technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 14
Lars Hildebrand

Praktikum zu

Einführung in die Informatik für LogWings, WiMas und MedPhys

Wintersemester 2022/23

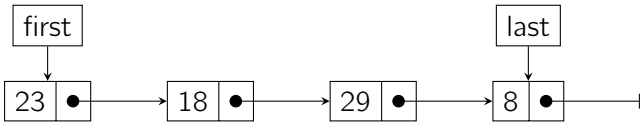
Übungsblatt 13

Besprechung:
30.01–03.02.2023
(KW 5)

Vorbereitende Aufgaben

Aufgabe 13.1: Einfach verkettete Liste

Folgendes Diagramm zeigt eine verkettete Liste mit Kopf (first) und Fußzeiger (last):



Führen Sie auf dieser Liste folgende drei Operationen hintereinander aus und repräsentieren Sie dabei die nach jedem Schritt entstandene Liste wie oben:

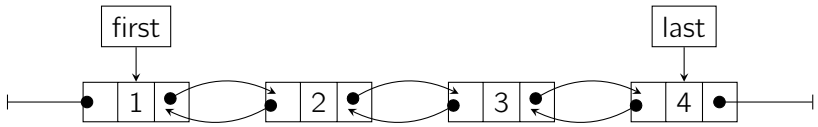
a) Entfernen des Elements mit der Zahl 29

b) Anfügen der Zahl 12 an das Ende der Liste

c) Entfernen des Elements mit der Zahl 23

Aufgabe 13.2: Doppelt verkettete Listen

Folgendes Diagramm zeigt eine doppelt verkettete Liste mit Kopf (first) und Fußzeiger (last):



Führen Sie auf dieser Liste folgende drei Operationen hintereinander aus und repräsentieren Sie dabei die nach jedem Schritt entstandene Liste wie oben:

a) Anfügen der Zahl 5 an das Ende der Liste

b) Entfernen des Elements mit der Zahl 2

c) Entfernen des Elements mit der Zahl 5

Präsenzaufgaben

Aufgabe 13.3: Listen – Implementierung 1

In dieser Aufgabe wollen wir eine einfach verkettete Liste implementieren, die **int**-Werte verwalten kann. Das erste Element hat dabei, wie bei Arrays, den Index 0, das zweite den Index 1, usw.

a) Erstellen Sie zwei neue Klassen **List** und **Element** im Paket **blatt13**. Die Klasse **List** besitzt zwei Referenzen auf Elementobjekte mit den Namen **first** für den Kopf und **last** für den Fuß. Im Konstruktor sollen beide auf **null** gesetzt werden. Der Konstruktor besitzt keine Parameter.

   Die Klasse **Element** soll zwei Attribute besitzen: Einen Zahlenwert **value** vom Typ **int** und eine Referenz **next** auf das nächste Element. Der Konstruktor eines Elements muss entsprechend einen Wert entgegennehmen und sein **value**-Attribut setzen. Die Referenz auf das nächste Element ist bei Instanziierung zunächst immer **null**.

b) Die Klasse **Element** soll eine Methode **setNext** besitzen, die das nächste Element auf ein übergebenes Element setzt, eine Methode **getNext**, die dieses Element zurückgibt und eine Methode **getValue**, die den gespeicherten Wert zurückgibt.

Die Klasse **List** soll dagegen eine Methode **add** besitzen, die eine Zahl entgegennimmt, um ein neues Element zu erzeugen und an das Ende der Liste anzufügen. Beachten Sie dabei die Referenzen des Kopfes und des Fußes entsprechend anzupassen. Ebenso muss die **next**-Referenz des aktuell letzten Elementes angepasst werden, falls es existiert.

c) Es fehlt die Möglichkeit, Daten aus der Liste auszulesen. Implementieren Sie eine Methode **get**, die einen Index *n* entgegennimmt und den Wert des Elementes mit dem Index *n* aus der Liste zurückgibt. Gehen Sie davon aus, dass nur auf vorhandene Indizes zugegriffen wird.

d) Testen Sie Ihre Implementierung an dieser Testklasse:

```
1  package blatt13;
2
3  public class ListTest1 {
4      public static void main(String[] args) {
5          List list = new List();
6
7          list.add(47);
8          list.add(237);
9          list.add(9);
10
11         if (list.get(0) != 47
12                 || list.get(1) != 237
13                 || list.get(2) != 9) {
14             System.out.println("There are definitely errors in your code");
15         } else {
16             System.out.println("All values seem to be in their place");
17         }
18     }
19 }
```

Aufgabe 13.4: Listen – Implementierung 2

a) Implementieren Sie eine Methode **remove**, die einen Index *n* entgegennehmen und das *n*-te Element aus der Liste entfernen soll. Dabei soll der Wert des entfernten Objektes zurückgegeben werden. Beachten Sie dabei, passenden Referenzen zwischenzuspeichern, um die Verkettung der Liste nicht zu unterbrechen.

b) Wie kann man die Größe einer Liste erfahren? Welche Vorteile gibt es bei den jeweiligen Vorgehensweise?

c) Fügen Sie der Klasse **List** das Attribut **size** hinzu, in dem die Größe der Liste gespeichert werden soll. Implementieren Sie zudem eine Methode **getSize**, die die Größe ihrer Liste zurückgibt. Passen Sie gegebenenfalls Methoden an, die die Größe ihrer Liste ändern.

d) Testen Sie Ihre Implementierung an dieser Testklasse:

```
1  package blatt13;
2
3  public class ListTest2 {
4      public static void main(String[] args) {
5          List list = new List();
6
7          list.add(47);
8          list.add(237);
9          list.add(9);
10         list.add(9001);
11
12         if (list.getSize() != 4) {
13             System.out.println("You have to work on your size function");
14         }
15
16         if (list.remove(0) != 47
17                 || list.remove(1) != 9
18                 || list.remove(1) != 9001
19                 || list.get(0) != 237) {
20             System.out.println("There are definitely errors in your code");
21         } else {
22             System.out.println("Removing seems to work");
23         }
24
25         if (list.getSize() != 1) {
26             System.out.println("You have to work on your size function");
27         }
28     }
29 }
```

Ergänzende Aufgaben

Aufgabe 13.5: Liste von Objekten

Schreiben Sie eine alternative Implementierung der Liste, die in der Lage ist, Objekte wie z. B. **Vehicle** aus Blatt 10 zu verwalten.