TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einführung in die Informatik 2 für Ingenieure (MSE)*

Christoph Anneser (anneser@in.tum.de), Simon Ellmann (ellmann@in.tum.de)
http://db.in.tum.de/teaching/ss24/ei2/

Blatt Nr. 4
Dieses Blatt wird am Montag, den 13.05.2024 besprochen.
Zugehörige Vorlesungsaufzeichnung: Vorlesung 5

Aufgabe 1: Einfach verkettete Liste ⌨

Implementieren Sie eine generische, einfach verkettete Liste. Nutzen Sie hierfür die folgende Vorlage, die Sie auf gitlab.db.in.tum.de finden. Klicken Sie hierfür auf das Download-Symbol und laden Sie sich das Projekt als Zip-Datei herunter. Entpacken Sie die Datei und öffnen Sie sie in Ihrer IDE. Machen Sie sich mit den Dateien und der Struktur des Projekts vertraut:

```
LinkedList
├── src
│   ├── main
│   │   └── java
│   │       ├── LinkedList.java // todo: Ihre Implementierung
│   │       └── List.java // Interface
│   └── test
│       └── java
│           └── LinkedListTest.java // die Tests
└── pom.xml // Maven Konfigurationsdatei, mehr Infos maven.apache.org
```

Weitere Informationen erhalten Sie in der Zentralübung.

Implementieren Sie nun die mit todo gekennzeichneten Methoden in der Klasse `LinkedList.java`.

Tipp 1: Erarbeiten Sie sich vor der Implementierung der Methoden ein Grundverständnis, wie die Liste funktionieren soll. Erstellen Sie z.B. Skizzen von Listen, wie Sie nach bestimmten Operationen im Speicher aussehen könnte (vgl. Abbildung 1).

Tipp 2: Benutzen Sie für Vergleiche stets die `equals`-Methode. Eine generische Liste kann z.B. auch Punkt-Objekte speichern.

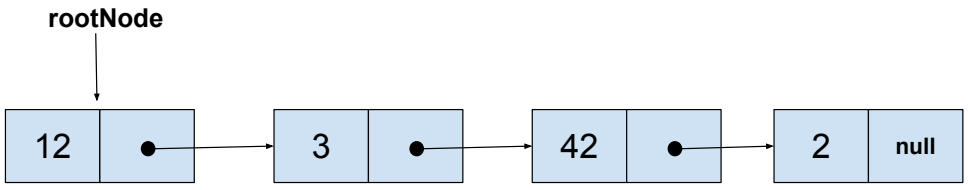

Abbildung 1: Interner Zustand einer Integer speichernden Liste, nachdem `list.add(12)`, `list.add(3)`, `list.add(42)`, `list.add(2)` aufgerufen wurden.

Aufgabe 2: Don't repeat yourself (DRY)

Auf der Webseite[1] finden Sie die Klasse `Warenhaus.java`.

a) Nennen Sie mindestens zwei Gründe, warum die Modellierung nicht optimal ist – beispielsweise, wenn sich der Mehrwertsteuersatz mal wieder ändern sollte.

b) Implementieren Sie den Aufzählungstyp `Mehrwertsteuersatz`, der zwei Werte für den normalen und den vergünstigten Mehrwertsteuersatz hat.

c) Ziehen Sie die Gemeinsamkeiten der Produkte in eine gemeinsame Oberklasse `Produkt`. Führen Sie dafür eine Methode ein, die für ein Produkt angibt, ob die vergünstigte Mehrwertsteuer anwendbar ist. Verwenden Sie diese Methode bei der Preisberechnung.

d) Führen Sie die Klasse `Einkaufskorb` ein, die eine Menge von Produkten verwaltet und deren Gesamtpreis bestimmen kann. Verwenden Sie diese Klasse in der main-Methode des Warenhauses.

Aufgabe 3: Werte vs. Objekte

In dieser Aufgabe schauen wir uns noch einmal den Unterschied zwischen Werten und Objekten an. Überlegen Sie sich, was die Ausgabe sein sollte und warum. Die Java Datei können Sie auch auf der Vorlesungswebsite herunterladen: `https://db.in.tum.de/teaching/ss24/ei2/material/WerteVsObjekte.java`

```
1  class PersonA {
2    int alter;
3    public PersonA(int alter) {
4      this.alter = alter;
5    }
6  }
7
8  class PersonB {
9    Alter alter;
10   public PersonB(Alter alter) {
11     this.alter = alter;
12   }
13 }
14
15 class Alter {
16   public int jahre;
17   public Alter(int jahre) {
18     this.jahre = jahre;
19   }
20 }
21
22 class WerteVsObjekte {
23   public static void main(String[] args) {
```

[1] Hier: `http://db.in.tum.de/teaching/ss24/ei2`

```
24      PersonA mickeyA = new PersonA(50);
25      PersonA donaldA = new PersonA(55);
26
27      donaldA.alter = mickeyA.alter;
28      mickeyA.alter = 51;
29      System.out.println("MickeyA ist " + mickeyA.alter);
30      System.out.println("DonaldA ist " + donaldA.alter);
31
32      System.out.println("====");
33
34
35      PersonB mickeyB = new PersonB(new Alter(50));
36      PersonB donaldB = new PersonB(new Alter(55));
37
38      donaldB.alter = mickeyB.alter;
39      mickeyB.alter.jahre = 51;
40      System.out.println("MickeyB ist " + mickeyB.alter.jahre);
41      System.out.println("DonaldB ist " + donaldB.alter.jahre);
42    }
43  }
```

Aufgabe 4: Operationen

(a) Welche drei Gruppen von Operationen haben wir eingeführt und welche Eigenschaften machen diese Gruppen aus?

(b) Warum sollte eine Beobachterfunktion das Objekt nicht verändern?

(c) Welcher Rückgabetyp macht bei einer Beobachterfunktion keinen Sinn?

(d) Welche Access Modifier gibt es und was bedeuten sie?

Optionale Zusatzaufgabe 5: Stack mit verketteter Liste ⌨

Implementieren Sie einen Stack für `int`-Werte mithilfe einer verketteten Liste. Der Stack sollte dabei die Methoden `push()` und `pop()` anbieten, mit denen Elemente auf den Stack gelegt bzw. von ihm heruntergenommen werden. Eine mögliche Modellierung ist in Abbildung 2 gezeigt. Wie könnte man diesen Stack generischer machen?

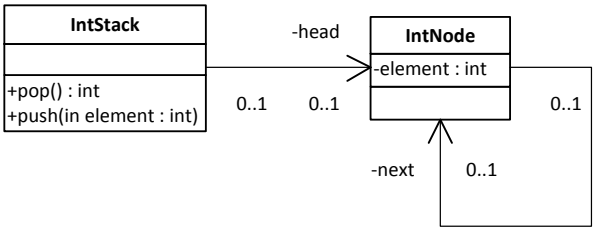

Abbildung 2: Stack mit verketteter Liste