Aufgaben zur Tischgruppe mit Sammlungsklassen

In dem nachfolgenden Beispiel wollen wir Sammlungsklassen anwenden.
Methodisch arbeiten wir wie im vorherigen Aufgabenteil.

Problemstellung: Modellieren einer Tischgruppe in einer Klasse

Zunächst müssen wir versuchen, die Aufgabe unmissverständlich zu definieren. Eine Tischgruppe soll aus einem rechteckigen Tisch mit einer variablen Anzahl von Stühlen darum bestehen. Dabei gilt es natürlich zu beachten, dass der Tisch länger wird, wenn mehr Stühle darum gruppiert werden sollen. An den kurzen Seiten des Tisches soll jeweils nur ein Stuhl stehen. An den langen Seiten soll jeweils dieselbe Zahl von Stühlen stehen. Die Stühle sollen die von uns bisher verwendeten Abmessungen von 40x40 Pixeln haben, zwischen ihnen soll jeweils ein Platz von 10 Pixeln verbleiben und denselben Abstand sollen sie auch vom Tisch haben.

- Zeichnen Sie mehrere Skizzen des Aufbaus solcher Tischgruppen und beschriften Sie diese mit den Längenangaben.

Für eine Tischgruppe mit 6 Stühlen sollten Sie ein Rechteck mit 110 Pixeln Länge [10+40+10+40+10] und 60 Pixeln Breite [10+40+10] verwendet haben.
Die Tischgruppe muss insgesamt eine Länge von 210 Pixeln [40+10 +110+10+40] und eine Breite von 160 Pixeln [40+10+60+10+40] haben.

- Berechnen Sie die Positionen der einzelnen Stühle.
  Geben Sie dazu jeweils die Position der linken oberen Ecke an.

Die Stuhlpositionen hängen natürlich davon ab, wie wir den Tisch ausrichten. Betrachten wir eine Ausrichtung der langen Kante in x-Richtung und verwenden die relativen Koordinaten in der Gruppe mit (0;0) für die linke obere Ecke der Gruppe, dann hat der Tisch eine linke obere Ecke von (50;50).
Der Stuhl an der linken kurzen Seite hat dann die Position (0;60), der gegenüber die Position (170;60).
An der oberen langen Seite sind die Positionen (60;0) und (110;0), unten entsprechend (60;120) und (110;120).

- Schreiben Sie die Schleifenanweisung zur Berechnung der x-Koordinaten der Stuhlpositionen an der oberen langen Kante für beliebige Längen der Tischgruppen auf.
- Geben Sie an, was bei den Stuhlpositionen an der unteren Kante anders sein muss.

Da vor dem Beginn der Schleife bekannt ist, wie oft sie ausgeführt werden soll, verwendet man eine for-Schleife. Ohne das Erzeugen der Objekte erhält man folgende Schleife:
→ für die obere Kante braucht man also folgende Inhalte:

```
for (int i=0;i<anzahl;i++)    60 + i*(40+10);
```

→ für die untere Kante ist die x-Koordinate gleich, es ändert sich allein die y-Koordinate. Sie hat statt 0 den Wert 120 [=50+60+10].

Definition und Speichern der Objekte im Konstruktor

Im vorigen Abschnitt haben wir gelernt, dass die Objekte im Konstruktor erzeugt und in einer Sammlungsstruktur gespeichert werden müssen, die wir im Kopf der Klasse deklarieren müssen, damit wir sie in der ganzen Klassendefinition verwenden können. Der Programmtext für die beiden Attribute lautete bei der Schrankwandklasse:

```
private ArrayList<Schrank> schraenke;
private int anzahl;
```

- Überlegen Sie, was in diesem Beispiel anders ist, welche Folgen das für diese beiden Zeilen hat und machen Sie einen Vorschlag zur Änderung

---

Wir haben in unserer Tischgruppe Objekte verschiedenen Typs. Im Fall der Schrankwand waren es alles Schrank-Objekte, so dass wir eine ArrayList für Schrank-Objekte verwenden konnten. Bei der Tischgruppe haben wir sowohl Stuhl-Objekte als auch ein Tisch-Objekt. Glücklicherweise haben wir unser Projekt aber inzwischen so modelliert, dass wir einen gemeinsamen Typ haben, nämlich den Typ Moebel.

- Schreiben Sie die Beziehungen zwischen Stuhl, Tisch und Moebel auf.

Jedes Stuhl-Objekt ist ein Moebel-Objekt.
Jedes Tisch-Objekt ist ein Moebel-Objekt. } ← Vererbung!
Daher können wir eine ArrayList von Moebel-Objekten verwenden. Die Zeile zur Deklaration der ArrayList könnte also lauten:

```
private ArrayList<Moebel> elemente;
```

- Schreiben Sie die gesamte Definition der Objekte im Konstruktor auf.

Eine mögliche Lösung ist:

```
// Definition der Sammlung
elemente = new ArrayList();
for (int i=0;i<anzahl;i++)
{
    // Stuehle an der oberen Kante
    elemente.add(new Stuhl(60 + i*(40+10), 0, farbe, 0));
    // Stuehle an der unteren Kante
    elemente.add(new Stuhl(60 + i*(40+10), 120, farbe, 180));
}
// der Stuhl links
elemente.add(new Stuhl(0, 60, farbe, 270));
// der Tisch
elemente.add(new Tisch(50, 50, farbe, 0, 10+anzahl*50, 60));
// der Stuhl rechts
elemente.add(new Stuhl(170, 60, farbe, 90));
```

Die Zeilen 5 bis 8 bilden den Inhalt der Schleifenanweisung, da sie wiederholt werden. Die anderen Objektdefinitionen werden nur einmal ausgeführt und gehören daher nicht zur Schleifenanweisung.

- Überlegen Sie, welche Folgen die Änderungen in der Methode gibAktuelleFigur()

haben.
Da die Zahl der Objekte, die gezeichnet werden müssen auch hier variabel ist, müssen wir diese Anzahl neu bestimmen.

- Berechnen Sie die Anzahl der zu zeichnenden Objekte in Abhängigkeit von der Anzahl der Stühle an der oberen Kante.
- Schreiben Sie die komplette Methode gibAktuelleFigur() auf.

Es sind zweimal **anzahl** viele Stühle an den langen Kanten, dazu zwei an den kurzen Seiten und natürlich auch der Tisch. Damit sind es insgesamt $2 \cdot anzahl + 3$ Objekte. Nutzt man die for-each-Schleife braucht man sich über die Anzahl keine Gedanken zu machen.

Die Methode mit "for-each" lautet:

```
protected Shape gibAktuelleFigur()
{
    GeneralPath tischgruppe = new GeneralPath();
    for (Moebel moebel : elemente)
      tischgruppe.append(moebel.gibAktuelleFigur(), false);
    return  transformiere(tischgruppe);
}
```