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 **`gibFigur()`** unterscheiden. Das haben wir durch die Einführung der Klasse Moebel und das Vererben aller anderen Methoden und der Attribute berücksichtigt. Erstellt man nun eine Klasse zu einem Moebelstück, dann wird nur noch der Konstruktor und die Methode **`gibFigur()`** in die neue Klasse kopiert und passend modifiziert.
Betrachtet man diesen Vorgang genauer, dann fällt auf, dass die neue Klasse immer noch gleiche Abschnitte enthält. Das wird man z.T. nicht vermeiden können, da z.B. der Konstruktor auch hier die Parameter an den Konstruktor der Klasse Moebel weiter reichen muss.
Auch die Methode **`gibFigur()`** enthält allerdings einen in allen Klassen gleichen Abschnitt und zwar den Teil, in dem die Transformation der konkreten Figur durchgeführt wird.

Jede Methode[2] soll für genau eine Aufgabe zuständig sein

Hier zeigt sich, dass die ursprüngliche Modellierung ungünstig gewesen ist, da die Methode eigentlich zwei Aufgaben erfüllt hat [wie auch der Kommentar dieser Methode zeigt]:

1. die Definition der konkreten Figur als GraphicsPath und
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(self, path)`** nennen können.
Diese Methode wäre dann:

```
    def Transformiere(self, path):
        """Transformiert den Pfad"""
        gc = Zeichenflaeche.GibZeichenflaeche().GibGC()
        gc.PushState()
        gc.Translate(self.x+self.b/2, self.y+self.t/2)
        gc.Rotate(radians(self.w))
        gc.Translate(-self.b/2, -self.t/2)
        transformation = gc.GetTransform()
        gc.PopState()
        path.Transform(transformation)
        return path
```

Sie besteht aus drei Transformationen.
Die erste ist eine Translation, also eine Verschiebung auf die Zielkoordinaten, hier das Zentrum der Rotation **`(self.x+self.b/2, self.y+self.t/2)`**.
Die zweite ist eine Rotation um dies Drehzentrum.
Danach muss noch wieder so zurück verschoben werden, dass die Position durch die Koordinaten **`(self.x, self.y)`** definiert sind.
Barnes/Kölling schreiben im BlueJ-Buch zu Kopplung und Kohäsion:
„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...“

1 Siehe dazu der Abschnitt 7.3 von Barnes / Kölling: JAVA lernen mit BlueJ
2 Diese Anforderung gilt entsprechend auch für Klassen.

Codeduplizieren sollte aufmerksam machen

Nun, 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 . Die Transformation ist eine in sich geschlossene Aufgabe, die für eine Figur bei der hier vorliegenden Modellierung des Zeichnens von Figuren zur Verfügung gestellt werden muss.
Der vorliegende erste Entwurf der Methode **`gibFigur()`** verstieß dadurch gegen das Prinzip der Kohäsion[1].

> **Konzept hohe Kohäsion:**
> 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 ...

Und was ist mit der Kopplung?

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

Wir sind noch nicht so weit, dass es sinnvoll ist, Fragen der losen Kopplung zwischen Klassen zu diskutieren. Das müsste noch kommen, zum jetzigen Zeitpunkt könnte man sich bestenfalls über die Kopplung zwischen den Möbelklassen und der Klasse Zeichenflaeche Gedanken machen.

1 Formulierungen nach Barnes/Kölling: Java lernen mit BlueJ