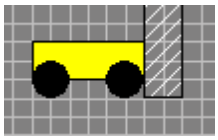
Die algorithmische Grundstruktur *Wiederholung*

Wiederholung mit Anfangsbedingung

1. Das Programm *Seifenkiste* aus Arbeitsblatt 01 soll dahingehend erweitert werden, dass eine Wand eingezeichnet wird und die Seifenkiste vom linken Rand aus nur noch bis zu dieser Wand fährt (seifenkiste6.eos).
   (Vorlagedatei: v01-seifenkiste2.eos)
   vgl. .\261-materialien\seifenkiste\06-seifenkiste6.eos



```
f:FENSTER
wand1:RECHTECK
seifenkiste1:GRUPPE
kiste1:RECHTECK
...
wand1.linksObenSetzen(300,200)
wand1.rechtsUntenSetzen(320,0)
wand1.füllartSetzen(schraffiert)
f.zeichne(seifenkiste1)
f.zeichne(wand1)
solange kiste1.rechts < wand1.links tue
   seifenkiste1.verschieben(1,0)
*solange
```

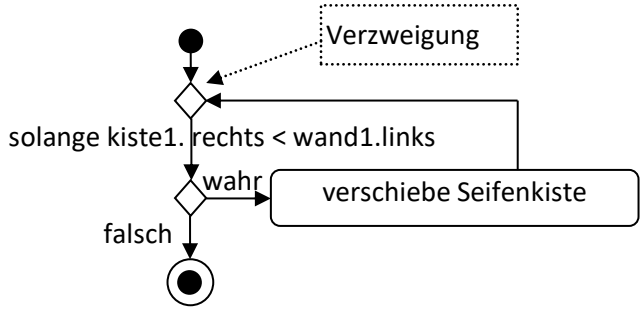
Man könnte abzählen oder berechnen, wie oft die Seifenkiste verschoben werden muss, bis sie an die „Wand“ anstößt. Diese Arbeit kann man sich aber sparen, wenn man statt der *Zählschleife* (`wiederhole n mal ... *wiederhole`) eine **Wiederholung mit Anfangsbedingung** verwendet:

```
solange <Bedingung> tue
  <Anweisungen>
*solange
```

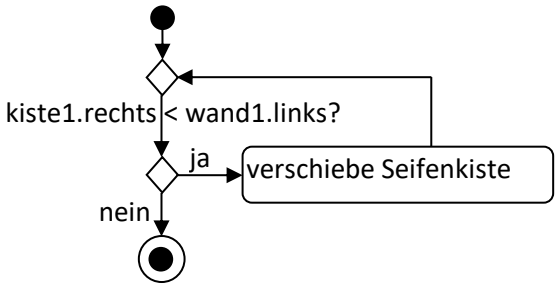
Dafür werden die Attributwerte abgefragt und in einer **Aussage** formuliert:
`kiste.rechts` enthält den x-Wert des rechten Rands der Seifenkiste.
In `wand.links` ist der x-Wert des linken Rands der Wand gespeichert.
Solange `kiste.rechts` kleiner als `wand.links` ist, fährt die Seifenkiste bzw. wenn beide Werte gleich sind, stößt die Seifenkiste an der Wand an.

Der große Vorteil ist, dass nicht von vorneherein feststehen muss, wie oft die Wiederholung durchlaufen werden soll.

Aktivitätsdiagramm für die Wiederholung mit Anfangsbedingung:



- **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 rechts)



Wichtige Verhältniszeichen für die Formulierung von Aussagen in EOS sind:

Darstellung in der Mathematik	Darstellung in der IT	Bedeutung
=, >, <	=, >, <	ist gleich, größer als, kleiner als
≥	>=	ist größer oder gleich
≤	<=	ist kleiner oder gleich
≠	<>	ist nicht gleich

Variablen

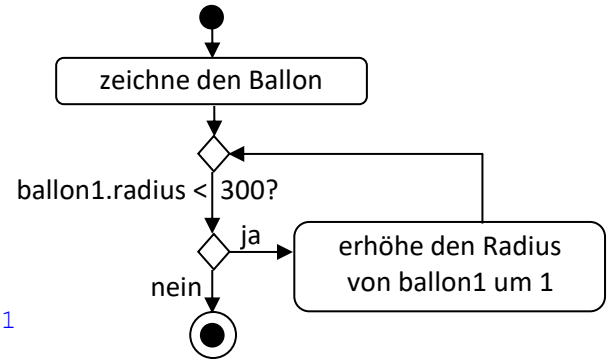
Attributwerte können nicht nur abgefragt, sondern auch **zugewiesen** werden. Die Schreibweise dafür ist: **Objekt.Attribut:=Wert**; Beispiel: `ballon1.radius:=ballon1.radius+1`

2. Öffne die Version deines EOS Programms, in der der Ballon „aufgepustet“ wird (ballon1.eos). Ändere die Zählschleife in eine Wiederholungsstruktur mit Anfangsbedingung ab und codiere eine Programmversion, in der die Attributwerte durch Wertzuweisung festgesetzt werden (ballon2.eos). (Vorlagedatei: v03-ballon1.eos); *vgl. .\261-materialien\ballon\02-ballon2.eos*
   Hinweise:
   - Mit der Methode `mittelpunktSetzen(mittex:integer,mittey:integer)` werden **zwei** Attributwerte zugewiesen. Diese Methode muss also durch zwei Wertzuweisungen ersetzt werden.
   - Die Methode `zeichne()` kann nicht durch eine Wertzuweisung ersetzt werden.

   Erstelle ein Aktivitätsdiagramm. Die Wertzuweisungen müssen nicht alle in einzelnen Anweisungen aufgeführt werden. Sie können in einer Aktion zusammengefasst werden.
   Wichtig ist die Wiederholungsstruktur mit Anfangsbedingung.

```
f:FENSTER
  ballon1:KREIS
  f.links:=100
  f.oben:=100
  f.breite:=700
  f.höhe:=700
  ballon1.radius:=14
  ballon1.randfarbe:=blau
  ballon1.füllfarbe:=blau
  ballon1.mittex:=0
  ballon1.mittey:=0
  f.zeichne(ballon1)
  solange ballon1.radius<300 tue
     ballon1.radius:=ballon1.radius+1
  *solange
```

Aktivitätsdiagramm:



- Man kann auch eigene Attribute erstellen, die nichts mit einem vorhandenen Objekt zu tun haben. In diesem Fall spricht man von **Variablen**.

3. Gib an, wie das Objekt `ballon1` der Klasse Kreis deklariert wird:

   `ballon1:KREIS`

4. Was könnte in der Methode `mittelpunktSetzen(mittex:integer,mittey:integer)` das Wort **Integer** bedeuten?
   Hinweis: Bei `mittex` und `mittey` handelt es sich um zwei **Variablen** der Klasse `Integer`. Das sind zwei Speicherbereiche, die unterschiedliche Werte annehmen können.

   *Das englische Wort **integer** heißt auf deutsch **ganzzahlig**.*

   Die Variablen können also ganzzahlige Werte annehmen.

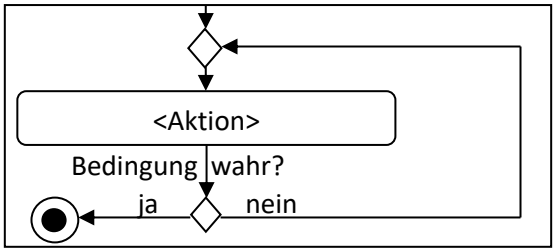
   Das Wort *integer* wird englisch ausgesprochen (etwa wie „intedscher“ mit der Betonung auf „in“). Es bedeutet in der IT auch etwas anderes als das deutsche Wort „integer“ (unbescholten), nämlich „ganzzahlig“).

- Ein Speicherbereich, der verschiedene Werte annehmen kann, wird **Variable** genannt.
- Zu der Variablen muss angegeben werden, welche *Art von* Wert zugewiesen wird. Das nennt man bei Variablen nicht Klasse, sondern **Datentyp**. In Programmiersprachen stehen eine Reihe verschiedener Datentypen zur Verfügung. Im Beispiel ist der Datentyp `Integer` (ganzzahlige Werte).
- Diesen Vorgang nennt man **Variablendeklaration**.

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.
Die Schreibweise in EOS lautet:

```
wiederhole
  <Anweisungen>
*wiederhole bis <Bedingung>
```



5. Das Programm aus Arbeitsblatt 01 Nr. 2 wird dahingehend erweitert, dass der blaue Ball innerhalb eines Spielfelds von dem Rand eines Rechtecks als „Bande“ abprallt. Außerdem wird das Objekt `ball1` etwas kleiner gezeichnet (mit dem Radius 10):

f:Fenster
Links=100 Oben=100 Breite=840 Höhe=400 hintergrundfarbe=hellblau
gitteraus()

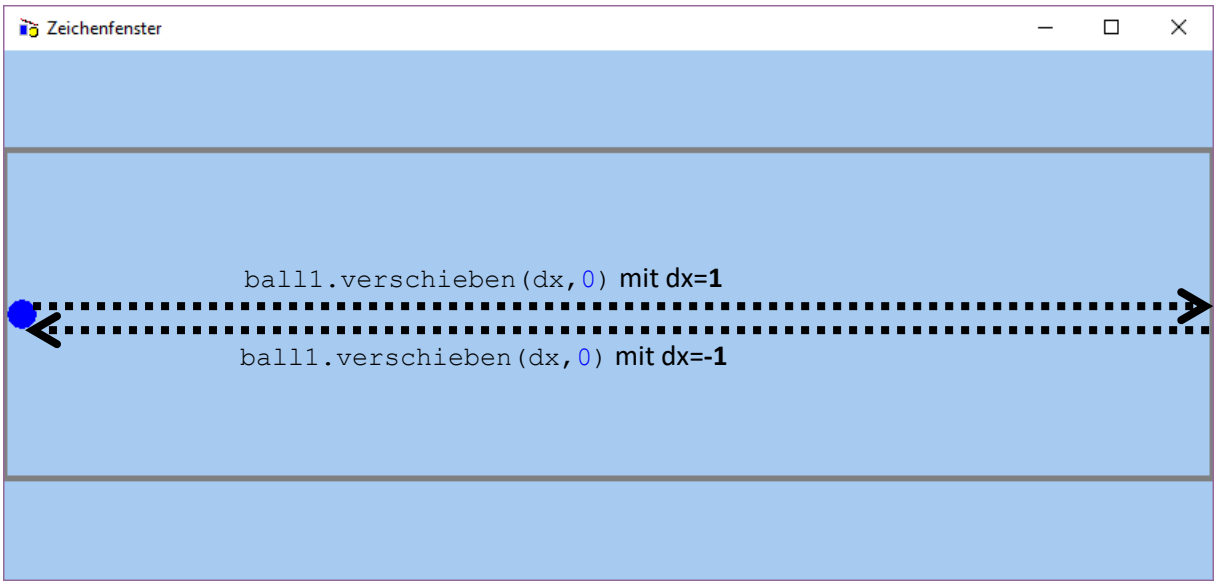
spielfeld:Rechteck
Links=-412 Oben=112 Rechts=412 Unten=-112 Randstärke=4 Randfarbe=grau Füllart=durchsichtig

ball1:Kreis
Mittex=-400 Mittey=0 Radius=10 Füllfarbe=blau Randfarbe=blau Randstärke=1

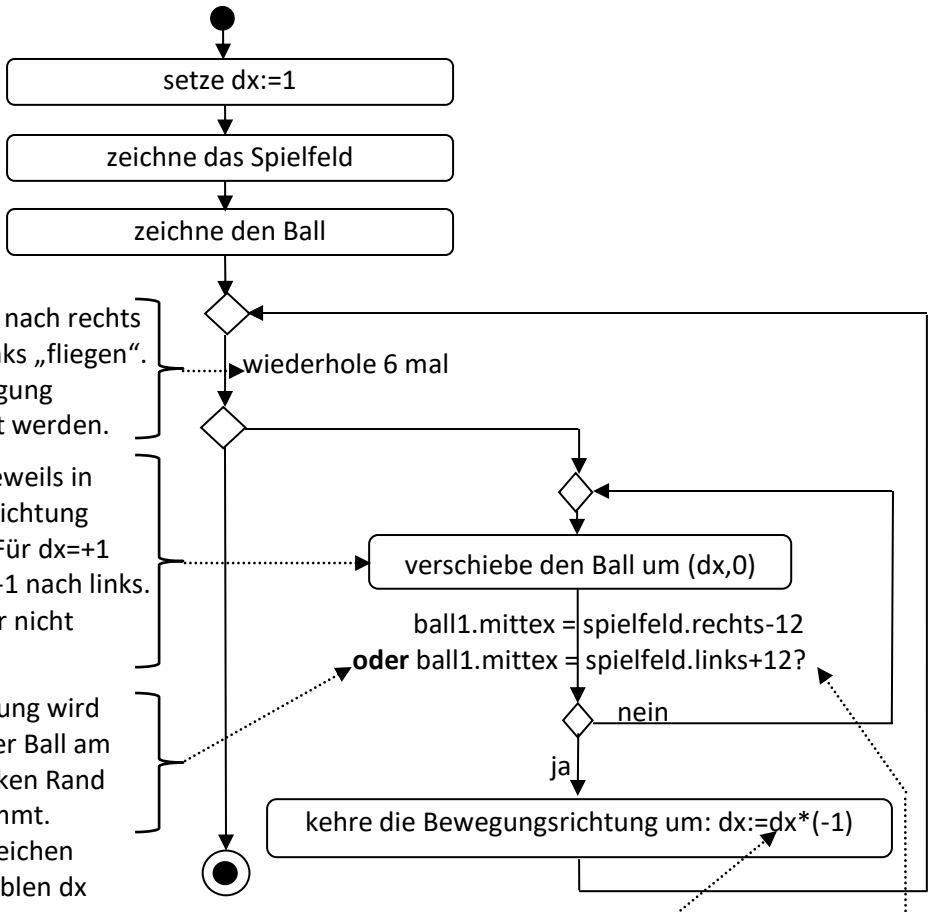
➢ Der Ball wird entweder um eins nach rechts oder um eins nach links verschoben. In der Anweisung `ball1.verschieben(x:Integer,y:Integer)` ist der Wert für x also entweder +1 oder -1. Um die Bewegung innerhalb *einer* Wiederholungsstruktur auszuführen, kann zur Speicherung der Bewegungsrichtung eine ganzzahlige Variable verwendet werden, z. B. **dx:Integer** (für „Delta x“).
Die Attribute für diese erste Version des „Ballspiels“ sind in dem Klassendiagramm rechts angegeben.
Anfangs muss die Variable dx den Wert 1 erhalten.
Der zugehörige Programmcode in EOS lautet:

```
...
dx:Integer
dx:=1
...
```

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



- Erstelle die Objekte in EOS. Verwende dazu die Version *ball2.eos*. Speichere das überarbeitete Programm als *ballspiel1.eos*. (Vorlagedatei: v02-ball2.eos)



- Der Ball soll dreimal nach rechts und dreimal nach links „fliegen". Also muss die Bewegung sechsmal ausgeführt werden.
- Dazu wird der Ball jeweils in Einerschritten in x-Richtung um dx verschoben: Für dx=+1 nach rechts, für dx=-1 nach links. In y-Richtung wird er nicht verschoben.
- Die Bewegungsrichtung wird umgekehrt, wenn der Ball am rechten **oder** am linken Rand des Spielfelds ankommt. Dazu muss das Vorzeichen der Zahl in der Variablen dx umgekehrt werden.
  Das kennst du aus der Mathematik: Die Zahl mit Minus Eins malnehmen.
  Der Ball hat einen Radius von 10 und das Spielfeld eine Randstärke von 4.
  Deshalb muss der Mittelpunkt des Balles 10+4:2 = 12 von der Mitte des Spielfelds entfernt sein.

- Ergänze den Programmcode inEOS.
  vgl. .\261-materialien\ball\06-ballspiel1.eos

6. Ergänze einen „Schläger", an dem der Ball abprallt, gemäß des Objektdiagramms rechts (ballspiel2.eos).
   vgl. .\261-materialien\ball\07-ballspiel2.eos

schlaeger1:Ellipse
Mittex=300 Mittey=0 Radiusx=10 Radiusy=90 Füllfarbe=grau

```
...
schlaeger1:ELLIPSE
...
schlaeger1.radiusxSetzen(10)
schlaeger1.radiusySetzen(90)
schlaeger1.mittelpunktSetzen(300,0)
schlaeger1.füllfarbeSetzen(grau)
```

- Gib die Bedingung für die Wiederholungsstruktur an, damit der Ball links an der Bande und rechts an dem „Schläger" abprallt. Ändere das EOS-Programm entsprechend ab und teste es.

```
ball1.mittex = schlaeger1.mittex oder ball1.mittex = spielfeld.links+12
```