelement mit Elementen vom Datentyp `int`, die im Objektattribut `element` gespeichert werden. Im Objektattribut `prev` wird ein Verweis auf den Vorgänger in der Liste und in `next` ein Verweis auf den Nachfolger in der Liste gespeichert. `SeqByDLL` ist der Listenkopf. `head` speichert das erste Element und `tail` speichert das letzte Element der Liste. Die Abbildung 2 zeigt eine schematische Darstellung so einer Liste.

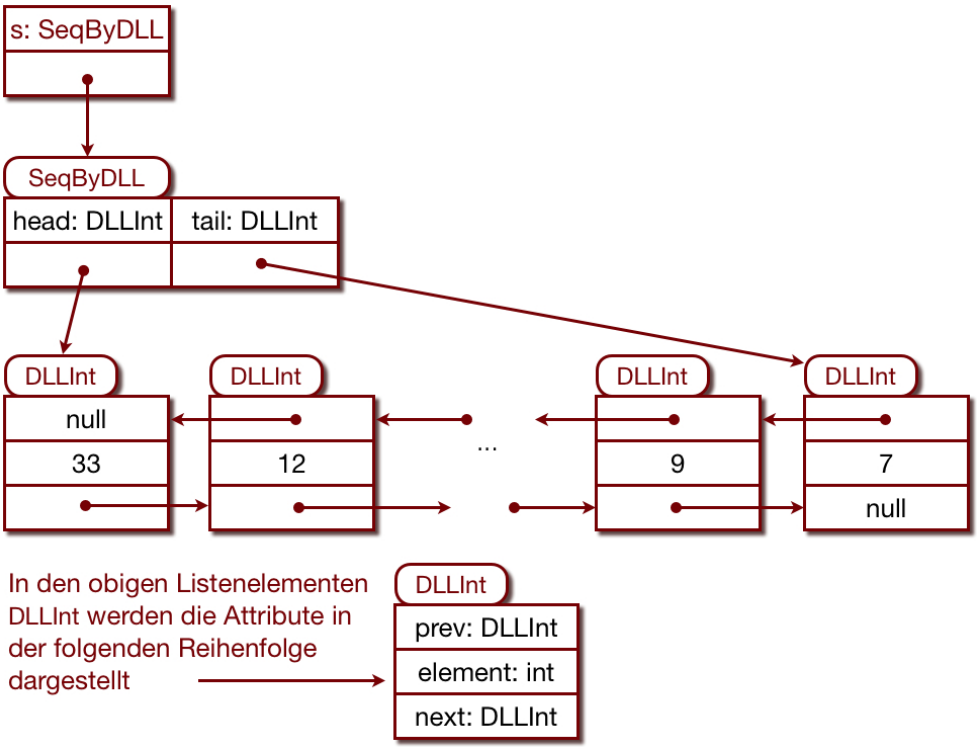

Abbildung 2: Doppelt verkettete Liste

Schreiben Sie eine Klasse `SeqByDLLOperation` mit den folgenden Klassenmethoden:

a) Schreiben Sie eine nicht-rekursive Klassenmethode
`public static String getString (SeqByDLL seq, boolean inverted)`,
die die Sequenz als `String` in der Form `->` $(e_1 : e_2 : ... : e_n)$ bzw. $(e_n : e_{n-1} : ... : e_1)$ `<-` zurückgibt. Wird der Wert `inverted` auf `false` gesetzt wird die Sequenz von Anfang bis Ende ausgegeben, wird der Wert auf `true` gesetzt, dann wird die Sequenz von hinten nach vorne ausgegeben. Für eine Sequenz mit den Elementen 1, 2, 13 und 5 wird also der folgende `String` erzeugt: `-> (1:2:13:5)`, falls `inverted` gleich `false` ist. Im Fall von `inverted` gleich `true` würde `(5:13:2:1) <-` ausgegeben.

   Testen Sie Ihre Klassenmethode mit einer leeren, einer ein- und einer zwei-elementigen Sequenz.

b) Schreiben Sie eine nicht-rekursive Klassenmethode
`public static SeqByDLL insert (int value, SeqByDLL seq)`,
die ein Listenelement mit dem Wert `value` erzeugt und das Element dann in die sortierte Eingabesequenz `seq` einsortiert. Die bisherigen Listenelemente `DLLInt` in der Sequenz sollen erhalten bleiben. Sie können davon ausgehen, dass die Eingabesequenz `seq` aufsteigend sortiert ist.

   Testen Sie Ihre Klassenmethode mit verschiedenen Eingabeparametern, z.B. mit einer leeren Eingabesequenz oder den Fällen, dass der Wert `value` kleiner oder größer als alle Werte der Eingabesequenz ist. Natürlich sollten Sie auch den Fall testen, dass der Wert `value` zwischen dem ersten und letzten Element einsortiert werden muss.