# 33. Bundesweiter Informatikwettbewerb – Aufgabe 2

*Team „ByteSector“*

# Teilaufgabe 1

Lösungsidee:

Da für die Gewichte und Balken nur die Eigenschaft Masse und für die Balken zusätzlich noch die Unterelemente und deren Positionen relevant sind, werden auch nur diese in der textuellen Darstellung wiedergegeben.

Die Gewichte werden nur durch ihre Masse dargestellt. Balken werden zusätzlich zu ihrer Masse noch durch eine eckige Klammer hinter dieser dargestellt. Diese Klammern beinhalten alle Subelemente, welche jeweils durch einen Vertikalstrich getrennt und mit dem Wert ihrer Position und einem anschließenden Unterstrich angeführt werden.

Ein Beispiel für einen Balken, der auf Position 5 ein Gewicht der Masse 4 und auf Position -10 einen weiteren Balken mit einem Gewicht der Masse 1 auf Position 2 und einem Gewicht der Masse 1 auf Position -2 hält:

6[5_4|-10_2[1_1|-1_1]]

Die Darstellung der Masse eines Balkens vor seinen Klammern ist zwar redundant, senkt allerdings die Zeit zur Erfassung seiner Masse zwecks Balanceüberprüfung erheblich.

Umsetzung:

Die Umsetzung erfolgt als eine rekursive Methode, welche in Java geschrieben ist. Hierbei wird im Falle eines Gewichtes die Masse und im Falle eines Balkens zusätzlich die Position und Masse aller Subelemente in oben genannter Form wiedergegeben. Das Abrufen der Masse der Subelemente erfolgt ebenfalls über diese Methode, was dazu führt, dass alle Balken auf allen Ebenen komplett dargestellt werden.

Der String, der zurückgegeben wird, wenn er auf den Hauptbalken, der die oberste Ebene des Mobiles widerspiegelt, angewandt wird, wird als Ergebnis in die Konsole ausgegeben.

```
/**Gibt die Masse eines Gewichtes aus. Ist das Gewicht ein Balken, wird zusätzlich noch die textuelle Darstellung
 * des Balkens in der Form '[position1_masse1|position2_masse2|...]" ausgegeben.*/
private String genMasse(Gewicht gewicht){
        // Rückgabestring
        String masse = "";

        // Masse des Gewichtes dem Rückgabestring hinzufügen
        masse += gewicht.getMasse();

        // Ist das Gewicht ein Balken, Subelementeklammer setzen
        if ((gewicht instanceof Balken)){
                // Klammer öffnen
                masse+="[";

                // Alle Subelemente durchgehen
                for(int i = 0; i < ((Balken)gewicht).getSubelemente().size(); i++){
                        // Für jedes Subelement die Position gefolgt von _ gefolgt vom Rückgabestring dieser Methode auf das Subelement dem Rückgabestring hinzufügen
                        masse += ((Balken)gewicht).getPositionen()[i] + "_" + genMasse(((Balken)gewicht).getSubelemente().get(i)) + "|";
                }

                // Letzten Trenntstrich entfernen
```

```
                masse = masse.substring(0, masse.length()-1);

                // Klammer schließen
                masse += "]";
        }

        // String zurückgeben
        return masse;
}
```

# Teilaufgabe 2

Lösungsidee:

Ein Mobile muss folgende Bedingungen erfüllen:

- alle Balken müssen balanciert sein
- maximal vier Subelemente pro Balken

Durch diese geringe Anzahl an Bedingungen, lässt sich die Aufgabe durch eine Methode lösen, die alle eingegebenen Gewichte in Gruppen mit bis zu vier Elementen zu Balken zusammenfasst und die Gewichte so platziert, dass die Balken stabil sind. Letzteres kann dadurch erreicht werden, dass Gewichte immer in Zweierpaaren an gegenüberliegende Seiten mit Abstand gleich der Masse des jeweils anderen Gewichts gehangen werden. Ist die Anzahl der Gewichte an einem Balken ungerade, wird eines der Gewichte in die Mitte des Balken, d.h. auf Position 0, gehangen, was die Balance des Balkens nicht beeinflusst.
Kommt es zu dem Fall, dass an einem Balken mit vier Gewichten durch die oben genannte Methode eine Position doppelt belegt wurde, wird die Position eines der betroffenen Gewichte zusammen mit der Position des anderen Gewichtes im Zweierpaar so lange um den selben Faktor erhöht, bis der Konflikt gelöst ist.

Die so erstellten Balken werden in einer weiteren Ausführung der Methode als Gewichte eingegeben und ebenfalls an neue Balken angehangen. Dies wird solange wiederholt, bis die Methode einen einzigen Balken zurückliefert, was das Mobile als fertigstellt gelten ließe.

Um die Abstände generell niedrig zu halten, wird angestrebt, alle Gewichte gleichmäßig auf die Balken zu verteilen. Dies wird dadurch erreicht, dass die Gewichte abwechselnd den zu erstellenden Balken zugeordnet werden.

Umsetzung:

Die Umsetzung erfolgt in einem Java-Programm, welches über eine Batch-Datei auf einem Windows-System gestartet werden kann.
Neben den Klassen **Balken** und **Gewicht** gibt es noch ein steuerndes Objekt, das der Klasse **Main** entstammt. Dieses erfasst zuallererst die Folge an Gewichten als durch Kommata getrennte Ganzzahlen, welche die Massen der Gewichte darstellen, und konvertiert diese in eine Liste an Objekten der Klasse **Gewicht**. Die oben besprochene Methode, die unter dem Namen *createBalken* umgesetzt wird, folgt, bis alle Balken und Gewichte zu einem Balken zusammengefasst sind. Schließlich wird dieser Balken in der textuellen Darstellung aus Teilaufgabe 1 wiedergegeben.

### Zusammenfassung von Gewichten zu Balken

```java
/** Verteilt alle eingegebenen Gewichte so auf neue erstellte Balken, dass diese balanciert sind, und gibt dann die Liste an Balken zurück.*/
private List<Gewicht> createBalken (List<Gewicht> gewichte){
	// Berechnen, wie viele Balken für die Unterbringung aller Gewichte erstellt werden müssen (=need)
	int anzahlBalken = (int) Math.ceil((double)gewichte.size() / 4);
	List<Gewicht> balken = new ArrayList<Gewicht>();

	// Alle erforderlichen Balken durchgehen
	for (int i = 0; i < anzahlBalken; i++){

		// Reihung an Gewichten erstellen, die später dem aktuellen Balken zugeteilt werden
		List<Gewicht> curGewichte = new ArrayList<Gewicht>();
		// Jedes 'Anzahl an Balken'-te Gewicht wird dem aktuellen Balken zugewiesen (=> Balken erhalten abwechselnd die Gewichte), damit die Gewichtsverteilung auf die einzelnen
Balken ähnlich ist.
		// Hierbei wird darauf geachtet, dass kein Balken mehr als vier Gewichte erhält
		//		Beispiel: Insgesamt 3 Balken, dies ist der 2
		//			=> Jedes dritte Gewicht (+ Offset von 1) wird diesem Balken zugeordnet
		for (int j = i; j < gewichte.size() && curGewichte.size() < 4; j += anzahlBalken){
			curGewichte.add(gewichte.get(j));
		}

		// Balken mit den aktuellen Gewichten auf Positionen, mit denen der Balken balanciert wäre, erstellen
		balken.add(new Balken(curGewichte, findPos(curGewichte)));
	}
	return balken;
}
```

### Einlesen und Konvertieren der Folge an Gewichten

```java
// Gewichte anfragen
System.out.println("Massen der Gewichte mit Komma getrennt eingeben (bsp. 1,2,3,44):");
Scanner sc = new Scanner(System.in);

// Benutzerinput splitten und konvertieren
int[] input = null;
try {
	String read[] = sc.nextLine().split(",");
	input = new int[read.length];
	for (int i = 0; i < input.length; i++)
		input[i] = Integer.parseInt(read[i]);

	sc.close();
} catch (Exception e){
	System.out.println("ERROR: Eingabe ungültig!");
	System.exit(0);
}

// Input auslesen
sortArrayInt(input);
List<Gewicht> gewichte = new ArrayList<Gewicht>();
for (int i = 0; i < input.length; i++){
	gewichte.add(new Gewicht(input[i]));
}
```

### Zusammenfassung aller Gewichte zu einem Balken

```
List<Gewicht> balken = createBalken(gewichte);

// Mobile erstellen: So lange Gewichte bzw. Balken an neue Balken hängen, bis ein einziger übrig bleibt
while (balken.size() > 1){
        balken = createBalken(balken);
}
```

Beispiele:

| Eingabe | Ausgabe |
|---|---|
| 1,1,2,2 | 6[2_2\|-2_2\|1_1\|-1_1] |
| 2,6,9,4,7,3,1,4,7 | 43[13_18[6_9\|-9_6\|0_3]\|-18_13[4_7\|-7_4\|0_2]\|0_12[4_7\|-7_4\|0_1]] |
| 1,43,6,235,2 | 287[45_242[6_235\|-235_6\|0_1]\|-242_45[2_43\|-43_2]] |
| 3645634,324658,782145,87654,2353476 | 7193567[2678134_4515433[782145_3645634\|-3645634_782145\|0_87654]<br>\|-4515433_2678134[324658_2353476\|-2353476_324658]] |