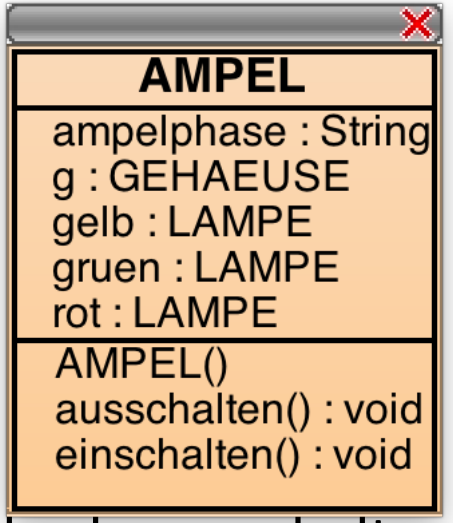
Übung 1 – Ampel

Öffne das BlueJ-Projekt „Ampel_Vorlage“ und öffne die Klasse AMPEL .

a)

Erzeuge ein Objekt von AMPEL und mach dich mithilfe des Objektinspektors und dem Quelltext über die Funktionsweise kundig.



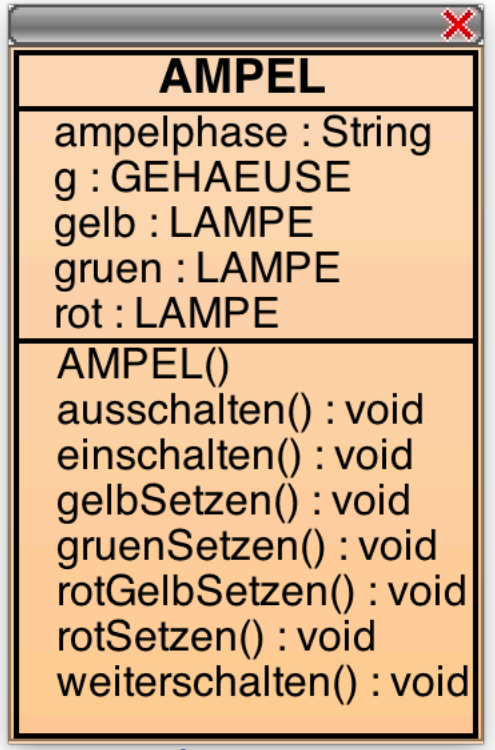
b)

Beschreibe in Worten, welche Objekte beim Ausführen der Methode ausschalten() beteiligt sind und wie sie miteinander kommunizieren.

Das aktuelle Objekt der Klasse AMPEL gibt an die Objekte gruen, gelb und rot der Klasse LAMPE die Botschaft, die Methode aus() aufzurufen.

Übung 1 – Ampel

c)
Implementiere die fehlenden Methoden, sodass eine einfache Simulation einer Ampelschaltung entsteht. Überlege, wie du die Funktionsweise der Methoden beschreiben könntest



Die Methoden gruenSetzen(), gelbSetzen(), rotSetzen() und rotGelbSetzen() arbeiten ähnlich wie die Methode ausschalten(). Zur Beschreibung eignet sich jeweils ein Sequenzdiagramm, da mehrere Objekte miteinander kommunizieren.

Bei der Methode weiterschalten() findet keine Kommunikation zwischen verschiedenen Objekten statt. Zur Beschreibung eignet sich ein Struktogramm oder ein Zustandsdiagramm.

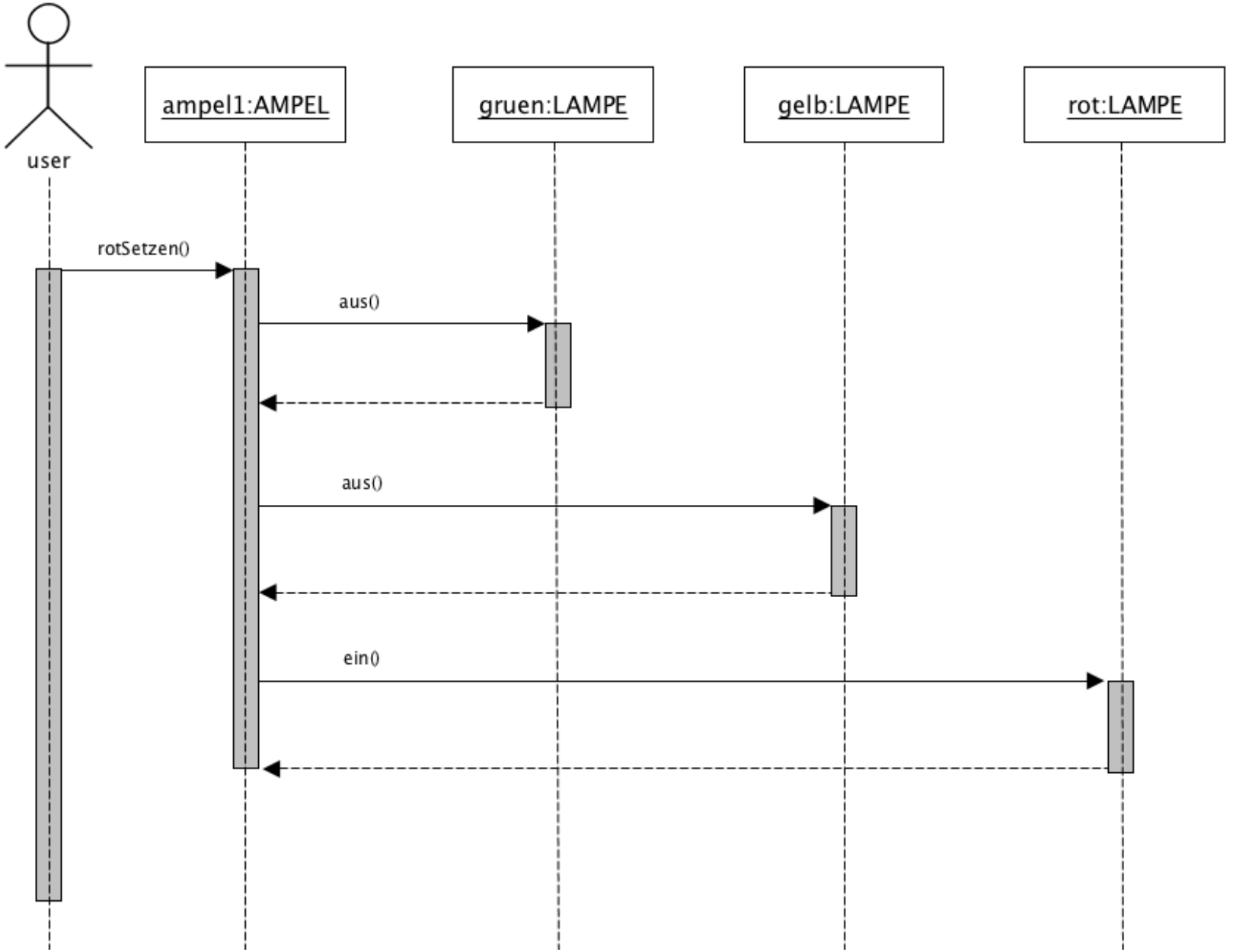
Übung 2 – Ampel

Erstelle ein Sequenzdiagramm für die Methode rotSetzen().

```
public void rotSetzen(){
    gruen.aus();
    gelb.aus();
    rot.an();
    ampelphase = "rot";
}
```

Übung 2 – Ampel Lösung

Sequenzdiagramm für die Methode rotSetzen().



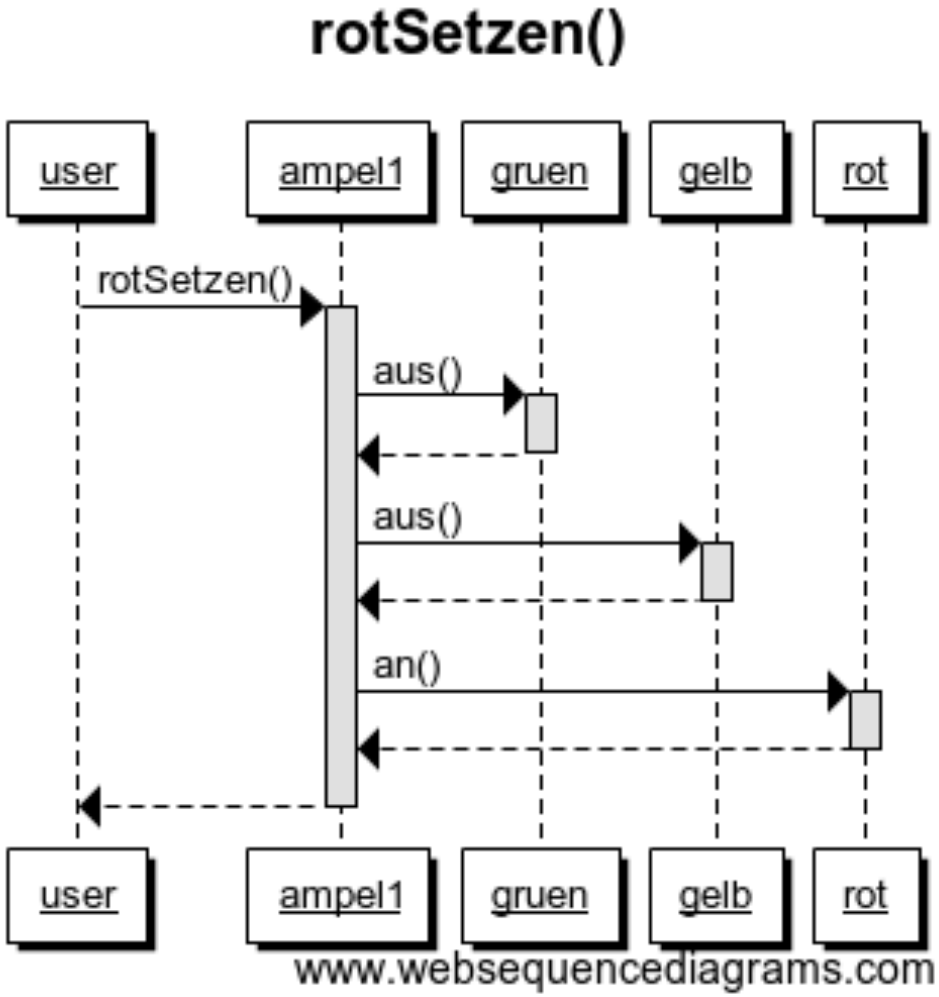
Übung 2 – Ampel Lösung

Sequenzdiagramm
für die Methode
rotSetzen().

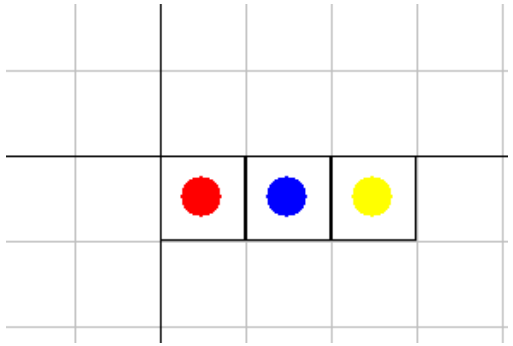
erstellt mit
websequencediagrams.com

Quelltext:

```
title rotSetzen()
user->+ampel1 : rotSetzen()
ampel1->+gruen: aus()
gruen-->-ampel1:
ampel1->+gelb: aus()
gelb-->-ampel1:
ampel1->+rot: an()
rot-->-ampel1:
ampel1-->-user:
```



Übung 3 – Spielautomat



Die Klasse Spielautomat erzeugt (ähnlich wie die Klasse Ampel) drei Lampen lampe1 (links), lampe2 (Mitte) und lampe3 (rechts) der Klasse LAMPE.

Die Position des Spielautomaten ist festgelegt durch die Position von lampe1.

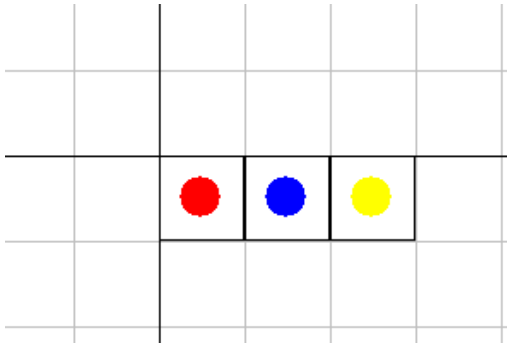
Die Position einer einzelnen Lampe ist festgelegt durch die Koordinaten der linken oberen Ecke. In der Zeichnung ist also die Position gleich (0,0).

(x-Achse nach rechts, y-Achse nach unten)

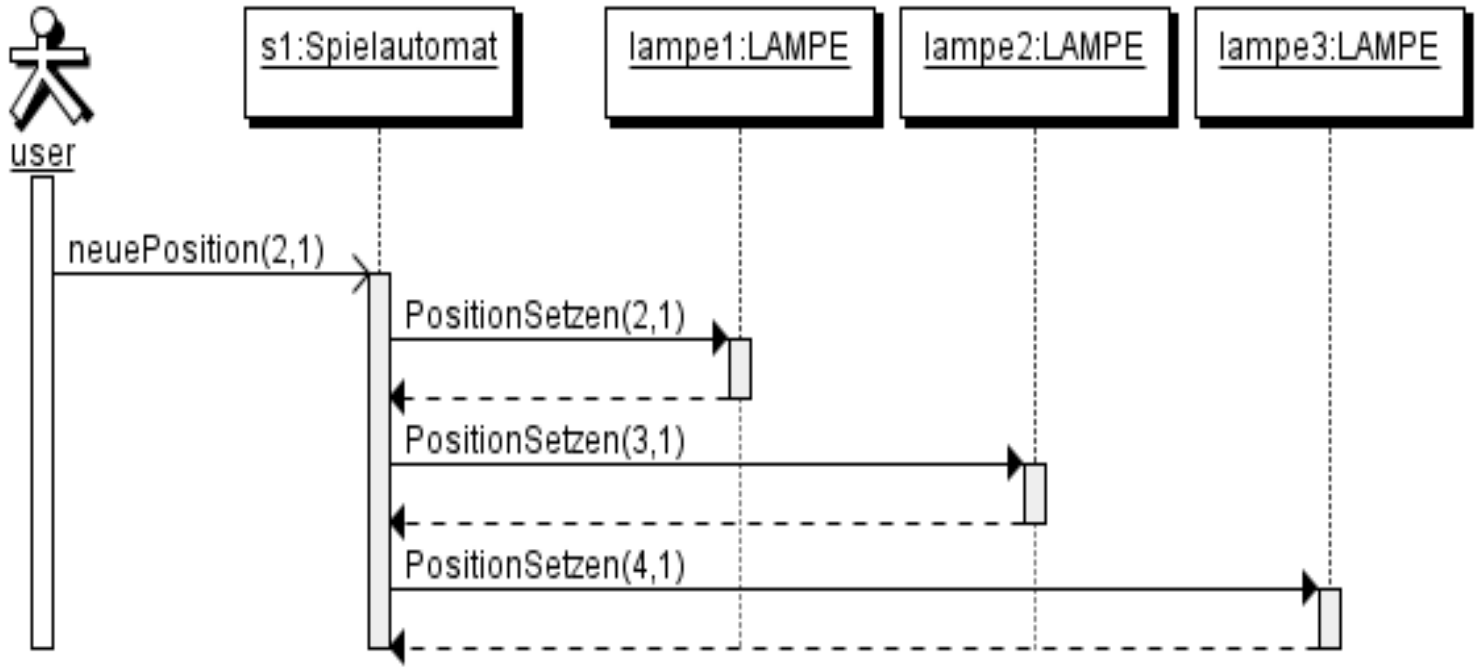
```
public class Spielautomat {
  int positionX, positionY;
  LAMPE lampe1, lampe2, lampe3;

  public Spielautomat(){
    lampe1= new LAMPE();
    lampe2 = new LAMPE();
    lampe3 = new LAMPE();

farbmusterSetzen("rot","blau","gelb");
    neuePosition(0,0);
  }
```

```
public void farbmusterSetzen(String farbe1, String
farbe2, String farbe3){
     lampe1.FarbeSetzen(farbe1);
     lampe2.FarbeSetzen(farbe2);
     lampe3.FarbeSetzen(farbe3);
  }

  public void neuePosition( int xNeu, int yNeu){
      positionX = xNeu;
      positionY = yNeu;
      lampe1.PositionSetzen(positionX,positionY);
      lampe2.PositionSetzen(positionX+1,positionY);
      lampe3.PositionSetzen(positionX+2,positionY);
    }
}
```

Übung 3 – Spielautomat Lösung

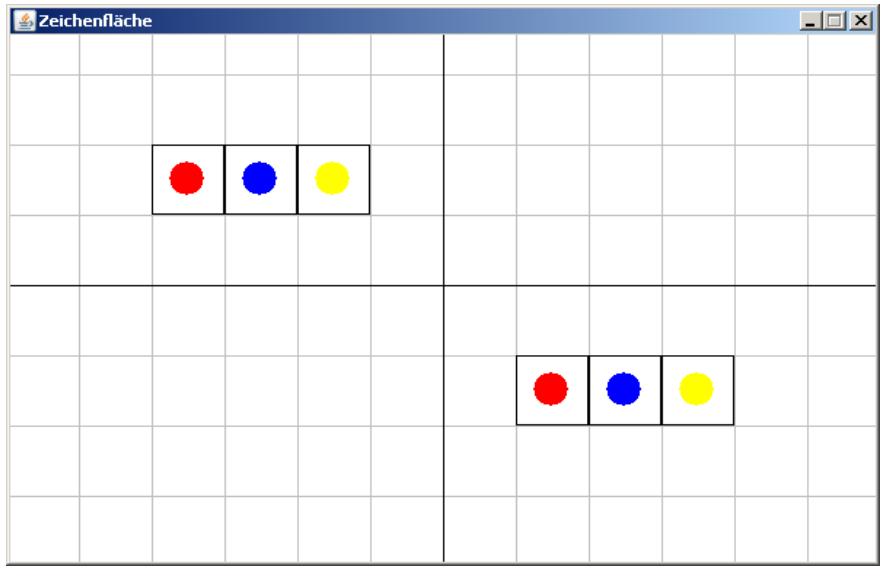


a)
Zeichne ein Sequenzdiagramm von s1.neuePosition(2,1) .
(s1 ist ein Objekt der Klasse Spielautomat)

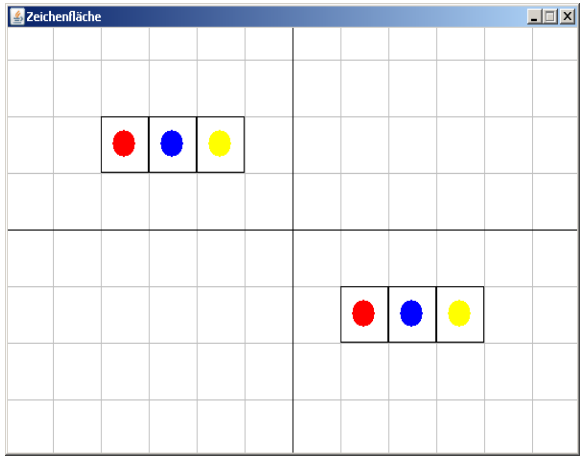


Übung 3 – Spielautomat

b)
Die Klasse Spielhalle enthält zwei Referenzattribute spA1 und spA2 der Klasse Spielautomat.
Im Konstruktor der Klasse werden zwei Spielautomaten (Farbmuster rot-blau-gelb) wie in der Zeichnung erzeugt.
In einer Methode werden die Lampen aller Spielautomaten auf die Farbe schwarz gesetzt.
Schreibe den Quelltext der Klasse Spielhalle.



Übung 3 – Spielautomat Lösung



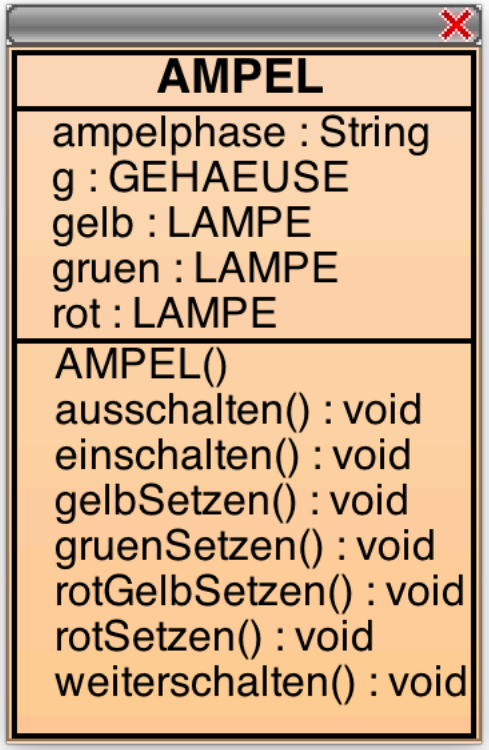
b)

```
public class Spielhalle {
    Spielautomat spA1, spA2;
      public Spielhalle(){
      spA1 = new Spielautomat();
      spA1.neuePosition(1,1);

      spA2 = new Spielautomat();
      spA2. neuePosition (-4,-2);
    }

    public void reset(){
      spA1.farbmusterSetzen("schwarz", "schwarz","schwarz");
      spA2.farbmusterSetzen("schwarz", "schwarz","schwarz");
    }
}
```

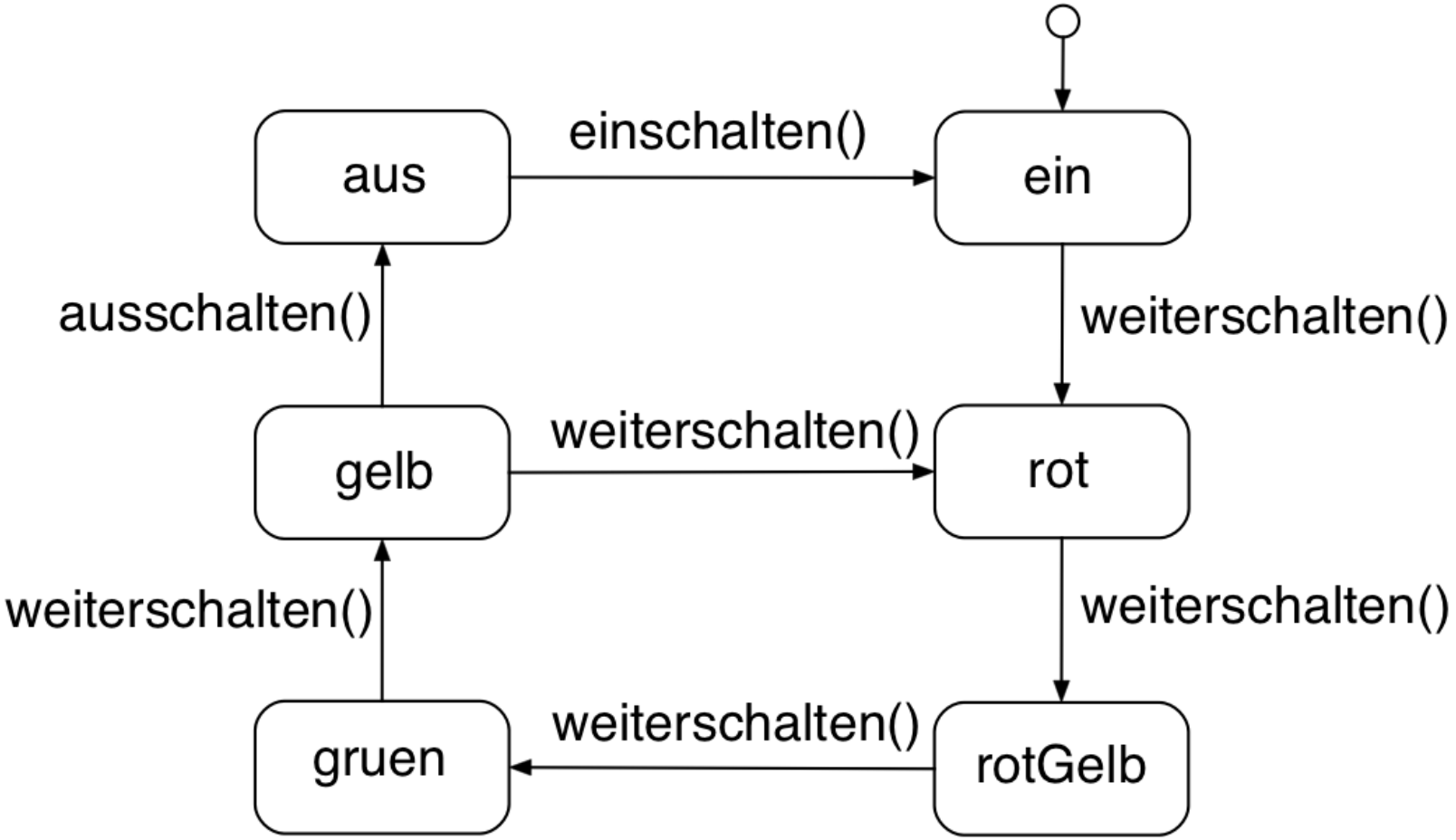
Übung 4 – Ampel Zustandsdiagramm

Die Methode ausschalten() soll die Ampel nur aus der Ampelphase „gelb“ in den Zustand „aus“ überführen.
Nur aus dem Zustand „aus“ soll die Methode einschalten() die Ampel wieder in den Zustand „ein“ überführen.

Ergänze das Zustandsdiagramm und die Implementierung.



Übung 4 – Ampel Zustandsdiagramm Lösung

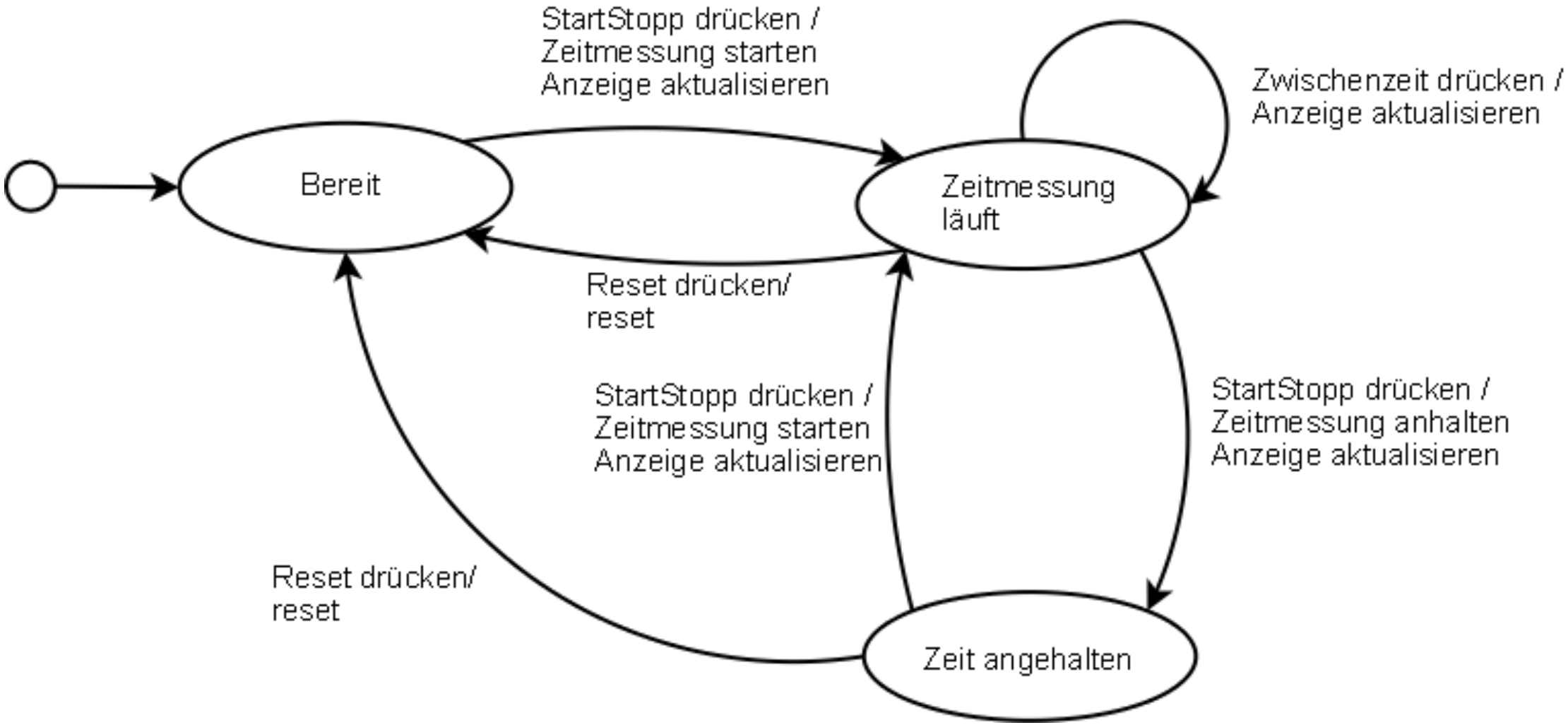


Übung 5 – Zustandsdiagramm, Stoppuhr

Zum Modellieren einer Stoppuhr verwendet man ein Zustandsdiagramm mit den Zuständen „Bereit“, „Zeitmessung läuft“ und „Zeit angehalten“.
Drückt man im Startzustand „Bereit“ die Taste „StartStopp“, startet die Zeitmessung und die Anzeige wird aktualisiert.
Drückt man im Zustand „Zeitmessung läuft“ die Taste „StartStopp“, wird die Zeitmessung angehalten und die Anzeige aktualisiert. Drückt man die Taste „StartStopp“ erneut, läuft die Zeitmessung weiter und die Anzeige wird aktualisiert.
Die Taste „Reset“ löst auf der Uhr die Anzeige „reset“ aus und setzt sie in den Startzustand.
Während der Zeitmessung kann man auch die Taste „Zwischenzeit“ betätigen. Dies aktualisiert die Anzeige.

Zeichne das Zustandsdiagramm.

Übung 5 – Zustandsdiagramm, Stoppuhr Lösung



Übung 6 – Zustandsdiagramm, Spülmaschine

Die Spülmaschine ist zu Beginn im Zustand **"Stand By"**.

Nach Wählen eines Programms ist sie im Zustand "**Programm gewählt".**

Durch Drücken der Taste Start wechselt sie nur dann in den Zustand **"In Betrieb",** wenn die Tür geschlossen ist. In diesem Fall wird der Wasserzulauf geöffnet und das Programm gestartet.

Öffnet man während des laufenden Programms die Tür, wird der Wasserzulauf gestoppt und das Programm angehalten. Die Maschine ist dann im Zustand "**Pause"** und durch Drücken der Taste "Abbrechen" gelangt man in den Zustand "Stand By".

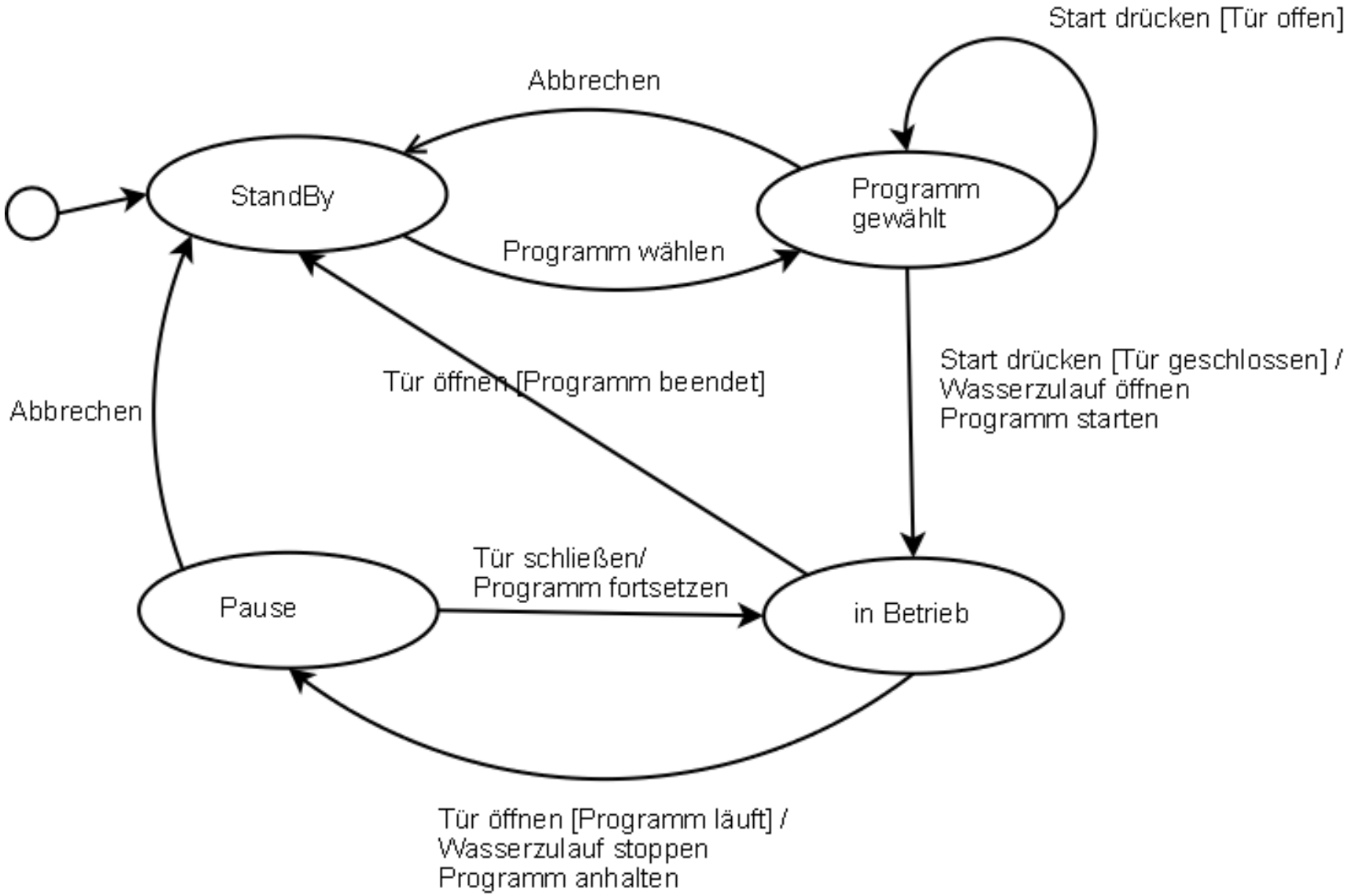
Das Abbrechen ist auch vom Zustand "Programm gewählt" möglich.

Schließt man im Zustand "Pause" wieder die Tür, wird das Programm fortgesetzt.

Öffnet man die Tür nach Beenden des Programms, wechselt die Maschine in den Zustand "Stand By".

Zeichne ein Zustandsdiagramm.

Übung 6 – Zustandsdiagramm, Spülmaschine Lösung



Übung 7 – Zustandsdiagramm, Kühlschrank

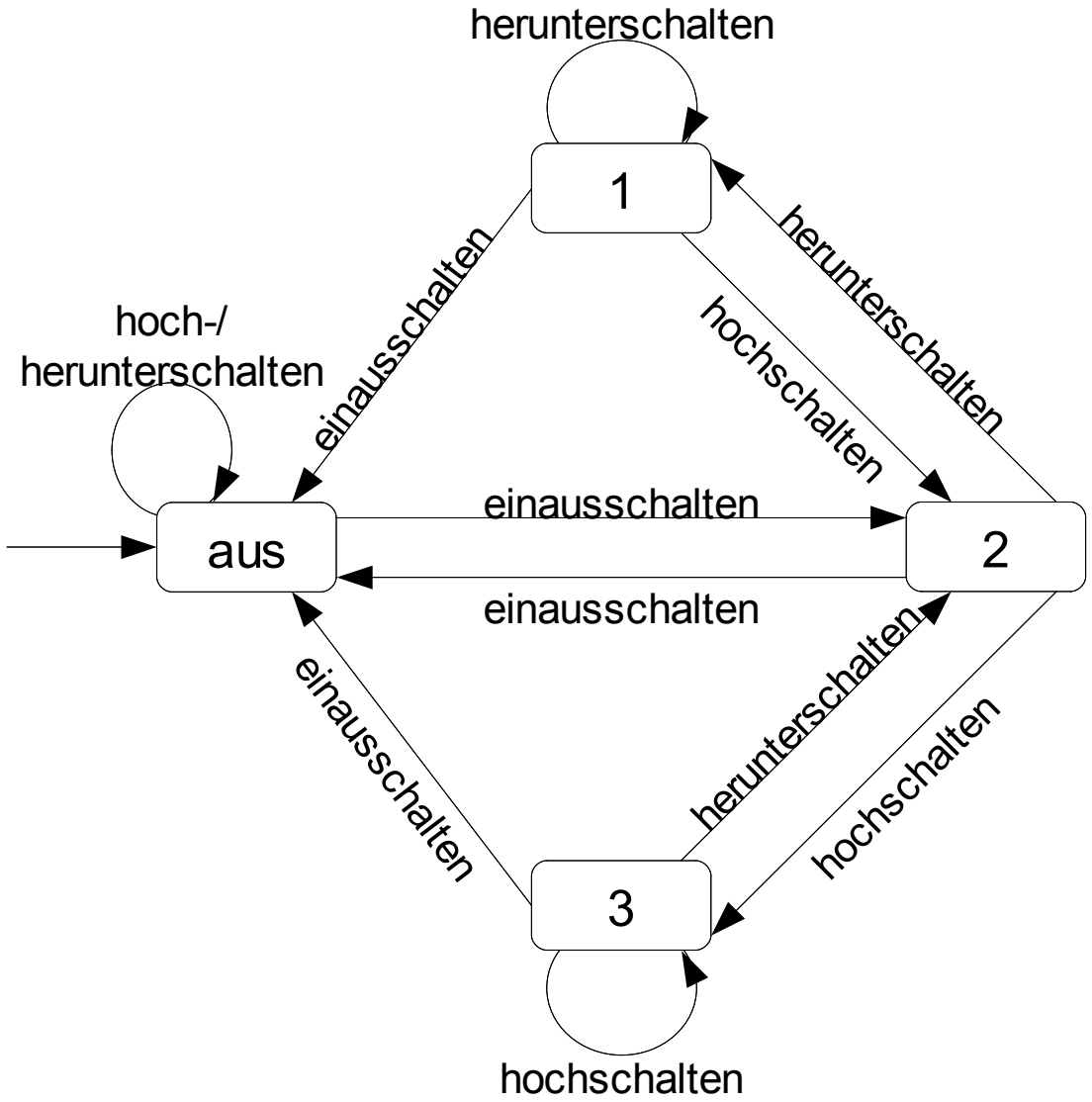
Ein einfacher Kühlschrank besitzt einen Ein-Aus-Schalter, der jederzeit betätigt werden kann. Im eingeschalteten Zustand kann er auf die Kühlstufen 1, 2 oder 3 eingestellt werden. Dafür gibt es die Tasten ▲ zum Hochschalten und ▼ zum Herunterschalten der Kühlstufe. Nach dem Einschalten befindet sich der Kühlschrank immer auf Kühlstufe 2.

a) Modelliere den beschriebenen Kühlschrank mithilfe eines Zustandsdiagramms.

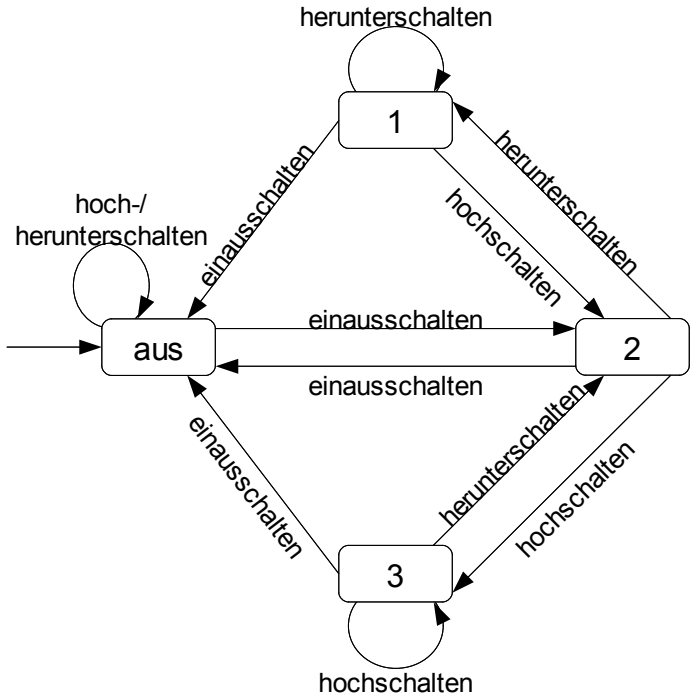
b) Erstelle eine Zustandsübergangstabelle :

Zustand \ auslösende Aktion	einschalten	hochschalten	herunterschalten
aus			
1			
2			
3			

Übung 7 – Zustandsdiagramm, Kühlschrank Lösung



Übung 7 – Zustandsdiagramm, Kühlschrank Lösung



Zustand \ auslösende Aktion	einschalten	hochschalten	herunterschalten
aus	2	aus	aus
1	aus	2	1
2	aus	3	1
3	aus	3	2

Übung 7 – Zustandsdiagramm, Kühlschrank (*)

c)

Implementiere den Kühlschrank durch Definition einer geeigneten Klasse, wobei die möglichen Zustände des Kühlschranks durch die Attribute stufe und eingeschaltet und die auslösenden Aktionen durch die Methoden hochschalten(), herunterschalten() und einausschalten() realisiert werden sollen.

Teste deine Implementierung, indem du die Zustandsübergänge nach unterschiedlichen Methodenaufrufen mit dem Zustandsdiagramm vergleichst. Definiere dazu eine Methode zustandAusgeben(), die den Zustand des Kühlschranks auf dem Bildschirm ausgibt.

Vgl. BlueJ Projekt kuehlschrank_1c