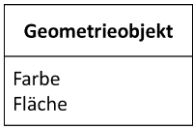
1 Das Klassenkonzept - Klasseneigenschaften

Das Wort „Klasse“ verdeutlicht, dass man bei der Definition von Strukturen in der objektorientierten Programmierung die einzelnen Elemente „klassifiziert“. Dies bedeutet, dass man nach Gemeinsamkeiten sucht und diese zusammenfasst. Überlegen Sie sich, wie die einzelnen Eigenschaften den Begriffen zuzuordnen sind:

	Farbe	Fläche	Volumen	Umfang	Kreis-radius	Kugelra-dius	Kanten-länge	Seiten-längeA	Seiten-längeB	Seiten-längeC	Rechteck-länge	Rechteck-breite
Geometrieobjekt												
Geo3DObjekt												
Geo2DObjekt												
Würfel												
Kugel												
Rechteck												
Dreieck												
Kreis												

Finden Sie heraus, welche Eigenschaften jeweils nur einem Begriff zuzuordnen sind und kreisen Sie diese Eigenschaften rot ein. Markieren Sie nun die Begriffe, deren übriggebliebenen Eigenschaften eine Teilmenge eines anderen Begriffes bilden. Wir können Gemeinsamkeiten bei den einzelnen Begriffen erkennen und eine Hierarchie bilden – wir können sie klassifizieren. Diese Eigenschaften sind Variablen, welche innerhalb der ganzen Klasse zugreifbar sind. Wir können diese Hierarchie in einem Klassendiagramm grafisch darstellen:



2 Methoden

Neben den Eigenschaften besitzen die Klassen auch Funktionalitäten – die Methoden. Auch hier gilt, dass wir gemeinsame Methoden feststellen können. Es gibt an dieser Stelle jedoch ein Problem. Gehen wir davon aus, wir wollen die Fläche berechnen. Der Wert der Fläche wurde der Klasse Geometrieobjekt zugeordnet. Die Berechnung wiederum kann nur auf der untersten Ebene erfolgen. Hier die Formeln für die Flächenberechnung der Klassen auf unterster Ebene:

Klasse:	**Formel für Fläche:**
Würfel	$A = 6 \cdot Kantenlänge^2$
Kugel	$A = 4 \cdot \pi \cdot Kugelradius^2$
Rechteck	$A = Rechtecklänge \cdot Rechteckbreite$
Dreieck	$A = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}$ für $s = \frac{1}{2} \cdot (a+b+c)$
Kreis	$A = \pi \cdot Kreisradius^2$

Alle in der Tabelle angegebenen Klassen benötigen also eine Funktionalität „`berechneFlaeche()`“, wodurch wir die Notwendigkeit zwar auf der obersten Ebene – der Klasse Geometrieobjekt – hinterlegen müssten, die Implementierung der einzelnen Berechnungsformeln aber auf der untersten Ebene erfolgen muss. Dieser Gedanke ist für das weitere Verständnis wichtig! Überlegen wir uns nun in der nebenstehenden Tabelle, welche Funktionalitäten für welche Klassen vorhanden sein sollten (O) und wo sie tatsächlich implementiert werden können (X).

	berechne-Fläche	berechne-Volumen	berechne-Umfang	erzeuge-Objekt
Geometrieobjekt				
Geo3DObjekt				
Geo2DObjekt				
Würfel				
Kugel				
Rechteck				
Dreieck				
Kreis				

Wir sehen also, dass die Flächenberechnung für alle Klassen sinnvoll sein könnte, die Volumenberechnung nur für die Klassen ab Geo3DObjekt und die Umfangberechnung ab Geo3DObjekt.

Die Objekterzeugung jedoch ist nur für die unteren fünf Klassen sinnvoll, da die Information „Erzeuge ein Geometrieobjekt“ zu unspezifisch ist. Der Rechner weiß schlichtweg nicht, welches Objekt er eigentlich erzeugen soll.

3 Vererbung

Die Vererbungsabhängigkeiten des Klassendiagramms werden im Java Code über das Schlüsselwort „extends“ umgesetzt. Das Wort „erweitern“ zeigt auch die wichtigste Idee hinter dem Hierarchiegedanken – eine Klasse bietet sinnvolle Funktionalität, die um zusätzliche, neue Funktionalität erweitert wird. Hier der Code unserer Klassen „Geometrieobjekt“, „Geo2DObjekt“ und „Kreis“:

```
public class Geometrieobjekt {
      int farbe;
      double flaeche;
}
```

```
public class Geo2DObjekt extends Geometrieobjekt {
      double umfang;
}
```

```
public class Kreis extends Geo2DObjekt {
      double radius;
}
```

In Java können Klassen immer nur eine einzige Elternklasse erweitern!