

1. Aufgabe: Datenstrukturen (36 Punkte)

- a) (8 Punkte) In der Informatik gibt es oft die Bezeichnungen FiFo (First in, First out) und LiFo (Last in, First out). Geben Sie an, welche Datenstrukturen nach diesen Prinzipien arbeiten und erklären Sie ihren Aufbau.
- b) (4 Punkte) In einen Info-LKW, dessen Ladefläche nur von hinten zugänglich ist, passen immer zwei I-Paletten nebeneinander. Begründen Sie, dass sich die Realisierung des LKWs in einer Klasse am einfachsten mit zwei Stapeln bewerkstelligen lässt.
- c) (12 Punkte) Realisieren Sie die Klasse, indem Sie den entsprechenden Java-Code angeben. Beachten Sie dabei auch die Klassenbeschreibung im Anhang. Sie dürfen davon ausgehen, dass die Klasse `IPalette` bereits definiert ist.
- d) (12 Punkte) Ein Mitarbeiter in einem Büro möchte zum Verwalten seiner Aufgaben ein Programm schreiben. Er möchte dazu auf eine der vier Datenstrukturen Liste, Schlange, Prioritätsschlange oder Stapel zurückgreifen. Bewerten Sie die Eignung der Datenstrukturen für diese Aufgabe.

Anhang

Dokumentation der Klasse `InfoLKW`

Konstruktor	<code>InfoLKW()</code> Erzeugt einen neuen LKW mit zwei leeren Stapeln.
Auftrag	<code>void einladen(IPalette palette, boolean links)</code> Fügt eine Palette dem LKW hinzu. Dabei gibt <code>links</code> an, ob dieses in den linken oder rechten Stapel geschehen soll.
Auftrag	<code>void umstellen(int anz, boolean links)</code> Lädt eine angegebene Anzahl von Paletten, falls vorhanden, beim Zutreffen des zweiten Parameters von links nach rechts. Sonst wird von rechts nach links umgestellt. Ist die Anzahl der vorhandenen Paletten kleiner, werden entsprechend weniger umgestellt.
Anfrage	<code>IPalette ausladen(boolean links)</code> Entfernt eine Palette, sofern vorhanden, aus dem linken oder rechten Stapel. Sie liefert <code>null</code> falls keine entsprechende Palette mehr vorhanden ist.



Lösungen:

Datenstrukturen (36 Punkte):

- a) (8 Punkte) Die Schlange arbeitet nach dem FiFo-Prinzip. An einem Ende werden neue Elemente hinzugefügt, am anderen Ende die Elemente entnommen. Der Stapel arbeitet nach dem LiFo-Prinzip. Es können Elemente nur an einem Ende Elemente hinzugefügt und wieder entfernt werden. Ein Vertauschen von Plätzen ist in beiden Fällen nicht möglich.

(2x 2P Zuordnung 2P Aufbau)

- b) (4 Punkte) Da die linke und die rechte Seite unabhängig voneinander sind, ist jeder ein eigener Stapel. Durch die Be- und Entladung nur an einer Seite wird das Stapelprinzip realisiert.

- c) (12 Punkte)

```
1 public class InfoLKW {
2     private Stack links;
3     private Stack rechts;
4
5     public InfoLKW() {
6         this.links = new Stack();
7         this.rechts = new Stack();
8     }
9
10    public void einladen(IPalette palette, boolean links) {
11        if (links) {
12            this.links.push(palette);
13        } else {
14            this.rechts.push(palette);
15        }
16    }
17
18    public void umstellen(int anz, boolean links) {
19        for (int i = 0; i < anz; i++) {
20            if (links) {
21                this.rechts.push(this.links.top());
22                this.links.pop();
23            } else {
24                this.links.push(this.rechts.top());
25                this.links.pop();
26            }
27        }
28    }
29
30    public IPalette ausladen(boolean links) {
31        IPalette palette;
```



```
32     if (links) {
33         palette = (IPalette) this.links.top();
34         this.links.pop();
35     } else {
36         palette = (IPalette) this.rechts.top();
37         this.rechts.pop();
38     }
39     return palette;
40 }
41 }
```

- d) (12 Punkte) Der Stapel ist ungeeignet, weil bei vielen Aufgaben solche die tiefer im Stapel liegen sehr lange nicht zur Bearbeitung kommen. Insgesamt ist die Schlange geeignet, da alle Aufgaben in der Reihenfolge des Eingangs bearbeitet werden. Sie wird aber den Fällen nicht gerecht, in denen Aufgaben dringender bearbeitet werden müssen. Daher ist in diesem Fall die Prioritätsschlange vorzuziehen. Mit der Liste lässt sich eine solche Organisation realisieren, sie bietet nicht die entsprechenden Methoden, um die einfach die Aufgaben nach Dringlichkeit zu sortieren. (4x 3P)

Ergebnis

sehr gut	2		
gut	4		
befriedigend	3	gesamt:	10
ausreichend	1	Schnitt:	2,30
mangelhaft	0		
ungenügend	0		

