# UML: Unified Modeling Language

Objektorientierte standardisierte Modellierungssprache

Ziele:

1. Zusammenhänge in der Softwareentwicklung grafisch zu veranschaulichen
2. Internationaler Standard,   
   damit ein Programmierer die Diagramme eines andern verstehen kann
3. Übergang von der grafischen Darstellung zur Implementierung ermöglichen

Darstellung durch verschiedene Typen von Diagrammen

Ähnlich wie PAPs oder Struktogramme,

jedoch umfangreicher u. detaillierter

Zwei Typen von Diagrammen:

1. Strukturdiagramme
2. Verhaltensdiagramme

Strukturdiagramme beschreiben -wie der Name vermuten lässt-

die Struktur von etwas, hier von Klassen oder Objekten

Verhaltensdiagramme beschreiben das Verhalten der Software

in Interaktion mit der Umwelt.

Beispiele laut Lehrplan:

Für Strukturdiagramme: Klassen- u. Objektdiagramme

Für Verhaltensdiagramme: Sequenz u. Use-Case oder Anwendungsfall-Diagramme

## Klassendiagramme

Klassendiagramme als wichtigster Vertreter von Strukturdiagrammen.

Es soll der Aufbau der Klassen dargestellt werden

sowie der Zusammenhang zwischen verschiedenen Klassen,

z.B. eine Vererbungsbeziehung.

Die Betrachtung kann -je nach Fokus- auf unterschiedlichem

Abstraktionsniveau stattfinden.

Ein Klasse wird dargestellt durch ein Rechteck:

1. Zuerst der Klassenname, fett gedruckt u. zentriert
2. Dann eine horizontale Trennlinie,  
   gefolgt von den Attributen
3. Weitere Trennlinie,  
   dann die Methoden

Auf hoher Abstraktionsstufe kann die Klasse auf den Klassennamen reduziert werden,

wenn nicht der Aufbau, sondern nur die Interaktion mit anderen Klassen

in diesem Diagramm veranschaulicht werden soll.

Auf niedrigem Abstraktionsniveau können alle Details der Attribute angegeben werden,

also neben Name u. Typ auch Zugriffsmodifizierer u.ä.

Attribute:

Die Angabe der Attribute erfolgt anders als bei Java:

Es soll ein int-Attribut "a" deklariert werden:

Java: "int a" in UML: "a: int"

Mit einem Plus- oder Minuszeichen (oder einer Raute oder geschwungenen Linie)

vor dem Attributsnamen kann der Zugriffsmodifizierer angegeben werden,

z.B. + für public, - für private

Methoden:

Methoden können mit Parameterliste u. Rückgabetyp verfeinert werden.

Auf hohem Abstraktionsniveau würde nur der Name stehen.

Hat die Methode eine Parameterliste,

würden die Parameter angegeben werden wie bei der Deklaration.

Habe z.B. eine Methode "m()" einen int-Parameter "i", würde dieser geschrieben als:

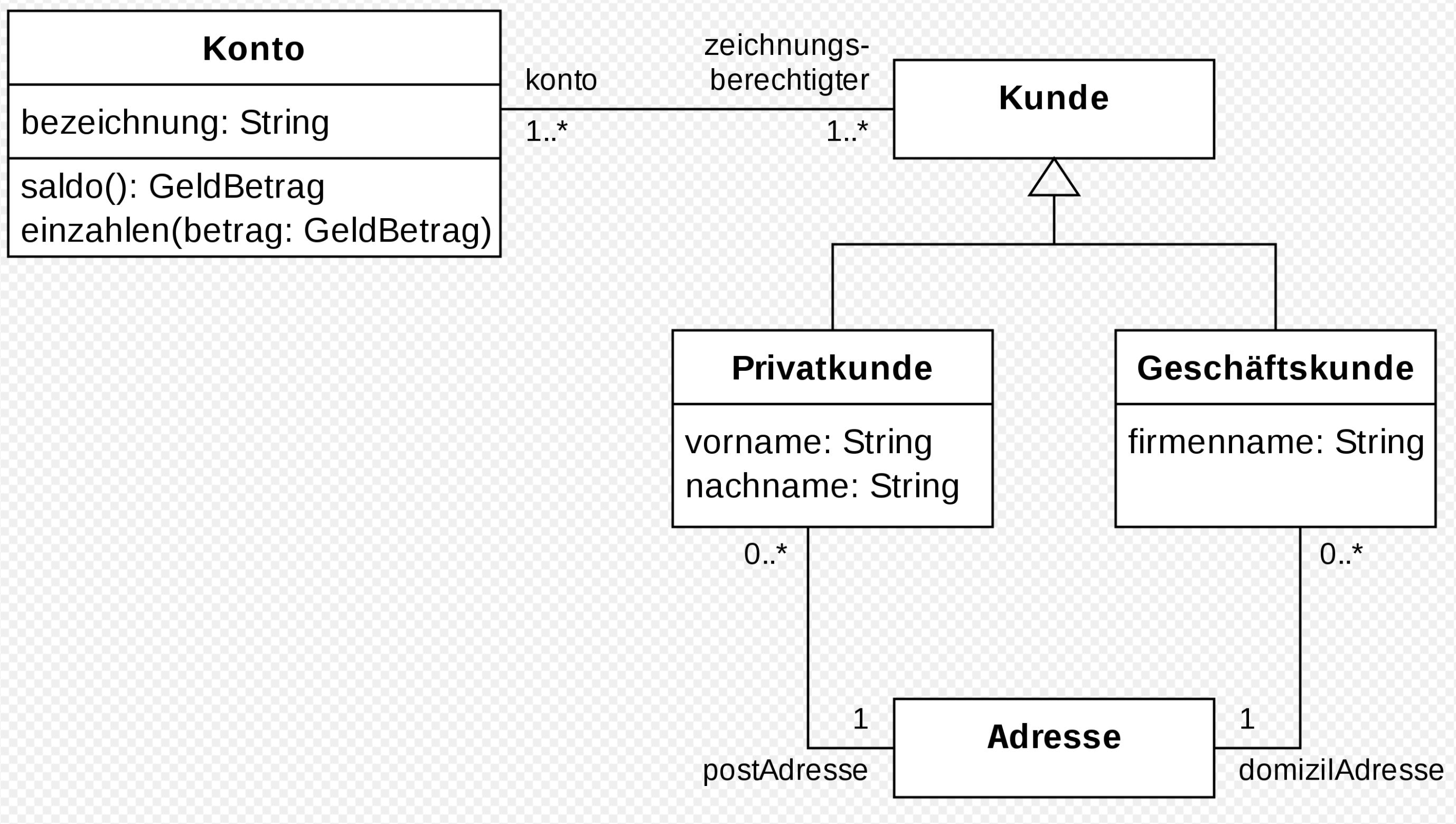
Java: m(int i) UML: m(i: int)

Habe eine Methode einen Rückgabewert, z.B. boolean, würde das geschrieben:

Java: boolean m(int i) UML: m(i:int) : boolean

Die Vererbungsbeziehung wird mit einer durchgezogenen Linie   
u. einem ungefüllten Pfeil in Richtung Superklasse dargestellt.

Beispiel für ein Klassendiagramm aus Wiki:



(Bild von Wikipedia)

Hier ist nicht nur eine Klasse dargestellt,

sondern mehrere, die in Beziehung miteinander stehen.

Betrachten wir dies oben an der Beziehung zwischen "Konto" u. "Kunde".

Die durchgezogene Linie besagt, dass es eine Beziehung gibt.

Details dazu:

* Die Bezeichnungen an den Seiten gibt an, wie die Beziehung zu lesen ist:  
  "Ein Konto hat einen zeichnungsberechtigten Kunden"
* Die Zahlen geben das quantitative Verhältnis an.  
  Möglich sind fixe Zahlen wie die 1, was besagen würde, dass ein 1:1-Verhältnis besteht.  
  Hier haben wir: 1..\*  
  Das besagt, dass wir 1 oder beliebig viele Verbindungen haben.  
  Hier: 1 Konto hat mindestens einen aber beliebig viele Zeichnungsberechtigte.  
  Umgekehrt: Jeder Kunde hat 1 oder beliebig viele Konten.  
  Hätte er keines, wäre es kein Kunde.

Daneben haben wir eine Vererbungsbeziehung:

Sowohl Privat- als auch Geschäftskunden sind Kunden.

Sie haben neben mglw. vorhandenen gemeinsamen Attributen beide ihre eigenen:

Hier: Für Privatkunden Name u. Vorname, für Geschäftskunden den Firmennamen.

Beide haben Sie eine Adresse:

Der Privatkunde eine postAdresse, der Geschäftskunde eine domizilAdresse.

Ein Privatkunde hat genau eine Postadresse, ein Geschäftskunde genau eine Domiziladresse.

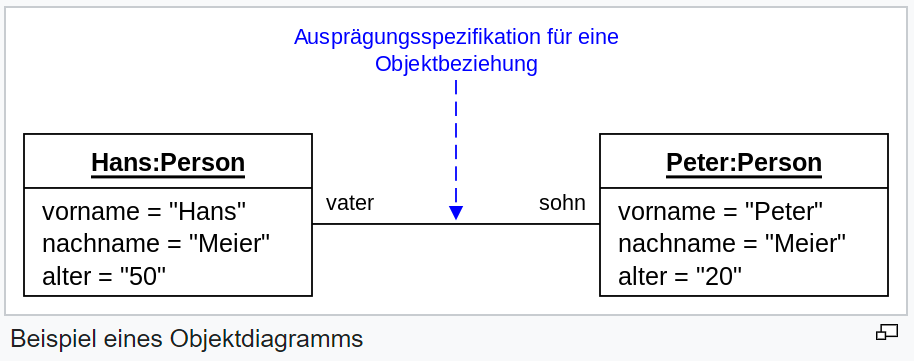
Umgekehrt: An einer Adresse können keine oder beliebig viele Geschäfts- oder Privatkunden wohnen.

## Objektdiagramme

Zweite Form von Strukturdiagrammen: Objektdiagramme

Sie sind ähnlich wie die Klassendiagramme.

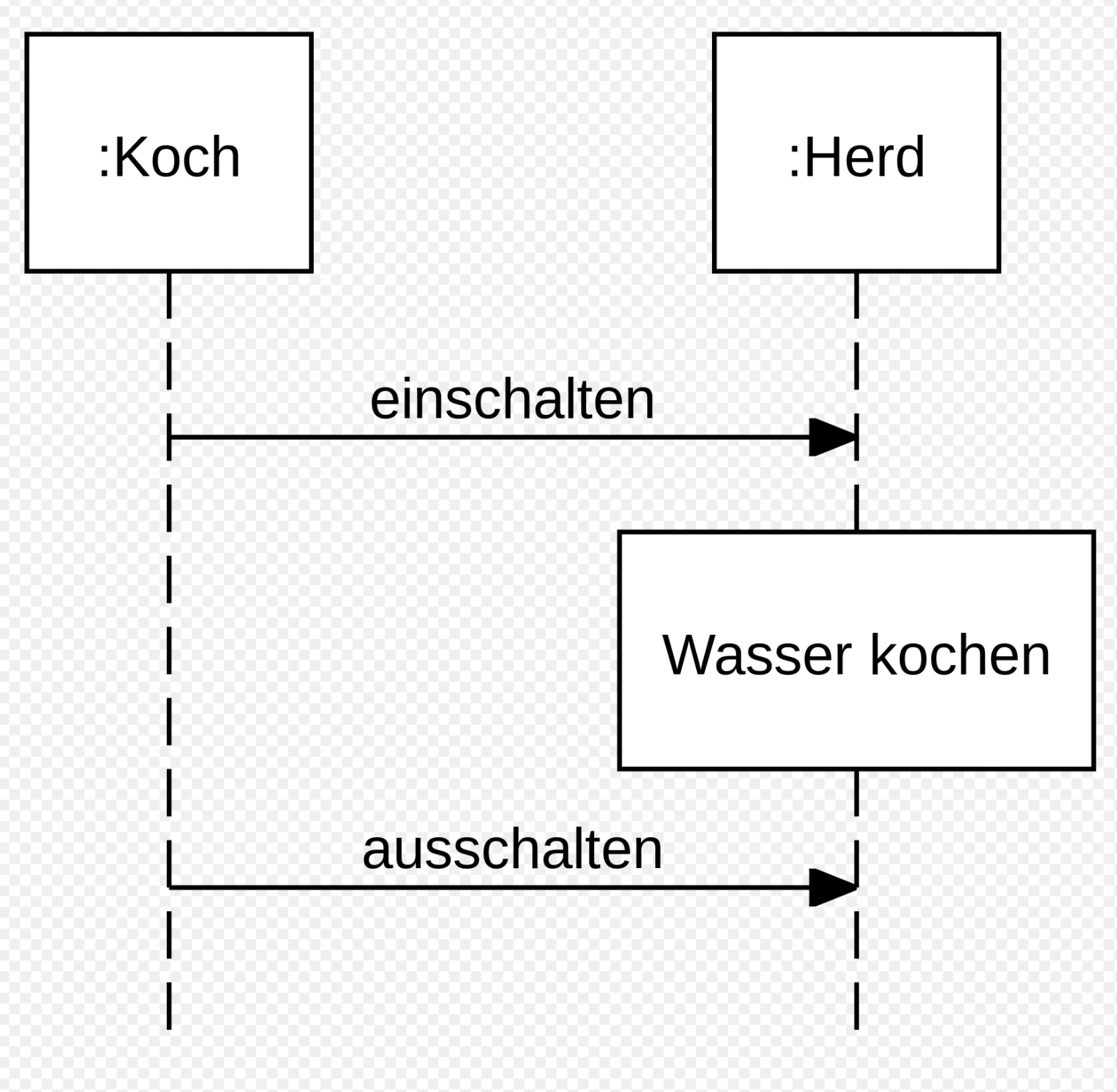
Siehe Wiki



(Bild von Wikipedia)

## Sequenzdiagramm

Beispiel für ein Sequenzdiagramm als erstem Vertreter von Verhaltensdiagrammen:



(Bild von Wikipedia)

Wir haben einen Koch u. einen Herd jeweils mit einer Lebenslinie.

Der Koch kann die Aktion "einschalten" auslösen.

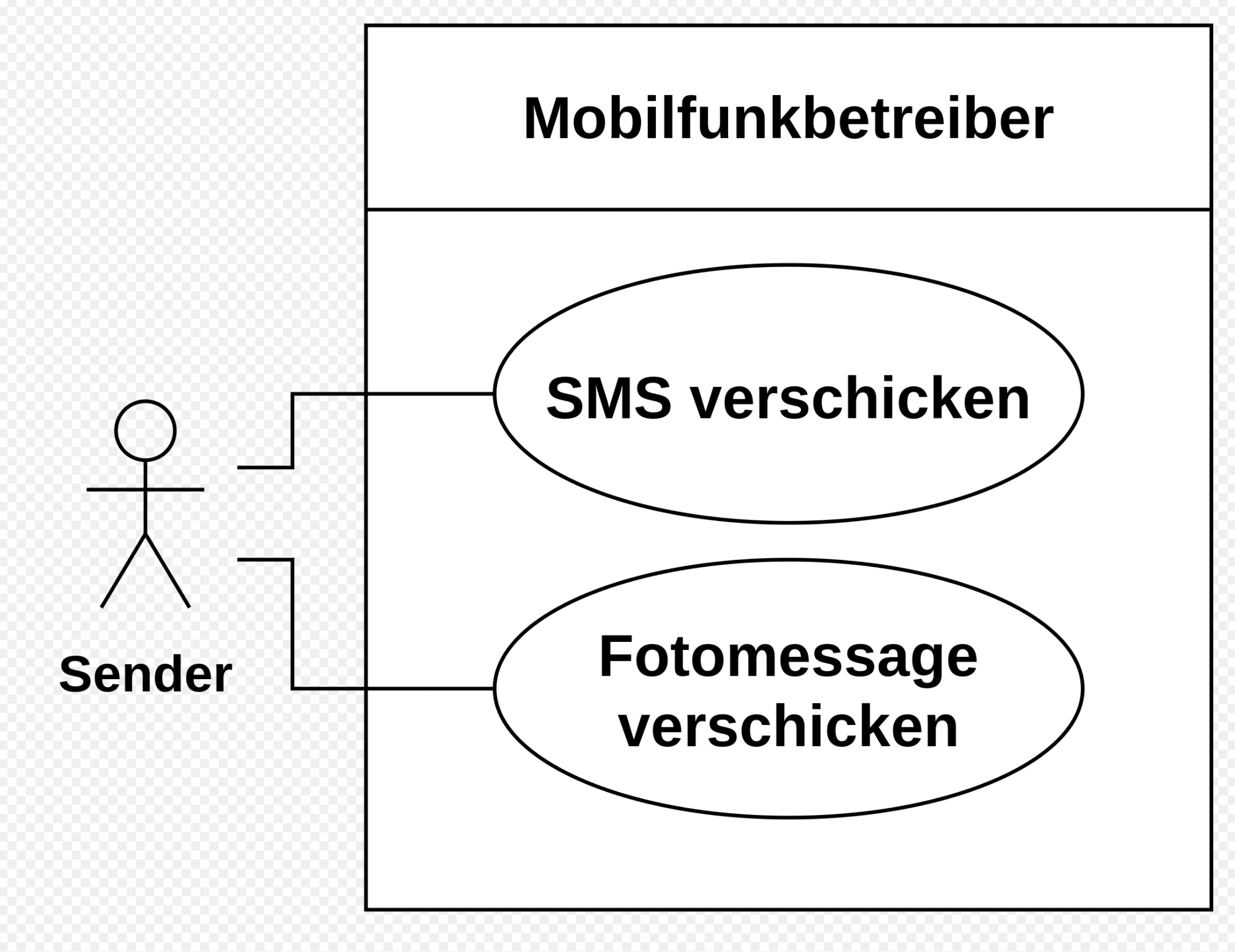
Auf dem Herd wird Wasser gekocht.

Danach schaltet der Koch den Herd aus.

## Use-Case oder Anwendungsfall-Diagramm

Beispiel für Use-Case oder Anwendungsfall-Diagramm als zweiten Vertreter

eines Verhaltensdiagramms:



(Bild von Wikipedia)

Hier stehen die Aktionen im Vordergrund,

die der Benutzer gegenüber einem System anfordern kann,

nicht jedoch der Aufbau des Systems.

Der Benutzer kann die Aktionen auslösen,

eine SMS oder eine MMS zu verschicken.