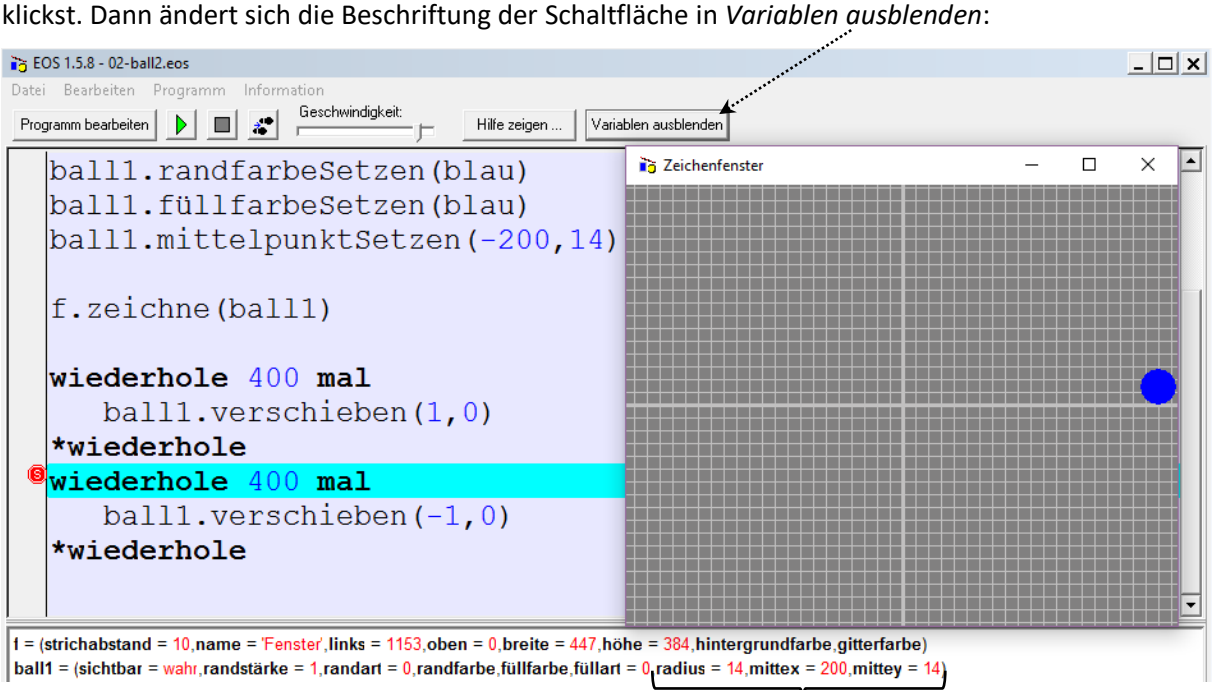
Attributwerte

Ball1:KREIS
Mittex=-200
Mittey=14
Radius=14
Füllfarbe=blau
Randfarbe=blau

Im Arbeitsblatt 2.6.1-01 hast du den im Objektdiagramm rechts gegebenen Ball nach rechts „rollen“ lassen (verschoben) und wieder zurück.

Hinweise:

- In EOS kannst du einen Stopppunkt setzen, indem du auf dem linken Rand an der Stelle klickst, an der das Programm anhalten soll.
- Durch Klick auf Play kannst du das Programm fortsetzen.
- Durch nochmaligen Klick auf den Stopppunkt kannst du diesen auch wieder löschen.
- Die aktuellen Attributwerte kannst du betrachten, indem du auf die Schaltfläche *Variablen zeigen* klickst. Dann ändert sich die Beschriftung der Schaltfläche in *Variablen ausblenden*:



1. Ergänze das Objektdiagramm des Balles in dem Moment, in dem er am rechten Rand ankommt.
   Die aktuellen Attributwerte findest du bei den Variablen.
   Was hat sich gegenüber dem Objektdiagramm oben geändert?

   Das Attribut Mittex hat nicht mehr den Wert -200, sondern 200.

   ➢ Attributwerte können sich während des Programmablaufs ändern.

Ball1:KREIS
Mittex=*200*
Mittey=*14*
Radius=*14*
Füllfarbe=blau
Randfarbe=blau

2. Wodurch wurde der Attributwert verändert?

   Die Methode `ball1.verschieben(1,0)` *wurde 400 mal angewendet.*

   Dabei wurde der Attributwert `ball1.Mittex` *jedesmal um 1 erhöht.*

3. Die Klasse Kreis verfügt über die Methode `mittexSetzen()`. Wenn man zu dem aktuellen Wert Eins addieren würde, hätte das denselben Effekt wie die Methode `ball1.verschieben(1,0)`. Formuliere den Befehl dafür in der objektorientierten Schreibweise von EOS.

   `ball1.mittexSetzen(ball1.mittex+1)`

4. Teste, ob diese Vorgehensweise funktioniert, indem du in deinem EOS-Programm *ball2.eos* aus dem Arbeitsblatt 01 die Methode `verschieben()` durch die Methode `mittexSetzen()` ersetzt (ball3.eos). (Vorlagedatei: v02-ball2.eos)
   vgl. .\261-materialien\ball\03-ball3.eos

5. Du kannst den Ball bestimmt auf dieselbe Art zehnmal 100 Pixel hoch „springen“ lassen (ball4.eos).
   vgl. .\261-materialien\ball\04-ball4.eos

6. Den blau gefüllten Kreis kann man auch als Ballon verwenden und „aufpusten“.
   - Erstelle die in den Objektdiagrammen gegebenen Objekte.
   - Erhöhe nach einer kurzen Wartezeit 300 mal schrittweise den Radius des Kreises, so dass es so aussieht als ob der Ballon aufgepustet würde (ballon1.eos).
     vgl. .\261-materialien\ballon\01-ballon1.eos

f:Fenster
Links=100 Oben=100 Breite=700 Höhe=700

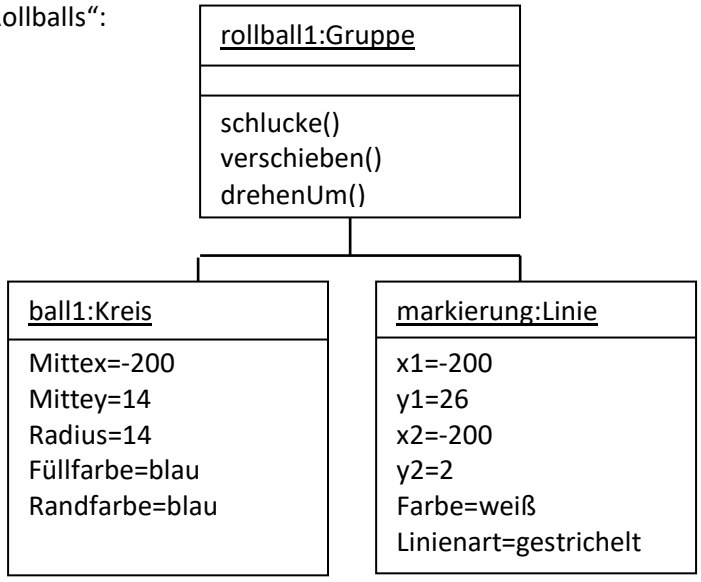
Ballon1:KREIS
Mittex=0 Mittey=0 Radius=14 Füllfarbe=blau Randfarbe=blau

7. Verwende wieder die ältere Version des Programms, in dem der Ball „rollt“ (ball2.eos).
   Nachdem der Kreis nur verschoben wird, entsteht nicht wirklich der Eindruck einer Rollbewegung.
   Das kann man ändern, indem eine Markierung in dem Ball gleichzeitig verschoben und gedreht wird.
   - Argumentiere, um welchen Winkel man den Ball bei einer Verschiebung von 1 Pixel drehen muss, damit der Eindruck des „Rollens“ ensteht.
     Hinweis: Für den Kreisumfang gilt: $u=2\pi r$

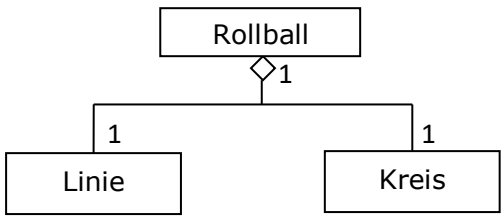
Wenn man das Objekt 360 mal um ein Grad dreht und gleichzeitig um einen Pixel verschiebt, gilt: *$360=2\cdot\pi\cdot r$; r wäre also $360:(2\cdot\pi)=57,3$* *Der Radius beträgt hier aber 14,* *also gilt: 57,3:14=4,09*	*Oder man berechnet das Ergebnis für x Grad direkt:* *$360:x=2\cdot\pi\cdot 14$ und damit:* *$x=360:(2\cdot\pi\cdot 14)$* *x=4,09*

   Man muss das Objekt bei einer Verschiebung von 1 Pixel um etwa 4 Grad drehen.

➢ Das Objektdiagramm des „Rollballs“:

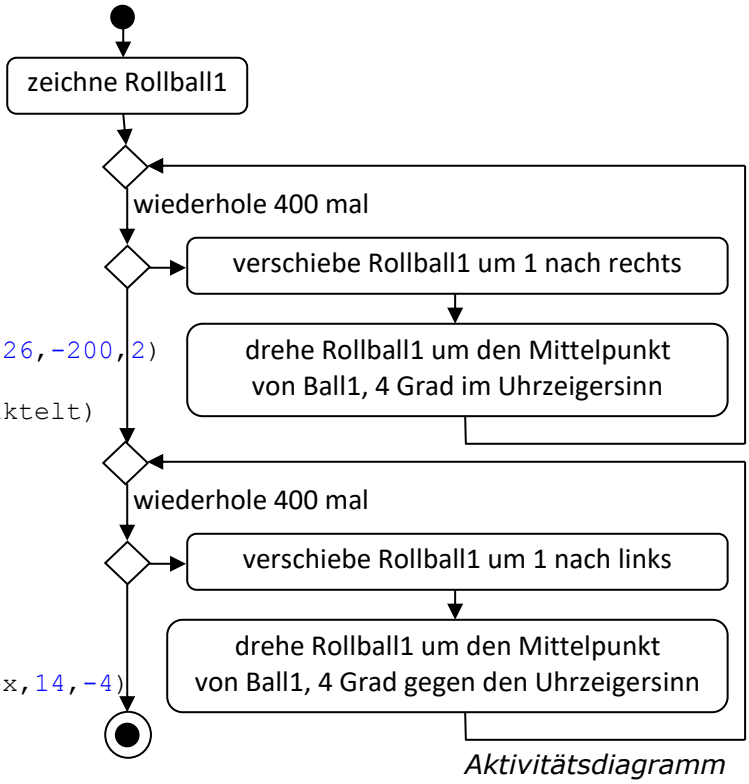


   - Modelliere rechts die Struktur des „Rollballs“ in einem Klassendiagramm.



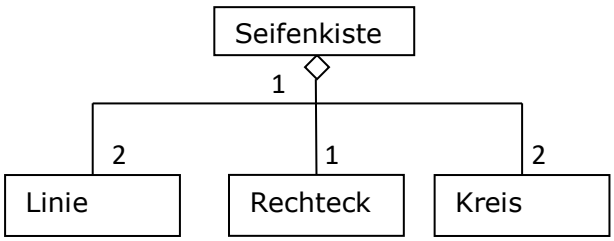
- Stelle die Handlungsanweisungen des „Rollballs“ in einem Aktivitätsdiagramm dar und codiere den Algorithmus in EOS (ball5.eos).
  vgl. .\261-materialien\ballon\05-ball5.eos

```
f:FENSTER

rollball1:GRUPPE
ball1:KREIS
markierung1:LINIE

ball1.radiusSetzen(14)
ball1.randfarbeSetzen(blau)
ball1.füllfarbeSetzen(blau)
ball1.mittelpunktSetzen(-200,14)

markierung1.endpunkteSetzen(-200,26,-200,2)
markierung1.farbeSetzen(weiß)
markierung1.linienartSetzen(gepunktelt)

rollball1.schlucke(ball1)
rollball1.schlucke(markierung1)

f.zeichne(rollball1)

wiederhole 400 mal
   rollball1.verschieben(1,0)
   rollball1.drehenUm(ball1.mittex,14,-4)
*wiederhole
wiederhole 400 mal
   rollball1.verschieben(-1,0)
   rollball1.drehenUm(ball1.mittex,14,4)
*wiederhole
```



Aktivitätsdiagramm

8. Ergänze bei der Seifenkiste weiße Linien als Markierung in den Rädern, damit es so aussieht, als ob sich die Räder drehen würden.
   - Modelliere die Struktur der Seifenkiste in einem Klassendiagramm.



Klassendiagramm

   - Berechnung des Drehwinkels:

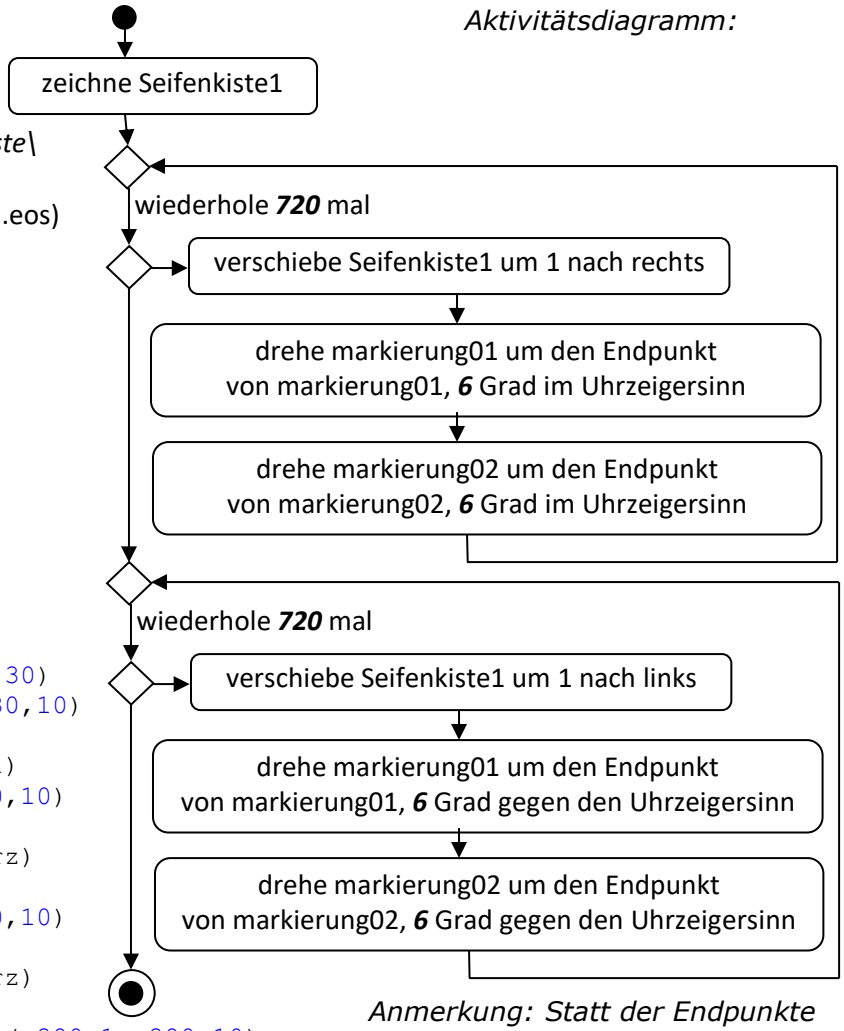
     x=360:(2·π·10)

     x=5,73

     Der Drehwinkel beträgt also etwa 6 Grad.

- Ergänze das Aktivitätsdiagramm und codiere den Algorithmus in EOS (seifenkiste5.eos).
  vgl. .\261-materialien\seifenkiste\05-seifenkiste5.eos
  (Vorlagedatei: v01-seifenkiste2.eos)

Aktivitätsdiagramm:



Anmerkung: Statt der Endpunkte der Markierungen könnte man auch die Mittelpunkte der Räder als Drehpunkt verwenden.

```
f:FENSTER

seifenkiste1:GRUPPE
kiste1:RECHTECK
rad01:KREIS
rad02:KREIS
markierung01:LINIE
markierung02:LINIE

f.linksSetzen(100)
f.obenSetzen(100)
f.breiteSetzen(800)
f.höheSetzen(600)
kiste1.linksObenSetzen(-390,30)
kiste1.rechtsUntenSetzen(-330,10)
kiste1.füllfarbeSetzen(gelb)
seifenkiste1.schlucke(kiste1)
rad01.mittelpunktSetzen(-380,10)
rad01.radiusSetzen(10)
rad01.füllfarbeSetzen(schwarz)
seifenkiste1.schlucke(rad01)
rad02.mittelpunktSetzen(-340,10)
rad02.radiusSetzen(10)
rad02.füllfarbeSetzen(schwarz)
seifenkiste1.schlucke(rad02)
markierung01.endpunkteSetzen(-380,1,-380,10)
markierung01.farbeSetzen(weiß)
seifenkiste1.schlucke(markierung01)
markierung02.endpunkteSetzen(-340,1,-340,10)
markierung02.farbeSetzen(weiß)
seifenkiste1.schlucke(markierung02)
f.zeichne(seifenkiste1)
wiederhole 3 mal
   wiederhole 720 mal
      seifenkiste1.verschieben(1,0)
      markierung01.drehenUm(markierung01.x2,10,-6)
      markierung02.drehenUm(markierung02.x2,10,-6)
   *wiederhole
   warte()
   wiederhole 720 mal
      seifenkiste1.verschieben(-1,0)
      markierung01.drehenUm(markierung01.x2,10,6)
      markierung02.drehenUm(markierung02.x2,10,6)
   *wiederhole
   warte()
*wiederhole
Methode warte
   wiederhole 2000 mal
      //keine Aktion
   *wiederhole
Ende
```