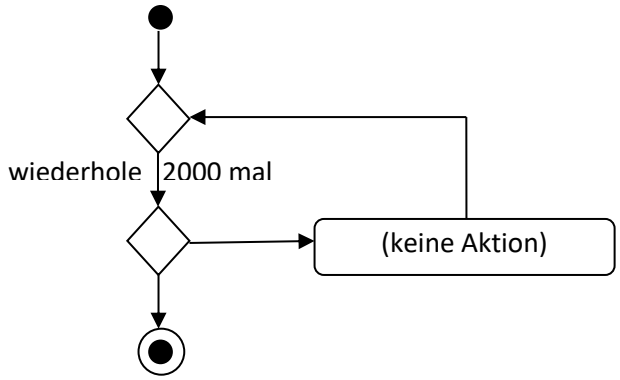
Übungen zur Programmierung in EOS

1. Die Seifenkiste soll jeweils kurz warten, wenn sie rechts und links die Bewegungsrichtung umkehrt.

   Eine Methode `warte()` gibt es in EOS nicht. Du kannst aber in EOS eigene Methoden erstellen. Dazu wird eine Wiederholungsstruktur erstellt, in der das Programm *nichts* tut, wodurch einige Zeit vergeht.

   Aktivitätsdiagramm für die Methode `warte()`:



   Die Anweisungen zur Programmierung einer **Methode** in EOS lauten:

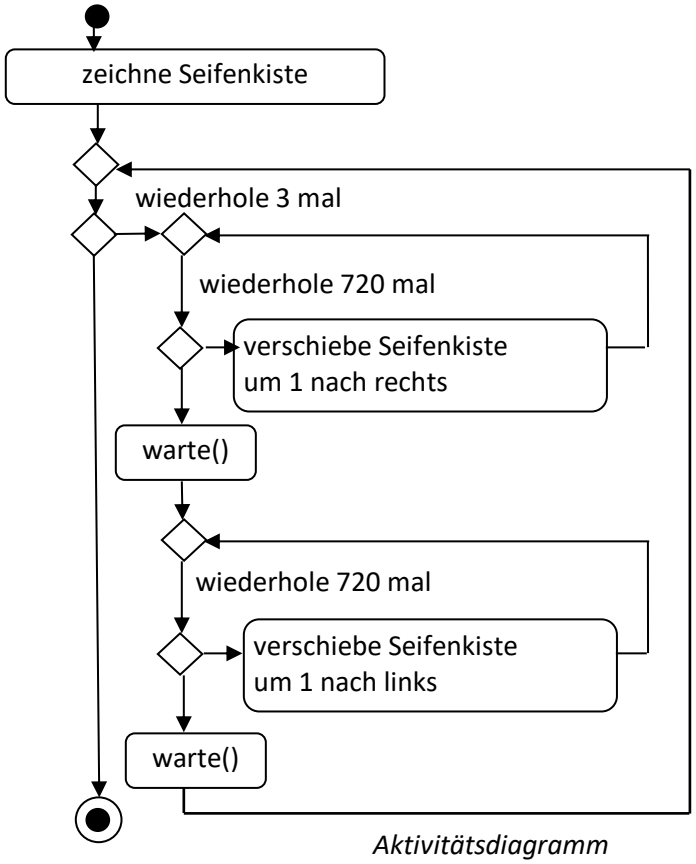
```
...
warte()
...
Methode warte
  wiederhole 2000 mal
   //keine Aktion
  *wiederhole
Ende
```

   Die Anweisung `Methode` bewirkt, dass die darin enthaltenen Befehle nur ausgeführt werden, wenn die Methode von einer anderen Stelle des Programms aus aufgerufen wird. Beendet wird die Methode mit der Anweisung `Ende`.

   Die Methode kann dann innerhalb des Programms beliebig oft aufgerufen werden, wie du aus dem Aktivitätsdiagramm rechts ersehen kannst.

   ○ Ergänze die Methode `warte()` in deinem EOS-Programm, am besten ganz am Ende. Dann kannst du die Methode innerhalb der vorhandenen Zählschleifen beliebig oft aufrufen (seifenkiste2.eos).

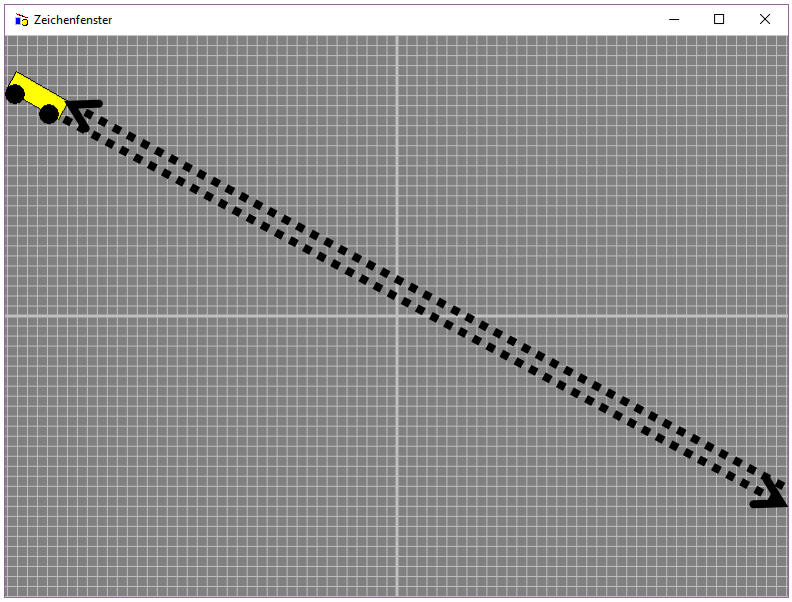
2. Optimiere das Programm, indem die Seifenkiste auf einer Straße fährt. Zur schöneren Gestaltung kannst du zum Beispiel auch das Gitternetz ausblenden und eine Hintergrundfarbe verwenden.



Aktivitätsdiagramm

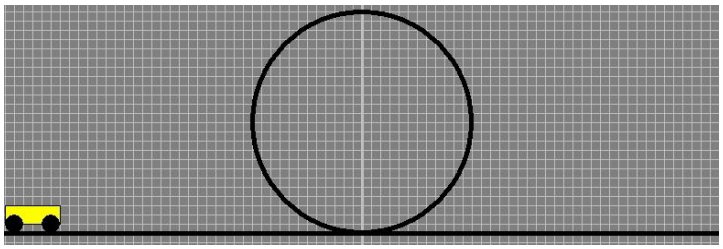
3. Die Seifenkiste soll schräg nach unten und wieder zurück fahren.
   - Analysiere das Problem, indem du den Diagonalen-schnittpunkt der Kiste herausfindest. Das ist der Drehpunkt.
   - Modelliere die Handlungs-anweisungen in einem Aktivitätsdiagramm.
   - Codiere den Algorithmus. (seifenkiste3.eos)

   Hinweis: Die Methode `drehenUm(x,y,Winkel)` dreht ein Objekt um den Drehpunkt (x|y).

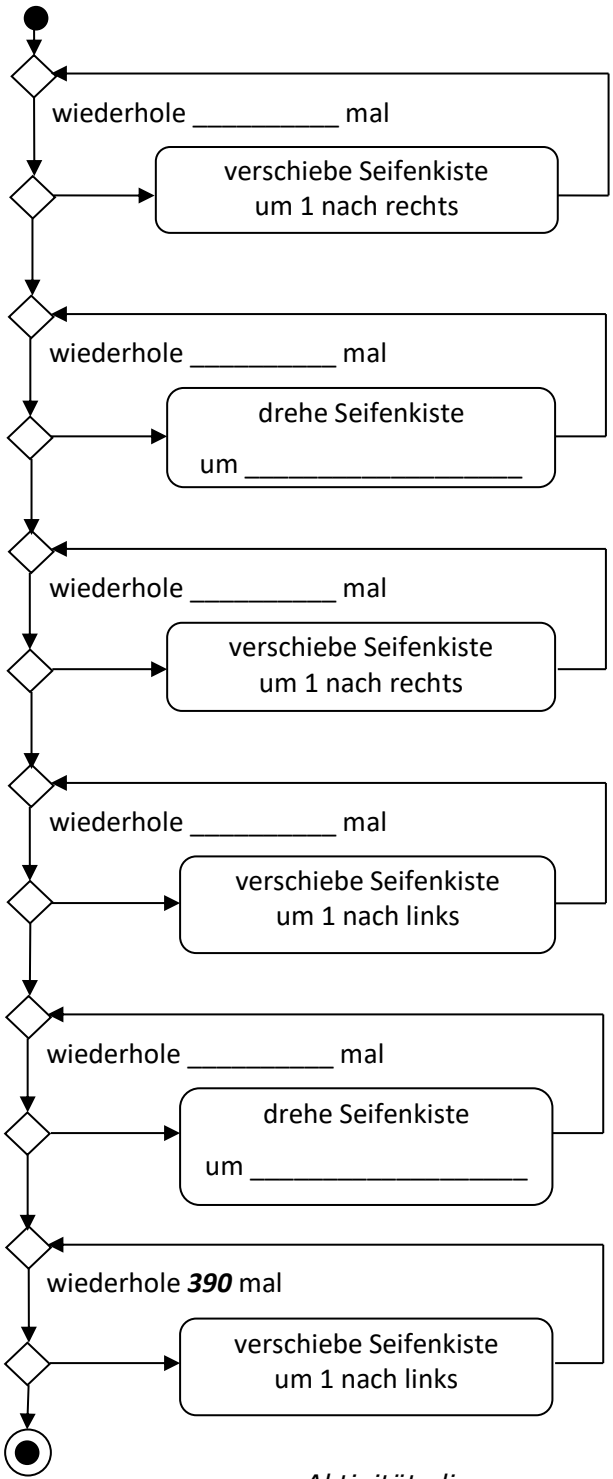


4. Zeichne eine Linie als Straße und einen Kreis als Looping, durch den die Seifenkiste fährt. Implementiere eine Lösung zu der Problemstellung.
   - Analysiere dazu den Weg, den die Seifenkiste abfahren muss; ergänze das Aktivitätsdiagramm.
   - Modelliere die Struktur in einem Klassendiagramm.
   - Codiere den Algorithmus (seifenkiste4.eos).

   Hinweis: Das Attribut `Füllart` verfügt für den Looping über den Attributwert `durchsichtig`.

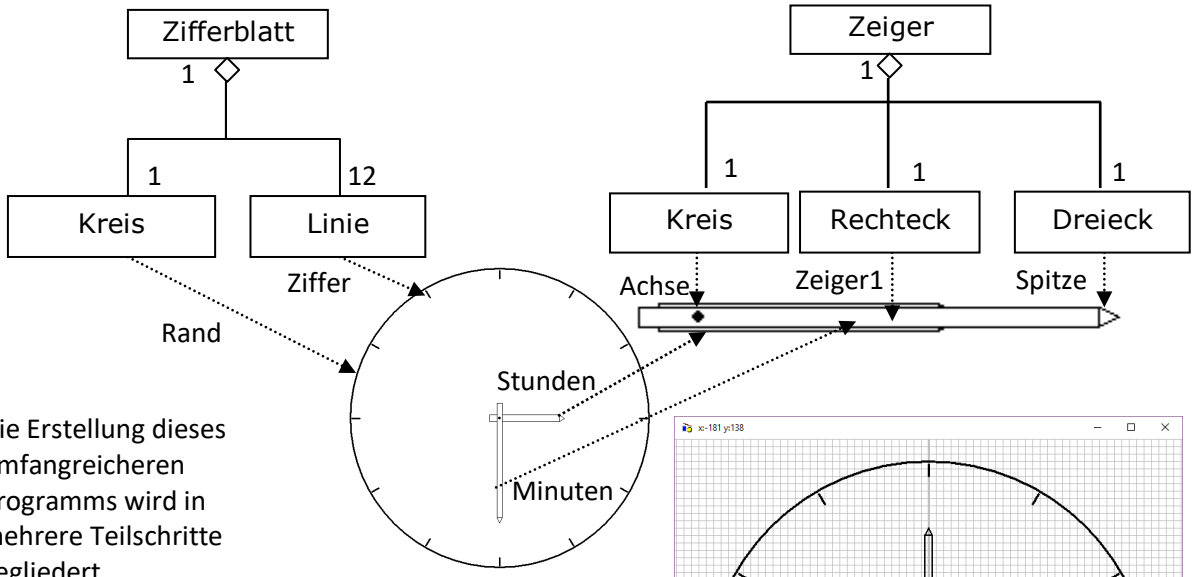


Klassenstruktur:



Aktivitätsdiagramm

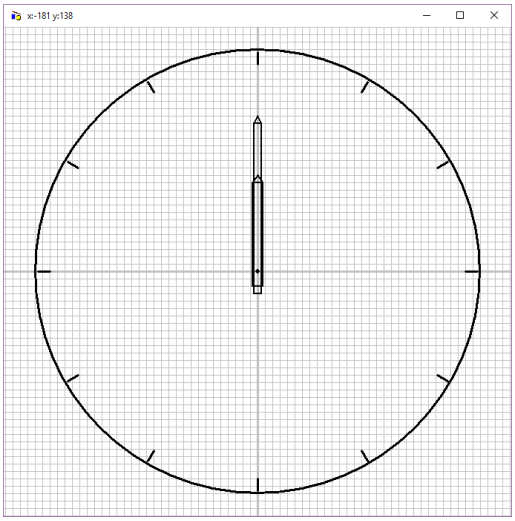
5. Entwickle ein EOS-Programm, in dem das Zifferblatt und die Zeiger einer `Uhr` gezeichnet werden. Dazu werden Objekte aus Klassen mit der folgenden Struktur erstellt:



Die Erstellung dieses umfangreicheren Programms wird in mehrere Teilschritte gegliedert.

a) Codiere zunächst lediglich das Fenster nach dem Objektdiagramm rechts. Speichere die Datei als `uhr1.eos`.

f:Fenster
Links=100 Oben=100 Breite=700 Höhe=700 Hintergrundfarbe=weiß
gitteraus()



➢ In umfangreicheren Programmen kann das gesamte Problem in Teilprobleme zerlegt werden. Dadurch wird der Algorithmus übersichtlicher gestaltet.
Dafür werden eigene Methoden festgelegt, zunächst die Methode `erstelleZifferblatt ()`.
Zuerst wird die Deklaration der Objekte für das Zifferblatt ergänzt:

Uhr2
Zifferblatt1:Gruppe Rand:Kreis Ziffer:Linie
erstelleZifferblatt ()

```
f:FENSTER
Zifferblatt1:GRUPPE
Rand:KREIS
Ziffer:LINIE

f.linksSetzen(100)
f.obenSetzen(100)
f.breiteSetzen(700)
f.höheSetzen(700)
f.hintergrundfarbeSetzen(weiß)
f.gitteraus()

erstelleZifferblatt()

methode erstelleZifferblatt
  ...
ende
```

Eine Methode wird nur ausgeführt, wenn sie aufgerufen wird.
➢ Ergänze die Anweisung dafür.

Zur Deklaration einer eigenen Methode in EOS werden die Programmzeilen zwischen die Anweisungen `methode` und `ende` geschrieben.
➢ Ergänze zunächst diese beiden Anweisungen.

Die Vorgehensweise zum Zeichnen des Zifferblatts ist auf der nächsten Seite beschrieben.

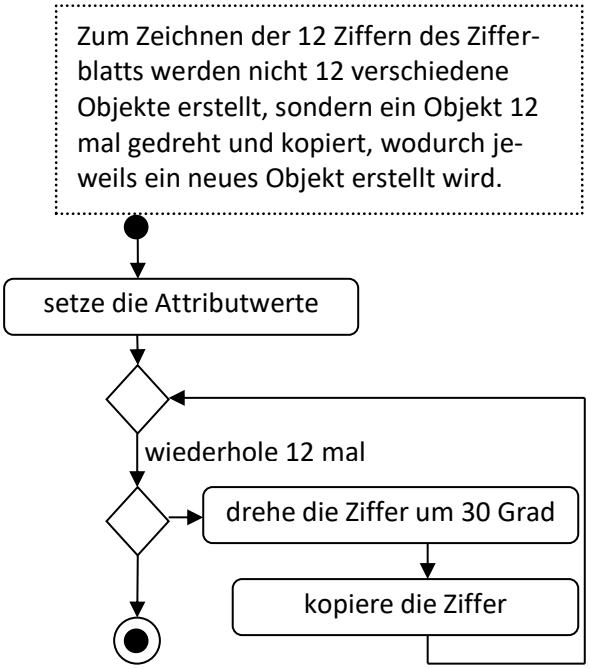
In der Methode `erstelleZifferblatt ()` sollen ein Kreis und 12 Linien für das Zifferblatt gezeichnet werden.
Dafür müssen **nicht** umständlich 12 einzelne Linien für die Ziffern erstellt werden:
Statt der Methode `schlucke()` kann die Methode `kopiere()` verwendet werden, um die Ziffern der Gruppe `Zifferblatt1` hinzuzufügen.
Im Gegensatz zu `schlucke()` fügt die Methode `kopiere()` nicht ein Objekt zu der Gruppe hinzu, sondern eine Kopie des Objekts mit den aktuellen Attributwerten.

b) Ergänze die Anweisungen innerhalb der Methode `erstelleZifferblatt ()` (`uhr2.eos`).

```
methode erstelleZifferblatt

    Rand.mittelpunktSetzen(0,0)
    Rand.füllfarbeSetzen(weiß)
    Rand.randstärkeSetzen(3)
    Rand.randfarbeSetzen(schwarz)
    Rand.radiusSetzen(300)
    Zifferblatt1.schlucke(Rand)

    Ziffer.linienStärkeSetzen(3)
    Ziffer.farbeSetzen(schwarz)
    Ziffer.endpunkteSetzen(0,280,0,296)
    wiederhole 12 mal
       Zifferblatt1.kopiere(Ziffer)
       Ziffer.drehenUm(0,0,30)
    *wiederhole
    f.zeichne(Zifferblatt1)
ende
```



c) In der Methode `erstelleZeiger ()` sollen der Stunden- und der Minutenzeiger gezeichnet werden.

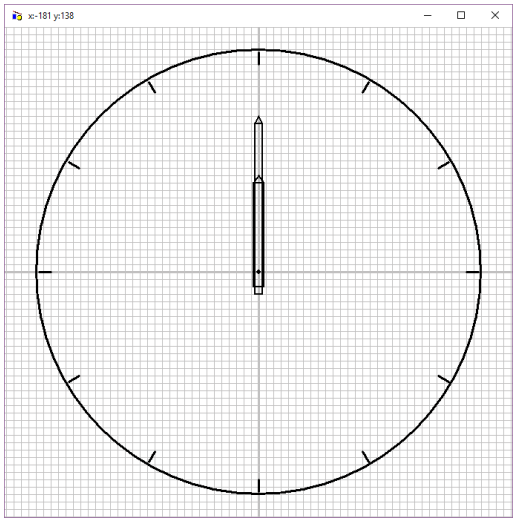
- Ergänze die Deklaration der Objekte ganz oben im Programm.
- Den Methodenaufruf ergänzt du nach der Anweisung `erstelleZifferblatt()`.
- Die Methode selbst wird ganz am Ende des Programms angefügt.
  Die Programmstruktur sollte dann so aussehen:

```
f:FENSTER
Zifferblatt1:GRUPPE
Stunden:Gruppe
Minuten:Gruppe
...
erstelleZifferblatt()
erstelleZeiger()
methode erstelleZifferblatt
   ...
ende
methode erstelleZeiger
   ...
ende
```

Uhr3
Zifferblatt1:Gruppe **Stunden:Gruppe** **Minuten:Gruppe** Rand:Kreis Ziffer:Linie **Zeiger1:Rechteck** **Spitze1:Dreieck** **Achse:Kreis** **Zeiger2:Rechteck** **Spitze2:Dreieck**
erstelleZifferblatt () **erstelleZeiger ()**

- Innerhalb der Methode werden dann später die Befehle zum Erstellen und Zeichnen der Zeiger eingetragen.

- Plane die Position und Größe der Objekte.
  Ein „Kästchen“ entspricht 10 Pixel.



Objektdiagramme zu Aufgabe b):

Rand:Kreis
Mittex=0 Mittey=0 Füllfarbe=weiß Randstärke=3 Randfarbe=schwarz Radius=300

Ziffer:Linie
x1=0 y1=280 x2=0 y2=296 Farbe=schwarz Linienstärke=3

Objektdiagramme zum Stundenzeiger:

Achse:Kreis
Mittex=0 Mittey=0 Füllfarbe=schwarz Randfarbe=schwarz Radius=3

Zeiger1:Rechteck
Links=-7 Oben=120 Rechts=7 Unten=-20 Randfarbe=schwarz Randstärke=1 Füllfarbe=weiß

Spitze1:Dreieck
x1=-7 y1=120 x2=7 y2=120 x3=0 y3=130 Randfarbe=schwarz Randstärke=1 Füllfarbe=weiß

Ergänze die Objektdiagramme zum Minutenzeiger.

Zeiger2: _____
Links=_____ Oben=_____ Rechts=_____ Unten=_____ Randfarbe=schwarz Randstärke=1 Füllfarbe=weiß

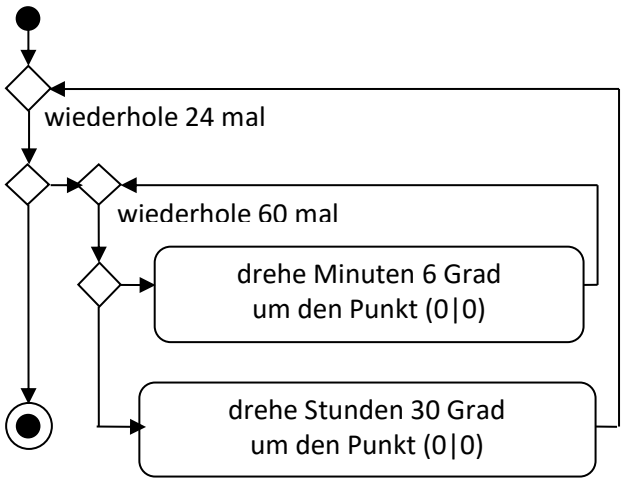
Spitze2: _____
_____ _____ _____ _____ _____ _____ Randfarbe=schwarz Randstärke=1 Füllfarbe=weiß

- Ergänze die Anweisungen innerhalb der Methode `erstelleZeiger ()` und speichere das Programm als `uhr3.eos`.

d) In der Methode `zähle` sollen die Stunden- und Minutenzeiger gedreht werden.
Erstelle zu dem Aktivitätsdiagramm den EOS-Programmcode für die Methode `zähle` (`uhr4.eos`).
Füge die neue Methode am Ende des Programms an und vergiss den Methodenaufruf nicht!

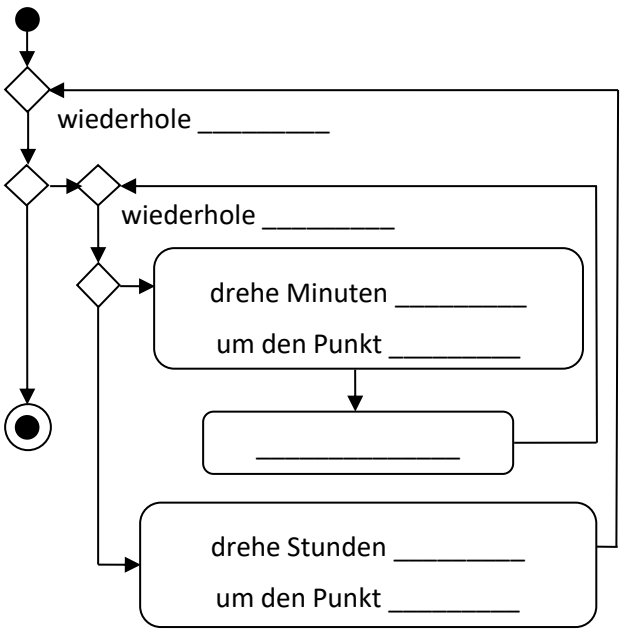
Teste dein Programm: Der Stundenzeiger muss nach jeder Umdrehung des Minutenzeigers um eine Stunde vorrücken und nach 24 Stunden wieder auf „zwölf Uhr“ stehen.

```
...
erstelleZifferblatt()
erstelleZeiger()
zähle()
...
```



Aktivitätsdiagramm: `zähle()`

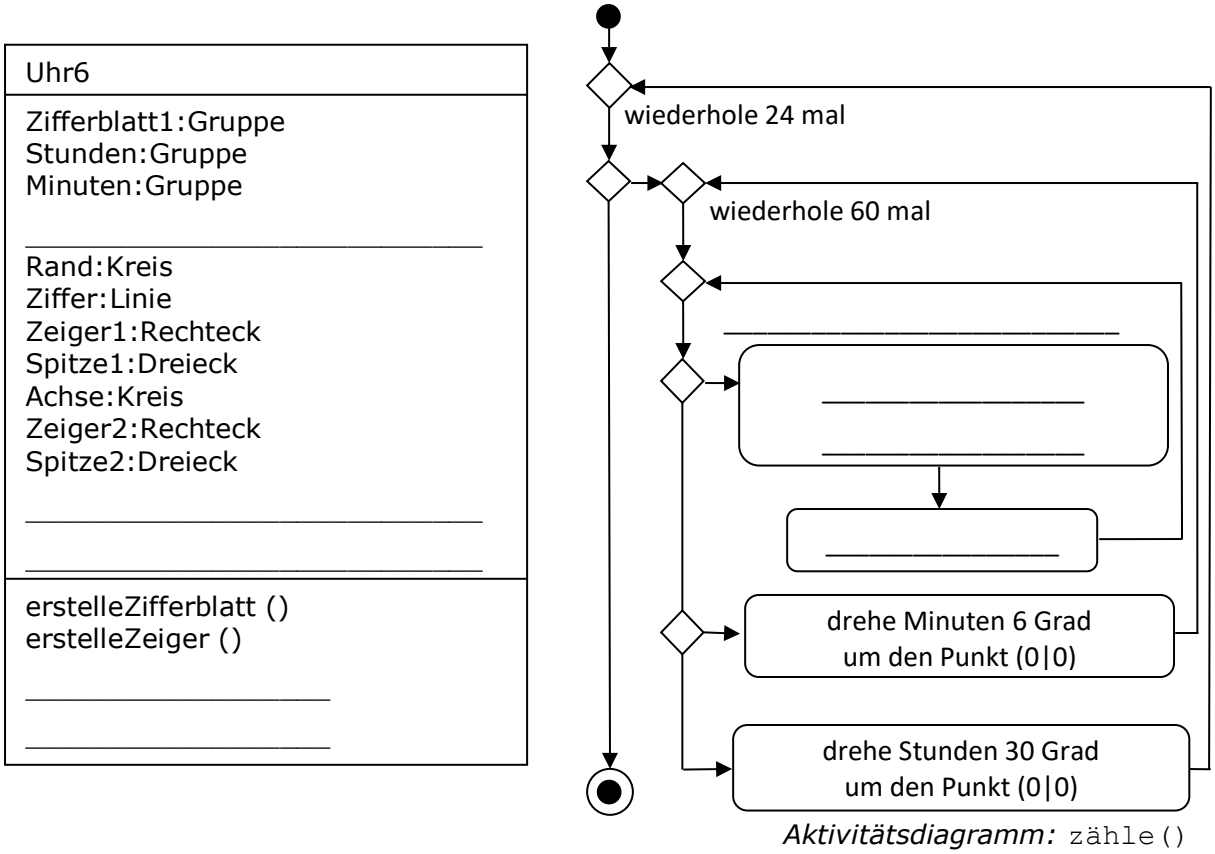
e) Die Bewegung des Minutenzeigers sieht in der Animation nicht gut aus.
Dieses Problem kann mit Hilfe der Methode `warte ()` behoben werden, indem an einer geeigneten Stelle im Programm eine kurze Pause gemacht wird.
- Ergänze das Aktivitätsdiagramm und den dafür erforderlichen Programmcode.
  Speichere das Programm als `uhr5.eos`.



Aktivitätsdiagramm: `zähle()`

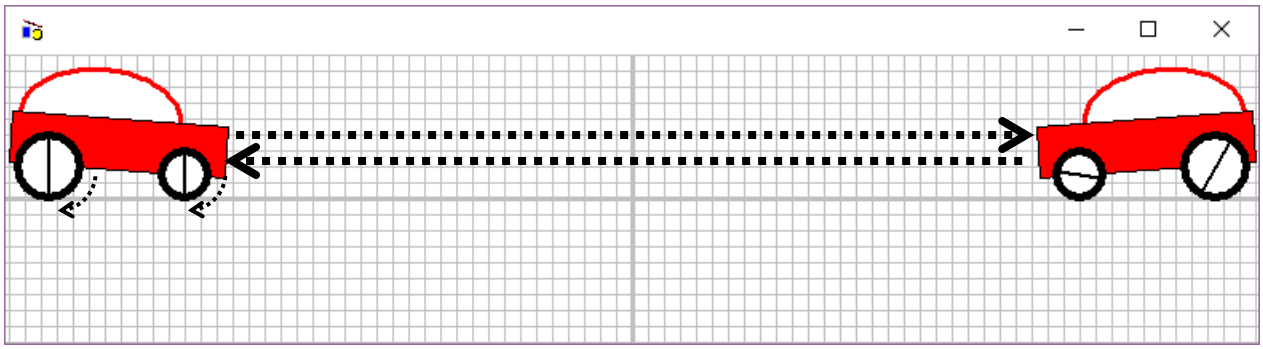
6. Ergänze in deinem EOS-Programm die Drehbewegung eines Sekundenzeigers (`uhr6.eos`).

   Ergänze das Klassendiagramm und das Aktivitätsdiagramm.



Aktivitätsdiagramm: `zähle()`

7. Erstelle in EOS einen „Monster Truck“ mit zwei unterschiedlich großen Rädern, der von links nach rechts und wieder zurück fährt. Mit Hilfe von zwei Linien in den Rädern soll eine Drehbewegung simuliert werden, wobei sich das kleine Rad etwas schneller drehen muss als das große Rad.



Hinweise:

- Um eine realistische Rotationsbewegung zu simulieren, sollte der Drehwinkel so gewählt werden, dass bei einer Drehung um 360 Grad der Kreisumfang einmal abgerollt wird.
  Für den Kreisumfang gilt: $u = 2\pi r$; für $u = 360$ gilt also $2\pi r = 360$ und damit $r = 57{,}3$.
  Bei einem Drehwinkel von 1 Grad und gleichzeitiger Verschiebung um 1 Pixel ist also ein Radius von 60 geeignet. Wenn beispielsweise ein Rad den Radius 20 und das andere Rad den Radius 15 hätte, müsste man jeweils um einen Winkel von 3 Grad bzw. 4 Grad drehen.
- Für die x-Koordinate des Drehpunkts der Linie muss nicht eine Zahl eingetragen werden. Dies kann auch der Attributwert für die x-Koordinate des Rades sein (z. B. `rad01.mittex`).
- Die Methode `spiegleX()` spiegelt die x-Werte eines Objekts.