Kohäsion und Kopplung

Mit diesen beiden Begriffen werden Eigenschaften von Klassenentwürfen beschrieben, bei denen man die Qualität von Entwürfen beschreibt[1].
Bei unseren neuen Möbelklassen haben wir herausgefunden, dass sie sich allein in der Methode **`gibAktuelleFigur()`** unterscheiden. Das haben wir korrigiert, allerdings enthält sie in der vorliegenden Form noch einen Abschnitt, der in allen Klassen gleich ist, nämlich den Teil, in dem die Transformation des konkreten Falles eines Shape durchgeführt wird. Hier zeigt sich, dass die ursprüngliche Modellierung ungünstig war, da die Methode eigentlich zwei Aufgaben erfüllte:

1. Die Definition der konkreten Figur als GeneralPath, Arc2D.Double, Ellipse2D.Double, Rectangle2D.Double, Line2D.Double, CubicCurve2D.Double, QuadCurve2D.Double, Polygon (o.ä.)
2. Die Transformation dieser konkreten Figur als Shape durch eine Verkettung von linearen Transformationen, um die gewünschte Lage und Orientierung zu erzielen.

Sinnvollerweise gliedert man den Code, der nun immer noch gemeinsam ist, in eine eigene Methode aus, die wir hier **`transformiere(Shape shape)`** nennen können.
Die Methode wäre dann:

```
protected Shape transformiere(Shape shape)
{
    AffineTransform t = new AffineTransform();
    t.translate(xPosition, yPosition);
    Rectangle2D umriss = shape.getBounds2D();
    t.rotate(Math.toRadians(orientierung),
                          umriss.getX()+umriss.getWidth()/2,
                          umriss.getY()+umriss.getHeight()/2);
    return  t.createTransformedShape(shape);
}
```

Sie besteht aus zwei Transformationen. Die erste ist eine Translation, also eine Verschiebung auf die Zielkoordinaten **`(xPosition, yPosition)`**. Die zweite ist eine Rotation um das von **`umriss`** gelieferte Drehzentrum.
Barnes/Kölling schreiben: „Der Begriff Kohäsion bezieht sich auf die Anzahl und Vielfalt der Aufgaben, für die eine einzelne Einheit in einer Anwendung zuständig ist... Idealerweise sollte eine Programmeinheit für genau eine in sich geschlossene Aufgabe zuständig sein..."
Genau das haben wir bei unserem ersten Entwurf verletzt. Hier zeigt sich wieder, dass hinter dem beim Codeduplizieren entstandenen Gefühl, *„das müsste raus"* mehr steckt (was übrigens nicht immer der Fall sein wird). Die Transformation ist für sich eine geschlossene Aufgabe, die für ein Shape – und das verwenden wir zur Darstellung unseres konkreten Möbels – zur Verfügung gestellt werden muss.
Der vorliegende erste Entwurf der Methode **`gibAktuelleFigur()`** verstieß gegen das Prinzip der Kohäsion.

> *Konzept hohe Kohäsion:* (Barnes/Kölling)
> *Der Begriff beschreibt, wie gut eine Programmeinheit eine logische Aufgabe oder Einheit abbildet. In einem System mit hoher Kohäsion ist jede Programmeinheit ... verantwortlich für genau eine wohldefinierte Aufgabe ...*

> *Konzept lose Kopplung:* (Barnes/Kölling)
> *Der Begriff beschreibt den Grad der Abhängigkeit zwischen Klassen. Wir streben eine möglichst lose Kopplung an – also ein System ...*

1 Siehe dazu der Abschnitt 7.3 von Barnes / Kölling: Objektorientierte Programmierung in JAVA