4 OO-Grundlagen

Bei der objektorientierten Programmierung steht, wie der Name bereits andeutet, das Objekt im Mittelpunkt der Betrachtung. Dabei bildet ein Objekt eine logische Einheit aus Variablen und Methoden. Auf dieser Basis lassen sich viele Problemstellungen auf natürliche Weise nachbilden.

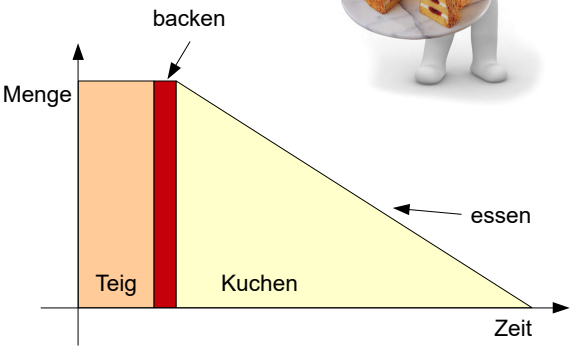
UML

UML ist eine vereinheitlichte Modellierungssprache für die objektorientierte Software-Entwicklung, durch die unterschiedlichste Diagramme definiert und unterschieden werden. Eines der gebräuchlichsten UML-Diagramme zur Veranschaulichung objektorientierter Zusammenhänge ist das UML-Klassendiagramm, dessen Komponenten in den nachfolgenden Kapiteln im entsprechenden Kontext gleich mit eingeführt werde. Das Thema UML selbst wird zu einem späteren Zeitpunkt im Kapitel 14 aufgegriffen und ausführlich behandelt.

4.1 Objekte / Klasse

Als einführendes Beispiel wird das Objekt **Kuchen** betrachtet. Dieses Objekt hat folgenden "Lebenszyklus":

Lebenszyklus



Der Einfachheit halber beginne der Lebenszyklus eines Kuchenobjektes mit dem Zeitpunkt, an dem alle benötigten Zutaten in eine Rührschüssel gegeben worden sind. Nach einer Phase des Knetens und Verrührens, in der der Kuchen nur als unfertiger Teig existiert, wird der Kuchen gebacken. Anschließend wird der Kuchen nach und nach gegessen, bis nichts mehr von ihm übrig ist. Diesen Lebenszyklus gilt es nun möglichst exakt mithilfe eines Objektes in Java nachzubilden.

4.1.1 Objektvariable

Objektvariable
Objektvariablen sind Variablen, die den Zustand eines Objektes beschreiben. Objektvariablen werden OHNE das Schlüsselwort **static** meist zu Beginn einer Klasse außerhalb von Methoden definiert.

In dem Kuchenbeispiel kann dies zum einen durch die Mengenangabe der Zutaten[1]

```
mehl, butter, eier, zucker,
```

aber auch durch zusätzliche Variablen wie

```
stuecke, teig
```

erfolgen. Die Variable **stuecke** (Stücke) gibt die Anzahl der noch vorhanden Reststücke eines Kuchens wieder. Die Variable **teig** kann über einen **boolschen** Wert die Information "noch Teig" oder "bereits fertig gebackener Kuchen" speichern.

4.1.2 Objektmethode

Objektmethode
Objektmethoden verändern die Objektvariablen und damit den Zustand eines Objektes. Objektmethoden werden OHNE das Schlüsselwort **static** innerhalb einer Klasse definiert.

Ein Kuchenobjekt sollte eine Methode

```
backen()
```

besitzen, mit der die Variable **teig** und damit der Zustand des Kuchens vom Zustand "Teig" in den Zustand "Kuchen" überführt werden kann. Eine zweite Methode:

```
essen()
```

verringert den Zahlenwert der Objektvariablen **stuecke** so, dass jederzeit die Anzahl der noch verbleibenden Kuchenstücke ermittelt werden kann.

4.1.3 Klasse

Klasse[2]
Eine Klasse beschriebt eine Menge von Objekten mit gleichen Eigenschaften.

Eine Klasse fasst lediglich alle Eigenschaften eines Objektes zusammen. Beispielsweise enthält die Klasse **Kuchen** alle Objektvariablen und Objektmethoden eines Kuchenobjektes. Sie ist damit vergleichbar mit einem Rezept aus einem Backbuch. Die Klasse **Kuchen** ist lediglich die Beschreibung, die Bauanleitung für unterschiedliche Kuchen mit gleichen Zutaten. Sie ist aber kein tatsächliches Kuchenobjekt mit konkreten Mengenangaben für die Zutaten. Damit wird eine neue Metaebene in die Programmierung eingezogen:

1 Versuchen Sie bitte nicht die in diesem Buch beschriebenen Kuchenobjekte real (durch tatsächliches Nachbacken) zu erzeugen! Wenn doch, sollten Sie zumindest noch die Zutat Backpulver ergänzen.
2 Eine Klasse entspricht in etwa einer Entität beim Datenbankentwurf.

4 OO-Grundlagen

Metaebene

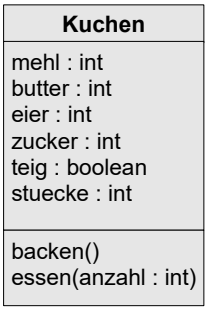
Mit den Klassen kommt eine Metaebene hinzu. Die in der Klasse zusammengefassten Objektvariablen und Methoden werden nicht mehr direkt angesprochen und ausgeführt, sondern dienen lediglich als Vorlage zum Erstellen von Objekten. Klassen, die nicht der Steuerung, sondern ausschließlich der Modellierung konkreter Objekte dienen, werden auch als Fachklassen bezeichnet.

Fachklasse

Fachklassen dienen der fachlichen Modellierung und Beschreibung konkreter Objekte auf der Metaebene.

Klassendarstellung in UML

Eine Klasse wird im UML-Klassendiagramm durch ein Rechteck dargestellt. In weiteren abgetrennten Bereichen sind Objektvariablen und -methoden definierbar. Die Datentypen folgen, durch Doppelpunkt getrennt den Variablen, der Typ des Rückgabewert den Methoden. Der Pseudodatentyp **void** ist in UML nicht definiert und wird daher grundsätzlich weggelassen.



Java-Programm

In Java sind die vorgestellten Objektvariablen und -methoden durch folgende Kasse darstellbar:

Klasse Kuchen

```
1  package grundlagen01;
2
3  public class Kuchen
4  {
5    int mehl, butter, eier, zucker;
6    boolean teig;
7    int stuecke;
8
9    public void backen()
10   {
11     teig = false;
12   }
13
14   public void essen (int anzahl)
15   {
16     stuecke = stuecke - anzahl;
17   }
18 }
```

Konkrete Ebene

Neben der Metaebene gibt es im Programm auch immer noch die konkrete Ebene. Hier können konkrete Objekte nach den Bauplänen der Funktionsklassen erzeugt werden. Auf der konkreten Programmebene befindet sich beispielsweise die **main()**-Funktion. Sie wird direkt beim Programmstart aufgerufen und ist somit die zentrale Steuerungsmethode, die die Programmabarbeitung kontrolliert und koordiniert. Doch selbst die **main()**-Funktion muss in Java innerhalb einer Klasse definiert werden. Sie unterscheidet sich aber durch das Schlüsselwort **static** von anderen objektbezogenen Methoden. Durch das Schlüsselwort **static** wird die **main()**-Funktion zur Klassenmethode, die eigenständig und objektunabhängig verwendet werden kann. Obwohl die **main()**-Funktion in jeder beliebigen Klasse stehen kann, sollten die Metaebene und die konkrete Programmebene nach Möglichkeit sauber voneinander getrennt werden. Es ist somit ratsam die **main()**-Funktion in eine eigene Klasse auszulagern. Eine solche Klasse wird auch als Steuerungsklasse bezeichnet.

Steuerungsklasse

Steuerungsklassen sind Klassen, die der Steuerung und Kontrolle einzelner Programmteile oder der zentralen Programmsteuerung dienen.

4.1.4 Objekt

Nach dem Bauplan einer Klasse können beliebig viele konkrete Objekte erzeugt werden. Da Objekte grundsätzlich zu den referenziellen Datentypen gehören erfolgt die Objekterzeugung mithilfe des **new**-Operators. Das Erzeugen der Objekte erfolgt auf der konkreten Programmebene beispielsweise in der **main()**-Funktion. Sollen beispielsweise zwei Kuchen erstellt werden, ein kleiner Kuchen mit folgenden Zutatenangaben:

```
Mehl 300g; Butter 250g; Zucker 200g; 4 Eier
```

und ein zweiter, großer Kuchen, mit denselben Zutaten, aber anderen Mengenangaben

```
Mehl 500g; Butter 500g; Zucker 500g; 6 Eier
```

dann kann die **main()**-Funktion (in einer separaten Klasse) wie folgt formuliert sein:

Steuerungsklasse mit main-Methode

```
1  package grundlagen01;
2
3  public class KuchenStart
4  {
5    public static void main(String[] args)
6    {
7      Kuchen klein = new Kuchen();
8      Kuchen gross = new Kuchen();
9
10     klein.mehl   = 300; klein.butter  = 250;
11     klein.zucker = 200; klein.eier    = 4;
12     klein.teig  = true; klein.stuecke = 12;
13
14     gross.mehl   = 500; gross.butter  = 500;
15     gross.zucker = 500; gross.eier    = 6;
16     gross.teig  = true; gross.stuecke = 16;
17   }
18 }
```

Die Zeile 7 und 8 des Programms erzeugen mithilfe des **new**-Operators die beiden Kuchen **klein** und **gross** und stellen entsprechend Speicherplatz zur Verfügung. Jeder der beiden Kuchen erhält dabei, gemäß den Vorgaben der Klasse **Kuchen**, seine eigene **mehl**, **butter**, **eier**, **zucker**, **teig** und **stuecke** Variable. Deshalb müssen die Objektvariablen auch objektbezogen angespro-

chen werden! Will man beispielsweise für den Kuchen **klein** die Mengenangabe für Mehl auf **350** ändern, so erreicht man die Objektvariable **mehl** über den Objektnamen **klein**.

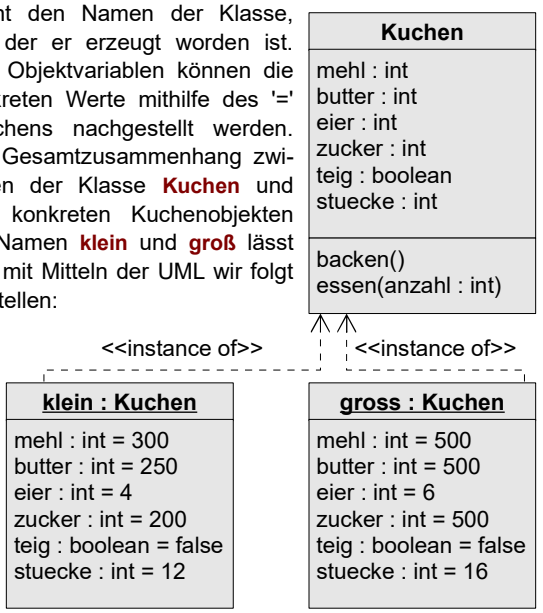
```
klein.mehl = 350;
```

Auch die Objektmethoden **backen()** und **essen()** werden eindeutig den Kuchenobjekten zugeordnet und sind objektbezogen aufzurufen. Möchte man beispielsweise den Kuchen **gross** backen und anschließend 3 Stücke davon essen, so erfolgen auch die Methodenaufrufe über den Objektnamen **gross**.

```
gross.backen();
gross.essen(3);
```

UML-Klassendiagramm / Objektdiagramm

In UML sind Objekte ähnlich wie Klassen durch ein Rechteck darzustellen. Allerdings wird der Objektname unterstrichen und enthält durch einen Doppelpunkt getrennt den Namen der Klasse, aus der er erzeugt worden ist. Den Objektvariablen können die konkreten Werte mithilfe des '=' -Zeichens nachgestellt werden. Der Gesamtzusammenhang zwischen der Klasse **Kuchen** und den konkreten Kuchenobjekten mit Namen **klein** und **groß** lässt sich mit Mitteln der UML wir folgt darstellen:



4.1.5 Ausgabe

Das bisher vorgestellte Beispielprogramm ist prinzipiell lauffähig, enthält aber keinerlei Bildschirmausgaben. Daher lässt sich die Funktionsweise nicht konkret nachvollziehen. Zur Kontrolle der Objektvariablen ist es sinnvoll eine eigene objektbezogene Methode

```
showObjektVar()
```

zu schreiben, die nichts anderes macht als alle Objektvariablen auf dem Bildschirm auszugeben.

4.1.6 Beispielprogramm

Kuchen

```
1  package grundlagen01;
2
3  public class Kuchen
4  {
5   int mehl, butter, eier, zucker, stuecke;
6   boolean teig;
7  //Objektmethoden
8   public void backen()
9   { teig = false;  }
10  public void essen (int anzahl)
11  { stuecke = stuecke - anzahl;  }
12
13  public void showObjektVar()
14  {
15   System.out.println("***** Kuchen *****");
16   System.out.println("Mehl: "    + mehl);
17   System.out.println("Butter: "  + butter);
18   System.out.println("Eier: "    + eier);
19   System.out.println("Zucker: "  + zucker);
20   if(teig)  System.out.println("Teig");
21   else      System.out.println("Kuchen");
22   System.out.println("Stücke: "+ stuecke);
23  }
24 }
```

```
25 package grundlagen01;
26
27 public class KuchenStart
28 {
29  public static void main(String[] args)
30  {
31   Kuchen klein = new Kuchen();
32   klein.mehl   = 300; klein.butter  = 250;
33   klein.zucker = 200; klein.eier    = 4;
34   klein.teig  = true; klein.stuecke = 12;
35   klein.showObjektVar();
36   Kuchen gross = new Kuchen();
37   gross.mehl   = 500; gross.butter  = 500;
38   gross.zucker = 500; gross.eier    = 6;
39   gross.teig  = true; gross.stuecke = 16;
40   gross.showObjektVar();
41   gross.backen();
42   gross.essen(3);
43   gross.showObjektVar();
44  }
45 }
```

```
1  ***** Kuchen *****
2  Mehl: 300
3  Butter: 250
4  Eier: 4
5  Zucker: 200
6  Teig
7  Stücke: 12
8  ***** Kuchen *****
9  Mehl: 500
10 Butter: 500
11 Eier: 6
12 Zucker: 500
13 Teig
14 Stücke: 16
15 ***** Kuchen *****
16 Mehl: 500
17 Butter: 500
18 Eier: 6
19 Zucker: 500
20 Kuchen
21 Stücke: 13
```

Aufgabe 22.4.1