Anmerkungen zu den Schrankwandprojekten

- Wichtiger erster Schritt beim Projekt Schrankwand ist die Erkenntnis, dass nicht das dreifache Kopieren des Codes von Schrank die Lösung ist, sondern dass die Klasse Schrankwand das nutzt, was die Klasse Schrank schon kann und sich daher drei Schrankobjekte beschafft.
- Eine rein statische Variante erzeugt also die Objekte schran1, schrank2 und schrank3 und greift in der Methode gibAktuelleFigur() auf diese drei Objekte zu und ruft an ihnen die Methode gibAktuelleFigur() auf, um die zurückgegebenen Shape-Objekte dem eigenen GeneralPath hinzuzufügen.
- Damit nicht bei jeder Aktion von Schrankwand jedesmal neue Schrankobjekte erzeugt werden müssen, sollte man das Erzeugen im Konstruktor erledigen.
- Wenn man mehrere Konstruktoren verwendet, sollte man – auch im sinne der Kohäsion – das Erzeugen aus den Konstruktoren in eine eigene Methode ausgliedern:

```
/**
 * initialisiert die Schrankobjekte.
 * Das sollte im Konstruktor geschehen,
 * wird aber wegen der zwei Konstruktoren
 * in eine eigene Methode ausgegliedert.
 */
private void initSchraenke(){
    schrank1= new Schrank(0, 0, farbe, 0, breite/3, tiefe);
    schrank2= new Schrank(breite/3, 0, farbe, 0, breite/3, tiefe);
    schrank3= new Schrank(2*breite/3, 0, farbe, 0, breite/3, tiefe);
}
```

- Das Zusammenbauen in den GeneralPath ist das sehr übersichtlich.

```
/**
 * Berechnet das zu zeichnende Shape anhand der gegebenen Daten
 */
protected Shape gibAktuelleFigur() {
    // einen GeneralPath definieren
    GeneralPath schrankwand = new GeneralPath();
    schrankwand.append(schrank1.gibAktuelleFigur(), false);
    schrankwand.append(schrank2.gibAktuelleFigur(), false);
    schrankwand.append(schrank3.gibAktuelleFigur(), false);
    return transformiere(schrankwand);
}
```

- Dieser statische Entwurf ist ungünstig. Sinnvollerweise führt man ein Attribut **`schraenke`** ein, in dem die drei Schrankobjekte unter einem gemeinsamen Namen abgespeichert werden können [eine Sammlungsstruktur]. Außerdem sollte man dem Konstruktor die Anzahl der zu erzeugenden Schrankobjekte übergeben können.
- Eine erste Variante dafür ist ein Array. Die Zugriffe erfolgen dann über eine Index-gesteuerte Schleife [eine Wiederholungsstruktur].
- Für die Verwendung aller Sammlungsstrukturen gilt, dass bei Java drei Schritte notwendig sind:
  - Deklaration der Sammlung z.B.: **`Schrank[] schraenke;`**
  - Definition [Initialisierung] des Sammlungsobjektes
    z.B.: **`schraenke=new Schrank[anzahl];`**
  - Definition der Objekte in der Sammlung:
    z.B.: **`schraenke[0]=new Schrank();`**
- Wird nicht nur einmalig auf die Objekte zugegriffen – wozu sonst sollte man sie

abspeichern wollen – ist es sinnvoll, das Sammlungsobjekt als Attribut zu halten. Dazu deklariert man es im Kopf der Klassendefinition und definiert [initialisiert] es selbst und seine zugehörigen Objekte im Konstruktor oder in einer vom Konstruktor [von den Konstruktoren] aufgerufenen Initialisierungsmethode.

- Eine sinnvolle Alternative dazu ist, die Objekte zunächst nicht zu definieren [initialisieren] und das erst dann zu tun, wenn sie das erstemal benötigt werden.
  Beispiel: `if (schraenke==null) schraenke=new Schrank[anzahl];`

Eine Beispiellösung

```
public class Schrankwand extends Moebel
{
  private Schrank[] schraenke;
  private int anzahl;

  /**
   * Erzeuge eine neue Schrankwand mit einer Standardfarbe und Standardgroesse
   * an einer Standardposition. (Standardkonstruktor)
   */
  public Schrankwand(int anzahl) {
    xPosition = 10;
    …
    initSchraenke(anzahl);
  }

  /**
   * Erzeuge eine neue Schrankwand.
   * Konstruktor, bei dem alle Attribute gesetzt werden können.
   */
  public Schrankwand(
       int xPosition,
       int yPosition,
       String farbe,
       int orientierung,
       int breite,
       int tiefe,
       int anzahl) {
    this.xPosition = xPosition;
    …
    initSchraenke(anzahl);
  }

  /**
   * initialisiert die Schrankobjekte.
   * Das sollte im Konstruktor geschehen,
   * wird aber wegen der zwei Konstruktoren
   * in eine eigene Methode ausgegliedert.
   */
  private void initSchraenke(int anzahl){
    this.anzahl=anzahl;
    schraenke=new Schrank[anzahl];
    for (int i=0; i<anzahl; i++)
      schraenke[i] =
      new Schrank(i*breite/anzahl, 0, farbe, 0, breite/anzahl, tiefe);
  }
```