Aufgaben für Gruppenarbeit zu Sammlungsklassen

Hinweis

Bearbeiten Sie die Aufgaben schrittweise. Wenn Sie dazu neigen, "sich selbst über's Ohr zu hauen" und lieber einmal nachsehen, wie es geht, knicken Sie besser die Blätter jeweils an den Trennlinien um.

- Öffnen Sie das Projekt **04-Schrankwand-3-ohne-Sammlung**.
- Öffnen Sie die Klassendefinition der Klasse Schrankwand.
- Entfernen Sie aus Gründen der Übersichtlichkeit den zweiten Konstruktor, so dass die Klassendefinition nur den Standardkonstruktor enthält.
- Erläutern Sie zunächst [schriftlich] den hier angegebenen Programmtext der Methode gibAktuelleFigur().
- Beschreiben Sie, welchen Mangel diese zwar funktionierende, aber eben mangelhafte Lösung hat.

```
    protected Shape gibAktuelleFigur() {
        GeneralPath schrankwand = new GeneralPath();
        for (int i=0; i<anzahl;i++)
            schrankwand.append(new Schrank(i*breite/anzahl, 0, farbe, 0, breite/
anzahl, tiefe).gibAktuelleFigur(), false);
        return transformiere(schrankwand);
    }
```

Wenn wir verhindern wollen, dass das Programm sich jedesmal beim Zeichnen anzahl-viele neue Schrankobjekte beschafft, müssen wir dafür sorgen, dass es diese nur einmal erzeugt.
Das könnte man einmal so lösen, dass sich das Schrankwandobjekt merkt, ob es schon einmal die Objekte erzeugt hat und sie nur dann erzeugt, wenn zum erstenmal gezeichnet werden soll.
Da man beim Erzeugen eines Schrankwandobjektes aber in der Regel davon ausgehen kann, dass die Schrankwand auch gezeichnet werden soll, kann man dies auch beim Erzeugen der Schrankwand selbst tun.

- Überlegen Sie sich, wo das Erzeugen der Schrankobjekte geschehen muss, wenn es jeweils beim Erzeugen des Schrankwandobjektes erfolgen soll.
- Fügen Sie den notwendigen Programmtext in das Projekt ein.

Zur Lösung

Es sollte Ihnen klar geworden sein, dass im Konstruktor der Klasse die Objekte erzeugt werden müssen. Im Konstruktor muss eine Schleifenanweisung – wie bisher in der Methode gibAktuelleFigur() vorhanden – eingefügt werden, die das Erzeugen der Schrankobjekte steuert. Fügt man allein das Erzeugen an dieser Stelle ein, könnte der Programmtext so aussehen:

```
for (int i=0; i<anzahl;i++)
    new Schrank(i*breite/anzahl, 0, farbe, 0, breite/anzahl, tiefe);
```

Leider reicht es nicht allein, sich Gedanken über das Erzeugen der Schrankobjekte zu machen. Zunächst einmal müssen wir die Objekte nicht nur erzeugen, sondern auch abspeichern. Dazu brauchen wir eine Variable zum Abspeichern der Schrankobjekte unter einem gemeinsamen Namen. Wir wollen diese Variable einmal **`schraenke`** nennen.

- Ergänzen Sie die Programmzeile um die Zuweisung zum Variablennamen schraenke.

Zur Lösung

Der Programmtext lautet nun:

```
        for (int i=0; i<anzahl;i++)
            schraenke[i] = new Schrank(i*breite/anzahl, 0, farbe, 0,
breite/anzahl, tiefe);
```

Ohne das in eckigen Klammern angegebene **[i]** würden wir die Schrankobjekte jeweils immer in dieselbe Variable **schraenke** schreiben und dabei jeweils den vorherigen Wert durch den aktuellen überschreiben, so dass der vorherige Wert verloren ist. **schraenke** muss also eine Variable sein, unter deren Namen mehrere Objekte abgespeichert werden können, sie muss also für eine Sammlung stehen, so dass die Schleifenanweisung die Schrankobjekte jeweils in die entsprechenden Speicherplätze hineinschreibt.

- Überlegen Sie, an welcher Stelle im Programmtext diese Variable angegeben [Fachausdruck: deklariert] werden muss.

Zur Lösung

Für die Sammlung gibt es in Java mehrere Möglichkeiten, in jedem Fall aber gilt: Im Kopf der Klassendefinition, also dort, wo wir die Attribute [Instanzvariablen] der Klasse angeben, muss ein neues Attribut eingefügt werden, das wir brauchen, um die erzeugten Objekte zu speichern.
Die entsprechende Programmzeile hängt davon ab, welchen Sammlungstyp wir verwenden. Ist das ein Array, dann lautet sie:

```
private Schrank[] schraenke;
```

- Fügen Sie die Zeile ein, übersetzen Sie das Programm und testen Sie es.

Zur Lösung

Das Übersetzen führt zwar nicht zu einer Fehlermeldung, das Austesten führt aber zu einer Fehlermeldung. Der Grund ist, dass wir dem Programm zwar mitgeteilt haben, welche Art [Deklaration] von Speicherplätzen es reservieren soll, aber nicht, wie viele davon. Im Konstruktor muss daher das Array schraenke in der passenden Größe definiert werden.
Da der Konstruktor die Anzahl der zu erzeugenden Schrankobjekte übergeben bekommt, kann man hier durch das Programm den notwendigen Speicherplatz für die Speicherung der Schrankobjekte reservieren lassen. Man nennt das die Definition des Arrays. Der dafür notwendige Programmtext muss vor der Schleife eingefügt werden.

```
schraenke = new Schrank[anzahl];
```

- Was fehlt noch?

Zur Lösung

Die Methode gibAktuelleFigur() muss noch so geändert werden, dass sie die Schrankobjekte nicht neu erzeugt, sondern die vorhandenen benutzt. Die Programmzeile lautet dann:

```
for (int i=0; i<anzahl;i++)
    schrankwand.append(schraenke[i].gibAktuelleFigur(), false);
```

Diese Variante ist damit fertig.

Ein echter Sammlungstyp: ArrayList

Als eine weitere Übung erarbeiten Sie ein neues Projekt mit einer Modifikation der Lösung, die einen echten Sammlungstyp verwendet, die ArrayList. Verwenden Sie dabei gleich die typisierte ArrayList.
Eine zusätzliche Anforderung ist hier noch, dass Sie das Paket ArrayList importieren müssen, so dass bei den Importen die folgende Zeile eingefügt werden muss:

```
import java.util.ArrayList;
```

- Speichern Sie das aktuelle Projekt unter einem neuen Namen ab.
- Laden Sie die Dokumentation der Klasse ArrayList [→ Javadoc].
- Versuchen Sie die notwendigen Änderungen schrittweise durchzuführen.

Deklaration:
private ArrayList<Schrank> schraenke;

Definition

```
schraenke = new ArrayList<Schrank>();
```

Es wird also der Konstruktor der Klasse ArrayList aufgerufen.
Alternativ auch:

```
schraenke = new ArrayList();
```

Die Typisierung ist nur bei der Deklaration notwendig.

Für das Hinzufügen der Objekte bietet die Klasse die Methode add an. Sie muss die Schrankobjekte mit den richtigen Positionswerten übergeben bekommen.

```
for (int i=0; i<anzahl;i++)
    schraenke.add(new Schrank(i*breite/anzahl,
                              0, farbe, 0, breite/anzahl, tiefe));
```

Die Angabe der Positionen erfolgt relativ zur Schrankwandposition.

Der Zugriff geht wie bei Arrays mit einem Index,beispielsweise i, allerdings muss über die spezielle Methode mit get(i) zugegriffen werden.

```
for (int i=0; i<anzahl;i++)
    schrankwand.append(schraenke.get(i).gibAktuelleFigur(), false);
```

Besser ist der Zugriff mit "for – each". Die entsprechende Anweisung lautet:

```
for (Schrank schrank : schraenke)
    schrankwand.append(schrank.gibAktuelleFigur(), false);
```

Der boolean-Wert false sorgt dafür, dass zwischen den einzelnen Schranksymbolen keine Verbindungslinien gezeichnet werden.