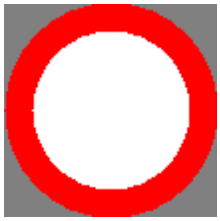
Modellierung und Codierung



Ein **Modell** ist die vereinfachte Beschreibung der Wirklichkeit.
Zum Beispiel Verkehrszeichen beschreiben auf einfache Art Vorschriften für Verhaltensweisen im Straßenverkehr.

Auch die Darstellung der Attribute eines Objekts in einem Objektdiagramm ist ein Modell:
Sachverhalte werden vereinfacht und beschrieben.

- Das Entwickeln einer vereinfachten Beschreibung der Wirklichkeit nennt man Modellierung.

Verbotszeichen:KREIS
Mittex=0 Mittey=0 Radius=50 Füllfarbe=Weiß Randfarbe=Rot Randstärke=15

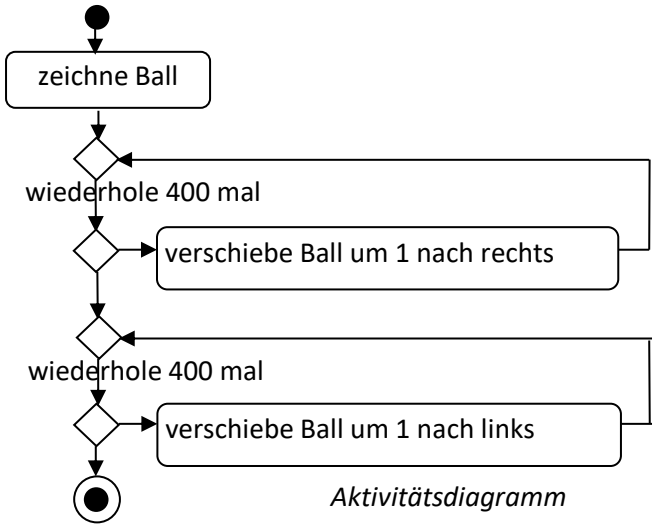
Objektdiagramm

Klassendiagramme und Aktivitätsdiagramme
Mit Hilfe von Diagrammen werden verschiedene Perspektiven von Modellen dargestellt.

- *Klassendiagramme* ermöglichen die Darstellung der Struktur eines Modells.
- In *Objektdiagrammen* werden aktuelle Zustände festgehalten.
- In *Aktivitätsdiagrammen* können Verhaltensweisen von Objekten beschrieben werden.
  Das Aktivitätsdiagramm zeigt den Ablauf der Programmanweisungen:

Kreis
Mittex Mittey Füllfarbe Randfarbe Randstärke
mittelpunktSetzen() radiusSetzen() füllfarbeSetzen() randfarbeSetzen() randstärkeSetzen()

Klassendiagramm

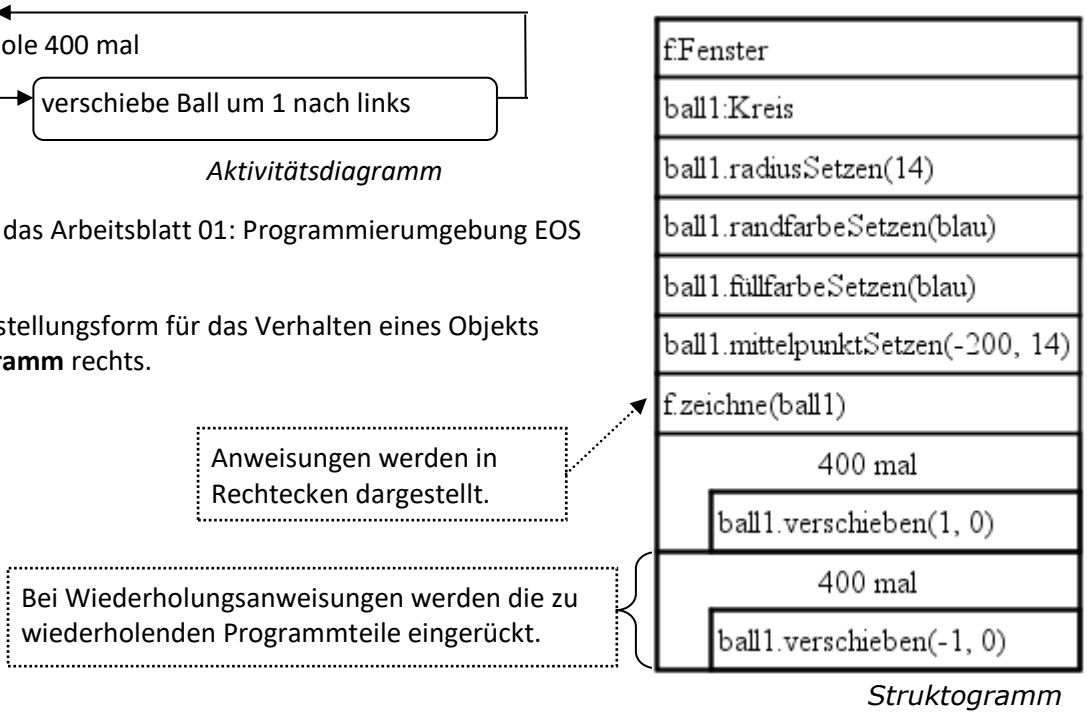


Aktivitätsdiagramm

A Bearbeite das Arbeitsblatt 01: Programmierumgebung EOS

Struktogramme

Eine andere Darstellungsform für das Verhalten eines Objekts ist das **Struktogramm** rechts.



Struktogramm

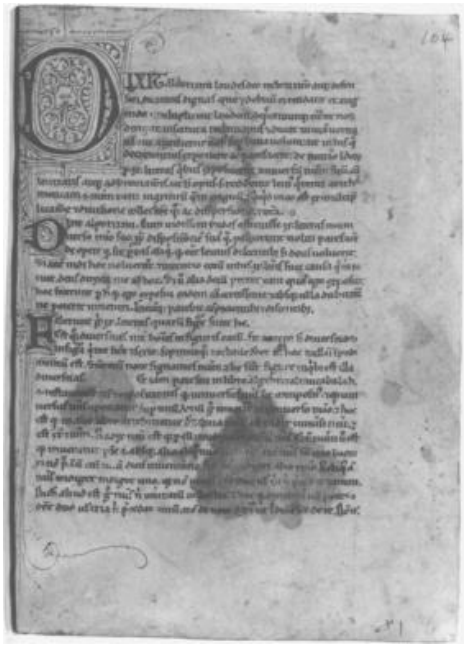
A Bearbeite das Arbeitsblatt 02: Wiederholung von Grundbegriffen

Algorithmus

Das Aktivitätsdiagramm enthält eine Handlungsvorschrift zur Lösung einer Art von Problemen, z. B. einen Ball „rollen“ zu lassen.

- Eine Handlungsvorschrift zur Lösung einer Klasse von Problemen bezeichnet man als **Algorithmus**.

Der Begriff Algorithmus stammt von dem Namen des Autors eines arabischen Lehrbuchs über das Rechnen mit indischen Ziffern (um 825): Muhammed Al Chwarizmi.
Die lateinische Übersetzung dieses Buchs begann mit den Worten „Dixit Algorismi“ („Algorismi hat gesagt“).
Über die Jahrhunderte hat sich daraus das Wort Algorithmus entwickelt.



Seite aus einer Übersetzung Muhammed Al Chwarizmis Lehrbuchs
Quelle: http://de.wikipedia.org/wiki/Bild:Dixit_algorizmi.png

Codierung

- Bei der *Programmierung* wird der Algorithmus in einer Folge von Anweisungen formuliert, die der Computer verarbeiten kann. Das nennt man **Codierung**.
  Unter *Implementierung* versteht man in der Softwareentwicklung das Erstellen eines Modells und Umsetzen in ein Computerprogramm.

Beispielsweise wurde im Arbeitsblatt 01 die Bewegung eines Balles mit Hilfe eines Programmierwerkzeugs *implementiert*. Dazu werden die folgenden Arbeitsschritte ausgeführt:

- Entwickeln von *Klassen-* bzw. *Objektdiagrammen* zur Darstellung der Struktur und der Attributwerte.
- Entwickeln von *Aktivitätsdiagrammen* zur Darstellung der Verhaltensweisen von Objekten bzw. der Algorithmen.
- Formulierung des Algorithmus in einer Folge von Anweisungen, die der Computer verarbeiten kann (*Codierung*).

Methoden

In umfangreicheren Programmen kann das gesamte Problem in Teilprobleme zerlegt werden. Dadurch wird der Algorithmus übersichtlicher gestaltet.
Dafür kann man eigene Methoden festlegen.
Eine Methode wird nur ausgeführt, wenn sie aufgerufen wird. Zur Deklaration einer eigenen Methode in EOS werden die Programmzeilen zwischen die Anweisungen `Methode ...` und `Ende` geschrieben.
Wenn die Anweisungen für eine Methode in einer Programmiersprache formuliert werden, spricht man auch von einer **Funktion**.

```
...
wiederhole 3 mal
   wiederhole 720 mal
      seifenkiste1.verschieben(1,0)
   *wiederhole
   warte()
   wiederhole 720 mal
      seifenkiste1.verschieben(-1,0)
   *wiederhole
   warte()
*wiederhole

Methode warte
   wiederhole 2000 mal
      //warte
   *wiederhole
Ende
```

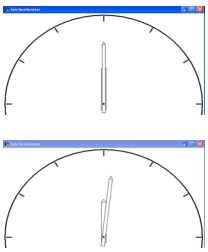
- Um Abläufe in **sinnvolle Teilschritte** zu gliedern, kann man eigene **Methoden** festlegen.

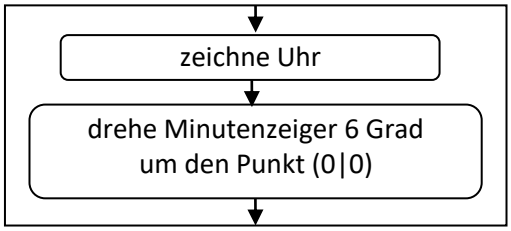
- Bearbeite das Arbeitsblatt 03: Übungen zur Programmierung in EOS
- Bearbeite das Arbeitsblatt 04: Attributwerte

Die algorithmischen Grundstrukturen *Sequenz* und *Wiederholung*

Sequenz

- Eine Folge von Anweisungen, die nacheinander abgearbeitet werden, nennt man **Sequenz**.

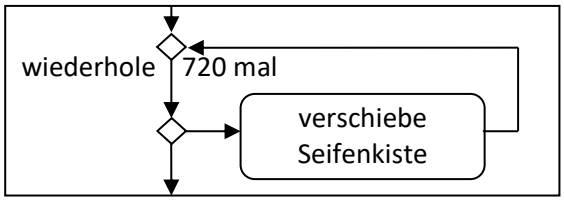




Sequenz

Wiederholungsstrukturen

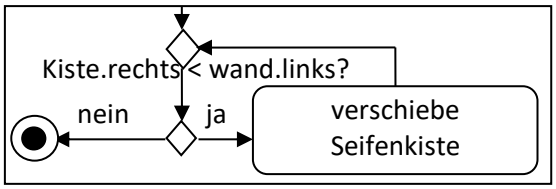
Wird ein Teil des Programmcodes mehrfach abgearbeitet, liegt eine **Wiederholungsstruktur** vor.



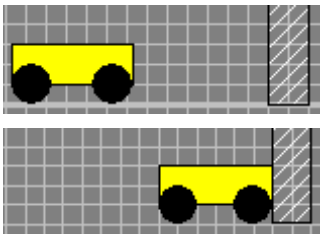
Zählschleife

- **Zählschleifen**:
  Bei einer Zählschleife ist die Anzahl der Wiederholungen festgelegt.

- **Wiederholung mit Anfangsbedingung**:
  Wenn der Programmablauf auf Grund einer *Bedingung* wiederholt wird, spricht man von einer **bedingten Wiederholung**.
  Prüft man die Bedingung zur Ausführung einer Sequenz *vor*her ab, liegt eine *Wiederholung mit Anfangsbedingung* vor.



Wiederholung mit Anfangsbedingung

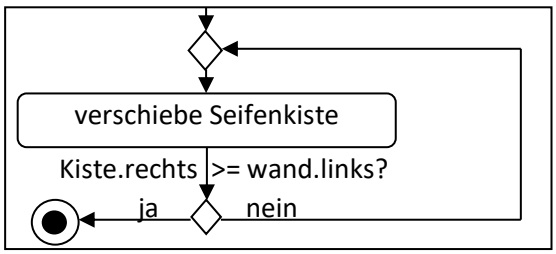


```
solange kiste.rechts < xwand.links tue
   seifenkiste.verschieben(1,0)
*solange
```

In einer **Bedingung** werden die Attributwerte abgefragt und in einer Aussage formuliert.
Aussagen sind Sätze, die Sachverhalte beschreiben und denen man als Wahrheitswert „wahr“ oder „falsch“ zuordnen kann. Die Aussage kann auch durch eine Frage ersetzt werden, die mit ja/nein zu beantworten ist. (siehe Aktivitätsdiagramm oben)

A Bearbeite das Arbeitsblatt 05: Die algorithmische Grundstruktur *Wiederholung* und Variablen

- **Wiederholung mit Endebedingung**:
  Prüft man die Bedingung zur Ausführung einer Folge von Anweisungen ab, *nachdem* die Sequenz bereits mindestens einmal durchlaufen wurde, liegt eine *Wiederholung mit Endebedingung* vor.



Im Beispiel würde die Seifenkiste mindestens einmal nach rechts verschoben, auch wenn sie schon an der Mauer stünde. Dieser Fall wird aber durch das Intervall für die Zufallszahl ausgeschlossen, weshalb der Algorithmus auch so zulässig ist.

```
wiederhole
   seifenkiste.verschieben(1,0)
*wiederhole bis kiste.rechts >= wand.links
```

Wiederholung mit Endebedingung

Variablen

Für die meisten Programmieranwendungen müssen Werte zur weiteren Verwendung gespeichert werden.
Dafür verwendet man **Variablen** (siehe Klassendiagramm rechts).

Ballspiel
dx:Integer f:Fenster spielfeld:Rechteck ball1:Kreis

- Ein Speicherbereich, der verschiedene Werte annehmen kann, wird **Variable** genannt.

In der *Variablendeklaration* wird festgelegt, welche *Art von* Wert zugewiesen wird (z. B. `dx:Integer`).
In Programmiersprachen steht eine Reihe verschiedener Klassen zur Verfügung, mit denen dieser **Datentyp** einer Variablen festgelegt werden kann. Im Beispiel wird die Klasse **Integer** als Datentyp für die Variable **dx** verwendet, weil *ganzzahlige* Werte zugewiesen werden sollen.

Wichtige Datentypen sind:

Datentyp	Erläuterung	Beispiele für Werte
Integer	Ganze Zahlen	2; -5; ...
Real	Kommazahlen	0,75; -2,5; ...
String	Zeichenketten	"Hallo"; 'PLZ: 81479'
Boolean	Wahrheitswert	wahr; falsch

A Bearbeite das Arbeitsblatt 06: Zufallszahlen

Im Beispiel *SeifenkisteMitWand* wird eine Zufallszahl als x-Koordinate des linken Rands einer Wand in der Variablen `xwand` gespeichert. Da die Zufallszahl ganzzahlig sein muss, ist der *Datentyp* `Integer`.
(vgl. Klassendiagramm rechts)
In Programmierwerkzeugen gibt es meist Methoden, die eine Zufallszahl bilden, z. B. `zufall()` oder auch engl. *random()*.

SeifenkisteMitWand
xwand:Integer seifenkiste1:Gruppe kiste1:Rechteck rad01:Kreis rad02:Kreis wand1:Rechteck

Da innerhalb von Computerprogrammen nur eindeutige Entscheidungen möglich sind, kann man sich das Erzeugen der Zufallszahl nicht wie das Werfen eines Würfels vorstellen. Vielmehr wird aus dem Wert eines zufälligen Ereignisses, zum Beispiel der aktuellen Uhrzeit, mit Hilfe eines geeigneten Algorithmus eine „Zufallszahl“ berechnet.

Parameter und Rückgabewerte

- Wenn ein Wert an eine Methode übergeben wird, bezeichnet man das als **Parameter**, z. B. `zufall(1,6)` erzeugt eine zufällige ganze Zahl von **1** bis **6**.
- Methoden können auch Werte an die aufrufende Methode zurückgeben, wo sie beispielsweise in einer Variablen gespeichert werden können.

```
...
   Wuerfel:=zufall(1,6)
...
```

Eine Zufallszahl von 1 bis 6 wird erzeugt und an die aufrufende Methode zurückgegeben.

Mit diesen Parametern könnte man beispielsweise den Computer „würfeln“ lassen.

Lokale Variablen

- In Programmiersprachen sind Variablen, die innerhalb einer Methode definiert werden, auch nur in der jeweiligen Methode bekannt. Auf die Werte dieser Variablen kann in anderen Methoden nicht zugegriffen werden, weshalb man von **lokalen Variablen** spricht.

Texte

Texte (Datentyp *String*) werden in Anführungszeichen ("Text") oder Hochkomma ('Text') geschrieben. Mit Hilfe des Additionszeichens (+) können mehrere Texte verkettet („aneinandergefügt“) werden.

Beispiel: `position.zeileHinzufügen ('x-Position der Wand: ' + xwand)`

Operationen
Daten gleichen Typs können miteinander verknüpft werden. Als Ergebnis entstehen daraus neue Daten. Das Verknüpfungszeichen (z. B. +) nennt man Operator. Wichtige Operationen sind:

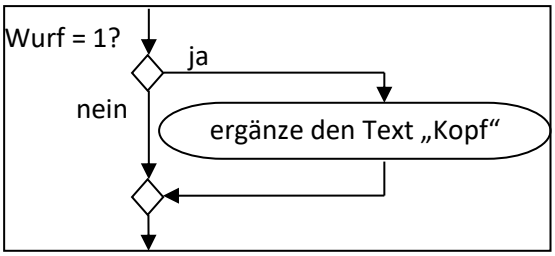
Mathematische bzw. sprachliche Darstellung	Darstellung in der IT	Bedeutung	Beispiel	Ergebnis
+, -	+, -	Summe und Differenz	6 + 3	9
"mal", "geteilt"	*, /	Produkt und Quotient	6 / 3	2
x^y	x^y	Potenz	2 ^ 4	16
+ (Texte)	'Text1'+'Text2'	Verkettung von Texten	'Haus'+'katze'	Hauskatze
			'6+3='+(6+3)	6+3=9
=	= oder ==	ist gleich	5 = 6	falsch
>, <	>, <	ist größer als, kleiner als	5 < 6	wahr
≥	>=	ist größer oder gleich	5 >= 6	falsch
≤	<=	ist kleiner oder gleich	5 <= 6	wahr
≠	<> oder !=	ist nicht gleich	5 != 6	wahr

Die algorithmische Grundstruktur *Auswahl*

A Bearbeite das Arbeitsblatt 07: Die algorithmische Grundstruktur *Auswahl*

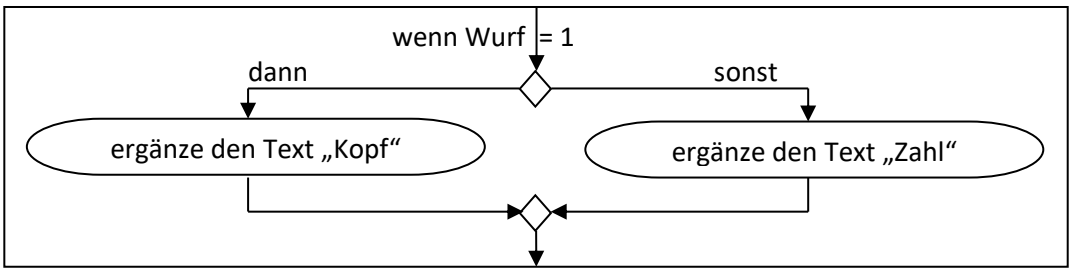
Wenn der Programmablauf auf Grund einer Bedingung verzweigt, spricht man von einer **Auswahlstruktur**.

- **Einseitige Auswahl**:
  Bei der *einseitigen Auswahl* wird eine Sequenz des Programms nur ausgeführt, wenn eine Aussage den Wahrheitswert `wahr` hat.
  Ist der Wahrheitswert `falsch`, wird die betreffende Sequenz nicht ausgeführt.



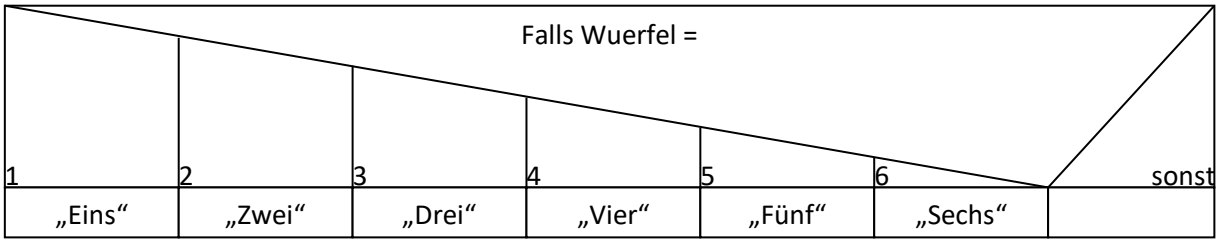
Einseitige Auswahl

- **Zweiseitige Auswahl**:
  Bei der *zweiseitigen Auswahl* wird entweder die eine oder die andere von zwei Sequenzen des Programms ausgeführt („*wenn-dann-sonst*“).



Zweiseitige Auswahl

- **Mehrseitige Auswahl**:
  Bei der *mehrseitigen Auswahl* wird genau eine aus mehreren Sequenzen des Programms ausgeführt. Das lässt sich in einem Struktogramm übersichtlicher darstellen als in einem Aktivitätsdiagramm:

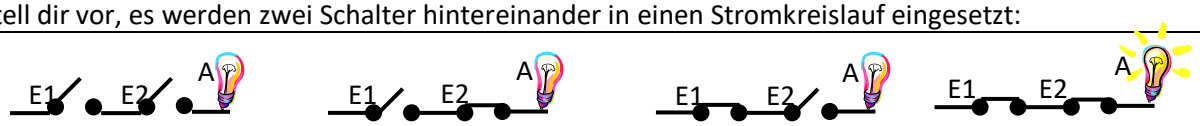


Mehrseitige Auswahl

Logische Operationen

- **Logische Funktionen** ermöglichen die Verknüpfung von Aussagen:
  (vgl. A8 - Binäre Grundschaltungen und Wertetabellen)

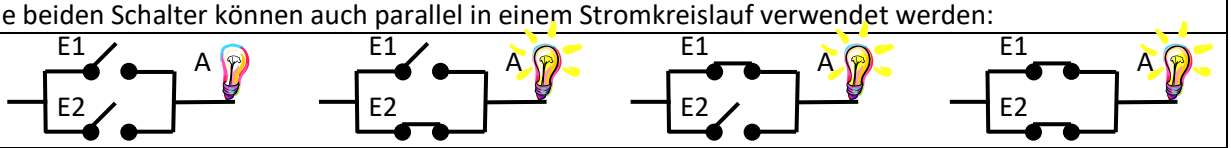
Stell dir vor, es werden zwei Schalter hintereinander in einen Stromkreislauf eingesetzt:



Die möglichen Aussagen über die Zustände der Schalter sind:			E1	E2	A
E1 offen (0)	E2 offen (0)	A aus (0)	0	0	0
E1 offen (0)	E2 geschlossen (1)	A aus (0)	0	1	0
E1 geschlossen (1)	E2 offen (0)	A aus (0)	1	0	0
E1 geschlossen (1)	E2 geschlossen (1)	A an (1)	1	1	1

So wie diese Schaltung funktioniert die **AND-Funktion**: Beide Schalter müssen geschlossen sein, damit die Lampe brennt. (Die Schreibweise für E1 **und** E2 lautet: **A = E1 ∧ E2**)

Die beiden Schalter können auch parallel in einem Stromkreislauf verwendet werden:



So wie diese Schaltung funktioniert die **OR-Funktion**:
Die Lampe brennt,
- sobald einer der beiden Schalter geschlossen ist
- oder beide Schalter geschlossen sind.

(Die Schreibweise für E1 **oder** E2 lautet: **A = E1 ∨ E2**)

E1	E2	A
0	0	0
0	1	1
1	0	1
1	1	1

Genauso können Wahrheitswerte mit *und* bzw. *oder* verknüpft werden. Die Wertetabellen sind identisch. Dafür muss die 0 durch *falsch* (f) und die 1 durch *wahr* (w) ersetzt werden.
Zum Beispiel die Aussage in der Auswahlstruktur

```
wenn ball1.mittex=schlaeger1.mittex und
     ball1.mittey>schlaeger1.mittey-90 und
     ball1.mittey<schlaeger1.mittey+90 dann
```

wird wahr, wenn **alle** Bedingungen wahr sind (vgl. Wertetabelle rechts).

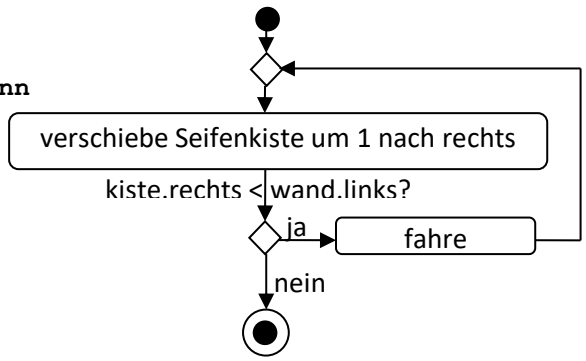
E1	E2	E1∧E2
f	f	f
f	w	f
w	f	f
w	w	w

A Bearbeite das Arbeitsblatt 08: Logische Operationen

Rekursion

- Als **Rekursion** bezeichnet man eine Programmiertechnik, in der eine Funktion sich selbst aufruft. Jeder Aufruf der rekursiven Funktion muss sich in endlich vielen Schritten auflösen lassen, sie darf nicht in eine Endlosschleife geraten.

```
methode fahre
    seifenkiste1.verschieben(1,0)
    wenn kiste1.rechts < wand1.links dann
        fahre ()
    *wenn
Ende
```



Aktivitätsdiagramm `fahre`

A Bearbeite das Arbeitsblatt 09: Rekursion