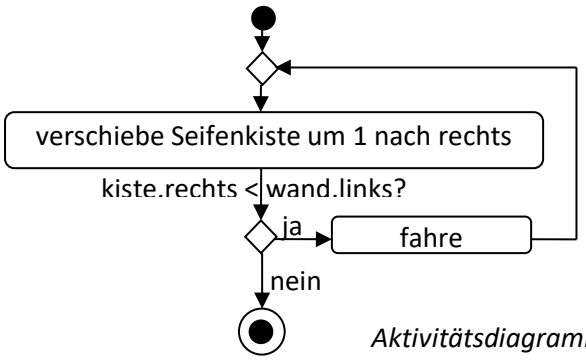
Rekursion

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

1. *Rekursive Programmierung* kann also nur mit Hilfe einer Methode durchgeführt werden.
   - Ergänze in der Version 7 oder 8 des EOS-Programms *seifenkiste* die beiden im Klassendiagramm gegebenen Methoden.
   - Ändere den Programmcode in der Methode *fahre()* nach dem Aktivitätsdiagramm ab und speichere das geänderte Programm als *seifenkiste-rekursiv.eos*.
     (Vorlagedatei: v09-seifenkiste8.eos)
     vgl. .\261-materialien\seifenkiste\09-seifenkiste-rekursiv.eos

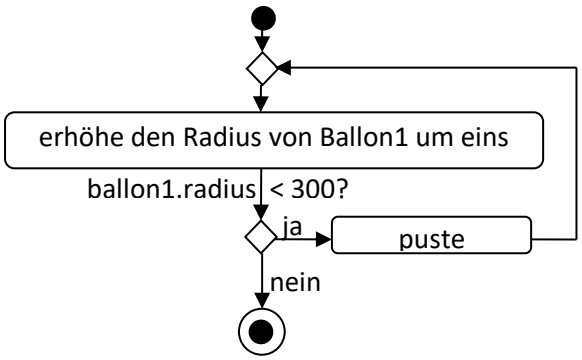
SeifenkisteMitWand
xwand:Integer seifenkiste1:Gruppe kiste1:Rechteck rad01:Kreis rad02:Kreis wand1:Rechteck
zeichneKiste() **fahre()**



Aktivitätsdiagramm `fahre`

2. Auch das Programm *ballon1.eos* (vgl. Arbeitsblatt 1.8-04, S. 2) kann rekursiv programmiert werden. Zur Erinnerung: Ein blauer „Ballon“ mit dem Radius 14 wurde auf einen Radius von 300 „aufgepustet“.
   - Modelliere den „Ballon“ in dem Klassendiagramm rechts. Ergänze eine geeignete Methode.
   - Stelle den Algorithmus für die neue Methode in einem Aktivitätsdiagramm dar.

Ballon
ballon1:Kreis
warte() ***puste()***



Aktivitätsdiagramm `puste`

   - Ändere den Programmcode ab (ballon3.eos).
     vgl. .\261-materialien\ballon\03-ballon3.eos

3. Warum kann man bei einer immer weiter laufenden Uhr nicht von einer Rekursion sprechen?

   Das Programm befindet sich in einer Endlosschleife. Es lässt sich also nicht in endlich vielen Schritten auflösen.

4. Nenne eine gewollte Endlosschleife.

   Zum Beispiel die Abfrage der Position des Mauszeigers.

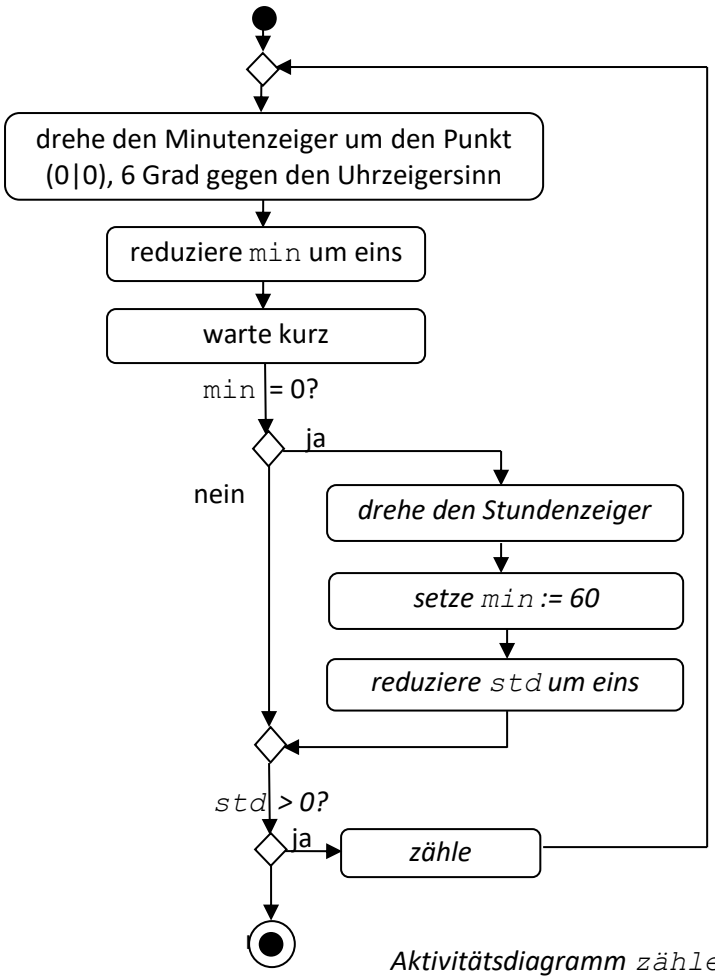
5. Warum kann man bei dem Programm zu Aufgabe 1 von einer Rekursion sprechen?

   Hier lässt sich jeder Aufruf der rekursiven Funktion in endlich vielen Schritten auflösen.

6. Wenn eine Uhr nicht immer weiterläuft, sondern auf Null zurückzählt, spricht man von einem *Timer*.
   Ein Timer ist für rekursive Programmierung geeignet.
   Dafür kann als Basis das Programm *uhr1.eos* verwendet werden.
   Es soll so abgeändert werden, dass 3 Stunden **zurück**gezählt werden.
   Zum Start ist also der Stundenzeiger auf „drei Uhr“ zu stellen.
   Stoppen soll der Timer, wenn der Stundenzeiger auf Null steht.
   Hinweis: Um die Stunden und Minuten mitzählen zu können, werden die Variablen `std` und `min` eingeführt.
   Rechts ist das Klassendiagramm des *Timers* gegeben.

Timer
Zifferblatt1:Gruppe Stunden:Gruppe Minuten:Gruppe Rand:Kreis Ziffer:Linie Zeiger1:Rechteck Spitze:Dreieck Achse:Kreis **std:Integer** **min:Integer**
erstelleUhr() warte() zähle()

   - Ergänze das Aktivitätsdiagramm dazu.



Aktivitätsdiagramm `zähle`

```
methode zähle
    Minuten.drehenUm(0,0,6)
    min:=min-1
    warte()
    wenn min=0 dann
        Stunden.drehenUm(0,0,30)
        min:=60
        std:=std-1
    *wenn
    wenn std>0 dann
        zähle()
    *wenn
ende
```

   - Codiere den Algorithmus in EOS und speichere die Datei als *timer.eos*.
     (Vorlagedatei: v10-uhr1.eos); *vgl. .\261-materialien\uhr\07-timer.eos*