

# BE22

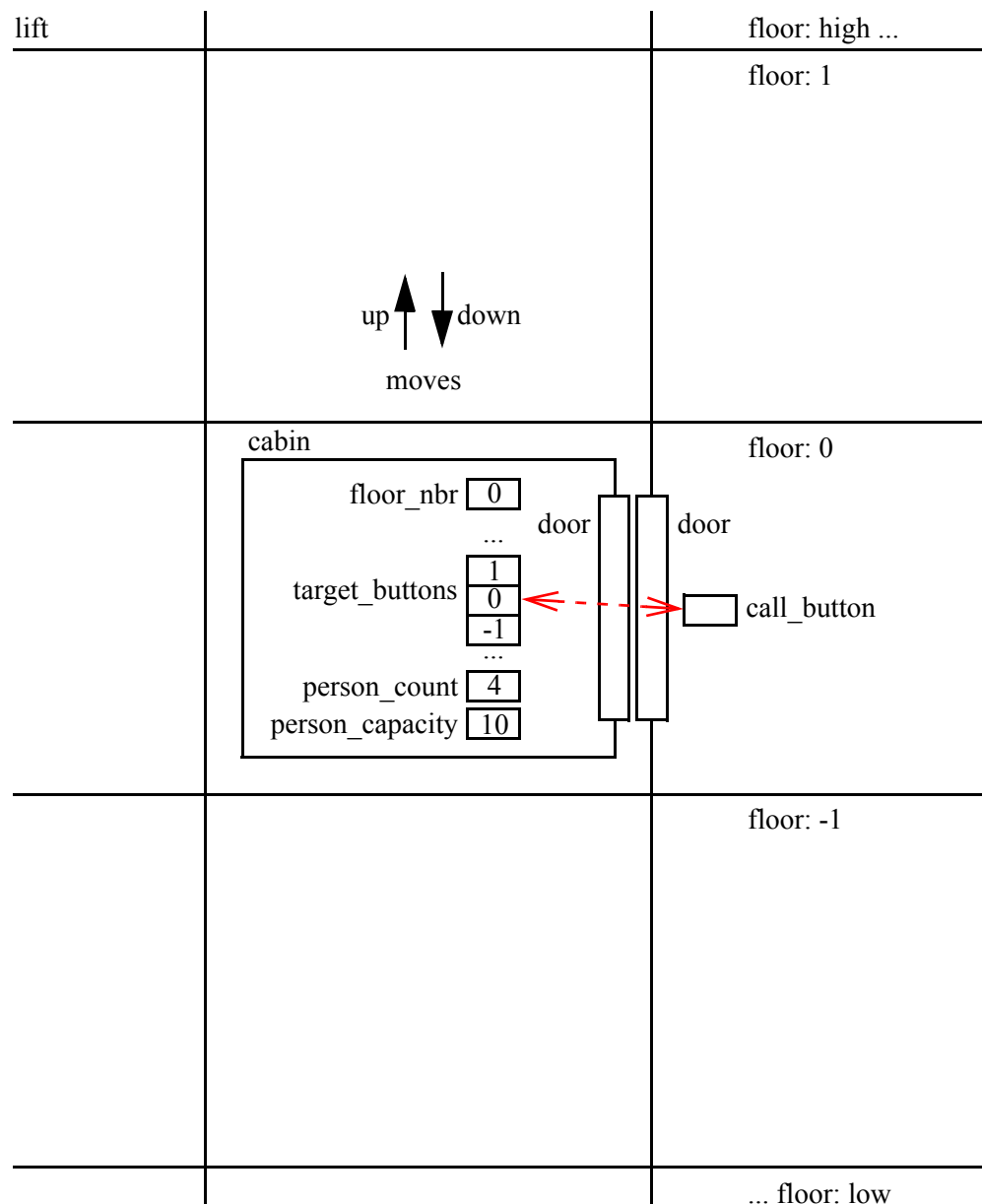
## Beispielentwurf zu PÜ22 Projektübungsblatt: Fahrstuhlsimulation

### BE22.1

### Physisches Modell des Problembereichs

Siehe *LS Leitlinien zum Softwareentwurf*, *Leitlinie LS.2*.

**Bild BE22.1**  
 Physisches Modell  
 eines Fahrstuhls



**BE22.2****Verbale Beschreibung des Problembereichs**

Siehe *LS Leitlinien zum Softwareentwurf, Leitlinie LS.4.*

Der Entwurf beruht auf folgenden Annahmen:

- (1) Ein Fahrstuhl ist ein Lift, ein Stockwerk eine Etage, ein Schalter ein Knopf.
- (2) Ein Lift besteht aus einer Anzahl Etagen und einer Kabine.
- (3) Die Etagen sind von der untersten bis zur obersten lückenlos durchnummeriert.
- (4) Jede Etage hat eine Tür und einen Rufknopf.  
Lifte mit zwei Rufknöpfen zur Auf- und Abfahrt werden nicht modelliert.
- (5) Die Kabine hat eine Tür, für jede Etage einen Zielknopf, und Anzeigen für die aktuelle Etage und die aktuelle und maximale Personenzahl.
- (6) Die Kabine steht auf der aktuellen Etage oder fährt auf- oder abwärts.  
Der Notfall, dass die Kabine zwischen zwei Etagen hängen bleibt, wird nicht modelliert.
- (7) Eine Tür ist geschlossen oder offen oder öffnet oder schließt sich.
- (8) Eine sich schließende Tür öffnet sich wieder, wenn ihre Lichtschranke durchbrochen wird. Das Wiederöffnen wird modelliert, aber ohne Lichtschranke.
- (9) Wenn die Kabine steht, sind alle Türen geschlossen, ausgenommen möglicherweise die Kabinentür und die Tür der aktuellen Etage.
- (10) Die Kabinentür und die Tür der aktuellen Etage bewegen sich höchstens, wenn die Kabine steht, und zwar stets synchron.
- (11) Wenn die Kabine fährt, sind alle Türen geschlossen.
- (12) Drücken eines Knopfs bedeutet, die entsprechende Etage in die Zielmenge der Kabine aufzunehmen.
- (13) Knöpfe werden von Personen gedrückt, vom Lift zurückgesetzt, und zeigen ihren Zustand an.
- (14) Hat die Kabine eine Zieletage erreicht, so wird sie aus der Zielmenge entfernt, die entsprechenden Knöpfe zurückgesetzt.
- (15) Die Kabine fährt so lange in dieselbe Richtung, bis alle Ziele in dieser Richtung erreicht sind; danach wechselt sie bei Bedarf die Fahrtrichtung.
- (16) Bei leerer Zielmenge bleibt die Kabine stehen.

Die Annahmen sind ggf. zu präzisieren und zu ergänzen.

**BE22.3****Entwurfsentscheidungen**

Siehe *LS Leitlinien zum Softwareentwurf, Leitlinie LS.4.*

- (1) Ein realer Lift hat mehr Teile als in BE22.2 beschrieben, z.B. eine Steuerkomponente und einen Motor. Diese werden nicht modelliert, stattdessen übernimmt z.B. die Kabine die Steuerung. Modelliert wird nur das funktionale Äußere eines Lifts, die Sicht der Fahrgäste auf die Funktionen des Lifts.
- (2) Zum Problembereich gehören als unabhängige Akteure ein Lift und beliebig viele Personen. Die inhärente Nebenläufigkeit wird auf ein sequenzielles Simulationsprogramm abgebildet.
- (3) Die Simulation lässt den Lift kleine Schritte gehen, damit sie dazwischen Personen Knöpfe drücken lassen kann.
- (4) Personen werden nicht als Objekte modelliert. Die Simulation spielt ihre Rolle.
- (5) Die Simulation wird randomisiert: Aktionen werden zufällig ausgewählt, um Nebenläufigkeit zu simulieren.

- (6) Etagenruftknopf und Kabinenzielknopf müssen synchron schalten. Dafür wird eine einfache, ggf. zunächst unrealistische, Lösung gesucht.
- (7) Für einen gegenseitigen Botschaftenaustausch zwischen den Knöpfen müssen sie sich gegenseitig kennen. Diese raffinierte Variante wird später modelliert, s. 22.6 S. 8.

Die Entwurfsentscheidungen sind ggf. zu präzisieren und zu ergänzen.

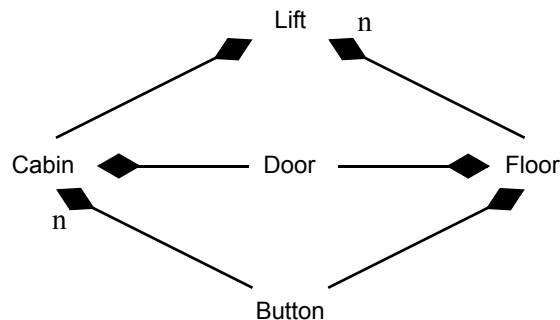
## BE22.4

### Klassendiagramm

Siehe *LS Leitlinien zum Softwareentwurf*, Leitlinien *LS.15* bis *LS.21*.

Bild BE22.2

Klassendiagramm Lift



## BE22.5

### Schnittstellenspezifikationen mit Verträgen

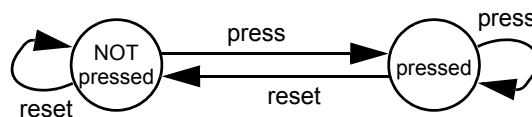
Siehe *LS Leitlinien zum Softwareentwurf*, Leitlinien *LS.28* bis *LS.42*.

### BE22.5.1

#### Knopf

Bild BE22.3

Zustandsdiagramm  
Knopf



Aktionen und Nachbedingungen dem Bild BE22.3 Zustandsdiagramm Knopf entnehmen.

Programm BE22.1

Schnittstelle Knopf

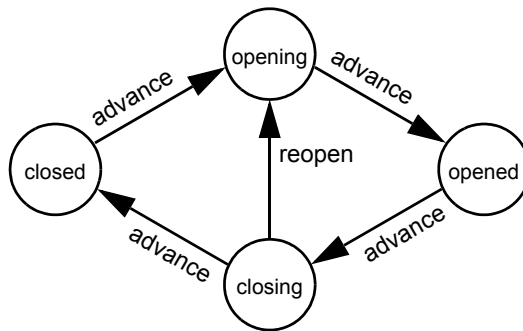
```

INTERFACE Button
  QUERIES
    pressed : BOOLEAN
  ACTIONS FOR Lift_Simulation -- simulating Person
    press
    POST
      pressed
    END
  ACTIONS FOR Cabin, Lift
    reset
    POST
      NOT pressed
    END
END Button
  
```

Der rot markierte Kunde **Lift** ist nur für die erste Version der unabhängigen Knöpfe erforderlich.

**BE22.5.2****Tür****Bild BE22.4**

Zustandsdiagramm Tür



Aktionen, Vor- und Nachbedingungen dem Bild BE22.4 Zustandsdiagramm Tür entnehmen.

**Programm BE22.2**

Schnittstelle Tür

```

INTERFACE Door
  QUERIES
    state : (closed, opening, opened, closing)
  ACTIONS FOR Lift
    advance
      POST
        OLD (state) = closed IMPLIES state = opening;
        OLD (state) = opening IMPLIES state = opened;
        OLD (state) = opened IMPLIES state = closing;
        OLD (state) = closing IMPLIES state = closed;
      END
    reopen
      PRE
        state = closing
      POST
        state = opening
      END
  END Door
  
```

**BE22.5.3****Etage****Programm BE22.3**

Schnittstelle Etage

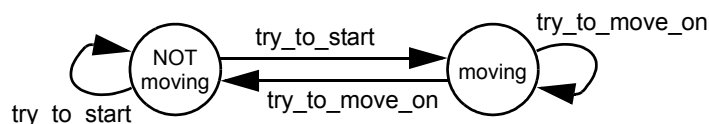
```

INTERFACE Floor
  QUERIES
    door : Door
    button : Button
  END Floor
  
```

**BE22.5.4****Kabine****Bild BE22.5**

Zustandsdiagramm Kabine

Kabine



Aktionen, Vor- und Nachbedingungen dem Bild BE22.5 Zustandsdiagramm Kabine entnehmen.

Statt einer separaten Initialisierungs- oder Erzeugungsprozedur oder eines Konstruktors verwenden wir die kompaktere Syntax:

**Programm BE22.4**  
Schnittstelle Kabine

```
INTERFACE Cabin
  (IN low, high : INTEGER PRE low < high;
   IN person_capacity : NATURAL PRE person_capacity > 4)
```

```
INTERFACE Cabin
  (IN low, high : INTEGER PRE low < high;
   IN person_capacity : NATURAL PRE person_capacity > 4)

  QUERIES -- elementary
    door          : Door
    buttons       : Array OF Button RANGE low .. high
    floor_nbr     : low .. high
    person_count,
    person_capacity : NATURAL
    moving,
    up            : BOOLEAN

  QUERIES -- derived
    up_requested,
    down_requested : BOOLEAN

  INVARIANTS
    person_count <= person_capacity;
    person_capacity > 4;
    moving IMPLIES door.state = closed;

    up_requested = EXISTS i IN floor_nbr + 1 .. high : buttons [i].pressed;
    down_requested = EXISTS i IN low .. floor_nbr - 1 : buttons [i].pressed

  ACTIONS FOR Lift
    try_to_start
      -- Check if targets are requested. If any, start moving.
      -- Change buttons [floor_nbr].pressed, moving, up.
      PRE
        NOT moving;
        door.state = closed
      POST
        door.state = closed;
        NOT buttons [floor_nbr].pressed;
        ((OLD (up) OR (NOT OLD (up) AND NOT down_requested)) AND up_requested)
          IMPLIES (moving AND up);
        ((NOT OLD (up) OR (OLD (up) AND NOT up_requested)) AND down_requested)
          IMPLIES (moving AND NOT up);
        (moving OR up # OLD (up)) IMPLIES (up_requested OR down_requested)
      END

    try_to_move_on
      -- Move to next floor in current direction. If it is a target, stop moving.
      -- If an end is reached, reverse direction.
      -- Change floor_nbr, moving, up.
      PRE
        moving;
        door.state = closed
      POST
        door.state = closed
        OLD (up) IMPLIES floor_nbr = OLD (floor_nbr) + 1;
        NOT OLD (up) IMPLIES floor_nbr = OLD (floor_nbr) - 1;
        buttons [floor_nbr].pressed IMPLIES NOT moving;
        floor_nbr IN {low, high} IMPLIES up = NOT OLD (up)
      END

  END Cabin
```

**BE22.5.5****Lift****Programm BE22.5**

Schnittstelle Lift

```

INTERFACE Lift
  (IN low, high : INTEGER PRE low < high;
   IN person_capacity : NATURAL PRE person_capacity > 4)

  QUERIES
    cabin    : Cabin (low, high, person_capacity)

    floors   : Array OF Floor RANGE low .. high

  INVARIANTS
    FOR_ALL i IN low .. high : (i # cabin.floor_nbr IMPLIES floors [i].door.state = closed);
    cabin.moving IMPLIES floors [cabin.floor_nbr].door.state = closed;
    NOT cabin.moving IMPLIES cabin.door.state = floors [cabin.floor_nbr].door.state

  ACTIONS FOR Lift_Simulation
    step_on
    DO
      IF cabin.moving THEN
        cabin.try_to_move_on
      ELSE
        INSPECT cabin.door.state
        | closed :
          SELECT
          | true :
            floors [cabin.floor_nbr].button.reset
            cabin.try_to_start
          | true :
            cabin.door.advance
            floors [cabin.floor_nbr].door.advance
          END
        | opening :
          cabin.door.advance
          floors [cabin.floor_nbr].door.advance
        | opened :
          SELECT
          | cabin.person_count < cabin.person_capacity :
            cabin.person_count.increment (random)
          | cabin.person_count > 0 :
            cabin.person_count.decrement (random)
          | true :
            cabin.door.advance
            floors [cabin.floor_nbr].door.advance
          END
        | closing :
          SELECT
          | true :
            cabin.door.advance
            floors [cabin.floor_nbr].door.advance
          | true :
            cabin.door.reopen
            floors [cabin.floor_nbr].door.reopen
          END
        END INSPECT
      END IF
    END step_on
  END Lift

```

Eine vertragliche Spezifikation der Aktion `step_on` von Lift ist mit einigen Implikationen möglich, aber umfangreich und unübersichtlich. Deshalb ziehen wir hier eine operationale Spezifikation in imperativem Pseudocode vor. Um die algorithmische Modellierung zu vereinfachen, nutzen wir Dijkstras nichtdeterministische Auswahlanweisung,

hier SELECT genannt, die zufällig einen Fall mit erfüllter Bedingung wählt.<sup>1</sup> Die INSPECT-Anweisung entspricht der CASE-, *inspect*- und *switch*-Anweisung von *Pascal*, *Eiffel* bzw. *C\**.

Die Zufallsauswahlen bei den SELECT-Abschnitten können realitätsferne Situationen bewirken. Beispielsweise gilt für Lift.step\_on im ersten SELECT-Abschnitt: Ist die Kabinentür geschlossen, so kann die Kabine starten oder die Türen können sich öffnen. Dadurch kann es vorkommen, dass die Kabine auf einer angeforderten Etage anhält (Cabin.try\_to\_move\_on) und sofort wieder startet (Cabin.try\_to\_start), ohne dass sich die Türen dazwischen öffnen.<sup>2</sup> Das ist einerseits nicht realistisch modelliert, andererseits fehlt die Anforderung, dass sich nach dem Anhalten der Kabine die Türen öffnen sollen, in der Problembeschreibung BE22.2 S. 2. Der zweite SELECT-Abschnitt in Lift.step\_on schwächelt ähnlich: Es kann vorkommen, dass sich die Türen öffnen und sofort wieder schließen, ohne dass dazwischen Personen ein- oder aussteigen. Wie sind Beschreibung und Entwurf zu ändern, um solche Mängel zu beseitigen und zusätzliche Anforderungen zu erfüllen?

Die rot markierte Zeile

```
floors [cabin.floor_nbr].button.reset
```

erscheint nur in der ersten Version der unabhängigen Knöpfe. Der Lift setzt den Etagenknopf vor dem Starten zurück, die Kabine ihren Knopf beim Starten.

## BE22.5.6

### Programm BE22.6

Schnittstelle

Liftsimulation

## Liftsimulation

```
INTERFACE Lift_Simulation
  ACTIONS FOR command line
    run -- make, main with arguments from command line
      (IN low, high : INTEGER PRE low < high;
       IN person_capacity : NATURAL PRE person_capacity > 4)
      DO
        lift : Lift (low, high, person_capacity)
        UNTIL user requests break LOOP
          SELECT
            -- person's actions:
            | true :
              i : low .. high := random
              lift.floors [i].button.press
              -- To synchronize the cabin button, this is not realistic, but simple:
              lift.cabin.buttons [i].press
            | lift.cabin.person_count > 0 :
              lift.cabin.buttons [random].press
            -- lift's actions:
            | true :
              lift.step_on
          END SELECT
        END LOOP
      END run
END Lift_Simulation
```

<sup>1</sup> Edsger W. Dijkstra: *Guarded Commands, Nondeterminacy and Formal Derivation of Programs*. Comm. ACM, Vol. 18, No. 8 (Aug. 1975) S. 453–457, <http://dl.acm.org/citation.cfm?id=360975> (Zugriff: 2011-11-14).

<sup>2</sup> Tobias Ableitner hat diese Schwäche erkannt; außerdem hat er in der Vorversion entdeckt, dass die dritte Invariante von Call\_Button fehlerhaft war und die vierte Nachbedingung von Cabin.try\_to\_move\_on in der *Eiffel*-Implementation zu einem Fehler führte. Ich bewundere seinen Scharfblick und danke ihm für die Hinweise (11.11.2011).

Auch der SELECT-Abschnitt in Lift\_Simulation.run kann zu einer unrealistischen Situation führen: Es werden keine Knöpfe gedrückt, wegen fehlender Ziele bleibt die Kabine stehen, auf derselben Etage wiederholt sich das Öffnen der Türen, das Aus- und Einsteigen von Personen, das Schließen und Öffnen der Türen. Dem Nachteil der Realitätsferne steht der Vorteil der Einfachheit des nichtdeterministischen Algorithmus gegenüber.

Die rot markierte Zeile

`lift.cabin.buttons [i].press`

erscheint nur in der ersten Version der unabhängigen Knöpfe. Die Liftsimulation in der Rolle einer Person drückt nicht nur den Etagenknopf, sondern auch den entsprechenden Kabinenknopf.

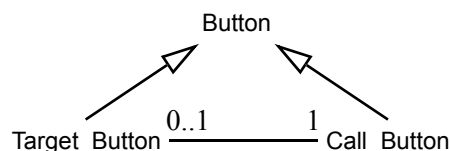
## BE22.6

## Synchronisation der Knöpfe

### BE22.6.1

### Entwurf

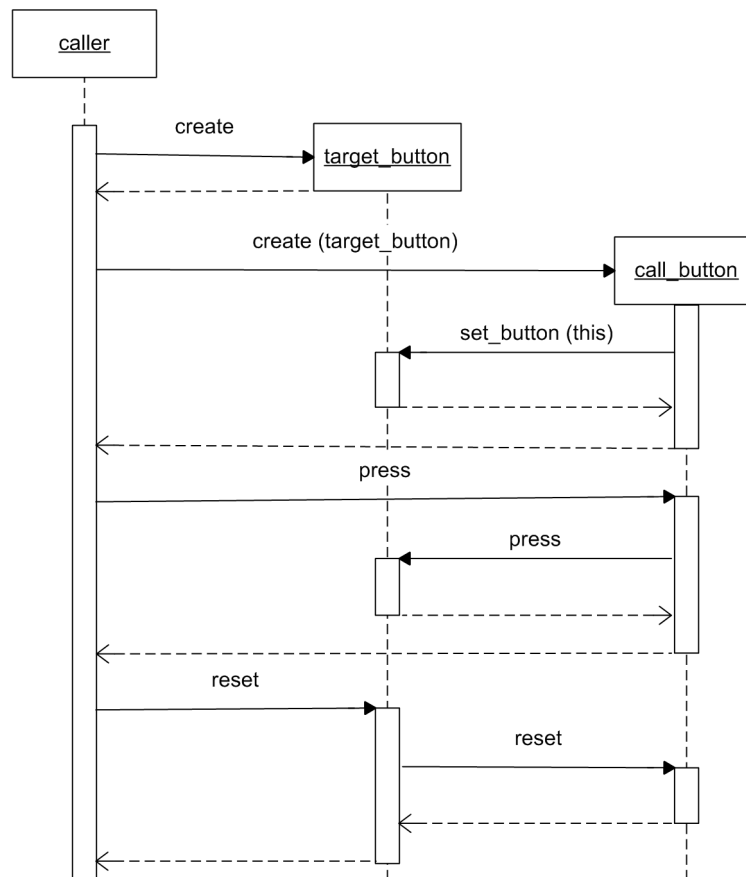
**Bild BE22.6**  
Klassendiagramm  
Knöpfe



Ideen

- (1) Target\_Button und Call\_Button kennen sich gegenseitig, genauer: jeweils ein Partnerobjekt.
- (2) Call\_Button.press ruft Call\_Button.target\_button.press.
- (3) Target\_Button.reset ruft Target\_Button.call\_button.reset.
- (4) Problem: Zirkulare Abhängigkeit zwischen Target\_Button und Call\_Button. Lösung: Target\_Button kennt nur Button, nicht Call\_Button. Call\_Button kann Target\_Button kennen.
- (5) Reihenfolge der Initialisierung: Ein neues Call\_Button-Objekt erhält bei der Initialisierung eine Referenz auf sein Target\_Button-Partnerobjekt, das zuvor schon erzeugt wurde, ohne sein Call\_Button-Partnerobjekt zu kennen. Das Call\_Button-Objekt macht sich dann bei seinem Target\_Button-Partnerobjekt bekannt.

**Bild BE22.7**  
Sequenzdiagramm  
Knöpfe<sup>1</sup>



## BE22.6.2

### Knopf

Änderung: Bei reset Kunde Lift entfernt, Kunde Button ergänzt.

**Programm BE22.7**  
Schnittstelle Knopf

```

INTERFACE Button
  QUERIES
    pressed : BOOLEAN
  ACTIONS FOR Lift_Simulation -- simulating Person
    press
    POST
      pressed
    END
  ACTIONS FOR Cabin, Button
    reset
    POST
      NOT pressed
    END
END Button
  
```

## BE22.6.3

### Zielknopf

Änderungen:

- (1) Zusätzliche Aktion set\_button für Button.
- (2) Zugriff auf press auf Button eingeschränkt, sonst geerbt.
- (3) Nachbedingung von reset verstärkt. Kunde Lift entfernt.

<sup>1</sup> Erstellt mit Microsoft Visio UML Sequence, importiert als GIF-Datei. Ein Versuch war's vielleicht wert.

**Programm BE22.8**  
Schnittstelle Zielknopf

```

INTERFACE Target_Button EXTENDS Button

    PROTECTED
        button : Button

    ACTIONS FOR Button
        set_button (IN new_button : Button)
            PRE
                new_button # void
            POST
                button = new_button
            END

        press

    ACTIONS FOR Cabin
        reset REDEFINED
        POST AND
            button # void IMPLIES NOT button.pressed
        END

END Target_Button

```

**BE22.6.4**

**Rufknopf**

Änderungen:

- (1) Zusätzliche Aktion init für Lift.
- (2) Nachbedingung von press verstärkt.

**Programm BE22.9**  
Schnittstelle Rufknopf

```

INTERFACE Call_Button EXTENDS Button

    PROTECTED
        target_button : Target_Button

    INVARIANTS
        target_button # void;
        target_button.button = this;
        pressed IMPLIES target_button.pressed

    ACTIONS FOR Lift
        init -- creation procedure, constructor
            (IN new_target_button : Target_Button)
            PRE
                new_target_button # void
            DO
                target_button := new_target_button
                target_button.set_button (this)
            POST
                target_button = new_target_button
            END

    ACTIONS FOR Lift_Simulation
        press REDEFINED
        DO
            super
            target_button.press
        POST AND
            target_button.pressed
        END

END Call_Button

```

**BE22.6.5****Etage**

**Programm BE22.10**  
Schnittstelle Etage

```
INTERFACE Floor (IN target_button : Target_Button PRE target_button # void)
  QUERIES
    door    : Door
    button  : Call_Button (target_button)
END Floor
```

**BE22.6.6****Kabine**

**Programm BE22.11**  
Schnittstelle Kabine

```
INTERFACE Cabin (...)
  QUERIES -- elementary
    ...
    buttons : Array OF Target_Button RANGE low .. high
    ...
END Cabin
```

**BE22.6.7****Lift**

**Programm BE22.12**  
Schnittstelle Lift

```
INTERFACE Lift
  QUERIES
    ...
    floors : Array OF Floor RANGE low .. high
  INVARIANTS
    ...
  ACTIONS FOR Lift_Simulation
    init -- creation procedure, constructor
      (IN low, high : INTEGER PRE low < high;
       IN person_capacity : NATURAL PRE person_capacity > 4)
    DO
      FOR_ALL i IN low .. high LOOP
        floors [i].init (cabin.buttons [i])
      END
    END init
    ...
  END Lift
```