Aufgabe 1 (Fusion)

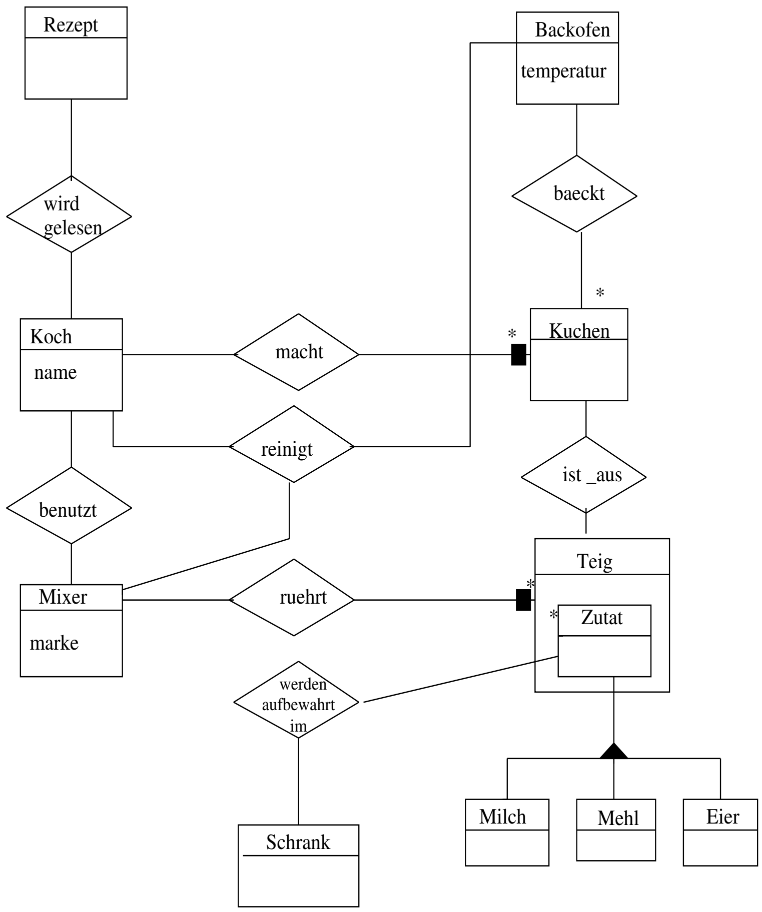
1. Erstellen Sie auf der Grundlage der nachfolgenden Beschreibung ein Objektmodell. *12 Punkte*
   Achten Sie darauf, daß Kardinalitäten und Totalitätsmarken und Attribute an geeignete Stellen angebracht werden.

   Beschreibung:
   Ein Koch, mit Namen Zuckerbäcker, bekommt den Auftrag einen einfachen Kuchen zu backen. Die Zutaten sind Milch, Eier und Mehl. Alle Zutaten sind vorrätig, es muß nichts eingekauft werden. Der Mixer ist von der Firma Mixgut. Der Kuchen wird mit einer bestimmten Temperatur gebacken.

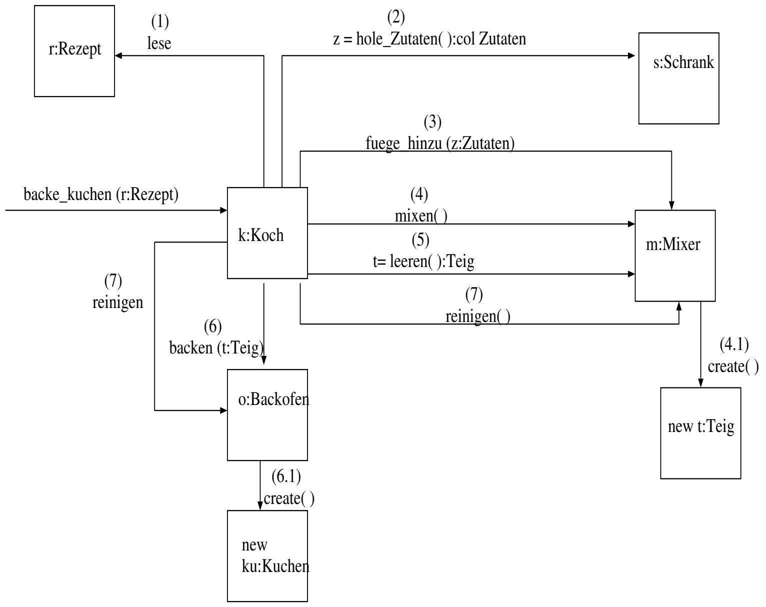
   So geht man vor:

   1.) Koch liest das Rezept
   2.) holt die Zutaten aus dem Schrank
   3.) gibt die Zutaten in den Mixer
   4.) Zutaten werden zu einem Teig verrührt
   5.) holt ihn aus dem Mixer
   6.) Kuchen wird im Ofen gebacken
   7.) reinigt Mixer und Ofen

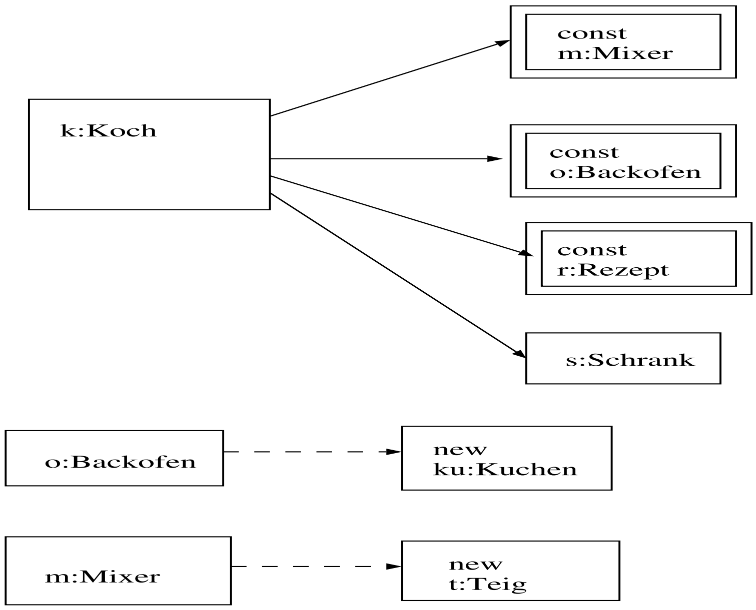
   Hinweis: Modellieren Sie nur die oben beschriebenen Informationen!



2. Entwickeln Sie die Sichtbarkeitsgraphen aus dem vorgegebene Objektinteraktionsgraphen und begründen Sie **in Stichworten** warum Sie diese Sichtbarkeitsreferenzen gewählt haben. *10 Punkte*



Lösung:



Backofen, Rezept und Mixer sind nicht veränderbar (konstant) und werden ausschließlich vom Koch benutzt. Dagegen steht der Schrank auch für andere Objekte zur Verfügung.

Alternative: Alle Objekte sind an den Klienten gebunden und exklusiv.

3. Ergänzen Sie die beiden Klassenbeschreibungen um die noch fehlenden Informationen. *8 Punkte*

class koch

endclass

class mixer

method create():Teig
endclass

Lösung:

class koch
attribute name:String
attribute const m: exclusive Mixer
attribute const o: exclusive Backofen
attribute const r: exclusive Rezept
attribute s: Schrank
method backe_kuchen(r:Rezept)
endclass

class mixer
attribute marke:String
method create():Teig
method fuege_hinzu(z:Zutaten)
method mixen()
method leeren()
method reinigen()
endclass

Aufgabe 2 (Z)

(Geschätzte Bearbeitungszeit: 30 min.)

1. Für welche Art von Systemen ist die Modellierung mit Z nicht geeignet? Begründen Sie *kurz*. *2 Punkte*

Lösung:

Reaktive Systeme, kontinuierliche Systeme, Echtzeitsysteme . . .

Der Modellierungsteil besteht aus zwei unabhängigen Teilen; im ersten Teil ost der *Zustand* für ein System zu spezifizieren, im zweiten Teil sind *Operationen* zu einen gegebenen Zustand anzugeben.

2. Modellieren Sie den Zustand einer (reduzierten) Mediothek. Als Basistypen sind gegeben: *7 Punkte*

   $[PERSON, MEDIUM]$

   Die Modellierung soll ausdrücken:

- Die Mediothek verwaltet einen Kundenbestand.
- Die Mediothek verwaltet einen Bestand von Medien, die *entweder* vorrätig *oder* ausgeliehen sind.
- Ein Kunde kann maximal zehn Medien auf einmal ausleihen.

Geben Sie eine mögliche Initialisierungsoperation an:

Mediothek

$$\begin{array}{l}
kunden : \mathbb{P}\, PERSON \\
bestand, vorr\ddot{a}tig : \mathbb{P}\, MEDIUM \\
ausgeliehen : MEDIUM \pfun PERSON \\
\hline
bestand = vorr\ddot{a}tig \cup \operatorname{dom} ausgeliehen \\
vorr\ddot{a}tig \cap \operatorname{dom} ausgeliehen = \varnothing \\
\operatorname{ran} ausgeliehen \subseteq kunden \\
\forall p : kunden \bullet \#(ausgeliehen \rhd \{p\}) \leq 10
\end{array}$$

Init_Mediothek

$$\begin{array}{l}
Mediothek' \\
\hline
kunden' = \varnothing \\
bestand' = \varnothing
\end{array}$$

3. Die Mississippi Shipping Company möchte für ihr Schiff *Mississippi Queen* ein einfaches Buchungssystem modelliert haben. Das Schiff besitzt eine bestimmte Kapazität an Passagieren und befährt eine Strecke mit mehreren Anlegestellen. Passagiere können mehrere Abschnitte buchen. Es soll sichergestellt werden, daß die Kapazität auf keinem Abschnitt überschritten wird. *16 Punkte*

$$[PERSON, ABSCHNITT]$$
$$kapazit\ddot{a}t : \mathbb{N}_1$$

Schiff

$$\begin{array}{l}
passagiere : \mathbb{P}\, PERSON \\
abschnitte : \mathbb{P}\, ABSCHNITT \\
gebucht : PERSON \leftrightarrow ABSCHNITT \\
\hline
passagiere = \operatorname{dom} gebucht \\
\operatorname{ran} gebucht \subseteq abschnitte \\
\forall a : abschnitte \bullet \#(gebucht \rhd \{a\}) \leq kapazit\ddot{a}t
\end{array}$$

Die Softwarefirma *Z&Co* konnte den Auftrag leider nicht vollständig ausführen, so daß von Ihnen noch folgende Operationen und Abfragen zu spezifizieren sind.

(a) Ein Passagier bucht genau einen Streckenabschnitt. Spezifizieren Sie für diese Operation den Normalfall und den Fehlerfall *Überbuchter_Abschnitt*. Setzen Sie die Spezifikation der Fehlerfälle *Ungültiger Abschnitt* und *BereitsGebucht* als gegeben voraus. Spezifizieren Sie dann die totale Operation *Buchung*.

$$REPORT ::= ok \mid fehler$$

BuchungOK

$$\begin{array}{|l}
\Delta Schiff \\
p? : PERSON \\
a? : ABSCHNITT \\
r! : REPORT \\
\hline
a? \in abschnitte \\
\{p? \mapsto a?\} \notin gebucht \\
\#(gebucht \rhd a?) < kapazität \\
gebucht' = gebucht \cup \{p? \mapsto a?\} \\
passagiere' = passagiere \cup \{p?\} \\
abschnitte' = abschnitte \\
r! = ok
\end{array}$$

Überbuchter_Abschnitt

$$\begin{array}{|l}
\Xi Schiff \\
p? : PERSON \\
a? : ABSCHNITT \\
r! : REPORT \\
\hline
a? \in \operatorname{ran} gebucht \\
\#(gebucht \rhd a?) \geq kapazität \\
r! = fehler
\end{array}$$

$$Buchung == BuchungOK \lor Überbuchter\_Abschnitt \lor Ungültiger\_Abschnitt \lor Bereits\_Gebucht$$

Die Meldung war nicht explizit gefordert.

(b) Ein Passagier storniert alle von ihm gebuchten Streckenabschnitte.

Stornierung

$$\begin{array}{|l}
\Delta Schiff \\
p? : PERSON \\
\hline
p? \in passagiere \\
gebucht' = \{p?\} \lhd gebucht \\
passagiere' = passagiere \setminus \{p?\} \\
abschnitte' = abschnitte
\end{array}$$

(c) Abfrage, wieviele Streckenabschnitte ein gegebener Passagier gebucht hat.

Gebuchte_Abschnitte

$\Xi Schiff$
$p? : ABSCHNITT$
$anzahl! : \mathbb{N}$

$p? \in passagiere$
$anzahl! = \#(\{p?\} \lhd gebucht)$

(d) Abfrage, auf welchen Streckenabschnitten das Schiff leer fährt.

Leere_Abschnitte

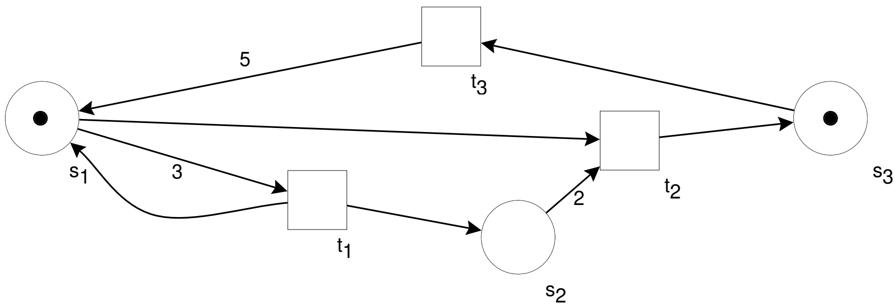
$\Xi Schiff$
$leere! : \mathbb{P}\,ABSCHNITT$

$leere! = abschnitte \setminus \operatorname{ran} gebucht$

Aufgabe 3 (Petri-Netze)

(Geschätzte Bearbeitungszeit: 15 min.)

Gegeben ist folgendes S/T-Netz: *14 Punkte*



1. Geben Sie für obiges Netz die Netzmatrix und die Anfangsmarkierung an.

	t_1	t_2	t_3	M_n
s_1	-2	-1	5	1
s_2	1	-2	0	0
s_3	0	1	-1	1

2. Ist das Netz lebendig? Leiten Sie Ihre Aussage her. $M_1 = M_n + t_3 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 5 \\ 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 6 \\ 0 \\ 0 \end{pmatrix}$

$$M_2 = M_1 + t_1 = \begin{pmatrix} 6 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \\ 0 \end{pmatrix}$$

$$M_3 = M_2 + t_1 = \begin{pmatrix} 4 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix}$$

$$M_4 = M_3 + t_2 = \begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix} + \begin{pmatrix} -1 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = M_N$$

Das Netz ist zyklisch, jede Transition hat gefeuert, jede Tranistion feuert im Zyklus, keine anderen Transitionen können schalten, also ist das Netz lebendig.

3. Geben Sie $*s_1$ an: $*s_1 = \{t_1, t_3\}$

4. Geben Sie $*t_1 \cap t_1*$ an: $*t_1 \cap t_1* = \{s_1\}$

Aufgabe 4 (Statecharts)

(Geschätzte Bearbeitungszeit: 20 min. zzgl. 10 min für's Lesen)

1. Was versteht man in Statecharts unter Broadcast-Kommunikation? *2 Punkte*

   Broadcast-Kommunikation: wenn ein Ereignis auftritt, bemerken dies alle (Unter-)Zustände des Systems. Ereignissse können nicht gespeichert werden, d.h. wenn eine Komponente im Moment des Auftretens des Ereignisses anderweitig beschäftigt ist, hat sie das Ereignis „verpaßt“.

2. Es ist ein Radiowecker in Statecharts zu modellieren. Die Anzeigen des Weckers, das Einstellen der Zeit und der Weckzeit und die Zuordnung von Radiostationen zu den Stationstasten des Radioteils des Weckers werden nicht betrachtet. *18 Punkte*

3. Der Wecker kann *aus* sein (dann weckt er nicht), oder *ein*geschaltet (dann weckt er, wenn die eingestellte Weckzeit erreicht ist).

4. Im Initialzustand ist der Wecker ausgeschaltet.

5. Durch Betätigen eines Schalters *a* kann der Wecker ein- und ausgeschaltet werden.

6. Das Wecken kann entweder durch einen Piepston oder durch Anstellen eines eingebauten Radios erfolgen. Die initiale Einstellung ist der Piepston.

7. Für das Radio sind 4 Stationstasten vorhanden, die verschiedenen Radiosendern zugeordnet sind . Initial wird Station 1 gewählt. Mittels Betätigen eines Knopfes *c* wird die Stationsnummer weitergeschaltet, wobei Betätigen von *c* in Station 4 wieder zu Station 1 zurückführt.

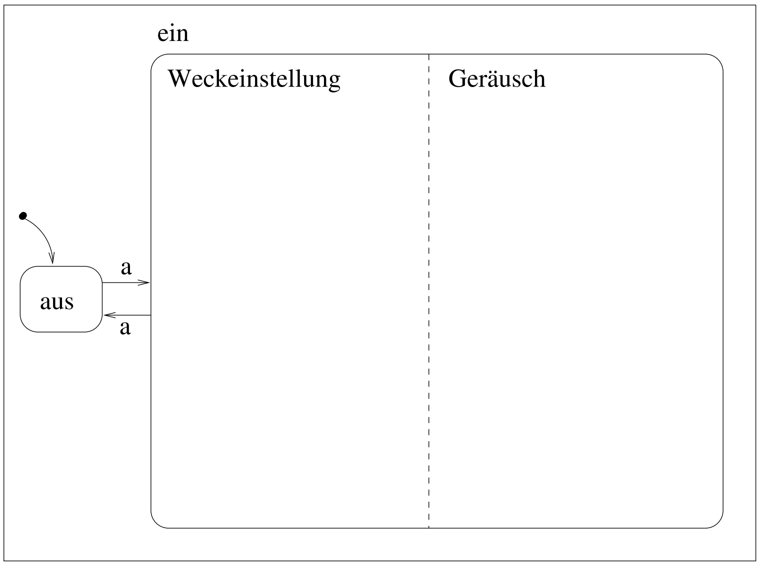
8. Die Wahl zwischen Piepston und Radio erfolgt mittels eines Schalters *b*. Wenn von Piepston auf Radio umgeschaltet wird, wird die zuletzt eingestellte Radiostation wieder eingestellt.

9. Im Normalzustand ist der Wecker leise. Wenn jedoch die eingestellte Weckzeit erreicht ist, gibt er ein Geräusch (sei es ein Piepsen, sei es ein Radioprogramm) von sich.

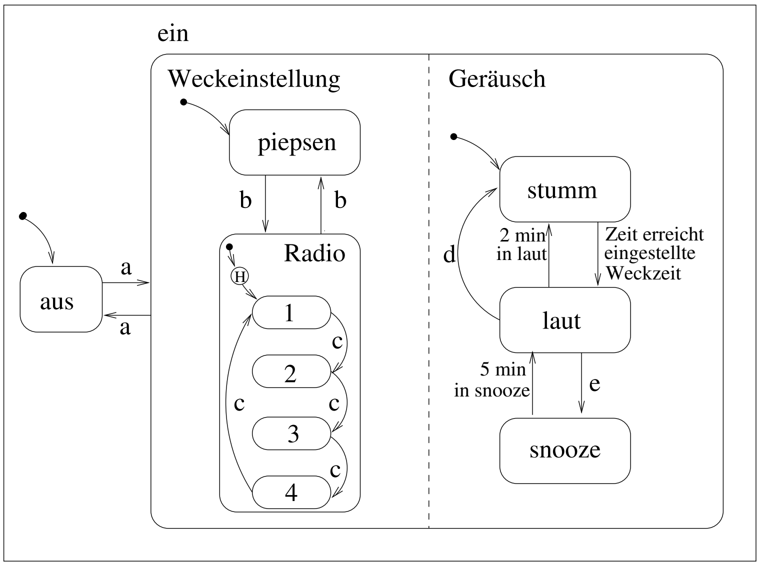
10. Betätigen eines Knopfes *d* schaltet das Geräusch wieder aus.
11. Wenn 2 Minuten lang kein Knopf gedrückt wurde, hört der Wecker wieder auf, Geräusche von sich zu geben.
12. Wenn Knopf *e* („Snooze“) gedrückt wird, gibt der Wecker nach Ablauf von 5 min wieder Geräusche von sich.

Vervollständigen Sie folgendes Statechart gemäß obiger Beschreibung, und verwenden Sie dazu die Ereignisse

- *a* : Schalter *a* (ein/aus) wurde betätigt
- *b* : Schalter *b* (Piepser/Radio) wurde betätigt
- *c* : Knopf *c* (Stationstaste) wurde gedrückt
- *d* : Knopf *d* (Geräusch aus) wurde gedrückt
- *e* : Knopf *e* (Snooze) wurde gedrückt
- *Zeit erreicht eingestellte Weckzeit*
- *n min in Zust*, wobei *n* eine Zahl und *Zust* ein Zustand ist



Vervollständigtes Statechart:



Aufgabe 5 (Abstrakte Datentypen)

(Geschätzte Bearbeitungszeit: 15 min.)

1. Was versteht man im Zusammenhang mit abstrakten Datentypen unter den Begriffen *5 Punkte*

   - Konstruktoren
   - Selektoren
   - Prädikate

   Konstruktoren: alle Elemente des Datentyps können durch Terme dargestellt werden, die ausschließlich aus Konstruktoren aufgebaut sind. Konstruktoren ergeben damit ein Induktionsprinzip für den ADT. Für Konstruktoren werden oftmals keine Axiome angegeben. Sie sind nur in Ausnahmefällen nötig, wenn z.B. die Konstruktoren nicht injektiv sind, wie bei Mengen.
   Selektoren sind Inverse der (nicht konstanten) Konstruktoren. Konstruktoren setzen also Teile zu einem Ganzen zusammen, Selektoren nehmen das Ganze wieder auseinander und liefern die Teile.
   Prädikate sind Funktionen mit Zielsorte **BOOL**, die also Eigenschaften von Elementen des ADT abtesten.

2. Spezifizieren Sie den generischen abstrakten Datentyp (type generator) der *Ternärbäume*. Das sind Bäume, die drei Unterbäume haben. In jedem Knoten eines Ternärbaumes ist ein Datenelement enthalten, dessen Typ generischer Parameter der Datentypdefinition ist. Folgende Funktionen sind zu definieren: *10 Punkte*

   - ein konstanter Konstruktor *empty* für den leeren Baum
   - eine Konstruktorfunktion *make*, die drei Bäume und ein Datenelement zu einem neuen Baum zusammensetzt
   - die entsprechenden Selektorfunktionen
   - ein Prädikat *isempty*, das abprüft, ob der Baum leer ist

- eine Funktion *count*, die die Anzahl der Datenelement in dem Baum zählt

```
type generator TTREE [ITEM]
  sorts TTREE, ITEM, NAT, BOOL
  functions empty:      -> TTREE
            make :      ITEM x TTREE x TTREE x TTREE -> TTREE
            get_item:   TTREE -> ITEM
            get_left:   TTREE -> TTREE
            get_middle: TTREE -> TTREE
            get_right:  TTREE -> TTREE
            is_empty:   TTREE -> BOOL
            count:      TTREE -> NAT
  axioms  t1, t2, t3: TTREE, i: ITEM
    get_item(empty) = undefined
    get_item(make(i, t1, t2, t3)) = i
    get_left(empty) = undefined
    get_left(make(i, t1, t2, t3)) = t1
    get_middle(empty) = undefined
    get_middle(make(i, t1, t2, t3)) = t1
    get_right(empty) = undefined
    get_right(make(i, t1, t2, t3)) = t1
    is_empty(empty) = true
    is_empty(make(i, t1, t2, t3)) = false
    count(empty) = 0
    count(make(i, t1, t2, t3)) = 1 + count(t1) + count(t2) + count(t3)
```