St. Petersburg State Institute of Fine Mechanics
and Optics (Technical University)

Department of Computer Technologies

A.S. Naumov, A.A. Shalyto

Elevator control system

Object-oriented programming
with explicit state selection

Project documentation

The project has been developed in the context of the
"Foundation for open project documentation"
http://is.ifmo.ru

St. Petersburg
2003

Table of Contents

Introduction

A SWITCH-technology has been proposed by A.A. Shalyto for the algorithmization and programming of logic control problems. This technology was further developed by the author together with N.I. Tukkel and applied to event and object-oriented programs. For more details about this technology and different examples of its usage see the following Web sites: http://is.ifmo.ru and http://www.softcraft.ru.

This technology is convenient for technical object control problems such as an elevator control program, considered in this paper. It is so, because control logic becomes more centralized in the source code. Another advantage of this approach is that the code becomes isomorphic to the transition graph, by which it was constructed. It allows to consider only transition graphs in order to understand the program logic (or behaviour) rather than to analyze source codes.

The Java programming language has been selected for implementation so that an application should be executed in the form of an applet. The term "aplet" is hereinafter referred to with one "p" due to the fact that such spelling has been taken on the Web site of http://www.sun.ru.

The object-oriented programming and the automata-based programming, used in this paper, are called by A.A. Shalyto and N.I. Tukkel as "object-oriented programming with explicit state selection". A peculiarity of the example is that only one of the classes represents all the automata. This contains seven automata, each of them is one of the methods of this class. Non-automata methods are not present in this class.

Two threads are used in the given implementation, in one of which all automata are started.

1. Problem definition

The aim of the paper is to develop an elevator control program. For the sake of project simplicity, the building is a 3-storey building.

There is two buttons on the panel, located on the each floor ("Up" and "Down"). Such panels located on extreme floors have only one button. In the elevator car there are buttons indicating floor numbers and also there is a "Stop" ("S") button, when being pressed it stops the elevator on the floor which is the nearest in the elevator moving direction. In addition, an elevator car is equipped by a weight indicator (not displayed visually) and a door closing timer in case of nobody being present in the car. There are also "Come in" and "Go out" buttons that shall be pressed to come into the car and to go out of it.

2. Class diagram

A program, which class diagram is given in Fig. 1, consists of two parts.

The first part consists of one class and two interfaces:

- *ElevatorAutomaton* – a class that encapsulates the operating logic of the elevator control system – it implements all automata (Section 3);
- *ElevatorLogInterface* – an interface that declares methods providing logging of the application operation logic (Section 4);
- *ElevatorVisualizerInterface* – an interface that declares all input variables and output actions (Section 5).

The second part consists of three classes:

- *ElevatorLog* – implements the *ElevatorLogInterface* interface and provides logging of the application operation logic (Section 6);
- *ElevatorVisualizer* – implements the *ElevatorVisualizerInterface* interface, emulates and visualises the elevator operation. It performs the start of automata and processing of events. It implements the methods of input variables and output actions (Section 7);
- *ElevatorApplet* – creates copies of the *ElevatorLog* and *ElevatorVisualizer* classes and places them on the aplet panel (Section 8).

This paper has shown how the control system can be separated from the object (elevator) control. The control system is implemented by the first part of the program and the control object is implemented by its second part.

Each of the above-listed classes and interfaces are described below, with a class that contains automata being described in greater detail.

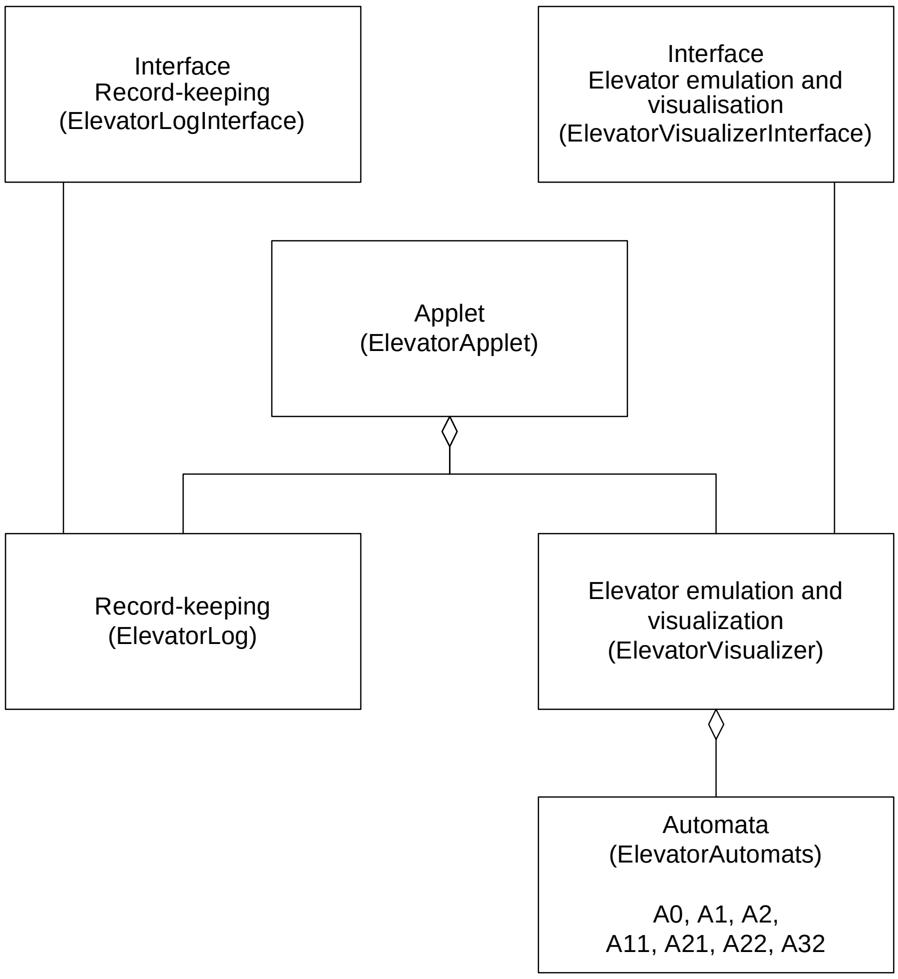


Fig. 1. Class diagram

3. "ElevatorAutomats" class

3.1. Description

The class encapsulates the operation logic for the elevator control system. The class constructor receives references to the *ElevatorLogInterface* and *ElevatorVisualizerInterface* interfaces. Seven automata are implemented in the class, i.e. public methods of the form of `public void Ann( int e )` where `nn` is the automaton number, `e` is the event.

The state of each automaton is stored in a protected variable `protected int ynn` where `nn` is the automaton number. The initial state of all automata is zero. On transition graphs, the predicate of "`ynn=s`" verifies, whether there is the automaton with the number of `nn` is in the state with the number of `s`.

Let us note that identical events, input variables and output actions can be used in different automata.

3.2. Numeration and list of events (e)

0	The common timer has operated
2	Switching-off of the lamp in the button (to be delivered from the A0 automaton)
3	Switching-off of the lamp in the button (to be delivered from the A0 automaton)
4	Enabling of switching-on of the lamp in the button (to be delivered from the A0 automaton)
8	Setting of the value of "Down" (to be delivered from the A0 automaton)
9	Setting of the value of "Up" (to be delivered from the A0 automaton)
11	The "Up" button on the first floor was pressed
21	The "Up" button on the second floor was pressed
22	The "Down" button on the second floor was pressed
32	The "Down" button on the third floor was pressed
41	The "1" button in the elevator car was pressed
42	The "2" button in the elevator car was pressed
43	The "3" button in the elevator car was pressed
44	The "S" (STOP) button in the elevator car
60	Arrival at the floor
80	The doors have opened
81	The doors have closed
90	The door closing timer has operated

3.3. Numeration and list of input variables (x)

51	A weight is present in the elevator car
61	The elevator is on the 1st floor
62	The elevator is on the 2nd floor
63	The elevator is on the 3rd floor

All variables being examined by automata are declared in the *ElevatorVisualizerInterface* interface (Section 5) and implemented in the *ElevatorVisualizer* class (Section 7).

3.4. Numeration and list of output actions (z)

110	Switch off the lamp in the "Up" button on the first floor
111	Switch on the lamp in the "Up" button on the first floor
210	Switch off the lamp in the "Up" button on the second floor
211	Switch on the lamp in the "Up" button on the second floor
220	Switch off the lamp in the "Down" button on the second floor

221	Switch on the lamp in the "Down" button on the second floor
320	Switch off the lamp in the "Down" button on the third floor
321	Switch on the lamp in the "Down" button on the third floor
400	Switch off the lamp in the button
411	Switch on the lamp in the "1" button in the elevator car
421	Switch on the lamp in the "2" button in the elevator car
431	Switch on the lamp in the "3" button in the elevator car
441	Switch on the lamp in the "S" (STOP) button in the elevator car
500	Switch off the lamp in the elevator car
501	Switch on the lamp in the elevator car
700	Stop the elevator car
701	Start the upward motion
702	Start the downward motion
800	Automatically stop the door motion
801	Open the doors
802	Close the doors
900	Switch off the door closing timer
901	Start the door closing timer

The output actions of automata are declared in the *ElevatorVisualizerInterface* interface (Section 5) and implemented in the *ElevatorVisualizer* class (Section 7).

3.5. "Elevator control" automaton (A0)

3.5.1. Description

It is the main elevator control automaton. It is responsible for opening and closing of doors, motion and stop of the elevator car, switching on and off of the lamp in the elevator car. It is the only automaton, which contains called automata, i.e. the calls of other automata with certain events occur from the states of this automaton. For example, the automata of A2, A11, A21, A22, A32 with the events of 2, 3, 3, 3, 3 will be sequentially ***called*** in state 3, which is designated as A: 2(2), 11(3), 21(3), 22(3), 32(3).

A diagram of links and a transition graph for this automaton are given in Fig. 2 and Fig. 3 respectively.

The automaton on the event of e0 is started, however this event does not take part in the transition conditions. Therefore, it is present in the link diagram and is not present in the transition graph.

In the general case, automata are implemented as follows.

In the first *switch* operator, transfers and their actions are implemented.

If the state is changed, then the actions in states will be implemented in the second *switch* operator (or in its analogue) as well as automata with the corresponding events will be called.

The program listings are given in Section 9.

3.5.2. Link diagram

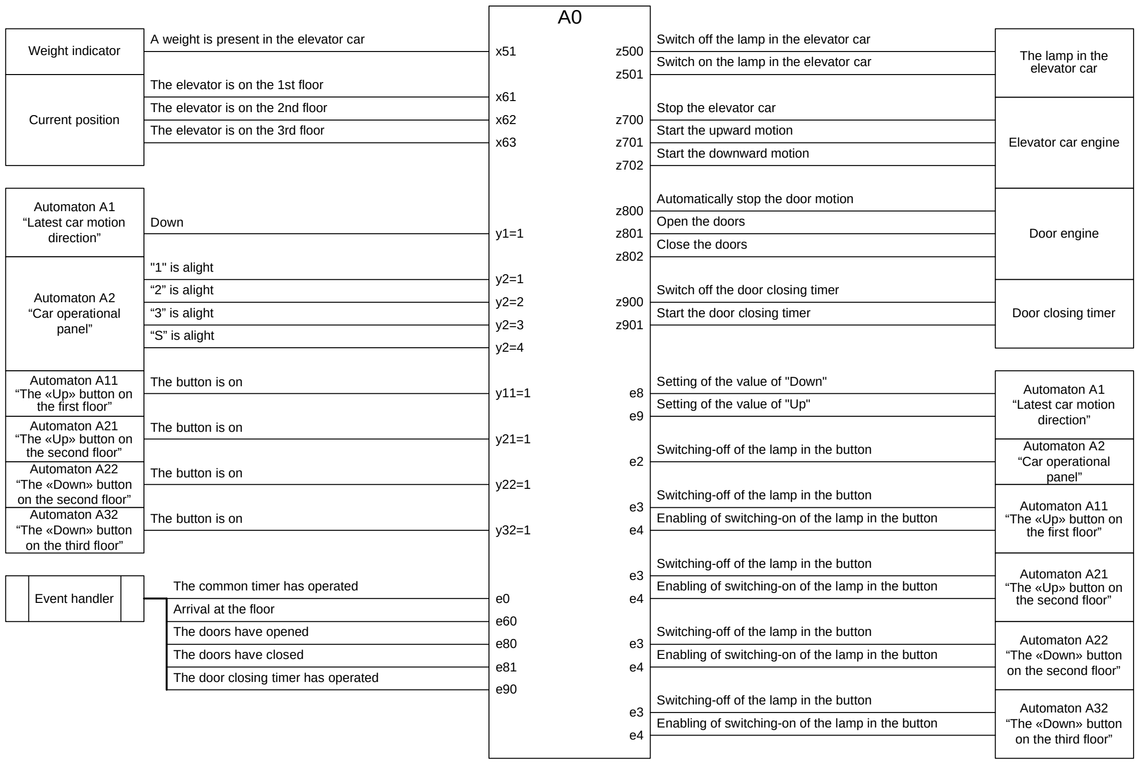
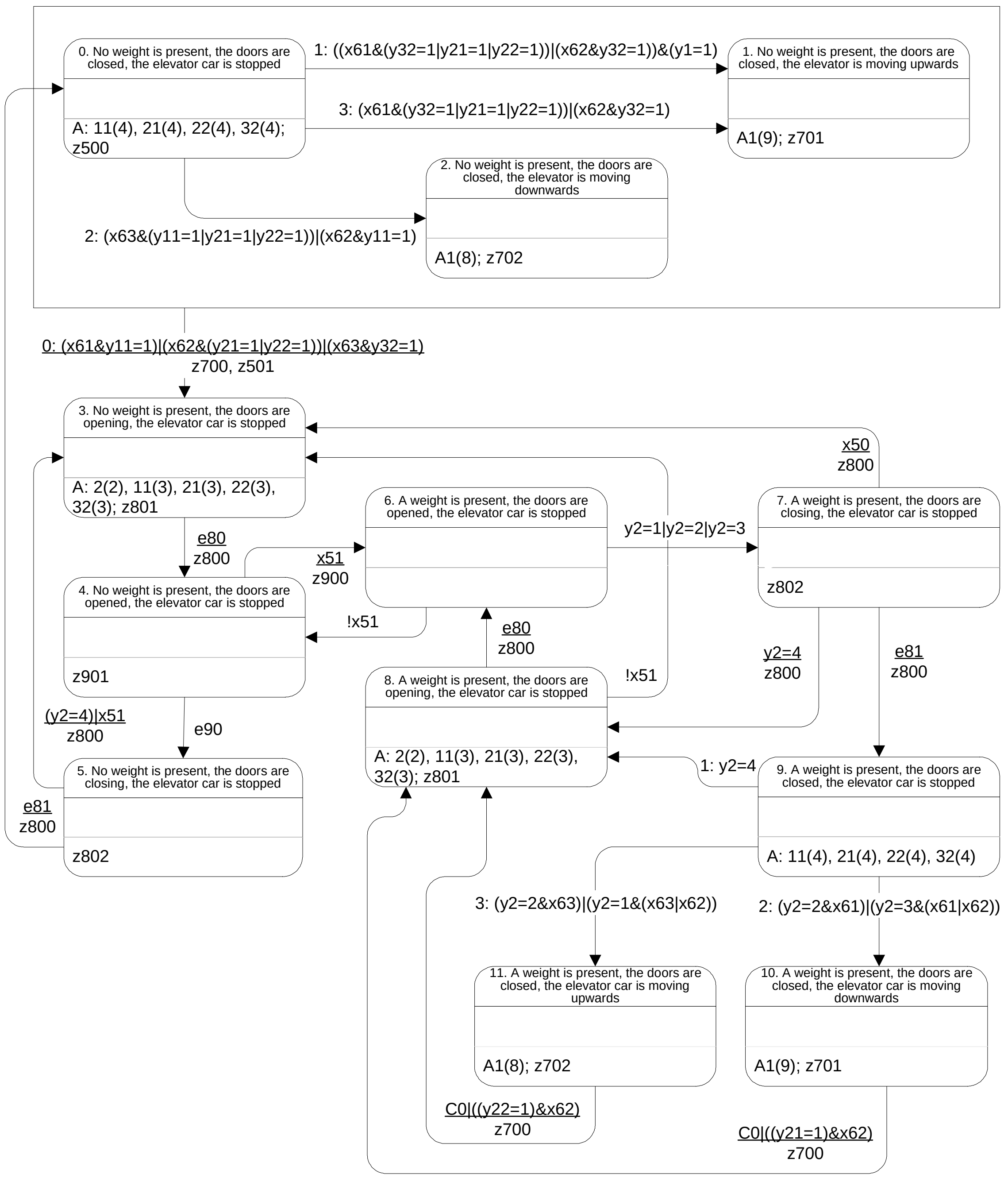

A0
Weight indicator
A weight is present in the elevator car
x51
Current position
The elevator is on the 1st floor
The elevator is on the 2nd floor
The elevator is on the 3rd floor
x61
x62
x63
Automaton A1 “Latest car motion direction”
Down
y1=1
Automaton A2 “Car operational panel”
"1" is alight
“2” is alight
“3” is alight
“S” is alight
y2=1
y2=2
y2=3
y2=4
Automaton A11 “The «Up» button on the first floor”
The button is on
y11=1
Automaton A21 “The «Up» button on the second floor”
The button is on
y21=1
Automaton A22 “The «Down» button on the second floor”
The button is on
y22=1
Automaton A32 “The «Down» button on the third floor”
The button is on
y32=1
Event handler
The common timer has operated
Arrival at the floor
The doors have opened
The doors have closed
The door closing timer has operated
e0
e60
e80
e81
e90
z500
z501
Switch off the lamp in the elevator car
Switch on the lamp in the elevator car
The lamp in the elevator car
z700
z701
z702
Stop the elevator car
Start the upward motion
Start the downward motion
Elevator car engine
z800
z801
z802
Automatically stop the door motion
Open the doors
Close the doors
Door engine
z900
z901
Switch off the door closing timer
Start the door closing timer
Door closing timer
e8
e9
Setting of the value of "Down"
Setting of the value of "Up"
Automaton A1 “Latest car motion direction”
e2
Switching-off of the lamp in the button
Automaton A2 “Car operational panel”
e3
e4
Switching-off of the lamp in the button
Enabling of switching-on of the lamp in the button
Automaton A11 “The «Up» button on the first floor”
e3
e4
Switching-off of the lamp in the button
Enabling of switching-on of the lamp in the button
Automaton A21 “The «Up» button on the second floor”
e3
e4
Switching-off of the lamp in the button
Enabling of switching-on of the lamp in the button
Automaton A22 “The «Down» button on the second floor”
e3
e4
Switching-off of the lamp in the button
Enabling of switching-on of the lamp in the button
Automaton A32 “The «Down» button on the third floor”

Fig. 2. A link diagram for the "Elevator control" automaton

3.5.3. Transition graph


0. No weight is present, the doors are closed, the elevator car is stopped
A: 11(4), 21(4), 22(4), 32(4); z500
1: ((x61&(y32=1|y21=1|y22=1))|(x62&y32=1))&(y1=1)
1. No weight is present, the doors are closed, the elevator is moving upwards
3: (x61&(y32=1|y21=1|y22=1))|(x62&y32=1)
A1(9); z701
2. No weight is present, the doors are closed, the elevator is moving downwards
2: (x63&(y11=1|y21=1|y22=1))|(x62&y11=1)
A1(8); z702
0: (x61&y11=1)|(x62&(y21=1|y22=1))|(x63&y32=1)
z700, z501
3. No weight is present, the doors are opening, the elevator car is stopped
A: 2(2), 11(3), 21(3), 22(3), 32(3); z801
x50
z800
6. A weight is present, the doors are opened, the elevator car is stopped
y2=1|y2=2|y2=3
7. A weight is present, the doors are closing, the elevator car is stopped
z802
e80
z800
x51
z900
4. No weight is present, the doors are opened, the elevator car is stopped
!x51
z901
e80
z800
!x51
y2=4
z800
e81
z800
8. A weight is present, the doors are opening, the elevator car is stopped
A: 2(2), 11(3), 21(3), 22(3), 32(3); z801
(y2=4)|x51
z800
e90
5. No weight is present, the doors are closing, the elevator car is stopped
e81
z800
z802
1: y2=4
9. A weight is present, the doors are closed, the elevator car is stopped
A: 11(4), 21(4), 22(4), 32(4)
3: (y2=2&x63)|(y2=1&(x63|x62))
2: (y2=2&x61)|(y2=3&(x61|x62))
11. A weight is present, the doors are closed, the elevator car is moving upwards
A1(8); z702
10. A weight is present, the doors are closed, the elevator car is moving downwards
A1(9); z701
C0|((y22=1)&x62)
z700
C0|((y21=1)&x62)
z700

C0 := ((y2=4)&(x61|x62|x63)))|((y2=1)&x61)|((y2=2)&x62)|((y2=3)&x63)

Fig. 3. A transition graph for the "Elevator control" automaton (A0)

3.6. "Latest car motion direction" automaton (A1)

3.6.1. Description

This automaton is used for determining the necessity of stopping on an intermediate floor. The automaton state corresponds to the direction of the latest elevator car motion ("Up" or "Down").

The creation of such a simple automaton is connected with the fact that the selection of the floor, on which a stop should be made, can generally be much more difficult.

A link diagram and a transition graph for this automaton are given in Fig. 4 and Fig. 5 respectively.

3.6.2. Link diagram

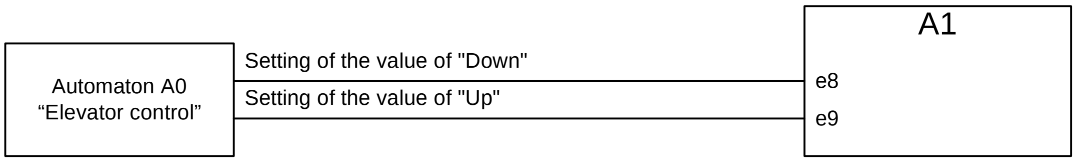


Fig. 4. A link diagram for the "Latest car motion direction" automaton

3.6.3. Transition graph

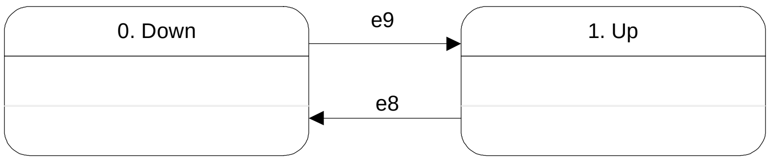


Fig. 5. A transition graph for the "Latest car motion direction" automaton (A1)

3.7. "Car operational panel" automaton (A2)

3.7.1. Description

This automaton is responsible for the processing of button, being pressed on the car operational panel, for switching button lamps on and off.

A link diagram and a transition graph for this automaton are given in Fig. 6 and Fig. 7 respectively.

3.7.2. Link diagram

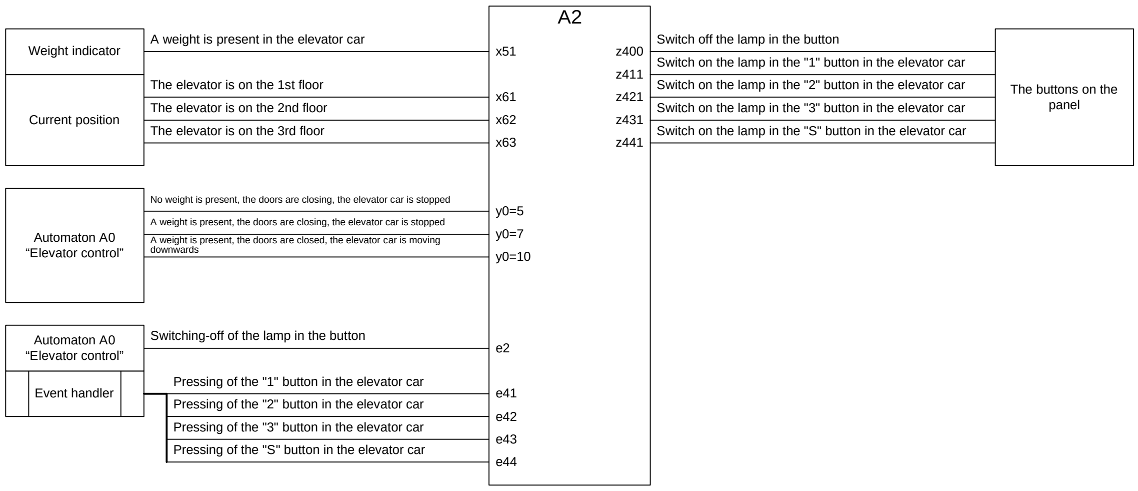

A2
Weight indicator
A weight is present in the elevator car
x51
Current position
The elevator is on the 1st floor
The elevator is on the 2nd floor
The elevator is on the 3rd floor
x61
x62
x63
Automaton A0 "Elevator control"
No weight is present, the doors are closing, the elevator car is stopped
A weight is present, the doors are closing, the elevator car is stopped
A weight is present, the doors are closed, the elevator car is moving downwards
y0=5
y0=7
y0=10
Automaton A0 "Elevator control"
Switching-off of the lamp in the button
e2
Event handler
Pressing of the "1" button in the elevator car
Pressing of the "2" button in the elevator car
Pressing of the "3" button in the elevator car
Pressing of the "S" button in the elevator car
e41
e42
e43
e44
z400
z411
z421
z431
z441
Switch off the lamp in the button
Switch on the lamp in the "1" button in the elevator car
Switch on the lamp in the "2" button in the elevator car
Switch on the lamp in the "3" button in the elevator car
Switch on the lamp in the "S" button in the elevator car
The buttons on the panel

Fig. 6. A link diagram for the "Car operational panel" automaton

3.7.3. Transition graph

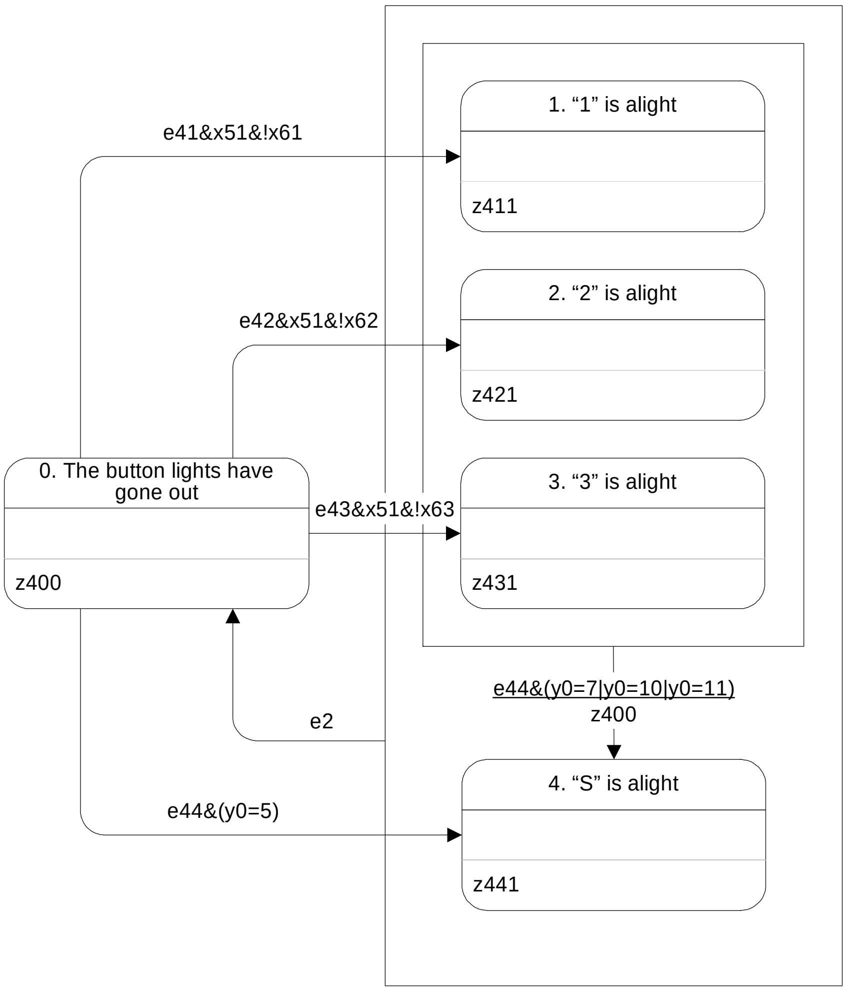


Fig. 7. A transition graph for the "Car operational panel" automaton (A2)

3.8. Call button automata (A11, A21, A22, A32)

3.8.1. Description

The automata of A11 (the "Up" button on the first floor), A21 (the "Up" button on the second floor), A22 (the "Down" button on the second floor), A32 (the "Down" button on the third floor) have a similar structure. In spite of the fact that these automata can be implemented by a single class or generated by a single basic class, it has not been done because of the automata simplicity.

A link diagram and a transition graph for these automata are given in Fig. 8…Fig. 15.

3.8.2. Link diagrams and transition graphs

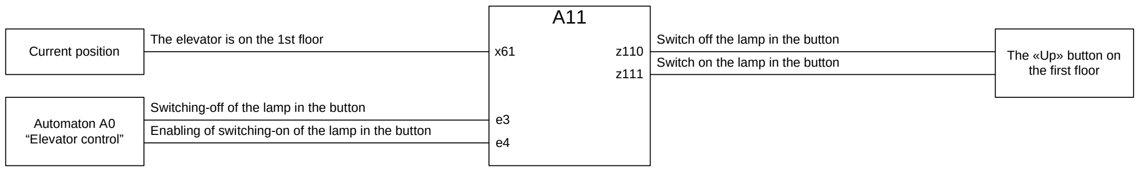

A11
Current position
The elevator is on the 1st floor
x61
z110
z111
Switch off the lamp in the button
Switch on the lamp in the button
The «Up» button on the first floor
Automaton A0 “Elevator control”
Switching-off of the lamp in the button
Enabling of switching-on of the lamp in the button
e3
e4

Fig. 8. A link diagram for the "Up" button of the first floor" automaton

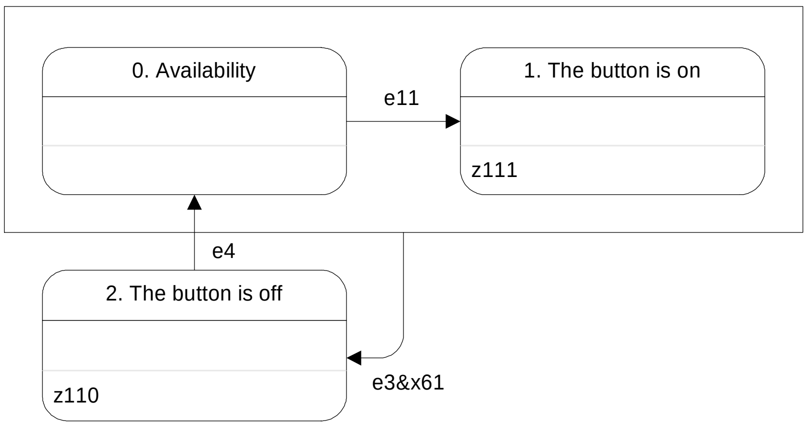


Fig. 9. A transition graph for the "Up" button of the first floor" automaton (A11)

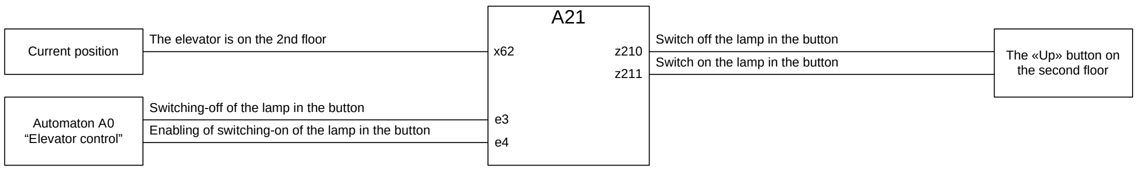

Current position
The elevator is on the 2nd floor
Automaton A0 "Elevator control"
Switching-off of the lamp in the button
Enabling of switching-on of the lamp in the button
A21
x62
e3
e4
z210
z211
Switch off the lamp in the button
Switch on the lamp in the button
The «Up» button on the second floor

Fig. 10. A link diagram for the "Up" button of the second floor" automaton

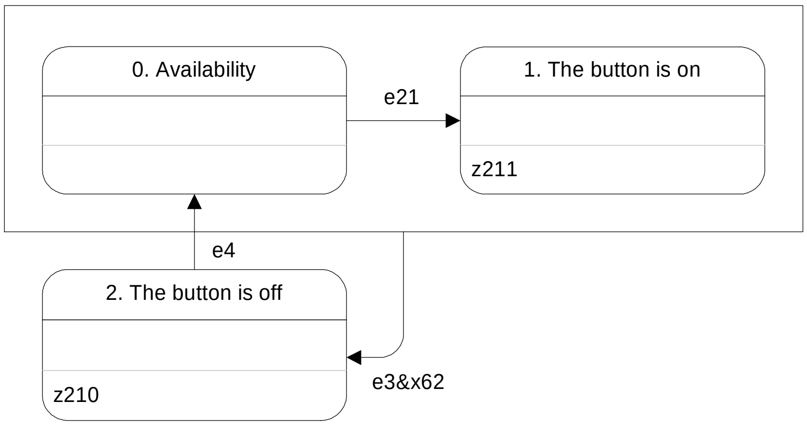


Fig. 11. A transition graph for the "Up" button of the second floor" automaton (A21)

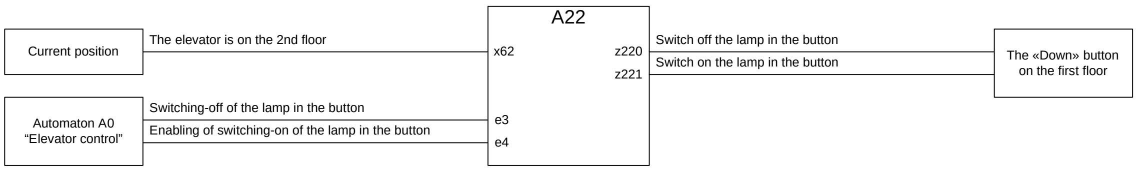

Current position
The elevator is on the 2nd floor
Automaton A0 "Elevator control"
Switching-off of the lamp in the button
Enabling of switching-on of the lamp in the button
A22
x62
e3
e4
z220
z221
Switch off the lamp in the button
Switch on the lamp in the button
The «Down» button on the first floor

Fig. 12. A link diagram for the "Down" button of the second floor" automaton

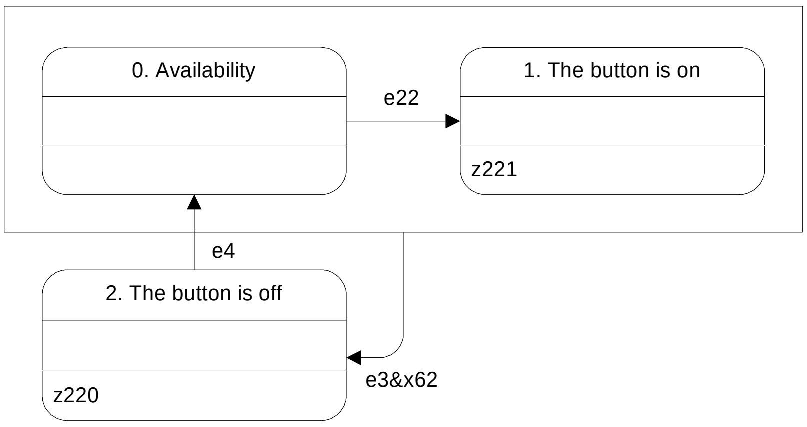


Fig. 13. A transition graph for the "Down" button of the second floor" automaton (A22)

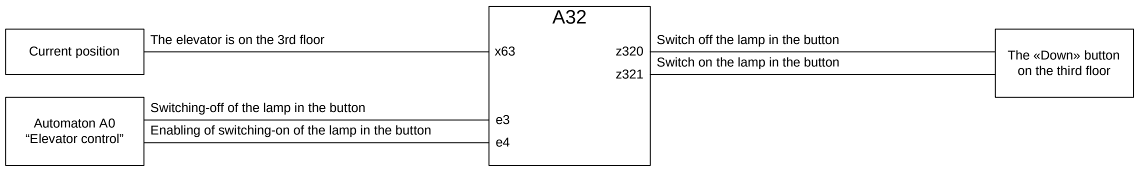

A32
Current position
The elevator is on the 3rd floor
x63
Automaton A0
“Elevator control”
Switching-off of the lamp in the button
Enabling of switching-on of the lamp in the button
e3
e4
z320
z321
Switch off the lamp in the button
Switch on the lamp in the button
The «Down» button on the third floor

Fig. 14. A link diagram for the "Down" button of the third floor" automaton

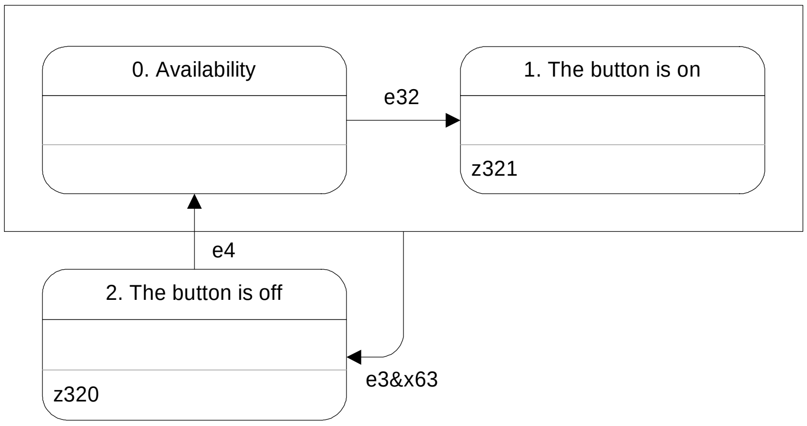


Fig. 15. A transition graph for the "Down" button of the third floor" automaton (A32)

4. "ElevatorLogInterface" interface

4.1. Description

This interface declares logging methods. The interface methods are used by the *ElevatorAutomats* class.

4.2. Method prototypes and their brief description

Let us give some method prototypes and their descriptions:

- `public void begin (int aut, int state, int event)` – adds a record into the log about a start of the `aut` automaton in the state of `state` with the event of `event`. It is called from the *ElevatorAutomats* class;
- `public void trans (int aut, int old_state, int new_state)` – adds a record into the log about a transfer of the `aut` automaton from the state of `old_state` into the state of `new_state`. It is called from the *ElevatorAutomats* class;
- `public void end (int aut, int state, int event)` – adds a record into the log about the `aut` automaton’s completion of processing of the event of `event` in the state of `state`. It is called from the *ElevatorAutomats* class;
- `public void input (int num, String str, boolean ret)` – adds a record into the log about a poll of the input variable of `num` co with the narrative of `str`. The input variable value is `ret`. This function shall be called from methods of the class implementing the *ElevatorVisualizerInterface* interface;

- `public void action (int num, String str)` – adds a record into the log about execution of the output action of `num` with the narrative of `str`. This function shall be called from methods of the class implementing the *ElevatorVisualizerInterface* interface;
- `public void error (int aut, String err)` – adds a record into the log about an error of `err` occurred during operation of the automaton of `aut`. It is called from the *ElevatorAutomats* class;
- `public void log (String str)` – adds a line of `str` into the log.

  The use of the log allows to "instantly" find errors in the program logic.

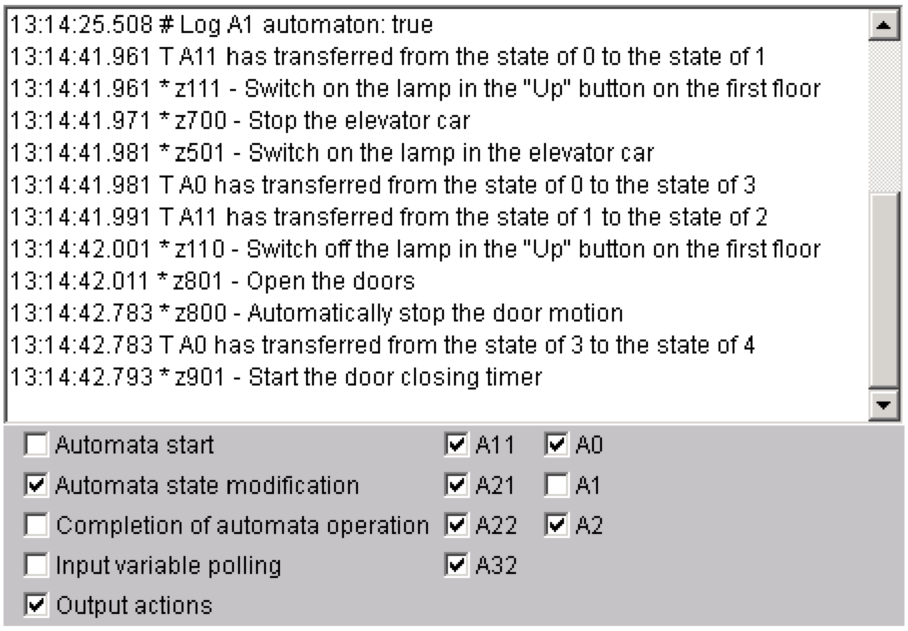
5. "ElevatorVisualizerInterace" interface

This interface declares methods, which are used to examine the input variables of the form `public boolean xnn()` where `nn` is the input variable number and for generating the output actions of the form `public void znn()` where `nn` is the output action number. These methods are used by the *ElevatorAutomats* class.

If it is required to modify the program, the class implementing this interface shall call a method of `ElevatorLogInterface.input()`in the methods of polling input variables and a method of `ElevatorLogInterface.action()`in the methods of generating output actions.

6. "ElevatorLog" class

This class implements the *ElevatorLogInterface* interface and mantains logging of the application operation. The log (Fig. 16) is generated by records, which are automatically added into the list (`java.awt.List` class). The log details are regulated by a set of flags shown in Fig. 16 (`java.awt.Checkbox` classes). Due to the fact that the A0 automaton is started by the common timer, it is necessary to switch off the start and completion of the automata operation as well as the input variable polling in order to prevent permanent logging.



```
13:14:25.508 # Log A1 automaton: true
13:14:41.961 T A11 has transferred from the state of 0 to the state of 1
13:14:41.961 * z111 - Switch on the lamp in the "Up" button on the first floor
13:14:41.971 * z700 - Stop the elevator car
13:14:41.981 * z501 - Switch on the lamp in the elevator car
13:14:41.981 T A0 has transferred from the state of 0 to the state of 3
13:14:41.991 T A11 has transferred from the state of 1 to the state of 2
13:14:42.001 * z110 - Switch off the lamp in the "Up" button on the first floor
13:14:42.011 * z801 - Open the doors
13:14:42.783 * z800 - Automatically stop the door motion
13:14:42.783 T A0 has transferred from the state of 3 to the state of 4
13:14:42.793 * z901 - Start the door closing timer
```

☐ Automata start ✓ A11 ✓ A0
✓ Automata state modification ✓ A21 ☐ A1
☐ Completion of automata operation ✓ A22 ✓ A2
☐ Input variable polling ✓ A32
✓ Output actions

Fig. 16. Logging

The log fragments are given in Section 10.

7. "ElevatorVisualizer" class

7.1. Description

This class implements the *ElevatorVisualizerInterface* interface, and it is the main class for elevator emulation and its visualisation.

The class also implements the library interface of `java.awt.event.MouseListener`. The method of `public void mousePressed(MouseEvent e)` processes of the mouse button clicks at the cursor position on the visualisation area and adds events into the queue implemented by the `java.util.Vector` class.

In addition, the class implements the library interface of `java.lang.Runnable`. The method of `public void run()` contains an "infinite" cycle, which, in case of the availability of an event in the queue, will start the corresponding automaton with such event or change the internal variables of the class copy, for example a Boolean variable determining the presence of a weight in the elevator car. Then one "tick" of the elevator operation is emulated and, if necessary, the automata with the corresponding events will be called. After that, the A0 automaton will be called with the event of e0 (common timer operation) and the visualisation area will be re-drawn.

It implements the functions of input variable and output actions.

7.2. "Consequences" of button presses

A button press is processed in the `public void mousePressed(MouseEvent e)` method and adds the following events to the queue:

- 11 – "Up" button on the first floor;
- 21 – "Up" button on the second floor;
- 22 – "Down" button on the second floor;
- 32 – "Down" button on the third floor;
- 41 – "1" button in the elevator car;
- 42 – "2" button in the elevator car;
- 43 – "3" button in the elevator car;
- 44 – "S" button in the elevator car;
- 50 – go out from the car (right arrow);
- 51 – come into the car (left arrow).

Fig. 17 shows an elevator operation visualiser. The buttons located near the elevator allow to call the elevator car from the corresponding floor. The visualiser lamp corresponds to the lamp in the elevator car. It does not shine if the doors are closed and the car is empty. In addition, there is an operational panel in the elevator car. The timer shows how much time is left to the door closing and the indicator shown the direction of possible person's motion direction (from the car or to the car).

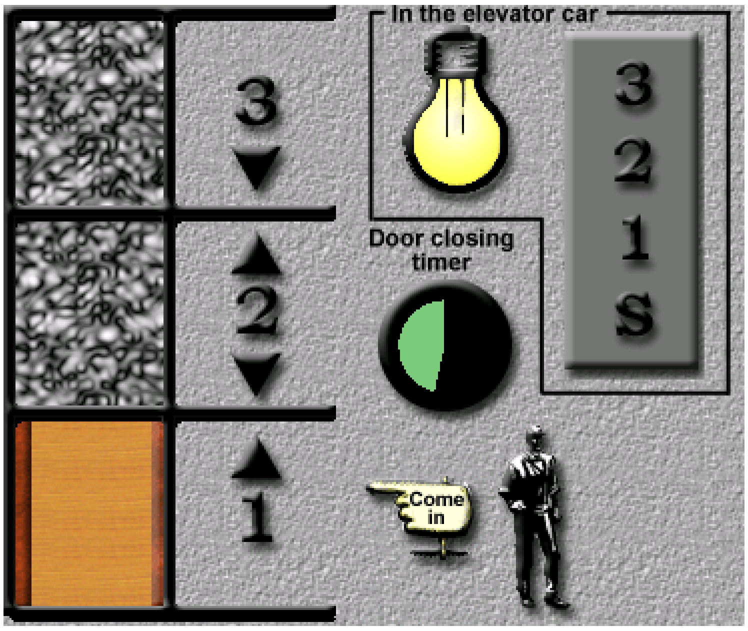


Fig. 17. Elevator operation visualisation

8. "ElevatorApplet" class

8.1. Description

This class extends the library class `java.applet.Applet` and overrides the methods `public void init()` and `public void destroy()`.

The first method creates copies of the classes of *ElevatorLog* (record-keeping) – a *log* variable and *ElevatorVisualizer* (elevator emulation and visualisation) – a *vis* variable, which are inherited from the library class of `java.awt.Canvas`. They are located in the applet area.

An additional thread, in which a copy of the *ElevatorVisualizer* class is started, is created in the main applet execution thread.

In the second method, the additional thread gets stopped.

A method of `public int getParamInt(String param, int def)` is determined, which tries to obtain the applet parameter of `param`, round it to a integer number and return the result. In case of a failure, the default value of `def` is returned.

8.2. Possible applet parameters

The *ElevatorLog* class uses the following parameter:

- LOG_HEIGHT – the logging area height in pixels. The default value is 410.

The *ElevatorVisualizer* class uses the following parameters:

- DELAY – a delay between the system emulation "ticks" in milliseconds. The default value is 50.

- DOORS_TIMER_MAX – the number of waiting "ticks" before closing of the doors. The default value is 30.
- DOORS_MOVE – the elevator door movement at opening/closing of the doors for one "tick" in pixels. The default value is 2.
- DOORS_MAX – the maximal movement of the elevator doors at opening. It shall be multiple to the DOORS_MOVE value and be within the range of 20-40. The default value is 30.
- ELEVATOR_MOVE – the movement of the elevator car during the up/down motion for one "tick" in pixels. It shall divide 100 without a remainder. The default value is 2.

9. Program listings

9.1. ElevatorAutomats.java

```
/**
 * Encapsulates elevator control automata
 */
public class ElevatorAutomats
{
  protected ElevatorVisualizerInterface vis;
  protected ElevatorLogInterface log;

  ElevatorAutomats( ElevatorVisualizerInterface vis,
               ElevatorLogInterface log )
  {
    this.vis = vis;
    this.log = log;
  }

  protected int y0 = 0;
  /**
  * Automaton A0
  */
  public void A0( int e )
  {
    int y_old = y0;
    boolean xx61;
    boolean xx62;
    boolean xx63;
    log.begin( 0, y0, e );
    switch ( y0 )
    {
    case 0:
        xx61 = vis.x61();
        xx62 = vis.x62();
        xx63 = vis.x63();
        if ( ( xx61 && y11 == 1 ) ||
             ( xx62 && ( y21 == 1 || y22 == 1 ) ) ||
             ( xx63 && y32 == 1 ) )
        {                   vis.z700(); vis.z501();           y0 = 3; }
        else if ( ( ( xx61 &&
                    ( y32 == 1 || y21 == 1 || y22 == 1 ) )
                        || ( xx62 && y32 == 1 ) ) && y1 == 1 )
        {                                                     y0 = 1; }
        else if ( ( xx63 &&
                    ( y11 == 1 || y21 == 1 || y22 == 1 ) )
                        || ( xx62 && y11 == 1 ) )
        {                                                     y0 = 2; }
        else if ( ( xx61 &&
                    ( y32 == 1 || y21 == 1 || y22 == 1 ) )
                        || ( xx62 && y32 == 1 ) )
        {                                                     y0 = 1; }
        break;

    case 1:
    case 2:
        if ( ( vis.x61() && y11 == 1 ) ||
             ( vis.x62() && ( y21 == 1 || y22 == 1 ) )
                 || ( vis.x63() && y32 == 1 ) )
        {                   vis.z700(); vis.z501();           y0 = 3; }
        break;
```

```
case 3:
    if ( e == 80 )
    {                   vis.z800();                      y0 = 4; }
    break;

case 4:
    if ( e == 90 )
    {                                                    y0 = 5; }
    else if ( vis.x51() )
    {                   vis.z900();                      y0 = 6; }
    break;

case 5:
    if ( e == 81 )
    {                   vis.z800();                      y0 = 0; }
    else if ( y2 == 4 || vis.x51() )
    {                   vis.z800();                      y0 = 3; }
    break;

case 6:
    if ( y2 == 1 || y2 == 2 || y2 == 3 )
    {                                                    y0 = 7; }
    else if ( !vis.x51() )
    {                                                    y0 = 4; }
    break;

case 7:
    if ( e == 81 )
    {                   vis.z800();                      y0 = 9; }
    else if ( y2 == 4 )
    {                   vis.z800();                      y0 = 8; }
    else if ( !vis.x51() )
    {                   vis.z800();                      y0 = 3; }
    break;

case 8:
    if ( !vis.x51() )
    {                                                    y0 = 3; }
    else if ( e == 80 )
    {                   vis.z800();                      y0 = 6; }
    break;

case 9:
    xx61 = vis.x61();
    xx62 = vis.x62();
    xx63 = vis.x63();
    if ( y2 ==4 )
    {                                                    y0 = 8; }
    else if ( ( y2 == 2 && xx61 ) ||
              ( y2 == 3 && ( xx61 || xx62 ) ) )
    {                                                    y0 = 10; }
    else if ( ( y2 == 2 && xx63 ) ||
              ( y2 == 1 && ( xx63 || xx62 ) ) )
    {                                                    y0 = 11; }
    break;

case 10:
    xx61 = vis.x61();
    xx62 = vis.x62();
    xx63 = vis.x63();
    if ( ( y2 == 4 && ( xx61 || xx62 || xx63 ) ) ||
         ( y2 == 1 && xx61 ) || ( y2 == 2 && xx62 )
      || ( y2 == 3 && xx63 ) || ( y21 == 1 && xx62 ) )
```

```
        {                   vis.z700();                       y0 = 8; }
        break;

    case 11:
        xx61 = vis.x61();
        xx62 = vis.x62();
        xx63 = vis.x63();
        if ( ( y2 == 4 && ( xx61 || xx62 || xx63 ) ) ||
             ( y2 == 1 && xx61 ) || ( y2 == 2 && xx62 ) ||
             ( y2 == 3 && xx63 ) || ( y22 == 1 && xx62 ) )
        {                   vis.z700();                       y0 = 8; }
        break;

    default:
        log.error( 0, "unknown state" );
    }

    if ( y_old != y0 )
    {
        log.trans( 0, y_old, y0 );
        switch ( y0 )
        {
        case 0:
            A11( 4 ); A21( 4 ); A22( 4 ); A32( 4 );
            vis.z500();
            break;

        case 1:
            A1( 9 );
            vis.z701();
            break;

        case 2:
            A1( 8 );
            vis.z702();
            break;

        case 3:
            A2( 2 ); A11( 3 ); A21( 3 ); A22( 3 ); A32( 3 );
            vis.z801();
            break;

        case 4:
            vis.z901();
            break;

        case 5:
            vis.z802();
            break;

        case 7:
            vis.z802();
            break;

        case 8:
            A2( 2 ); A11( 3 ); A21( 3 ); A22( 3 ); A32( 3 );
            vis.z801();
            break;

        case 9:
            A11( 4 ); A21( 4 ); A22( 4 ); A32( 4 );
            break;
```

```
      case 10:
          A1( 9 );
          vis.z701();
          break;

      case 11:
          A1( 8 );
          vis.z702();
          break;
      }
  }
  log.end( 0, y0, e );
}

protected int y1 = 0;
/**
* Automaton A1
*/
public void A1( int e )
{
  int y_old = y1;
  log.begin( 1, y1, e );
  if ( y1 == 0 && e == 9 )                                y1 = 1;
  else if ( y1 == 1 && e == 8 )                           y1 = 0;
  if ( y_old != y1 )
      log.trans( 1, y_old, y1 );
  log.end( 1, y1, e );
}

protected int y2 = 0;
/**
* Automaton A2
*/
public void A2( int e )
{
  int y_old = y2;
  log.begin( 2, y1, e );
  switch ( y2 )
  {
  case 0:
      boolean xx51 = vis.x51();
      if ( e == 41 && xx51 && !vis.x61() )
      {                                                   y2 = 1; }
      else if ( e == 42 && xx51 && !vis.x62() )
      {                                                   y2 = 2; }
      else if ( e == 43 && xx51 && !vis.x63() )
      {                                                   y2 = 3; }
      else if ( e == 44 && y0 == 5 )
      {                                                   y2 = 4; }
      break;

  case 1:
  case 2:
  case 3:
      if ( e == 44 && ( y0 == 7 || y0 == 10 || y0 == 11 ) )
      {                  vis.z400();                      y2 = 4; }
      else if ( e == 2)
      {                                                   y2 = 0; }
      break;

  case 4:
      if ( e == 2 )
      {                                                   y2 = 0; }
      break;
  default:
      log.error( 0, "unknown state" );
  }

  if ( y_old != y2 )
  {
      log.trans( 2, y_old, y2 );
      switch ( y2 )
      {
      case 0:
          vis.z400();
          break;

      case 1:
          vis.z411();
          break;

      case 2:
          vis.z421();
          break;

      case 3:
          vis.z431();
          break;

      case 4:
          vis.z441();
          break;
      }
  }
  log.end( 2, y2, e );
}

protected int y11 = 0;
/**
* Automaton A11
*/
public void A11( int e )
{
  int y_old = y11;
  log.begin( 11, y11, e );
  switch ( y11 )
  {
  case 0:
      if ( e == 11 )
      {                                                    y11 = 1; }
      else if ( e == 3 && vis.x61() )
      {                                                    y11 = 2; }
      break;

  case 1:
      if ( e == 3 && vis.x61() )
      {                                                    y11 = 2; }
      break;

  case 2:
      if ( e == 4 )
      {                                                    y11 = 0; }
      break;
  }
  if ( y_old != y11 )
  {
      log.trans( 11, y_old, y11 );
      if ( y11 == 1 )        vis.z111();
      else if ( y11 == 2 )   vis.z110();
  }
  log.end( 11, y11, e );
}

protected int y21 = 0;
/**
* Automaton A21
*/
public void A21( int e )
{
  int y_old = y21;
  log.begin( 21, y21, e );
  switch ( y21 )
  {
  case 0:
      if ( e == 21 )
      {                                                    y21 = 1; }
      else if ( e == 3 && vis.x62() )
      {                                                    y21 = 2; }
      break;

  case 1:
      if ( e == 3 && vis.x62() )
      {                                                    y21 = 2; }
      break;

  case 2:
      if ( e == 4 )
      {                                                    y21 = 0; }
      break;
  }

  if ( y_old != y21 )
  {
      log.trans( 21, y_old, y21 );
      if ( y21 == 1 )        vis.z211();
      else if ( y21 == 2 )   vis.z210();
  }
  log.end( 21, y21, e );
}

protected int y22 = 0;
/**
* Automaton A22
*/
public void A22( int e )
{
  int y_old = y22;
  log.begin( 22, y22, e );
  switch ( y22 )
  {
  case 0:
      if ( e == 22 )
      {                                                    y22 = 1; }
      else if ( e == 3 && vis.x62() )
      {                                                    y22 = 2; }
      break;
```

```
    case 1:
        if ( e == 3 && vis.x62() )
        {                                                     y22 = 2; }
        break;

    case 2:
        if ( e == 4 )
        {                                                     y22 = 0; }
        break;
    }

    if ( y_old != y22 )
    {
        log.trans( 22, y_old, y22 );
        if ( y22 == 1 )        vis.z221();
        else if ( y22 == 2 )   vis.z220();
    }
    log.end( 22, y22, e );
  }

  protected int y32 = 0;
  /**
  * Automaton A32
  */
  public void A32( int e )
  {
    int y_old = y32;
    log.begin( 32, y32, e );
    switch ( y32 )
    {
    case 0:
        if ( e == 32 )
        {                                                     y32 = 1; }
        else if ( e == 3 && vis.x63() )
        {                                                     y32 = 2; }
        break;

    case 1:
        if ( e == 3 && vis.x63() )
        {                                                     y32 = 2; }
        break;

    case 2:
        if ( e == 4 )
        {                                                     y32 = 0; }
        break;

    }
    if ( y_old != y32 )
    {
        log.trans( 32, y_old, y32 );
        if ( y32 == 1 )        vis.z321();
        else if ( y32 == 2 )   vis.z320();
    }
    log.end( 32, y32, e );
  }
}
```

9.2. ElevatorLogInterface.java

```
/**
 * Interface of the log-maintaining class
 */
public interface ElevatorLogInterface
{
    /**
    * Beginning of automaton operation
    */
    public void begin( int aut, int state, int event );
    /**
    * Transfer between the states
    */
    public void trans( int aut, int old_state, int new_state );
    /**
    * Ending of automaton operation
    */
    public void end( int aut, int state, int event );
    /**
    * Input variable polling
    */
    public void input( int num, String str, boolean ret );
    /**
    * Execution of an action
    */
    public void action( int num, String str );
    /**
    * Error message
    */
    public void error( int aut, String err );
    /**
    * Output of the line into the log
    */
    public void log( String str );
}
```

9.3. ElevatorVisualizerInterface.java

```
/**

 * Interface of the elevator operation emulator
 * and visualiser. It determines all input
 * variable and output actions being used in the problem
 */
public interface ElevatorVisualizerInterface
{
    /**
     * Input variable: a weight is present in the elevator car
     */
    public boolean x51();

    /**
     * Input variable: the elevator is on the 1st floor
     */
    public boolean x61();

    /**
     * Input variable: the elevator is on the 2nd floor
     */
    public boolean x62();
```

```
/**
 * Input variable: the elevator is on the 3rd floor
 */
public boolean x63();

/**
 * Output action:
 * switch off the lamp in the «Up» button on the first floor
 */
public void z110();

/**
 * Output action:
 * switch on the lamp in the «Up» button on the first floor
 */
public void z111();

/**
 * Output action:
 * switch off the lamp in the «Up» button on the second floor
 */
public void z210();

/**
 * Output action:
 * switch on the lamp in the «Up» button on the second floor
 */
public void z211();

/**
 * Output action:
 * switch off the lamp in the «Down» button on the second floor
 */
public void z220();

/**
 * Output action:
 * switch on the lamp in the «Down» button on the second floor
 */
public void z221();

/**
 * Output action:
 * switch off the lamp in the «Down» button on the third floor
 */
public void z320();

/**
 * Output action:
 * switch on the lamp in the «Down» button on the third floor
 */
public void z321();

/**
 * Output action: switch off the lamp in the button
 */
public void z400();

/**
 * Output action: switch on the lamp in the “1” button
 */
public void z411();
```

```
/**
 * Output action: switch on the lamp in the “2” button
 */
public void z421();

/**
 * Output action: switch on the lamp in the “3” button
 */
public void z431();

/**
 * Output action: switch on the lamp in the “S” button
 */
public void z441();

/**
 * Output action: switch off the lamp in the elevator car
 */
public void z500();

/**
 * Output action: switch on the lamp in the elevator car
 */
public void z501();

/**
 * Output action: stop the elevator car
 */
public void z700();

/**
 * Output action: start the upward motion
 */
public void z701();

/**
 * Output action: start the downward motion
 */
public void z702();

/**
 * Output action: automatically stop the door motion
 */
public void z800();

/**
 * Output action: open the doors
 */
public void z801();

/**
 * Output action: close the doors
 */
public void z802();

/**
 * Output action: switch off the door closing timer
 */
public void z900();
```

```
    /**
     * Output action: start the door closing timer
     */
    public void z901();
}
```

9.4. ElevatorLog.java

```
import java.awt.Checkbox;
import java.awt.List;
import java.awt.Panel;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.Vector;

/**
 * Realisation of the log-maintaining class
 */
public class ElevatorLog
    extends Panel
    implements ElevatorLogInterface, ItemListener
{
    protected boolean logBegin;
    protected boolean logTrans;
    protected boolean logEnd;
    protected boolean logInput;
    protected boolean logAction;
    protected Vector logAut;

    protected int HEIGHT;
    final static int HEIGHT_DEFAULT = 410;

    protected List logList = new List();
    protected Checkbox chLogBegin, chLogEnd, chLogTrans, chLogInput,
        chLogAction, chLogA0, chLogA1, chLogA2, chLogA11, chLogA21,
        chLogA22, chLogA32;
    protected ElevatorApplet mainApplet;

    ElevatorLog( ElevatorApplet appl )
    {
        mainApplet = appl;
        HEIGHT = mainApplet.getParamInt( "LOG_HEIGHT",
                                         HEIGHT_DEFAULT );
        setSize( 450, HEIGHT );
        setLayout( null );
        logList.setBounds( 0, 0, 450, HEIGHT-100 );
        add( logList );
        chLogBegin = new Checkbox( "Automata start" );
        chLogBegin.setBounds( 10, HEIGHT-100, 200, 20 );
        chLogBegin.addItemListener( this );
        add( chLogBegin );
        chLogTrans = new Checkbox( "Automata state modification" );
        chLogTrans.setBounds( 10, HEIGHT-80, 200, 20 );
        chLogTrans.addItemListener( this );
        add( chLogTrans );
        chLogEnd = new Checkbox( "Completion of automata operation" );
        chLogEnd.setBounds( 10, HEIGHT-60, 200, 20 );
        chLogEnd.addItemListener( this );
        add( chLogEnd );
        chLogInput = new Checkbox( "Input variable polling" );
        chLogInput.setBounds( 10, HEIGHT-40, 200, 20 );
        chLogInput.addItemListener( this );
        add( chLogInput );
        chLogAction = new Checkbox( "Output actions" );
        chLogAction.setBounds( 10, HEIGHT-20, 200, 20 );
        chLogAction.addItemListener( this );
        add( chLogAction );
        chLogA11 = new Checkbox( "A11" );
        chLogA11.setBounds( 220, HEIGHT-100, 40, 20 );
        chLogA11.addItemListener( this );
        add( chLogA11 );
        chLogA21 = new Checkbox( "A21" );
        chLogA21.setBounds( 220, HEIGHT-80, 40, 20 );
        chLogA21.addItemListener( this );
        add( chLogA21 );
        chLogA22 = new Checkbox( "A22" );
        chLogA22.setBounds( 220, HEIGHT-60, 40, 20 );
        chLogA22.addItemListener( this );
        add( chLogA22 );
        chLogA32 = new Checkbox( "A32" );
        chLogA32.setBounds( 220, HEIGHT-40, 40, 20 );
        chLogA32.addItemListener( this );
        add( chLogA32 );
        chLogA0 = new Checkbox( "A0" );
        chLogA0.setBounds( 270, HEIGHT-100, 40, 20 );
        chLogA0.addItemListener( this );
        add( chLogA0 );
        chLogA1 = new Checkbox( "A1" );
        chLogA1.setBounds( 270, HEIGHT-80, 40, 20 );
        chLogA1.addItemListener( this );
        add( chLogA1 );
        chLogA2 = new Checkbox( "A2" );
        chLogA2.setBounds( 270, HEIGHT-60, 40, 20 );
        chLogA2.addItemListener( this );
        add( chLogA2 );
        logAut = new Vector();
    }

    /**
     * Processing of the "Flag management" external event
     */
    public void itemStateChanged( ItemEvent e )
    {
        boolean enable = ((Checkbox)e.getSource()).getState();
        if ( e.getSource().equals( chLogBegin ) )
        {
            log( "Log automata start: " +
                 enable );
            logBegin = enable;
        }
        else if ( e.getSource().equals( chLogTrans ) )
        {
            log( "Log automata state modification: " +
                 enable );
            logTrans = enable;
        }
        else if ( e.getSource().equals( chLogEnd ) )
        {
            log( "Log automata operation completion: " +
                 enable );
            logEnd = enable;
        }
        else if ( e.getSource().equals( chLogInput ) )
        {
            log( "Log input variable polling: " +
                 enable );
            logInput = enable;
        }
        else if ( e.getSource().equals( chLogAction ) )
        {
            log( "Log output actions: " +
                 enable );
            logAction = enable;
        }
        else if ( e.getSource().equals( chLogA0 ) )
            logAutChange( 0, enable );
        else if ( e.getSource().equals( chLogA1 ) )
            logAutChange( 1, enable );
        else if ( e.getSource().equals( chLogA2 ) )
            logAutChange( 2, enable );
        else if ( e.getSource().equals( chLogA11 ) )
            logAutChange( 11, enable );
        else if ( e.getSource().equals( chLogA21 ) )
            logAutChange( 21, enable );
        else if ( e.getSource().equals( chLogA22 ) )
            logAutChange( 22, enable );
        else if ( e.getSource().equals( chLogA32 ) )
            logAutChange( 32, enable );
    }

    /**
     * Error message
     */
    public void error( int aut, String err )
    {
        write( "! A" + aut + ": ERROR: " + err );
    }

    /**
     * Beginning of automaton operation
     */
    public void begin( int aut, int state, int event )
    {
        if ( logBegin )
            aut( aut, "{ A" + aut + ": being in the state " + state +
                " has been started with the event of e" + event );
    }

    /**
     * Transfer between the states
     */
    public void trans( int aut, int old_state, int new_state )
    {
        if ( logTrans )
            aut( aut, "T A" + aut + ": has transferred from the ” +
            “state of " + old_state + " to the state of " + new_state
            );
    }
```

```
/**
 * Ending of automaton operation
 */
public void end( int aut, int state, int event )
{
    if ( logEnd )
        aut( aut, "} A" + aut + ": has completed processing of “ +
        “the event of e" + event + " in the state of " + state );
}

/**
 * Input variable polling
 */
public void input( int num, String str, boolean ret )
{
    if ( logInput )
        write( "> x" + num + " - " + str + " - return " + ret );
}

/**
 * Execution of an action
 */
public void action( int num, String str )
{
    if ( logAction )
        write( "* z" + num + " - "  + str );
}

/**
 * Output of the line into the log
 */
public void log( String str )
{
    write( "# " + str );
}

protected void write( String str )
{
    Calendar cal = new GregorianCalendar();
    String hr = "" + cal.get( Calendar.HOUR_OF_DAY );
    while ( hr.length() < 2 ) { hr = "0" + hr; }
    String mn = "" + cal.get( Calendar.MINUTE );
    while ( mn.length() < 2 ) { mn = "0" + mn; }
    String sc = "" + cal.get( Calendar.SECOND );
    while ( sc.length() < 2 ) { sc = "0" + sc; }
    String ms = "" + cal.get( Calendar.MILLISECOND );
    while ( ms.length() < 3 ) { ms = "0" + ms; }
    logList.add( hr + ":" + mn + ":" + sc + "." + ms + " " +
                str );
    logList.makeVisible( logList.getItemCount()-1 );
}

protected void aut( int aut, String str )
{
    if ( logAut.contains( new Integer( aut ) ) )
        write( str );
}

protected void logAutChange( int aut, boolean enable )
{
    log( "Log A" + aut + " automaton: " + enable );
    Integer a = new Integer( aut );
```

```
        if ( enable )
        {
            if ( !logAut.contains( a ) )
                logAut.addElement( a );
        }
        else
        {
            logAut.removeElement( a );
        }
    }

}
```

9.5. ElevatorVisualizer.java

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Canvas;
import java.awt.Graphics;
import java.awt.List;
import java.awt.MediaTracker;
import java.awt.Image;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.util.Vector;

/**
 * Realisation of the elevator emulator and visualiser
 */
public class ElevatorVisualizer
    extends Canvas
    implements ElevatorVisualizerInterface, MouseListener, Runnable
{
    final static int DELAY_DEFAULT = 50;
    protected int DELAY;
    final static int DOORS_TIMER_MAX_DEFAULT = 30;
    protected int DOORS_TIMER_MAX;
    final static int DOORS_MAX_DEFAULT = 30;
    protected int DOORS_MAX;
    final static int DOORS_MOVE_DEFAULT = 2;
    protected int DOORS_MOVE;
    final static int ELEVATOR_MOVE_DEFAULT = 2;
    protected int ELEVATOR_MOVE;
    final static int ELEVATOR_1_FLOOR = 0;
    final static int ELEVATOR_2_FLOOR = 100;
    final static int ELEVATOR_3_FLOOR = 200;
    protected Vector events = new Vector();
    protected int doorsPosition = 0;
    protected int elevatorPosition = 0;
    protected int doorsMove = 0;
    protected int elevatorMove = 0;
    protected boolean isLampOn;
    protected int doorsTimer = -1;
    protected int panelState = 0;
    protected boolean xx51;
    protected boolean zz11, zz21, zz22, zz32;
    protected ElevatorApplet mainApplet;
    protected Image imBackground,
        imElevatorWallPaper,
        imElevatorDoor,
        imMan,
        imSkeleton,
        imLampOff,
        imLampOn,
        imDoorsTimer,
        im3DownOn,
        im2UpOn,
        im2DownOn,
        im1UpOn,
        imPanel0,
        imPanel1,
        imPanel2,
        imPanel3,
        imPanelS,
        imGoLeft,
        imGoRight;

    protected ElevatorLogInterface log;
    protected ElevatorAutomats aut;

    ElevatorVisualizer( ElevatorApplet appl,
                        ElevatorLogInterface log )
    {
        mainApplet = appl;
        setSize( 370, 310 );
        addMouseListener( this );
        aut = new ElevatorAutomats( this, log );
        this.log = log;
        DELAY = mainApplet.getParamInt( "DELAY", DELAY_DEFAULT );
        DOORS_TIMER_MAX = mainApplet.getParamInt( "DOORS_TIMER_MAX",
                                           DOORS_TIMER_MAX_DEFAULT );
        DOORS_MAX = mainApplet.getParamInt( "DOORS_MAX",
                                           DOORS_MAX_DEFAULT );
        DOORS_MOVE = mainApplet.getParamInt( "DOORS_MOVE",
                                           DOORS_MOVE_DEFAULT );
        ELEVATOR_MOVE = mainApplet.getParamInt( "ELEVATOR_MOVE",
                                           ELEVATOR_MOVE_DEFAULT );

        // Download of all graphical fragments
        MediaTracker t = new MediaTracker( this );
        imBackground = mainApplet.getImage(
                                mainApplet.getDocumentBase(),
                                "Background.gif" );
        t.addImage( imBackground, 0 );
        imElevatorWallPaper = mainApplet.getImage(
                                mainApplet.getDocumentBase(),
                                "ElevatorWallPaper.gif" );
        t.addImage( imElevatorWallPaper, 0 );
        imElevatorDoor = mainApplet.getImage(
                                mainApplet.getDocumentBase(),
                                "ElevatorDoor.gif" );
        t.addImage( imElevatorDoor, 0 );
        imMan = mainApplet.getImage(
                                mainApplet.getDocumentBase(),
                                "Man.gif" );
        t.addImage( imMan, 0 );
        imSkeleton = mainApplet.getImage(
                                mainApplet.getDocumentBase(),
                                "Skeleton.gif" );
        t.addImage( imSkeleton, 0 );
        imLampOff = mainApplet.getImage(
                                mainApplet.getDocumentBase(),
                                "LampOff.gif" );
        t.addImage( imLampOff, 0 );
        imLampOn = mainApplet.getImage(
                                mainApplet.getDocumentBase(),
                                "LampOn.gif" );
    t.addImage( imLampOn, 0 );
    imDoorsTimer = mainApplet.getImage(
                          mainApplet.getDocumentBase(),
                          "DoorsTimer.gif" );
    t.addImage( imDoorsTimer, 0 );
    im3DownOn = mainApplet.getImage(
                          mainApplet.getDocumentBase(),
                          "3DownOn.gif" );
    t.addImage( im3DownOn, 0 );
    im2UpOn = mainApplet.getImage(
                          mainApplet.getDocumentBase(),
                          "2UpOn.gif" );
    t.addImage( im2UpOn, 0 );
    im2DownOn = mainApplet.getImage(
                          mainApplet.getDocumentBase(),
                          "2DownOn.gif" );
    t.addImage( im2DownOn, 0 );
    im1UpOn = mainApplet.getImage(
                          mainApplet.getDocumentBase(),
                          "1UpOn.gif" );
    t.addImage( im1UpOn, 0 );
    imPanel0 = mainApplet.getImage(
                          mainApplet.getDocumentBase(),
                          "Panel0.gif" );
    t.addImage( imPanel0, 0 );
    imPanel1 = mainApplet.getImage(
                          mainApplet.getDocumentBase(),
                          "Panel1.gif" );
    t.addImage( imPanel1, 0 );
    imPanel2 = mainApplet.getImage(
                          mainApplet.getDocumentBase(),
                          "Panel2.gif" );
    t.addImage( imPanel2, 0 );
    imPanel3 = mainApplet.getImage(
                          mainApplet.getDocumentBase(),
                          "Panel3.gif" );
    t.addImage( imPanel3, 0 );
    imPanelS = mainApplet.getImage(
                          mainApplet.getDocumentBase(),
                          "PanelS.gif" );
    t.addImage( imPanelS, 0 );
    imGoLeft = mainApplet.getImage(
                          mainApplet.getDocumentBase(),
                          "GoLeft.gif" );
    t.addImage( imGoLeft, 0 );
    imGoRight = mainApplet.getImage(
                          mainApplet.getDocumentBase(),
                          "GoRight.gif" );
    t.addImage( imGoRight, 0 );
    try
    {
        t.waitForID( 0 );
    }
    catch ( InterruptedException e )
    {
    }
}

/**
 * Processing of "Mouse click" events
 * and their addition to the queue
 */
public void mousePressed( MouseEvent e )
{
    int x = e.getX();
    int y = e.getY();
    if ( x>=111 && x<=143 && y>=64 && y<=96 )
        addEvent( 32 );
    else if ( x>=111 && x<=143 && y>=108 && y<=140 )
        addEvent( 21 );
    else if ( x>=111 && x<=143 && y>=168 && y<=200 )
        addEvent( 22 );
    else if ( x>=111 && x<=143 && y>=211 && y<=243 )
        addEvent( 11 );
    else if ( x>=297 && x<=332 && y>=23  && y<=58 )
        addEvent( 43 );
    else if ( x>=297 && x<=332 && y>=61  && y<=96 )
        addEvent( 42 );
    else if ( x>=297 && x<=332 && y>=100  && y<=135 )
        addEvent( 41 );
    else if ( x>=297 && x<=332 && y>=138  && y<=173 )
        addEvent( 44 );
    else if ( x>=177 && x<=242 && y>=238  && y<=288 )
        addEvent( 51 );
    else if ( x>=291 && x<=356 && y>=233  && y<=283 )
        addEvent( 50 );
}
public void mouseReleased( MouseEvent e ) {}
public void mouseClicked( MouseEvent e ) {}
public void mouseEntered( MouseEvent e ) {}
public void mouseExited( MouseEvent e ) {}

public void update( Graphics g )
{
    paint( g );
}

/**
 * Visualisation area generation
 */
public void paint( Graphics g0  )
{
    Image im = createImage( 370, 310 );
    Graphics g = im.getGraphics();

    g.drawImage( imBackground, 0, 0, this );
    if ( doorsPosition > 0 )
    {
        g.drawImage( imElevatorWallPaper,
                     3, 202 - elevatorPosition, this );
        if ( xx51 )
        {
            g.drawImage( imMan,
                         25, 210 - elevatorPosition, this );
            g.drawImage( imGoRight, 295, 235, this );
        }
        else
            g.drawImage( imGoLeft, 179, 209, this );
    }
    g.drawImage( imElevatorDoor,
                 3, 202 - elevatorPosition,
                 44 - doorsPosition, 303 - elevatorPosition,
                 doorsPosition, 0, 40, 101, this );
    g.drawImage( imElevatorDoor,
                 44 + doorsPosition, 202 - elevatorPosition,
                 85, 303 - elevatorPosition,
                 0, 0, 40 - doorsPosition, 101, this );
    g.drawImage( imSkeleton, 0, 0, this );
    if ( isLampOn )
    {
        g.drawImage( imLampOn, 200, 10, this );
        switch ( panelState )
        {
        case 0:
            g.drawImage( imPanel0, 280, 13, this );
            break;
        case 1:
            g.drawImage( imPanel1, 280, 13, this );
            break;
        case 2:
            g.drawImage( imPanel2, 280, 13, this );
            break;
        case 3:
            g.drawImage( imPanel3, 280, 13, this );
            break;
        case 4:
            g.drawImage( imPanelS, 280, 13, this );
            break;
        }
    }
    else
    {
        g.drawImage( imLampOff, 200, 10, this );
        g.drawImage( imPanel0, 280, 13, this );
    }
    if ( doorsTimer >-1 )
    {
        g.drawImage( imDoorsTimer, 170, 110, this );
        g.setColor( new Color( 127, 200, 127 ) );
        g.fillArc( 197, 147, 47, 47,
                   90, 360*doorsTimer/DOORS_TIMER_MAX );
    }
    if ( zz11 )
        g.drawImage( im1UpOn, 113, 214, this );
    if ( zz21 )
        g.drawImage( im2UpOn, 113, 112, this );
    if ( zz22 )
        g.drawImage( im2DownOn, 113, 174, this );
    if ( zz32 )
        g.drawImage( im3DownOn, 113, 70, this );

    g0.drawImage( im, 0, 0, this );
}

/**
 * Main visualiser operating cycle
 */
public void run()
{
    while ( true )
    {
        // Sampling of an event from the queue
        int ev = getNextEvent();

        // Processing of the event selected
        switch ( ev )
        {
        case 11:
            // Call of the A11 automaton with the event of 11
            aut.A11( 11 );
            break;
```

```
case 21:
    aut.A21( 21 );
    break;

case 22:
    aut.A22( 22 );
    break;

case 32:
    aut.A32( 32 );
    break;

case 41:
case 42:
case 43:
case 44:
    if ( isLampOn )
        aut.A2( ev );
    break;

case 50:
case 51:
    if ( doorsPosition > 0 )
        xx51 = ( ev == 51 );
    break;
}

// Automata calls by flags to be generated by the elevator
if ( doorsTimer >-1 )
    doorsTimer--;

if ( doorsTimer == 0 )
    aut.A0( 90 );

boolean flag = doorsPosition > 0 &&
               doorsPosition < DOORS_MAX ;

doorsPosition += doorsMove;
if ( flag && doorsPosition == DOORS_MAX )
    aut.A0( 80 );

if ( flag && doorsPosition == 0 )
    aut.A0( 81 );

flag = elevatorPosition != ELEVATOR_1_FLOOR &&
       elevatorPosition != ELEVATOR_2_FLOOR &&
       elevatorPosition != ELEVATOR_3_FLOOR;

elevatorPosition += elevatorMove;
if ( flag && ( elevatorPosition == ELEVATOR_1_FLOOR ||
               elevatorPosition == ELEVATOR_2_FLOOR ||
               elevatorPosition == ELEVATOR_3_FLOOR ))
    aut.A0( 60 );

aut.A0( 0 );

repaint();
```

```
            try
            {
                Thread.sleep( DELAY );
            }
            catch ( InterruptedException e )
            {
                break;
            }
        }
    }

    /**
     * Add an event to the queue
     */
    synchronized public void addEvent( int ev )
    {
        events.addElement( new Integer( ev ) );
    }

    /**
     * Retrieve an event from the queue
     */
    synchronized protected int getNextEvent()
    {
        int ev = 0;
        if ( !events.isEmpty() )
        {
            ev = ( (Integer)events.firstElement() ).intValue();
            events.removeElementAt( 0 );
        }
        return ev;
    }

    /**
     * Input variable: a weight is present in the elevator car
     */
    public boolean x51()
    {
        log.input( 51, "a weight is present in the elevator car", xx51
);
        return xx51;
    }

    /**
     * Input variable: the elevator is on the 1st floor
     */
    public boolean x61()
    {
        boolean ret = elevatorPosition == ELEVATOR_1_FLOOR;
        log.input( 61, "the elevator is on the 1st floor", ret );
        return ret;
    }

    /**
     * Input variable: the elevator is on the 2nd floor
     */
    public boolean x62()
    {
        boolean ret = elevatorPosition == ELEVATOR_2_FLOOR;
        log.input( 62, "the elevator is on the 2nd floor", ret );
        return ret;
    }
    /**
     * Input variable: the elevator is on the 3rd floor
     */
    public boolean x63()
    {
        boolean ret = elevatorPosition == ELEVATOR_3_FLOOR;
        log.input( 63, "the elevator is on the 3rd floor", ret );
        return ret;
    }

    /**
     * Output action:
     * switch off the lamp in the «Up» button on the first floor
     */
    public void z110()
    {
        log.action( 110,
            "switch off the lamp in the \”Up\” button on the first
floor” );
        zz11 = false;
    }

    /**
     * Output action:
     * switch on the lamp in the «Up» button on the first floor
     */
    public void z111()
    {
        log.action( 111,
            "switch on the lamp in the \”Up\” button on the first
floor” );
        zz11 = true;
    }

    /**
     * Output action:
     * switch off the lamp in the «Up» button on the second floor
     */
    public void z210()
    {
        log.action( 210,
            "switch off the lamp in the \”Up\” button on the second
floor” );
        zz21 = false;
    }

    /**
     * Output action:
     * switch on the lamp in the «Up» button on the second floor
     */
    public void z211()
    {
        log.action( 211,
            "switch on the lamp in the \”Up\” button on the second
floor” );
        zz21 = true;
    }

    /**
     * Output action:
     * switch off the lamp in the «Down» button on the second floor
     */
    public void z220()
    {
        log.action( 220,
            "switch off the lamp in the \”Down\” button on the second
floor” );
        zz22 = false;
    }

    /**
     * Output action:
     * switch on the lamp in the «Down» button on the second floor
     */
    public void z221()
    {
        log.action( 221,
            "switch on the lamp in the \”Down\” button on the second
floor” );
        zz22 = true;
    }

    /**
     * Output action:
     * switch off the lamp in the «Down» button on the third floor
     */
    public void z320()
    {
        log.action( 320,
            "switch off the lamp in the \”Down\” button on the third
floor” );
        zz32 = false;
    }

    /**
     * Output action:
     * switch on the lamp in the «Down» button on the third floor
     */
    public void z321()
    {
        log.action( 321,
            "switch on the lamp in the \”Down\” button on the third
floor” );
        zz32 = true;
    }

    /**
     * Output action: switch off the lamp in the “1” button
     */
    public void z400()
    {
        log.action( 400, "switch off the lamp in the \"1\" button" );
        panelState = 0;
    }

    /**
     * Output action: switch on the lamp in the "1" button
     */
    public void z411()
    {
        log.action( 411, "switch on the lamp in the \"1\" button" );
        panelState = 1;
    }

    /**
     * Output action: switch on the lamp in the "2" button
     */
    public void z421()
```

```
{
    log.action( 421, "switch on the lamp in the \"2\" button" );
    panelState = 2;
}

/**
 * Output action: switch on the lamp in the "3" button
 */
public void z431()
{
    log.action( 431, "switch on the lamp in the \"3\" button" );
    panelState = 3;
}

/**
 * Output action: switch on the lamp in the "S" button
 */
public void z441()
{
    log.action( 441, "switch on the lamp in the \"S\" button" );
    panelState = 4;
}

/**
 * Output action: switch off the lamp in the elevator car
 */
public void z500()
{
    log.action( 500, "switch off the lamp in the elevator car" );
    isLampOn = false;
}

/**
 * Output action: switch on the lamp in the elevator car
 */
public void z501()
{
    log.action( 501, "switch on the lamp in the elevator car" );
    isLampOn = true;
}

/**
 * Output action: stop the elevator car
 */
public void z700()
{
    log.action( 700, "stop the elevator car" );
    elevatorMove = 0;
}

/**
 * Output action: start the upward motion
 */
public void z701()
{
    log.action( 701, "start the upward motion" );
    elevatorMove = ELEVATOR_MOVE;
}

/**
 * Output action: start the downward motion
 */
public void z702()
{
        log.action( 702, "start the downward motion" );
        elevatorMove = -ELEVATOR_MOVE;
    }

    /**
     * Output action: automatically stop the door motion
     */
    public void z800()
    {
        log.action( 800, "automatically stop the door motion" );
        doorsMove = 0;
    }

    /**
     * Output action: open the doors
     */
    public void z801()
    {
        log.action( 801, "open the doors" );
        doorsMove = DOORS_MOVE;
    }

    /**
     * Output action: close the doors
     */
    public void z802()
    {
        log.action( 802, "close the doors" );
        doorsMove = -DOORS_MOVE;
    }

    /**
     * Output action: switch off the door closing timer
     */
    public void z900()
    {
        log.action( 900, "switch off the door closing timer" );
        doorsTimer = -1;
    }

    /**
     * Output action: start the door closing timer
     */
    public void z901()
    {
        log.action( 901, "start the door closing timer" );
        doorsTimer = DOORS_TIMER_MAX;
    }
}
```

9.6. ElevatorApplet.java

```
import java.applet.Applet;

/**
 * Applet
 */
public class ElevatorApplet
    extends Applet
{
    protected ElevatorVisualizer vis;
    protected Thread visThread;
    protected ElevatorLog log;
    public int getParamInt( String param, int def )
    {
        String temp = getParameter( param );
        if ( temp == null )
            return def;
        try {
            return Integer.parseInt( temp );
        }
        catch ( NumberFormatException e )
        {
            return def;
        }
    }

    public void init()
    {
        setLayout( null );
        log = new ElevatorLog( this );
        vis = new ElevatorVisualizer( this, log );
        vis.setLocation( 0, 0 );
        add( vis );
        log.setLocation( 370, 0 );
        add( log );
        visThread = new Thread( vis );
        visThread.start();
    }

    public void destroy()
    {
        visThread.stop();
    }
}
```

9.7. index.html

```
<!-- An aplet starting HTML-file -->
<HTML>
<BODY>
<APPLET CODE="ElevatorApplet.class" WIDTH=820 HEIGHT=310>
<PARAM NAME="DELAY" VALUE="50">
<PARAM NAME="DOORS_TIMER_MAX" VALUE="30">
<PARAM NAME="DOORS_MAX" VALUE="30">
<PARAM NAME="DOORS_MOVE" VALUE="2">
<PARAM NAME="ELEVATOR_MOVE" VALUE="2">
<PARAM NAME="LOG_HEIGHT" VALUE="310">
</APPLET>
</BODY>
</HTML>
```

10. Log fragments

The following notation conventions are used in the logs:

`#`	–	a click on log management flags;
`T`	–	automaton transfer;
`*`	–	output actions;
`{`	–	beginning of automaton operation;

`}` – ending of automaton operation;

`>` – input variables.

In the logs given below, only numbers are indicated for the states whereas the state descriptions are not specified in order to reduce the log size.

Each log reflects the following script:

- the elevator car is called by pressing the button on the first floor;
- the lamp in the elevator car comes on at the door opening moment;
- a person comes into the doors being opening;
- the "3" button is pressed on the car operational panel;
- upon full opening of the doors, they will close again and the elevator car will move upwards;
- the "Down" button is pressed on the second floor and the lamp in it comes on;
- the elevator car is passing by the second floor and reached the third floor;
- the doors get open on the third floor and the person goes out;
- the timer for automatic door closing gets started;
- upon operation of the timer, the doors get closed;
- at the door closing moment, the lamp in the elevator car goes out;
- the elevator goes to the second floor;
- the doors get open on the second floor, the lamp in the elevator car comes on and the lamp in the "Down" button goes out;
- the timer for automatic door closing gets started;
- upon operation of the timer, the doors get closed;
- at the door closing moment, the lamp in the elevator car goes out.

10.1. Example of a simplified log

```
20:36:17.354 # Log automata state modification: true
20:36:18.145 # Log output actions: true
20:36:18.876 # Log A11 automaton: true
20:36:19.587 # Log A21 automaton: true
20:36:19.878 # Log A22 automaton: true
20:36:20.478 # Log A32 automaton: true
20:36:21.189 # Log A0 automaton: true
20:36:21.530 # Log A1 automaton: true
20:36:21.930 # Log A2 automaton: true
20:36:23.933 T A11: has transferred from the state of 0 to the state
of 1
20:36:23.933 * z111 - switch on the lamp in the «Up» button on the
first floor
20:36:23.933 * z700 - stop the elevator car
20:36:23.943 * z501 - switch on the lamp in the elevator car
20:36:23.943 T A0: has transferred from the state of 0 to the state of
3
20:36:23.943 T A11: has transferred from the state of 1 to the state
of 2
```

```
20:36:23.953 * z110 - switch off the lamp in the «Up» button on the
first floor
20:36:23.953 * z801 - open the doors
20:36:24.724 * z800 - automatically stop the door motion
20:36:24.724 T A0: has transferred from the state of 3 to the state of
4
20:36:24.724 * z901 - start the door closing timer
20:36:24.734 * z900 - switch off the door closing timer
20:36:24.734 T A0: has transferred from the state of 4 to the state of
6
20:36:27.068 T A2: has transferred from the state of 0 to the state of
3
20:36:27.068 * z431 - switch on the lamp in the "3" button
20:36:27.088 T A0: has transferred from the state of 6 to the state of
7
20:36:27.098 * z802 - close the doors
20:36:27.869 * z800 - automatically stop the door motion
20:36:27.869 T A0: has transferred from the state of 7 to the state of
9
20:36:27.889 T A11: has transferred from the state of 2 to the state
of 0
20:36:27.899 T A0: has transferred from the state of 9 to the state of
10
20:36:27.909 T A1: has transferred from the state of 0 to the state of
1
20:36:27.919 * z701 - start the upward motion
20:36:28.490 T A22: has transferred from the state of 0 to the state
of 1
20:36:28.490 * z221 - switch on the lamp in the «Down» button on the
second floor
20:36:32.986 * z700 - stop the elevator car
20:36:32.986 T A0: has transferred from the state of 10 to the state
of 8
20:36:33.006 T A2: has transferred from the state of 3 to the state of
0
20:36:33.016 * z400 - switch off the lamp in the button
20:36:33.026 T A32: has transferred from the state of 0 to the state
of 2
20:36:33.036 * z320 - switch off the lamp in the «Down» button on the
third floor
20:36:33.056 * z801 - open the doors
20:36:33.577 T A0: has transferred from the state of 8 to the state of
3
20:36:33.587 * z801 - open the doors
20:36:33.868 * z800 - automatically stop the door motion
20:36:33.868 T A0: has transferred from the state of 3 to the state of
4
20:36:33.888 * z901 - start the door closing timer
20:36:35.420 T A0: has transferred from the state of 4 to the state of
5
20:36:35.420 * z802 - close the doors
20:36:36.151 * z800 - automatically stop the door motion
20:36:36.151 T A0: has transferred from the state of 5 to the state of
0
20:36:36.171 T A32: has transferred from the state of 2 to the state
of 0
20:36:36.181 * z500 - switch off the lamp in the elevator car
20:36:36.191 T A0: has transferred from the state of 0 to the state of
2
20:36:36.201 T A1: has transferred from the state of 1 to the state of
0
20:36:36.221 * z702 - start the downward motion
20:36:38.745 * z700 - stop the elevator car
20:36:38.755 * z501 - switch on the lamp in the elevator car
```

```
20:36:38.765 T A0: has transferred from the state of 2 to the state of
3
20:36:38.775 T A21: has transferred from the state of 0 to the state
of 2
20:36:38.795 * z210 - switch off the lamp in the «Up» button on the
second floor
20:36:38.805 T A22: has transferred from the state of 1 to the state
of 2
20:36:38.825 * z220 - switch off the lamp in the «Down» button on the
second floor
20:36:38.835 * z801 - open the doors
20:36:39.616 * z800 - automatically stop the door motion
20:36:39.626 T A0: has transferred from the state of 3 to the state of
4
20:36:39.646 * z901 - start the door closing timer
20:36:41.168 T A0: has transferred from the state of 4 to the state of
5
20:36:41.178 * z802 - close the doors
20:36:41.909 * z800 - automatically stop the door motion
20:36:41.919 T A0: has transferred from the state of 5 to the state of
0
20:36:41.939 T A21: has transferred from the state of 2 to the state
of 0
20:36:41.949 T A22: has transferred from the state of 2 to the state
of 0
20:36:41.969 * z500 - switch off the lamp in the elevator car
```

10.2. Example of a full log

The log fragments accurately repeating in time have been manually replaced by dots.

```
20:37:46.091 # Log automata start: true
20:37:46.402 # Log automata state modification: true
20:37:46.722 # Log automata operation completion: true
20:37:47.053 # Log input variable polling: true
20:37:47.343 # Log output actions: true
20:37:47.994 # Log A11 automaton: true
20:37:48.285 # Log A21 automaton: true
20:37:48.665 # Log A22 automaton: true
20:37:48.956 # Log A32 automaton: true
20:37:49.587 # Log A0 automaton: true
20:37:49.847 # Log A1 automaton: true
20:37:50.197 # Log A2 automaton: true
20:37:50.838 { A0: being in the state 0 has been started with the
event of e0
20:37:50.838 > x61 - the elevator is on the 1st floor - return true
20:37:50.848 > x62 - the elevator is on the 2nd floor - return false
20:37:50.848 > x63 - the elevator is on the 3rd floor - return false
20:37:50.848 } A0: has completed processing of the event of e0 in the
state of 0
...
20:37:51.619 { A0: being in the state 0 has been started with the
event of e0
20:37:51.629 > x61 - the elevator is on the 1st floor - return true
20:37:51.639 > x62 - the elevator is on the 2nd floor - return false
20:37:51.649 > x63 - the elevator is on the 3rd floor - return false
20:37:51.669 } A0: has completed processing of the event of e0 in the
state of 0
20:37:51.730 { A11: being in the state 0 has been started with the
event of e11
20:37:51.740 T A11: has transferred from the state of 0 to the state
of 1
20:37:51.750 * zz111 - switch on the lamp in the «Up» button on the
first floor
20:37:51.770 } A11: has completed processing of the event of e11 in
the state of 1
20:37:51.780 { A0: being in the state 0 has been started with the
event of e0
20:37:51.800 > x61 - the elevator is on the 1st floor - return true
20:37:51.810 > x62 - the elevator is on the 2nd floor - return false
20:37:51.820 > x63 - the elevator is on the 3rd floor - return false
20:37:51.840 * z700 - stop the elevator car
20:37:51.850 * z501 - switch on the lamp in the elevator car
20:37:51.870 T A0: has transferred from the state of 0 to the state of
3
20:37:51.880 { A2: being in the state 0 has been started with the
event of e2
20:37:51.900 > x51 - a weight is present in the elevator car - return
false
20:37:51.920 } A2: has completed processing of the event of e2 in the
state of 0
20:37:51.930 { A11: being in the state 1 has been started with the
event of e3
20:37:51.950 > x61 - the elevator is on the 1st floor - return true
20:37:51.960 T A11: has transferred from the state of 1 to the state
of 2
20:37:51.980 * z110 - switch off the lamp in the «Up» button on the
first floor
20:37:52.000 } A11: has completed processing of the event of e3 in the
state of 2
20:37:52.010 { A21: being in the state 0 has been started with the
event of e3
20:37:52.030 > x62 - the elevator is on the 2nd floor - return false
20:37:52.040 } A21: has completed processing of the event of e3 in the
state of 0
20:37:52.060 { A22: being in the state 0 has been started with the
event of e3
20:37:52.080 > x62 - the elevator is on the 2nd floor - return false
20:37:52.090 } A22: has completed processing of the event of e3 in the
state of 0
20:37:52.110 { A32: being in the state 0 has been started with the
event of e3
20:37:52.130 > x63 - the elevator is on the 3rd floor - return false
20:37:52.140 } A32: has completed processing of the event of e3 in the
state of 0
20:37:52.160 * z801 - open the doors
20:37:52.180 } A0: has completed processing of the event of e0 in the
state of 3
20:37:52.240 { A0: being in the state 3 has been started with the
event of e0
20:37:52.250 } A0: has completed processing of the event of e0 in the
state of 3
...
20:37:53.382 { A0: being in the state 3 has been started with the
event of e0
20:37:53.402 } A0: has completed processing of the event of e0 in the
state of 3
20:37:53.472 { A0: being in the state 3 has been started with the
event of e80
20:37:53.482 * z800 - automatically stop the door motion
20:37:53.502 T A0: has transferred from the state of 3 to the state of
4
20:37:53.532 * z901 - start the door closing timer
20:37:53.552 } A0: has completed processing of the event of e80 in the
state of 4
```

```
20:37:53.572 { A0: being in the state 4 has been started with the
event of e0
20:37:53.592 > x51 - a weight is present in the elevator car - return
true
20:37:53.612 * z900 - switch off the door closing timer
20:37:53.632 T A0: has transferred from the state of 4 to the state of
6
20:37:53.662 } A0: has completed processing of the event of e0 in the
state of 6
20:37:53.742 { A0: being in the state 6 has been started with the
event of e0
20:37:53.752 > x51 - a weight is present in the elevator car - return
true
20:37:53.783 } A0: has completed processing of the event of e0 in the
state of 6
...
20:37:54.223 { A0: being in the state 6 has been started with the
event of e0
20:37:54.233 > x51 - a weight is present in the elevator car - return
true
20:37:54.263 } A0: has completed processing of the event of e0 in the
state of 6
20:37:54.343 { A2: being in the state 0 has been started with the
event of e43
20:37:54.353 > x51 - a weight is present in the elevator car - return
true
20:37:54.383 > x63 - the elevator is on the 3rd floor - return false
20:37:54.413 T A2: has transferred from the state of 0 to the state of
3
20:37:54.433 * z431 - switch on the lamp in the "3" button
20:37:54.454 } A2: has completed processing of the event of e43 in the
state of 3
20:37:54.484 { A0: being in the state 6 has been started with the
event of e0
20:37:54.504 T A0: has transferred from the state of 6 to the state of
7
20:37:54.544 * z802 - close the doors
20:37:54.584 } A0: has completed processing of the event of e0 in the
state of 7
20:37:54.674 { A0: being in the state 7 has been started with the
event of e0
20:37:54.694 > x51 - a weight is present in the elevator car - return
true
20:37:54.724 } A0: has completed processing of the event of e0 in the
state of 7
...
20:37:56.296 { A0: being in the state 7 has been started with the
event of e0
20:37:56.316 > x51 - a weight is present in the elevator car - return
true
20:37:56.346 } A0: has completed processing of the event of e0 in the
state of 7
20:37:56.436 { A22: being in the state 0 has been started with the
event of e22
20:37:56.456 T A22: has transferred from the state of 0 to the state
of 1
20:37:56.486 * z221 - switch on the lamp in the «Down» button on the
second floor
20:37:56.516 } A22: has completed processing of the event of e22 in
the state of 1
20:37:56.547 { A0: being in the state 7 has been started with the
event of e0
20:37:56.577 > x51 - a weight is present in the elevator car - return
true
```

```
20:37:56.607 } A0: has completed processing of the event of e0 in the
state of 7
20:37:56.697 { A0: being in the state 7 has been started with the
event of e81
20:37:56.717 * z800 - automatically stop the door motion
20:37:56.747 T A0: has transferred from the state of 7 to the state of
9
20:37:56.777 { A11: being in the state 2 has been started with the
event of e4
20:37:56.817 T A11: has transferred from the state of 2 to the state
of 0
20:37:56.857 } A11: has completed processing of the event of e4 in the
state of 0
20:37:56.897 { A21: being in the state 0 has been started with the
event of e4
20:37:56.937 } A21: has completed processing of the event of e4 in the
state of 0
20:37:56.977 { A22: being in the state 1 has been started with the
event of e4
20:37:57.007 } A22: has completed processing of the event of e4 in the
state of 1
20:37:57.047 { A32: being in the state 0 has been started with the
event of e4
20:37:57.077 } A32: has completed processing of the event of e4 in the
state of 0
20:37:57.107 } A0: has completed processing of the event of e81 in the
state of 9
20:37:57.137 { A0: being in the state 9 has been started with the
event of e0
20:37:57.167 > x61 - the elevator is on the 1st floor - return true
20:37:57.197 > x62 - the elevator is on the 2nd floor - return false
20:37:57.227 > x63 - the elevator is on the 3rd floor - return false
20:37:57.258 T A0: has transferred from the state of 9 to the state of
10
20:37:57.288 { A1: being in the state 0 has been started with the
event of e9
20:37:57.318 T A1: has transferred from the state of 0 to the state of
1
20:37:57.348 } A1: has completed processing of the event of e9 in the
state of 1
20:37:57.388 * z701 - start the upward motion
20:37:57.418 } A0: has completed processing of the event of e0 in the
state of 10
20:37:57.508 { A0: being in the state 10 has been started with the
event of e0
20:37:57.528 > x61 - the elevator is on the 1st floor - return false
20:37:57.568 > x62 - the elevator is on the 2nd floor - return false
20:37:57.608 > x63 - the elevator is on the 3rd floor - return false
20:37:57.658 } A0: has completed processing of the event of e0 in the
state of 10
...
20:38:33.410 { A0: being in the state 10 has been started with the
event of e0
20:38:33.490 > x61 - the elevator is on the 1st floor - return false
20:38:33.590 > x62 - the elevator is on the 2nd floor - return false
20:38:33.680 > x63 - the elevator is on the 3rd floor - return false
20:38:33.770 } A0: has completed processing of the event of e0 in the
state of 10
20:38:33.900 { A0: being in the state 10 has been started with the
event of e60
20:38:33.980 > x61 - the elevator is on the 1st floor - return false
20:38:34.060 > x62 - the elevator is on the 2nd floor - return false
20:38:34.151 > x63 - the elevator is on the 3rd floor - return true
20:38:34.271 * z700 - stop the elevator car
```

```
20:38:34.381 T A0: has transferred from the state of 10 to the state
of 8
20:38:34.461 { A2: being in the state 1 has been started with the
event of e2
20:38:34.551 T A2: has transferred from the state of 3 to the state of
0
20:38:34.641 * z400 - switch off the lamp in the button
20:38:34.721 } A2: has completed processing of the event of e2 in the
state of 0
20:38:34.812 { A11: being in the state 0 has been started with the
event of e3
20:38:34.912 > x61 - the elevator is on the 1st floor - return false
20:38:35.012 } A11: has completed processing of the event of e3 in the
state of 0
20:38:35.122 { A21: being in the state 0 has been started with the
event of e3
20:38:35.212 > x62 - the elevator is on the 2nd floor - return false
20:38:35.292 } A21: has completed processing of the event of e3 in the
state of 0
20:38:35.382 { A22: being in the state 1 has been started with the
event of e3
20:38:35.472 > x62 - the elevator is on the 2nd floor - return false
20:38:35.563 } A22: has completed processing of the event of e3 in the
state of 1
20:38:35.643 { A32: being in the state 0 has been started with the
event of e3
20:38:35.773 > x63 - the elevator is on the 3rd floor - return true
20:38:35.873 T A32: has transferred from the state of 0 to the state
of 2
20:38:35.963 * z320 - switch off the lamp in the «Down» button on the
third floor
20:38:36.053 } A32: has completed processing of the event of e3 in the
state of 2
20:38:36.143 * z801 - open the doors
20:38:36.224 } A0: has completed processing of the event of e60 in the
state of 8
20:38:36.314 { A0: being in the state 8 has been started with the
event of e0
20:38:36.404 > x51 - a weight is present in the elevator car - return
true
20:38:36.514 } A0: has completed processing of the event of e0 in the
state of 8
...
20:38:38.697 { A0: being in the state 8 has been started with the
event of e0
20:38:38.787 > x51 - a weight is present in the elevator car - return
true
20:38:38.897 } A0: has completed processing of the event of e0 in the
state of 8
20:38:39.078 { A0: being in the state 8 has been started with the
event of e0
20:38:39.168 > x51 - a weight is present in the elevator car - return
false
20:38:39.258 T A0: has transferred from the state of 8 to the state of
3
20:38:39.348 { A2: being in the state 1 has been started with the
event of e2
20:38:39.438 > x51 - a weight is present in the elevator car - return
false
20:38:39.528 } A2: has completed processing of the event of e2 in the
state of 0
20:38:39.648 { A11: being in the state 0 has been started with the
event of e3
20:38:39.779 > x61 - the elevator is on the 1st floor - return false
20:38:39.869 } A11: has completed processing of the event of e3 in the
state of 0
20:38:39.959 { A21: being in the state 0 has been started with the
event of e3
20:38:40.049 > x62 - the elevator is on the 2nd floor - return false
20:38:40.149 } A21: has completed processing of the event of e3 in the
state of 0
20:38:40.239 { A22: being in the state 1 has been started with the
event of e3
20:38:40.329 > x62 - the elevator is on the 2nd floor - return false
20:38:40.470 } A22: has completed processing of the event of e3 in the
state of 1
20:38:40.580 { A32: being in the state 2 has been started with the
event of e3
20:38:40.680 } A32: has completed processing of the event of e3 in the
state of 2
20:38:40.770 * z801 - open the doors
20:38:40.870 } A0: has completed processing of the event of e0 in the
state of 3
20:38:41.020 { A0: being in the state 3 has been started with the
event of e0
...
20:38:42.382 } A0: has completed processing of the event of e0 in the
state of 3
20:38:42.533 { A0: being in the state 3 has been started with the
event of e80
20:38:42.623 * z800 - automatically stop the door motion
20:38:42.713 T A0: has transferred from the state of 3 to the state of
4
20:38:42.823 * z901 - start the door closing timer
20:38:42.953 } A0: has completed processing of the event of e80 in the
state of 4
20:38:43.053 { A0: being in the state 4 has been started with the
event of e0
20:38:43.144 > x51 - a weight is present in the elevator car - return
false
20:38:43.244 } A0: has completed processing of the event of e0 in the
state of 4
...
20:38:55.692 { A0: being in the state 4 has been started with the
event of e0
20:38:55.822 > x51 - a weight is present in the elevator car - return
false
20:38:55.942 } A0: has completed processing of the event of e0 in the
state of 4
20:38:56.102 { A0: being in the state 4 has been started with the
event of e90
20:38:56.212 T A0: has transferred from the state of 4 to the state of
5
20:38:56.423 * z802 - close the doors
20:38:56.543 } A0: has completed processing of the event of e90 in the
state of 5
20:38:56.653 { A0: being in the state 5 has been started with the
event of e0
20:38:56.773 > x51 - a weight is present in the elevator car - return
false
20:38:56.883 } A0: has completed processing of the event of e0 in the
state of 5
...
20:39:02.261 { A0: being in the state 5 has been started with the
event of e0
20:39:02.371 > x51 - a weight is present in the elevator car - return
false
```

```
20:39:02.501 } A0: has completed processing of the event of e0 in the
state of 5
20:39:02.652 { A0: being in the state 5 has been started with the
event of e81
20:39:02.762 * z800 - automatically stop the door motion
20:39:02.872 T A0: has transferred from the state of 5 to the state of
0
20:39:02.982 { A11: being in the state 0 has been started with the
event of e4
20:39:03.092 } A11: has completed processing of the event of e4 in the
state of 0
20:39:03.202 { A21: being in the state 0 has been started with the
event of e4
20:39:03.323 } A21: has completed processing of the event of e4 in the
state of 0
20:39:03.443 { A22: being in the state 1 has been started with the
event of e4
20:39:03.553 } A22: has completed processing of the event of e4 in the
state of 1
20:39:03.663 { A32: being in the state 2 has been started with the
event of e4
20:39:03.783 T A32: has transferred from the state of 2 to the state
of 0
20:39:03.893 } A32: has completed processing of the event of e4 in the
state of 0
20:39:04.014 * z500 - switch off the lamp in the elevator car
20:39:04.124 } A0: has completed processing of the event of e81 in the
state of 0
20:39:04.244 { A0: being in the state 0 has been started with the
event of e0
20:39:04.354 > x61 - the elevator is on the 1st floor - return false
20:39:04.464 > x62 - the elevator is on the 2nd floor - return false
20:39:04.574 > x63 - the elevator is on the 3rd floor - return true
20:39:04.684 T A0: has transferred from the state of 0 to the state of
2
20:39:04.795 { A1: being in the state 1 has been started with the
event of e8
20:39:04.905 T A1: has transferred from the state of 1 to the state of
0
20:39:05.015 } A1: has completed processing of the event of e8 in the
state of 0
20:39:05.125 * z702 - start the downward motion
20:39:05.235 } A0: has completed processing of the event of e0 in the
state of 2
20:39:05.396 { A0: being in the state 2 has been started with the
event of e0
20:39:05.496 > x61 - the elevator is on the 1st floor - return false
20:39:05.606 > x62 - the elevator is on the 2nd floor - return false
20:39:05.726 > x63 - the elevator is on the 3rd floor - return false
20:39:05.836 } A0: has completed processing of the event of e0 in the
state of 2
...
20:39:38.303 { A0: being in the state 2 has been started with the
event of e0
20:39:38.433 > x61 - the elevator is on the 1st floor - return false
20:39:38.573 > x62 - the elevator is on the 2nd floor - return false
20:39:38.713 > x63 - the elevator is on the 3rd floor - return false
20:39:38.854 } A0: has completed processing of the event of e0 in the
state of 2
20:39:39.044 { A0: being in the state 2 has been started with the
event of e60
20:39:39.174 > x61 - the elevator is on the 1st floor - return false
20:39:39.314 > x62 - the elevator is on the 2nd floor - return true
20:39:39.454 * z700 - stop the elevator car
```

```
20:39:39.595 * z501 - switch on the lamp in the elevator car
20:39:39.735 T A0: has transferred from the state of 2 to the state of
3
20:39:39.865 { A2: being in the state 0 has been started with the
event of e2
20:39:40.005 > x51 - a weight is present in the elevator car - return
false
20:39:40.155 } A2: has completed processing of the event of e2 in the
state of 0
20:39:40.296 { A11: being in the state 0 has been started with the
event of e3
20:39:40.436 > x61 - the elevator is on the 1st floor - return false
20:39:40.576 } A11: has completed processing of the event of e3 in the
state of 0
20:39:40.726 { A21: being in the state 0 has been started with the
event of e3
20:39:40.877 > x62 - the elevator is on the 2nd floor - return true
20:39:41.007 T A21: has transferred from the state of 0 to the state
of 2
20:39:41.157 * z210 - switch off the lamp in the «Up» button on the
second floor
20:39:41.287 } A21: has completed processing of the event of e3 in the
state of 2
20:39:41.427 { A22: being in the state 1 has been started with the
event of e3
20:39:41.568 > x62 - the elevator is on the 2nd floor - return true
20:39:41.708 T A22: has transferred from the state of 1 to the state
of 2
20:39:41.848 * z220 - switch off the lamp in the «Down» button on the
second floor
20:39:41.998 } A22: has completed processing of the event of e3 in the
state of 2
20:39:42.138 { A32: being in the state 0 has been started with the
event of e3
20:39:42.279 > x63 - the elevator is on the 3rd floor - return false
20:39:42.429 } A32: has completed processing of the event of e3 in the
state of 0
20:39:42.569 * z801 - open the doors
20:39:42.719 } A0: has completed processing of the event of e60 in the
state of 3
20:39:42.859 { A0: being in the state 3 has been started with the
event of e0
...
20:39:47.716 { A0: being in the state 3 has been started with the
event of e0
20:39:47.867 } A0: has completed processing of the event of e0 in the
state of 3
20:39:48.067 { A0: being in the state 3 has been started with the
event of e80
20:39:48.217 * z800 - automatically stop the door motion
20:39:48.367 T A0: has transferred from the state of 3 to the state of
4
20:39:48.518 * z901 - start the door closing timer
20:39:48.658 } A0: has completed processing of the event of e80 in the
state of 4
20:39:48.808 { A0: being in the state 4 has been started with the
event of e0
20:39:48.958 > x51 - a weight is present in the elevator car - return
false
20:39:49.108 } A0: has completed processing of the event of e0 in the
state of 4
...
20:40:03.769 { A0: being in the state 4 has been started with the
event of e0
```

```
20:40:03.930 > x51 - a weight is present in the elevator car - return
false
20:40:04.080 } A0: has completed processing of the event of e0 in the
state of 4
20:40:04.300 { A0: being in the state 4 has been started with the
event of e90
20:40:04.450 T A0: has transferred from the state of 4 to the state of
5
20:40:04.631 * z802 - close the doors
20:40:04.791 } A0: has completed processing of the event of e90 in the
state of 5
20:40:04.951 { A0: being in the state 5 has been started with the
event of e0
20:40:05.111 > x51 - a weight is present in the elevator car - return
false
20:40:05.272 } A0: has completed processing of the event of e0 in the
state of 5
...
20:40:12.021 { A0: being in the state 5 has been started with the
event of e0
20:40:12.172 > x51 - a weight is present in the elevator car - return
false
20:40:12.332 } A0: has completed processing of the event of e0 in the
state of 5
20:40:12.562 { A0: being in the state 5 has been started with the
event of e81
20:40:12.722 * z800 - automatically stop the door motion
20:40:12.893 T A0: has transferred from the state of 5 to the state of
0
20:40:13.063 { A11: being in the state 0 has been started with the
event of e4
20:40:13.233 } A11: has completed processing of the event of e4 in the
state of 0
20:40:13.393 { A21: being in the state 2 has been started with the
event of e4
20:40:13.554 T A21: has transferred from the state of 2 to the state
of 0
20:40:13.714 } A21: has completed processing of the event of e4 in the
state of 0
20:40:13.884 { A22: being in the state 2 has been started with the
event of e4
20:40:14.044 T A22: has transferred from the state of 2 to the state
of 0
20:40:14.204 } A22: has completed processing of the event of e4 in the
state of 0
20:40:14.365 { A32: being in the state 0 has been started with the
event of e4
20:40:14.525 } A32: has completed processing of the event of e4 in the
state of 0
20:40:14.685 * z500 - switch off the lamp in the elevator car
20:40:14.855 } A0: has completed processing of the event of e81 in the
state of 0
20:40:15.016 { A0: being in the state 0 has been started with the
event of e0
20:40:15.176 > x61 - the elevator is on the 1st floor - return false
20:40:15.336 > x62 - the elevator is on the 2nd floor - return true
20:40:15.506 > x63 - the elevator is on the 3rd floor - return false
20:40:15.667 } A0: has completed processing of the event of e0 in the
state of 0
20:40:15.877 { A0: being in the state 0 has been started with the
event of e0
20:40:16.027 > x61 - the elevator is on the 1st floor - return false
20:40:16.197 > x62 - the elevator is on the 2nd floor - return true
20:40:16.368 > x63 - the elevator is on the 3rd floor - return false
```

```
20:40:16.528 } A0: has completed processing of the event of e0 in the
state of 0
```