Санкт-Петербургский государственный институт точной механики и оптики (технический университет)

Кафедра “Компьютерные технологии”

А.С. Наумов, А.А. Шалыто

Система управления лифтом

Объектно-ориентированное программирование с явным выделением состояний

Проектная документация

Проект создан в рамках
“Движения за открытую проектную документацию”
http://is.ifmo.ru

Санкт-Петербург
2003

Содержание

Введение

Для алгоритмизации и программирования задач логического управления А.А.Шалыто была предложена SWITCH-технология, которая применительно к событийным и объектно-ориентированным программам была развита им совместно с Н.И.Туккелем. Подробно ознакомиться с этой технологией и с конкретными примерами ее использования можно на сайтах http://is.ifmo.ru и http://www.softcraft.ru.

Эта технология удобна для задач управления техническими объектами, такими как, например лифт, рассматриваемый в настоящей работе. Это связано с тем, что при использовании автоматного подхода, в частности, удается повысить централизацию логики управления в программном коде. Другое достоинство этого подхода состоит в том, что код является изоморфным графу переходов, по которому он строился. Это позволяет для понимания логики работы программ (их поведения) не обращаться к текстам программ, а рассматривать лишь графы переходов.

Для того чтобы приложение было выполнено в виде аплета, при реализации выбран язык Java. Термин “аплет” здесь и в дальнейшем пишется с одной буквой “п” в виду того, что такое написание принято на сайте http://www.sun.ru.

В данной работе совместно применяются объектно-ориентированное и автоматное программирование, названное А.А.Шалыто и Н.И.Туккелем “объектно-ориентированное программирование с явным выделением состояний”. Особенность рассматриваемого примера состоит в том, что среди используемых классов только один является автоматным. Он содержит семь автоматов, каждый из которых является одним из методов этого класса. Неавтоматные методы в этом классе отсутствуют.

В данной реализации используются два потока, в одном из которых запускаются все автоматы.

1. Постановка задачи

Цель работы состоит в создании программы управления лифтом. Для упрощения проекта дом выбран трехэтажным.

На панелях вызова, расположенных на каждом этаже, предусмотрены по две кнопки (“Вверх” и ”Вниз”). На крайних этажах на этих панелях расположено по одной кнопке. В кабине лифта, кроме кнопок с указанием номеров этажей, имеется также кнопка “Стоп” (“S”), обеспечивающая остановку лифта на ближайшем по ходу движения этаже. Кроме того, в лифте предусмотрен индикатор груза (визуально не отображается) и автоматический таймер закрытия дверей в случае, если в кабине никого нет. Также имеются кнопки “Войти” и “Выйти”, которые должны быть нажаты для входа в лифт и выхода из него.

2. Диаграмма классов

Программа, диаграмма классов которой приведена на рис. 1, содержит две части.

Первая часть состоит из одного класса и двух интерфейсов:

- *ElevatorAutomats* – класс инкапсулирует логику работы системы управления лифтом – реализует все автоматы (разд. 3);
- *ElevatorLogInterface* – интерфейс объявляет методы, обеспечивающие протоколирование логики работы приложения (разд. 4);
- *ElevatorVisualizerInterface* – интерфейс объявляет все входные переменные и выходные воздействия (разд. 5).

Вторая часть состоит из трех классов:

- *ElevatorLog* – реализует интерфейс *ElevatorLogInterface* и протоколирует логику работу приложения (разд. 6);
- *ElevatorVisualizer* – реализует интерфейс *ElevatorVisualizerInterface*, эмулирует и визуализирует работу лифта. Осуществляет запуск автоматов и обработку событий. Реализует функции входных переменных и выходных воздействий (разд. 7);
- *ElevatorApplet* – создает экземпляры классов *ElevatorLog* и *ElevatorVisualizer* и размещает их на панели аплета (разд. 8).

В данной работе показано, как отделить систему управления от объекта управления (лифта). При этом система управления реализуется первой частью программы, а объект управления – второй.

Ниже описывается каждый из перечисленных классов и интерфейсов, причем наиболее подробно описан класс, содержащий автоматы.

Рис. 1. Диаграмма классов

3. Класс “ElevatorAutomats”

3.1. Словесное описание

Класс инкапсулирует логику работы системы управления лифтом. Конструктор класса принимает ссылки на интерфейсы *ElevatorLogInterface* и *ElevatorVisualizerInterface*. В классе реализованы семь автоматов – открытых методов вида `public void Ann( int e )`, где `nn` – номер автомата, `e` – событие.

Состояние каждого автомата хранится в защищенной переменной `protected int ynn`, где `nn` – номер автомата. Начальное состояние всех автоматов – нулевое. На графах переходов предикат “`ynn=s`” проверяет, находится ли автомат с номером `nn` в состоянии с номером `s`.

Отметим, что в разных автоматах могут использоваться одинаковые события, входные переменные и выходные воздействия.

3.2. Нумерация и перечень событий (e)

0 Сработал общий таймер

2 Выключение лампы в кнопке (передается из автомата A0)

3 Выключение лампы в кнопке (передается из автомата A0)

4 Разрешение на включение лампы в кнопке (передается из автомата A0)

8 Установить значение “Вниз” (передается из автомата A0)

9 Установить значение “Вверх” (передается из автомата A0)

11 Нажатие кнопки “Вверх” на первом этаже

21 Нажатие кнопки “Вверх” на втором этаже

22 Нажатие кнопки “Вниз” на втором этаже

32 Нажатие кнопки “Вниз” на третьем этаже

41 Нажатие кнопки “1” в кабине лифта

42 Нажатие кнопки “2” в кабине лифта

43 Нажатие кнопки “3” в кабине лифта

44 Нажатие кнопки “S” (STOP) в кабине лифта

60 Прибытие на этаж

80 Двери открылись

81 Двери закрылись

90 Сработал таймер закрытия дверей

3.3. Нумерация и перечень входных переменных (x)

51 В лифте есть груз

61 Лифт на 1 этаже

62 Лифт на 2 этаже

63 Лифт на 3 этаже

Входные переменные, опрашиваемые автоматами, объявлены в интерфейсе *ElevatorVisualizerInterface* (разд. 5) и реализованы в классе *ElevatorVisualizer* (разд. 7).

3.4. Нумерация и перечень выходных воздействий (z)

110 Выключить лампу в кнопке “Вверх” на первом этаже

111 Включить лампу в кнопке “Вверх” на первом этаже

210 Выключить лампу в кнопке “Вверх” на втором этаже

211 Включить лампу в кнопке “Вверх” на втором этаже

220 Выключить лампу в кнопке “Вниз” на втором этаже

221 Включить лампу в кнопке “Вниз” на втором этаже

320 Выключить лампу в кнопке “Вниз” на третьем этаже

321 Включить лампу в кнопке “Вниз” на третьем этаже

400 Выключить лампу в кнопке

411 Включить лампу в кнопке “1” в кабине лифта

421 Включить лампу в кнопке “2” в кабине лифта

431 Включить лампу в кнопке “3” в кабине лифта

441 Включить лампу в кнопке “S” (STOP) в кабине лифта

500 Выключить лампу в кабине лифта

501 Включить лампу в кабине лифта

700 Остановить лифт

701 Поехать вверх

702 Поехать вниз

800 Автоматически остановить движение дверей

801 Открывать двери

802 Закрывать двери

900 Выключить таймер закрытия дверей

901 Запустить таймер закрытия дверей

Выходные воздействия, вызываемые из автоматов, объявлены в интерфейсе *ElevatorVisualizerInterface* (разд. 5) и реализованы в классе *ElevatorVisualizer* (разд. 7).

3.5. Автомат “Управление лифтом” (A0)

3.5.1. Словесное описание

Основной автомат управления лифтом. Он отвечает за открытие и закрытие дверей, движение и остановку кабины, включение и выключение лампы в кабине. Единственный автомат, который содержит вызываемые автоматы – из состояний этого автомата происходят вызовы других автоматов с определенными событиями. Например, в состоянии 3 последовательно ***вызываются*** автоматы A2, A11, A21, A22, A32 с событиями 2, 3, 3, 3, 3, соответственно, что обозначено как A: 2(2), 11(3), 21(3), 22(3), 32(3).

Схема связей этого автомата приведена на рис. 2. Граф переходов – на рис. 3.

Автомат по событию e0 запускается, однако это событие не участвует в условиях переходов. Поэтому оно присутствует на схеме связей, а на графе переходов его нет.

Автоматы в общем случае реализуются следующим образом.

В первом операторе *switch* реализуются переходы и выполняются действия на них.

Если состояние изменилось, то во втором операторе *switch* (или в его аналоге) выполняются действия в вершинах, а также вызываются автоматы с соответствующими событиями.

Листинги программ приведены в разд. 9.

3.5.2. Схема связей

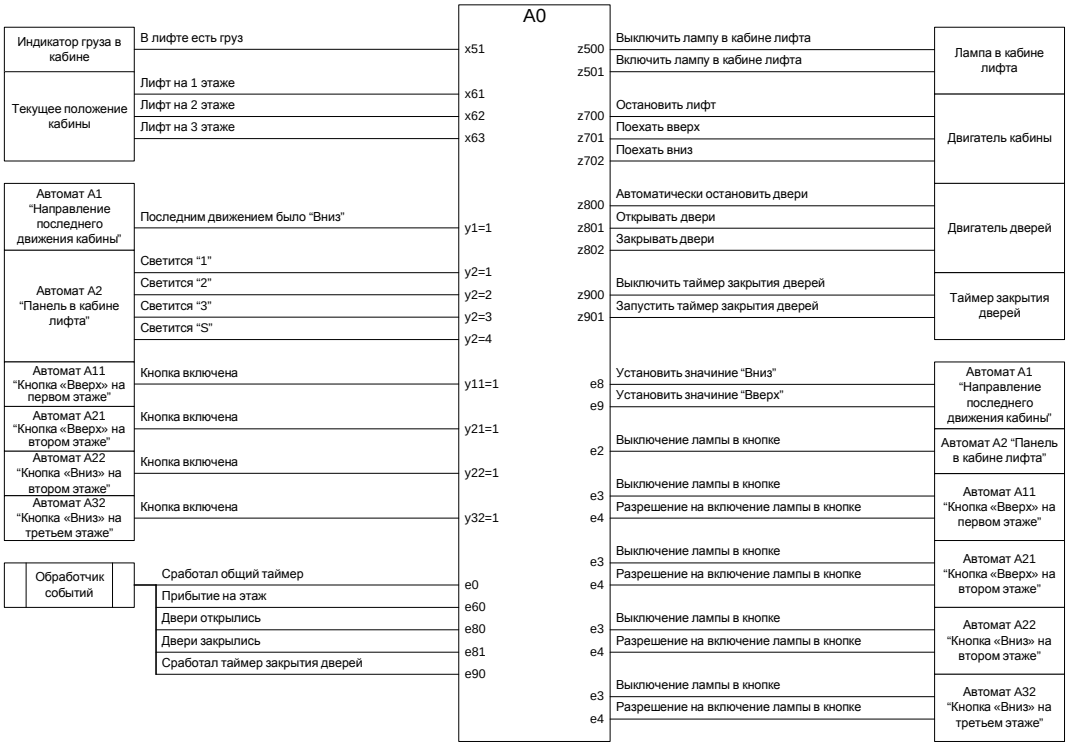

Рис. 2. Схема связей автомата “Управление лифтом”

3.5.3. Граф переходов

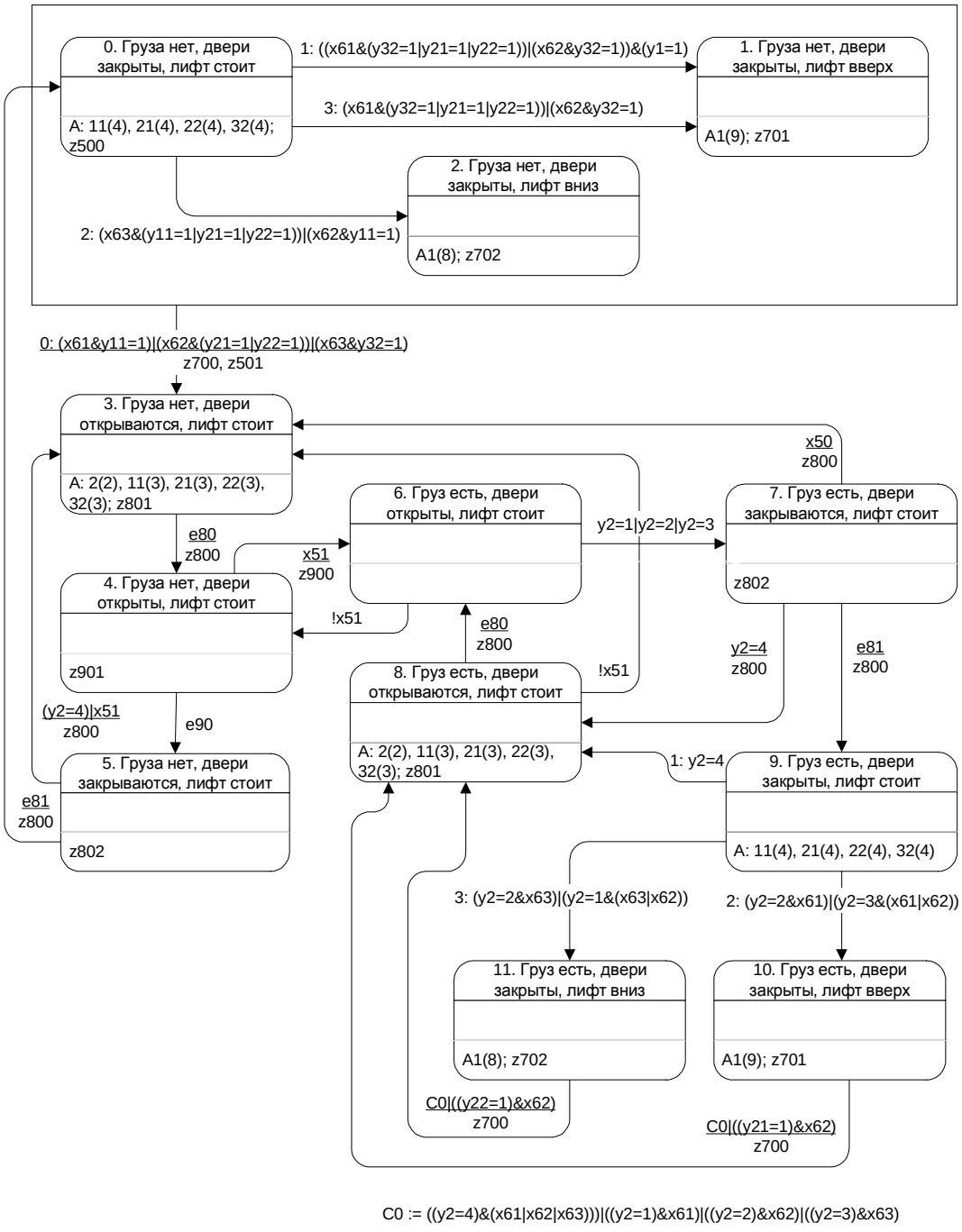

Рис. 3. Граф переходов автомата “Управление лифтом” (A0)

3.6. Автомат “Направление последнего движения кабины” (A1)

3.6.1. Словесное описание

Автомат используется для определения необходимости остановки на промежуточном этаже. Состояние автомата определяет направление последнего движения кабины (“Вверх” или “Вниз”).

Формирование такого простого автомата связано с тем, что выбор этажа, на котором следует остановиться, в общем случае может быть значительно сложнее.

Схема связей этого автомата приведена на рис. 4. Граф переходов – на рис. 5.

3.6.2. Схема связей

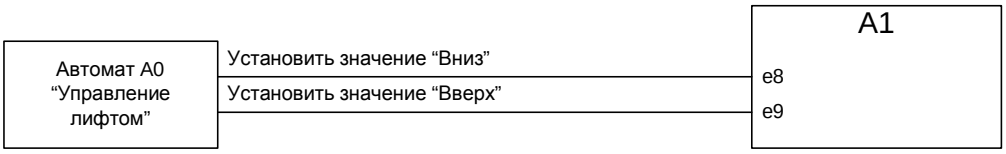

Рис. 4. Схема связей автомата “Направление последнего движения кабины”

3.6.3. Граф переходов

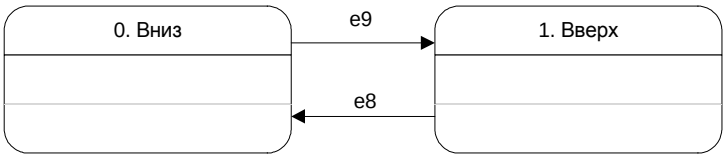

Рис. 5. Граф переходов автомата “Направление последнего движения кабины” (A1)

3.7. Автомат “Панель в кабине лифта” (A2)

3.7.1. Словесное описание

Автомат отвечает за обработку нажатий кнопок панели в кабине, включение и выключение ламп в кнопках.

Схема связей этого автомата приведена на рис. 6. Граф переходов – на рис. 7.

3.7.2. Схема связей

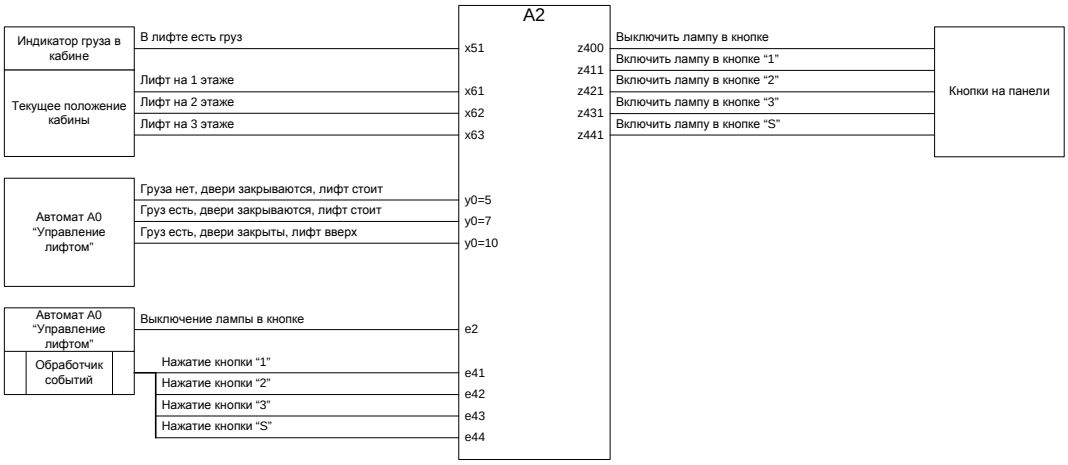

Рис. 6. Схема связей автомата "Панель в кабине лифта"

3.7.3. Граф переходов

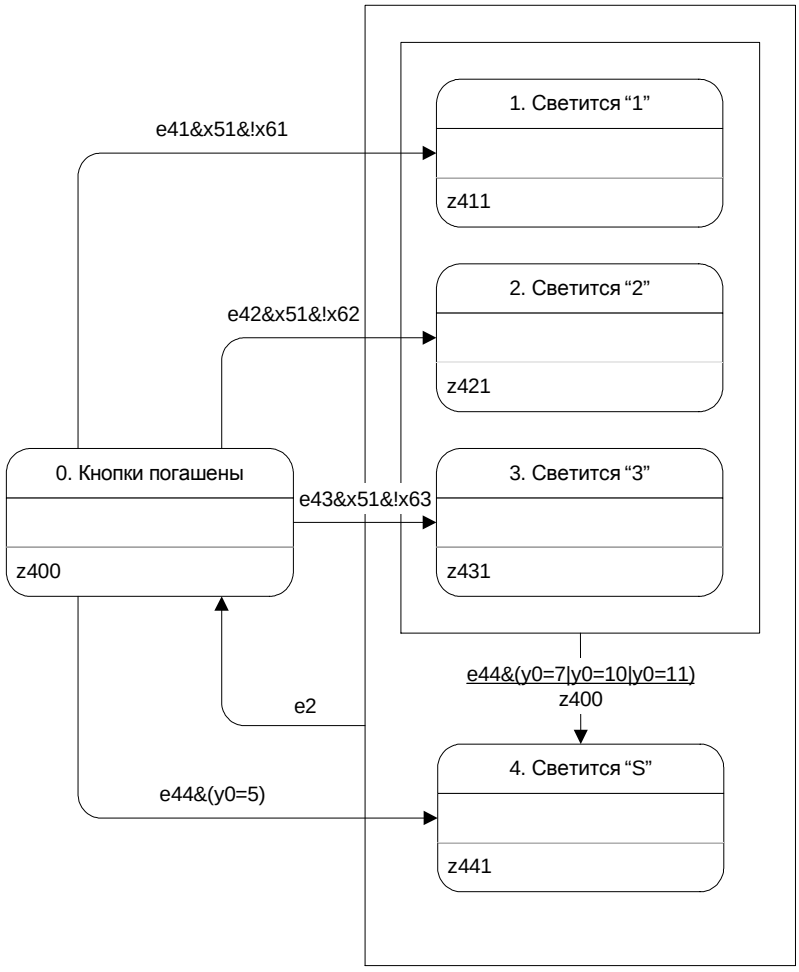

Рис. 7. Граф переходов автомата "Панель в кабине лифта" (A2)

3.8. Автоматы кнопок вызова (A11, A21, A22, A32)

3.8.1. Словесное описание

Автоматы A11 (кнопка “Вверх” на первом этаже), A21 (кнопка “Вверх” на втором этаже), A22 (кнопка “Вниз” на втором этаже), A32 (кнопка “Вниз” на третьем этаже) имеют схожую структуру. Несмотря на то, что эти автоматы могли быть реализованы одним классом, или порождены от одного базового класса, это из-за простоты автоматов сделано не было.

Схемы связей и графы переходов этих автоматов приведены на рис. 8 – рис. 15.

3.8.2. Схемы связей и графы переходов

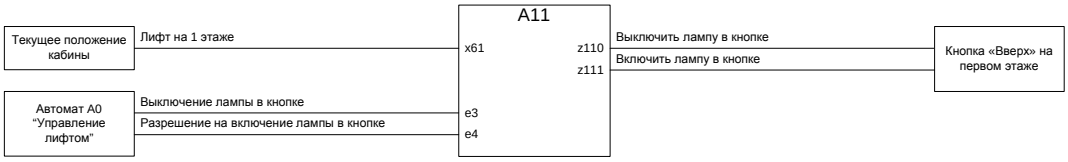

Рис. 8. Схема связей автомата “Кнопка «Вверх» на первом этаже”

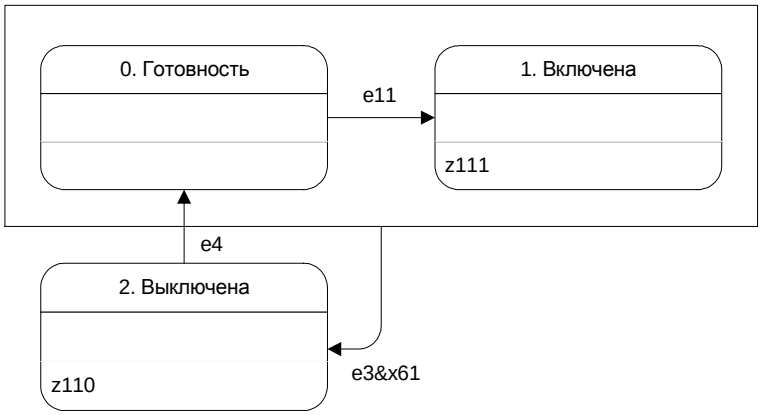

Рис. 9. Граф переходов автомата “Кнопка «Вверх» на первом этаже” (A11)

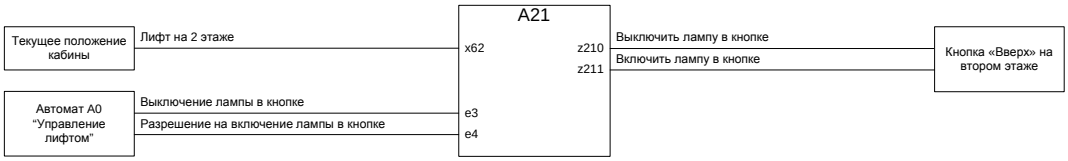

Рис. 10. Схема связей автомата “Кнопка «Вверх» на втором этаже”

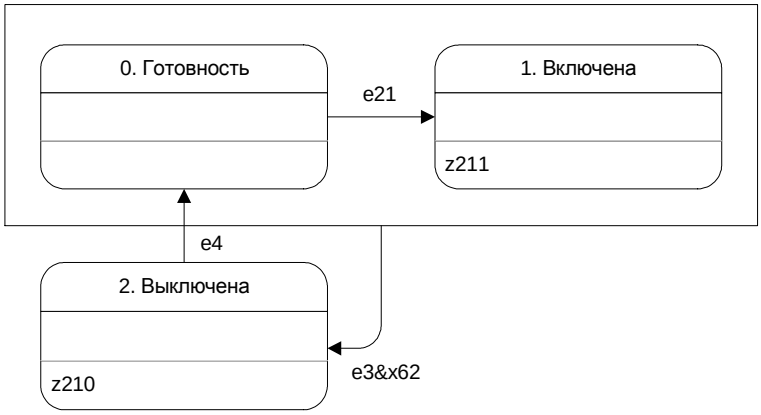

Рис. 11. Граф переходов автомата “Кнопка «Вверх» на втором этаже” (А21)

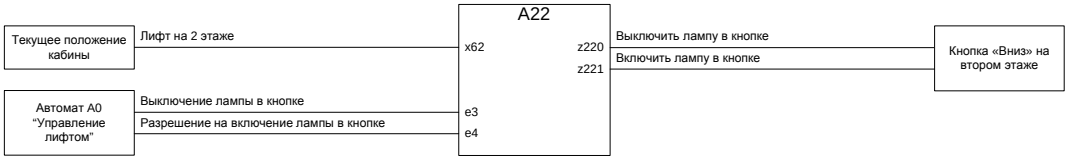

Рис. 12. Схема связей автомата “Кнопка «Вниз» на втором этаже”

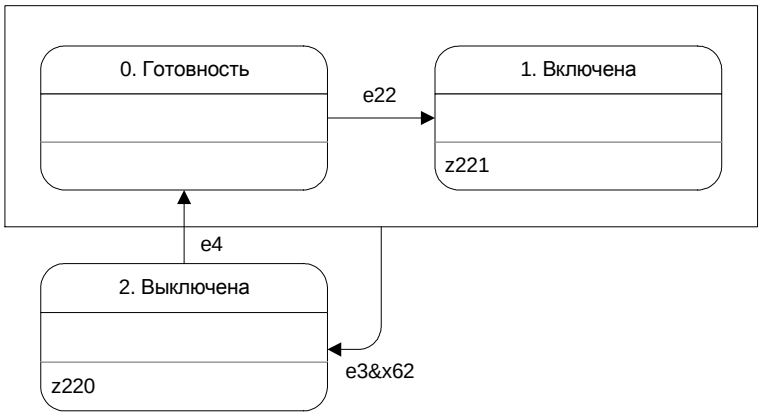

Рис. 13. Граф переходов автомата “Кнопка «Вниз» на втором этаже” (А22)

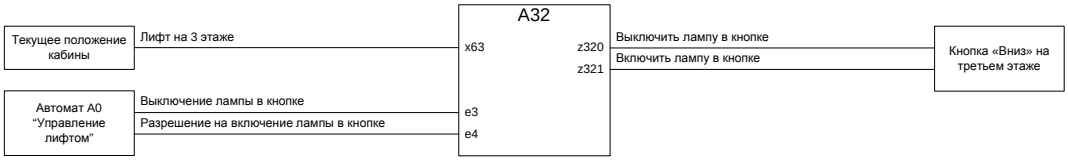

Рис. 14. Схема связей автомата “Кнопка «Вниз» на третьем этаже”

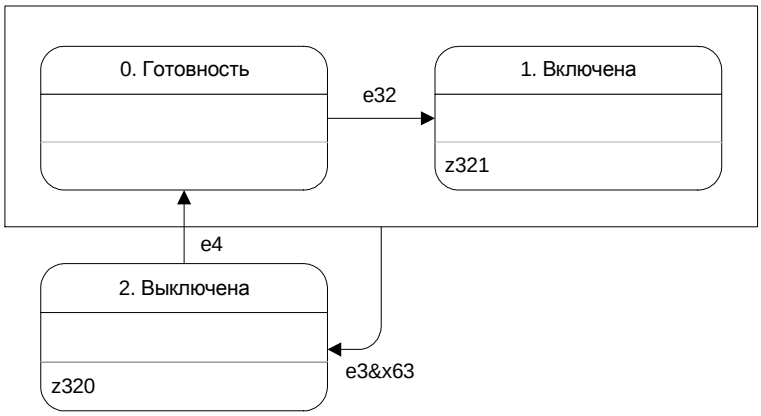

Рис. 15. Граф переходов автомата “Кнопка «Вниз» на третьем этаже” (A32)

4. Интерфейс “ElevatorLogInterface”

4.1. Словесное описание

Интерфейс объявляет методы протоколирования. Методы интерфейса используются классом *ElevatorAutomats*.

4.2. Прототипы методов и их краткое описание

Приведем прототипы методов и их описание:

- `public void begin( int aut, int state, int event )` – добавляет в протокол запись о запуске автомата `aut` в состоянии `state` с событием `event`. Вызывается из класса *ElevatorAutomats*;
- `public void trans( int aut, int old_state, int new_state )` – добавляет в протокол запись о переходе автомата `aut` из состояния `old_state` в состояние `new_state`. Вызывается из класса *ElevatorAutomats*;
- `public void end( int aut, int state, int event )` – добавляет в протокол запись о завершении автоматом `aut` обработки события `event` в состоянии `state`. Вызывается из класса *ElevatorAutomats*;
- `public void input( int num, String str, boolean ret )` – добавляет в протокол запись об опросе входной переменной `num` со словесным описанием `str`. Значение входной переменной – `ret`. Эта функция должна вызываться из методов класса, реализующего интерфейс *ElevatorVisualizerInterface*;

- `public void action( int num, String str )` – добавляет в протокол запись о выполнении выходного воздействия `num` со словесным описанием `str`. Эта функция должна вызываться из методов класса, реализующего интерфейс *ElevatorVisualizerInterface*;
- `public void error( int aut, String err )` – добавляет в протокол запись об ошибке `err`, произошедшей при работе автомата `aut`. Вызывается из класса *ElevatorAutomats*;
- `public void log( String str )` – добавляет в протокол строку `str`.

Использование протокола позволяет находить ошибки в логике программы “мгновенно”.

5. Интерфейс “ElevatorVisualizerInterace”

Интерфейс объявляет методы опроса входных переменных `public boolean xnn()`, где `nn` – номер входной переменной, и методы формирования выходных воздействий `public void znn()`, где `nn` – номер выходного воздействия. Указанные методы используются классом *ElevatorAutomats*.

При необходимости модификации программы класс, реализующий этот интерфейс, в методах опроса входных переменных должен вызывать метод `ElevatorLogInterface.input()`, а в методах формирования выходных воздействий – `ElevatorLogInterface.action()`.

6. Класс “ElevatorLog”

Класс реализует интерфейс *ElevatorLogInterface* и протоколирует работу приложения. Протокол (рис. 16), образуют записи, которые автоматически добавляются в список (класс `java.awt.List`). Подробность протокола регулируется указанным на рис. 16 набором флажков (классы `java.awt.Checkbox`). В силу того, что автомат A0 запускается по общему таймеру, чтобы не было постоянного протоколирования, необходимо отключить запуск и завершение работы автоматов, а также опрос входных переменных.

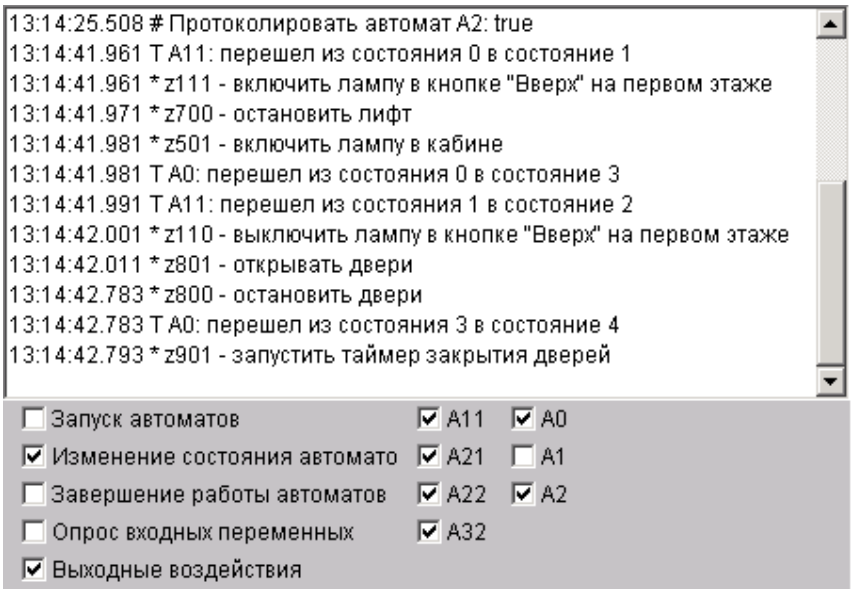

Рис. 16. Протоколирование

Фрагменты протоколов приведены в разд. 10.

7. Класс “ElevatorVisualizer”

7.1. Словесное описание

Класс реализует интерфейс *ElevatorVisualizerInterface* и является основным классом эмуляции лифта и его визуализации.

Класс реализует также библиотечный интерфейс `java.awt.event.MouseListener`. Метод `public void mousePressed( MouseEvent e )` обрабатывает нажатия кнопок мыши при размещении курсора на области визуализации и добавляет события в очередь, реализованную классом `java.util.Vector`.

Кроме этого, класс реализует библиотечный интерфейс `java.lang.Runnable`. Метод `public void run()` содержит “бесконечный” цикл, который в случае наличия в очереди события запускает с этим событием соответствующий автомат или изменяет внутренние переменные экземпляра класса, например булеву переменную, определяющую наличие в лифте груза. Затем эмулируется один “такт” работы лифта и в случае необходимости вызываются автоматы с соответствующими событиями. После этого вызывается автомат A0 с событием e0 (срабатывание общего таймера) и перерисовывается область визуализации.

Реализует функции входных переменных и выходных воздействий.

7.2. “Последствия” нажатия кнопок

Нажатие на кнопки обрабатывается в методе `public void mousePressed( MouseEvent e )` и добавляет в очередь события:

- 11 – кнопка “Вверх” на первом этаже;
- 21 – кнопка “Вверх” на втором этаже;
- 22 – кнопка “Вниз” на втором этаже;
- 32 – кнопка “Вниз” на третьем этаже;
- 41 – кнопка “1” в кабине лифта;
- 42 – кнопка “2” в кабине лифта;
- 43 – кнопка “3” в кабине лифта;
- 44 – кнопка “S” в кабине лифта;
- 50 – выйти из кабины (стрелка вправо);
- 51 – войти в кабину (стрелка влево).

На рис. 17 показан визуализатор работы лифта. Кнопки, расположенные рядом с лифтом, позволяют вызывать лифт на соответствующий этаж. Лампа визуализатора соответствует лампе в кабине лифта. Она не горит, если двери закрыты и в кабине никого нет. В кабине лифта также имеется панель управления. Таймер показывает, сколько времени осталось до закрытия дверей, а указатель – направление возможного перемещения человека (из лифта и в лифт).

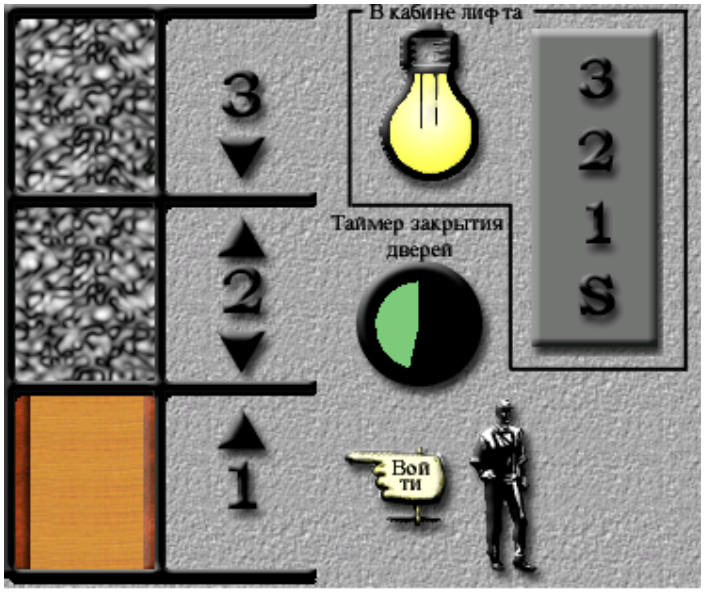

Рис. 17. Визуализация работы лифта

8. Класс “ElevatorApplet”

8.1. Словесное описание

Класс наследуется от библиотечного класса `java.applet.Applet` и переопределяет поведение методов `public void init()` и `public void destroy()`.

В первом методе создаются экземпляры классов *ElevatorLog* (ведение протокола) – переменная *log* и *ElevatorVisualizer* (эмуляция и визуализация лифта) – переменная *vis*, которые наследуются от библиотечного класса `java.awt.Canvas`. Они размещаются в области аплета.

В основном потоке выполнения аплета создается дополнительный поток, в котором запускается экземпляр класса *ElevatorVisualizer*.

Во втором методе дополнительный поток останавливается.

Определен метод `public int getParamInt( String param, int def )`, который пытается получить параметр аплета `param`, привести его к целому числу и возвращает результат. В случае неудачи возвращается значение по умолчанию – `def`.

8.2. Возможные параметры аплета

Классом *ElevatorLog* используется параметр:

- LOG_HEIGHT – высота области протоколирования в пикселях. Значение по умолчанию: 410.

Классом *ElevatorVisualizer* используются параметры:

- DELAY – задержка между “тактами” эмуляции системы в миллисекундах. Значение по умолчанию: 50.
- DOORS_TIMER_MAX – количество “тактов” ожидания перед закрытием дверей. Значение по умолчанию: 30.
- DOORS_MOVE – перемещение дверей лифта при открытии/закрытии за один “такт” в пикселях. Значение по умолчанию: 2.
- DOORS_MAX – максимальное перемещение дверей лифта при открытии. Оно должно быть кратно DOORS_MOVE и находиться в диапазоне 20-40. Значение по умолчанию: 30.
- ELEVATOR_MOVE – перемещение кабины лифта при движении вверх/вниз за один “такт” в пикселях. Оно должно делить нацело число 100. Значение по умолчанию: 2.

9. Листинги программ

9.1. ElevatorAutomats.java

```
/**
 * Инкапсулирует автоматы управления лифтом
 */
public class ElevatorAutomats
{
  protected ElevatorVisualizerInterface vis;
  protected ElevatorLogInterface log;

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

  protected int y0 = 0;
  /**
  * Автомат 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, "неизвестное состояние" );
}

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;
/**
* Автомат 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;
/**
* Автомат 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, "неизвестное состояние" );
  }

  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;
/**
* Автомат 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;
/**
* Автомат 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;
/**
* Автомат 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;
  /**
  * Автомат 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

```
/**
 * Интерфейс класса, ведущего протокол
 */
public interface ElevatorLogInterface
{
    /**
    * Начало работы автомата
    */
    public void begin( int aut, int state, int event );
    /**
    * Переход между состояниями
    */
    public void trans( int aut, int old_state, int new_state );
    /**
    * Окончание работы автомата
    */
    public void end( int aut, int state, int event );
    /**
    * Опрос входной переменной
    */
    public void input( int num, String str, boolean ret );
    /**
    * Выполнение действия
    */
    public void action( int num, String str );
    /**
    * Сообщение об ошибке
    */
    public void error( int aut, String err );
    /**
    * Вывод строки в протокол
    */
    public void log( String str );
}
```

9.3. ElevatorVisualizerInterface.java

```
/**
 * Интерфейс эмулятора и визуализатора работы лифта
 * Определяет все входные переменные и
 * выходные воздействия, используемые в задаче
 */
public interface ElevatorVisualizerInterface
{
    /**
     * Входная переменная: в лифте есть груз
     */
    public boolean x51();

    /**
     * Входная переменная: лифт на 1 этаже
     */
    public boolean x61();

    /**
     * Входная переменная: лифт на 2 этаже
     */
    public boolean x62();
```

```
/**
 * Входная переменная: лифт на 3 этаже
 */
public boolean x63();

/**
 * Выходное воздействие:
 * выключить лампу в кнопке “Вверх” на первом этаже
 */
public void z110();

/**
 * Выходное воздействие:
 * включить лампу в кнопке “Вверх” на первом этаже
 */
public void z111();

/**
 * Выходное воздействие:
 * выключить лампу в кнопке “Вверх” на втором этаже
 */
public void z210();

/**
 * Выходное воздействие:
 * включить лампу в кнопке “Вверх” на втором этаже
 */
public void z211();

/**
 * Выходное воздействие:
 * выключить лампу в кнопке “Вниз” на втором этаже
 */
public void z220();

/**
 * Выходное воздействие:
 * включить лампу в кнопке “Вниз” на втором этаже
 */
public void z221();

/**
 * Выходное воздействие:
 * выключить лампу в кнопке “Вниз” на третьем этаже
 */
public void z320();

/**
 * Выходное воздействие:
 * включить лампу в кнопке “Вниз” на третьем этаже
 */
public void z321();

/**
 * Выходное воздействие: выключить лампу в кнопке
 */
public void z400();

/**
 * Выходное воздействие: включить лампу в кнопке “1”
 */
public void z411();
```

```
/**
 * Выходное воздействие: включить лампу в кнопке “2”
 */
public void z421();

/**
 * Выходное воздействие: включить лампу в кнопке “3”
 */
public void z431();

/**
 * Выходное воздействие: включить лампу в кнопке “S”
 */
public void z441();

/**
 * Выходное воздействие: выключить лампу в кабине
 */
public void z500();

/**
 * Выходное воздействие: включить лампу в кабине
 */
public void z501();

/**
 * Выходное воздействие: остановить лифт
 */
public void z700();

/**
 * Выходное воздействие: поехать вверх
 */
public void z701();

/**
 * Выходное воздействие: поехать вниз
 */
public void z702();

/**
 * Выходное воздействие: автоматически остановить двери
 */
public void z800();

/**
 * Выходное воздействие: открывать двери
 */
public void z801();

/**
 * Выходное воздействие: закрывать двери
 */
public void z802();

/**
 * Выходное воздействие: выключить таймер закрытия дверей
 */
public void z900();
```

```
    /**
     * Выходное воздействие: запустить таймер закрытия дверей
     */
    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;

/**
 * Реализация класса, ведущего протокол
 */
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( "Запуск автоматов" );
        chLogBegin.setBounds( 10, HEIGHT-100, 200, 20 );
        chLogBegin.addItemListener( this );
        add( chLogBegin );
        chLogTrans = new Checkbox( "Изменение состояния автоматов" );
        chLogTrans.setBounds( 10, HEIGHT-80, 200, 20 );
        chLogTrans.addItemListener( this );
        add( chLogTrans );
        chLogEnd = new Checkbox( "Завершение работы автоматов" );
        chLogEnd.setBounds( 10, HEIGHT-60, 200, 20 );
        chLogEnd.addItemListener( this );
        add( chLogEnd );
        chLogInput = new Checkbox( "Опрос входных переменных" );
        chLogInput.setBounds( 10, HEIGHT-40, 200, 20 );
```

```
    chLogInput.addItemListener( this );
    add( chLogInput );
    chLogAction = new Checkbox( "Выходные воздействия" );
    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();
}

/**
 * Обработка внешнего события “Управление флажком”
 */
public void itemStateChanged( ItemEvent e )
{
    boolean enable = ((Checkbox)e.getSource()).getState();
    if ( e.getSource().equals( chLogBegin ) )
    {
        log( "Протоколировать запуск автоматов: " +
             enable );
        logBegin = enable;
    }
    else if ( e.getSource().equals( chLogTrans ) )
    {
        log( "Протоколировать изменение состояния автоматов: " +
             enable );
        logTrans = enable;
    }
    else if ( e.getSource().equals( chLogEnd ) )
    {
        log( "Протоколировать завершение работы автоматов: " +
             enable );
        logEnd = enable;
    }
```

```
    else if ( e.getSource().equals( chLogInput ) )
    {
        log( "Протоколировать опрос входных переменных: " +
             enable );
        logInput = enable;
    }
    else if ( e.getSource().equals( chLogAction ) )
    {
        log( "Протоколировать выходные воздействия: " +
             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 );
}

/**
 * Сообщение об ошибке
 */
public void error( int aut, String err )
{
    write( "! A" + aut + ": ОШИБКА: " + err );
}

/**
 * Начало работы автомата
 */
public void begin( int aut, int state, int event )
{
    if ( logBegin )
        aut( aut, "{ A" + aut + ": в состоянии " + state +
                  " запущен с событием e" + event );
}

/**
 * Переход между состояниями
 */
public void trans( int aut, int old_state, int new_state )
{
    if ( logTrans )
        aut( aut, "T A" + aut + ": перешел из состояния " +
                  old_state + " в состояние " + new_state );
}
```

```
/**
 * Окончание работы автомата
 */
public void end( int aut, int state, int event )
{
    if ( logEnd )
        aut( aut, "} A" + aut + ": завершил обработку события e" +
                  event + " в состоянии " + state );
}

/**
 * Опрос входной переменной
 */
public void input( int num, String str, boolean ret )
{
    if ( logInput )
        write( "> x" + num + " - " + str + " - вернул " + ret );
}

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

/**
 * Вывод строки в протокол
 */
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( "Протоколировать автомат A" + aut + ": " + 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;

/**
 * Реализация эмулятора и визуализатора лифта
 */
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 );

    //Загрузка всех графических фрагментов
    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 )
    {
    }
}

/**
 * Обработка событий “Клики мыши” и добавление их в очередь
 */
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 );
}

/**
 * Формирование области визуализации
 */
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 );
}

/**
 * Основной цикл работы визуализатора
 */
public void run()
{
    while ( true )
    {
        //Выборка события из очереди
        int ev = getNextEvent();

        //Обработка выбранного события
        switch ( ev )
        {
        case 11:
            aut.A11( 11 ); //Вызов автомата A11 с событием 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;
}

//Вызовы автоматов по флагам, формируемым лифтом
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;
        }
    }
}

/**
 * Добавить событие в очередь
 */
synchronized public void addEvent( int ev )
{
    events.addElement( new Integer( ev ) );
}

/**
 * Извлечь событие из очереди
 */
synchronized protected int getNextEvent()
{
    int ev = 0;
    if ( !events.isEmpty() )
    {
        ev = ( (Integer)events.firstElement() ).intValue();
        events.removeElementAt( 0 );
    }
    return ev;
}

/**
 * Входная переменная: в лифте есть груз
 */
public boolean x51()
{
    log.input( 51, "в лифте есть груз", xx51 );
    return xx51;
}

/**
 * Входная переменная: лифт на 1 этаже
 */
public boolean x61()
{
    boolean ret = elevatorPosition == ELEVATOR_1_FLOOR;
    log.input( 61, "лифт на 1 этаже", ret );
    return ret;
}

/**
 * Входная переменная: лифт на 2 этаже
 */
public boolean x62()
{
    boolean ret = elevatorPosition == ELEVATOR_2_FLOOR;
    log.input( 62, "лифт на 2 этаже", ret );
    return ret;
}
```

```
/**
 * Входная переменная: лифт на 3 этаже
 */
public boolean x63()
{
    boolean ret = elevatorPosition == ELEVATOR_3_FLOOR;
    log.input( 63, "лифт на 3 этаже", ret );
    return ret;
}

/**
 * Выходное воздействие:
 * выключить лампу в кнопке "Вверх" на первом этаже
 */
public void z110()
{
    log.action( 110,
        "выключить лампу в кнопке \"Вверх\" на первом этаже" );
    zz11 = false;
}

/**
 * Выходное воздействие:
 * включить лампу в кнопке "Вверх" на первом этаже
 */
public void z111()
{
    log.action( 111,
        "включить лампу в кнопке \"Вверх\" на первом этаже" );
    zz11 = true;
}

/**
 * Выходное воздействие:
 * выключить лампу в кнопке "Вверх" на втором этаже
 */
public void z210()
{
    log.action( 210,
        "выключить лампу в кнопке \"Вверх\" на втором этаже" );
    zz21 = false;
}

/**
 * Выходное воздействие:
 * включить лампу в кнопке "Вверх" на втором этаже
 */
public void z211()
{
    log.action( 211,
        "включить лампу в кнопке \"Вверх\" на втором этаже" );
    zz21 = true;
}

/**
 * Выходное воздействие:
 * выключить лампу в кнопке "Вниз" на втором этаже
 */
public void z220()
{
    log.action( 220,
        "выключить лампу в кнопке \"Вниз\" на втором этаже" );
    zz22 = false;
}
```

```
/**
 * Выходное воздействие:
 * включить лампу в кнопке "Вниз" на втором этаже
 */
public void z221()
{
    log.action( 221,
        "включить лампу в кнопке \"Вниз\" на втором этаже" );
    zz22 = true;
}

/**
 * Выходное воздействие:
 * выключить лампу в кнопке "Вниз" на третьем этаже
 */
public void z320()
{
    log.action( 320,
        "выключить лампу в кнопке \"Вниз\" на третьем этаже" );
    zz32 = false;
}

/**
 * Выходное воздействие:
 * включить лампу в кнопке "Вниз" на третьем этаже
 */
public void z321()
{
    log.action( 321,
        "включить лампу в кнопке \"Вниз\" на третьем этаже" );
    zz32 = true;
}

/**
 * Выходное воздействие: выключить лампу в кнопке "1"
 */
public void z400()
{
    log.action( 400, "выключить лампу в кнопке\"1\"" );
    panelState = 0;
}

/**
 * Выходное воздействие: включить лампу в кнопке "1"
 */
public void z411()
{
    log.action( 411, "включить лампу в кнопке \"1\"" );
    panelState = 1;
}

/**
 * Выходное воздействие: включить лампу в кнопке "2"
 */
public void z421()
{
    log.action( 421, "включить лампу в кнопке \"2\"" );
    panelState = 2;
}
```

```
/**
 * Выходное воздействие: включить лампу в кнопке "3"
 */
public void z431()
{
    log.action( 431, "включить лампу в кнопке \"3\"" );
    panelState = 3;
}

/**
 * Выходное воздействие: включить лампу в кнопке "S"
 */
public void z441()
{
    log.action( 441, "включить лампу в кнопке \"S\"" );
    panelState = 4;
}

/**
 * Выходное воздействие: выключить лампу в кабине
 */
public void z500()
{
    log.action( 500, "выключить лампу в кабине" );
    isLampOn = false;
}

/**
 * Выходное воздействие: включить лампу в кабине
 */
public void z501()
{
    log.action( 501, "включить лампу в кабине" );
    isLampOn = true;
}

/**
 * Выходное воздействие: остановить лифт
 */
public void z700()
{
    log.action( 700, "остановить лифт" );
    elevatorMove = 0;
}

/**
 * Выходное воздействие: поехать вверх
 */
public void z701()
{
    log.action( 701, "поехать вверх" );
    elevatorMove = ELEVATOR_MOVE;
}

/**
 * Выходное воздействие: поехать вниз
 */
public void z702()
{
    log.action( 702, "поехать вниз" );
    elevatorMove = -ELEVATOR_MOVE;
}
```

```
    /**
     * Выходное воздействие: автоматически остановить двери
     */
    public void z800()
    {
        log.action( 800, "автоматически остановить двери" );
        doorsMove = 0;
    }

    /**
     * Выходное воздействие: открывать двери
     */
    public void z801()
    {
        log.action( 801, "открывать двери" );
        doorsMove = DOORS_MOVE;
    }

    /**
     * Выходное воздействие: закрывать двери
     */
    public void z802()
    {
        log.action( 802, "закрывать двери" );
        doorsMove = -DOORS_MOVE;
    }

    /**
     * Выходное воздействие: выключить таймер закрытия дверей
     */
    public void z900()
    {
        log.action( 900, "выключить таймер закрытия дверей" );
        doorsTimer = -1;
    }

    /**
     * Выходное воздействие: запустить таймер закрытия дверей
     */
    public void z901()
    {
        log.action( 901, "запустить таймер закрытия дверей" );
        doorsTimer = DOORS_TIMER_MAX;
    }
}
```

9.6. ElevatorApplet.java

```
import java.applet.Applet;

/**
 * Аплет
 */
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

```
<!-- Html-файл для запуска аплета -->
<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. Фрагменты протокола

В протоколах используются следующие обозначения:

#	–	клик на флажках управления протоколированием;
T	–	переход автомата ;
*	–	выходные воздействия;
{	–	начало работы автомата;
}	–	окончание работы автомата.
>	–	входные переменные.

В приведенных ниже протоколах для состояний указаны только их номера, в то время как названия состояний с целью сокращения размеров протоколов не указываются.

Каждый протокол отражает следующий сценарий:

- на первом этаже по нажатию кнопки вызывается лифт;
- в момент открытия дверей загорается лампа в кабине;
- в открывающиеся двери заходит человек;
- на панели в кабине нажимается кнопка “3”;
- после полного открытия дверей они снова закрываются и лифт движется вверх;
- на втором этаже нажимается кнопка “Вниз” и в ней загорается лампа;
- лифт проходит мимо второго этажа и доезжает до третьего;
- на третьем этаже открываются двери и выходит человек;
- запускается таймер автоматического закрытия дверей;
- после срабатывания таймера двери закрываются;
- в момент закрытия дверей выключается лампа в кабине;
- лифт идет на второй этаж;
- на втором этаже двери открываются, включается лампа в кабине и выключается лампа в кнопке “Вниз”;
- запускается таймер автоматического закрытия дверей;
- после срабатывания таймера двери закрываются;
- в момент закрытия дверей выключается лампа в кабине.

10.1. Пример упрощенного протокола

```
20:36:17.354 # Протоколировать изменение состояния автоматов: true
20:36:18.145 # Протоколировать выходные воздействия: true
20:36:18.876 # Протоколировать автомат A11: true
20:36:19.587 # Протоколировать автомат A21: true
20:36:19.878 # Протоколировать автомат A22: true
20:36:20.478 # Протоколировать автомат A32: true
20:36:21.189 # Протоколировать автомат A0: true
20:36:21.530 # Протоколировать автомат A1: true
20:36:21.930 # Протоколировать автомат A2: true
20:36:23.933 T A11: перешел из состояния 0 в состояние 1
20:36:23.933 * z111 - включить лампу в кнопке "Вверх" на первом этаже
20:36:23.933 * z700 - остановить лифт
20:36:23.943 * z501 - включить лампу в кабине
20:36:23.943 T A0: перешел из состояния 0 в состояние 3
20:36:23.943 T A11: перешел из состояния 1 в состояние 2
20:36:23.953 * z110 - выключить лампу в кнопке "Вверх" на первом этаже
20:36:23.953 * z801 - открывать двери
20:36:24.724 * z800 - автоматически остановить двери
20:36:24.724 T A0: перешел из состояния 3 в состояние 4
20:36:24.724 * z901 - запустить таймер закрытия дверей
20:36:24.734 * z900 - выключить таймер закрытия дверей
20:36:24.734 T A0: перешел из состояния 4 в состояние 6
```

```
20:36:27.068 T A2: перешел из состояния 0 в состояние 3
20:36:27.068 * z431 - включить лампу в кнопке "3"
20:36:27.088 T A0: перешел из состояния 6 в состояние 7
20:36:27.098 * z802 - закрывать двери
20:36:27.869 * z800 - автоматически остановить двери
20:36:27.869 T A0: перешел из состояния 7 в состояние 9
20:36:27.889 T A11: перешел из состояния 2 в состояние 0
20:36:27.899 T A0: перешел из состояния 9 в состояние 10
20:36:27.909 T A1: перешел из состояния 0 в состояние 1
20:36:27.919 * z701 - поехать вверх
20:36:28.490 T A22: перешел из состояния 0 в состояние 1
20:36:28.490 * z221 - включить лампу в кнопке "Вниз" на втором этаже
20:36:32.986 * z700 - остановить лифт
20:36:32.986 T A0: перешел из состояния 10 в состояние 8
20:36:33.006 T A2: перешел из состояния 3 в состояние 0
20:36:33.016 * z400 - выключить лампу в кнопке
20:36:33.026 T A32: перешел из состояния 0 в состояние 2
20:36:33.036 * z320 - выключить лампу в кнопке "Вниз" на третьем этаже
20:36:33.056 * z801 - открывать двери
20:36:33.577 T A0: перешел из состояния 8 в состояние 3
20:36:33.587 * z801 - открывать двери
20:36:33.868 * z800 - автоматически остановить двери
20:36:33.868 T A0: перешел из состояния 3 в состояние 4
20:36:33.888 * z901 - запустить таймер закрытия дверей
20:36:35.420 T A0: перешел из состояния 4 в состояние 5
20:36:35.420 * z802 - закрывать двери
20:36:36.151 * z800 - автоматически остановить двери
20:36:36.151 T A0: перешел из состояния 5 в состояние 0
20:36:36.171 T A32: перешел из состояния 2 в состояние 0
20:36:36.181 * z500 - выключить лампу в кабине
20:36:36.191 T A0: перешел из состояния 0 в состояние 2
20:36:36.201 T A1: перешел из состояния 1 в состояние 0
20:36:36.221 * z702 - поехать вниз
20:36:38.745 * z700 - остановить лифт
20:36:38.755 * z501 - включить лампу в кабине
20:36:38.765 T A0: перешел из состояния 2 в состояние 3
20:36:38.775 T A21: перешел из состояния 0 в состояние 2
20:36:38.795 * z210 - выключить лампу в кнопке "Вверх" на втором этаже
20:36:38.805 T A22: перешел из состояния 1 в состояние 2
20:36:38.825 * z220 - выключить лампу в кнопке "Вниз" на втором этаже
20:36:38.835 * z801 - открывать двери
20:36:39.616 * z800 - автоматически остановить двери
20:36:39.626 T A0: перешел из состояния 3 в состояние 4
20:36:39.646 * z901 - запустить таймер закрытия дверей
20:36:41.168 T A0: перешел из состояния 4 в состояние 5
20:36:41.178 * z802 - закрывать двери
20:36:41.909 * z800 - автоматически остановить двери
20:36:41.919 T A0: перешел из состояния 5 в состояние 0
20:36:41.939 T A21: перешел из состояния 2 в состояние 0
20:36:41.949 T A22: перешел из состояния 2 в состояние 0
20:36:41.969 * z500 - выключить лампу в кабине
```

10.2. Пример полного протокола

Повторяющиеся в точности до времени фрагменты протокола вручную заменены многоточием.

```
20:37:46.091 # Протоколировать запуск автоматов: true
20:37:46.402 # Протоколировать изменение состояния автоматов: true
20:37:46.722 # Протоколировать завершение работы автоматов: true
20:37:47.053 # Протоколировать опрос входных переменных: true
20:37:47.343 # Протоколировать выходные воздействия: true
20:37:47.994 # Протоколировать автомат A11: true
```

```
20:37:48.285 # Протоколировать автомат A21: true
20:37:48.665 # Протоколировать автомат A22: true
20:37:48.956 # Протоколировать автомат A32: true
20:37:49.587 # Протоколировать автомат A0: true
20:37:49.847 # Протоколировать автомат A1: true
20:37:50.197 # Протоколировать автомат A2: true
20:37:50.838 { A0: в состоянии 0 запущен с событием e0
20:37:50.838 > x61 - лифт на 1 этаже - вернул true
20:37:50.848 > x62 - лифт на 2 этаже - вернул false
20:37:50.848 > x63 - лифт на 3 этаже - вернул false
20:37:50.848 } A0: завершил обработку события e0 в состоянии 0
...
20:37:51.619 { A0: в состоянии 0 запущен с событием e0
20:37:51.629 > x61 - лифт на 1 этаже - вернул true
20:37:51.639 > x62 - лифт на 2 этаже - вернул false
20:37:51.649 > x63 - лифт на 3 этаже - вернул false
20:37:51.669 } A0: завершил обработку события e0 в состоянии 0
20:37:51.730 { A11: в состоянии 0 запущен с событием e11
20:37:51.740 T A11: перешел из состояния 0 в состояние 1
20:37:51.750 * zz111 - включить лампу в кнопке "Вверх" на первом этаже
20:37:51.770 } A11: завершил обработку события e11 в состоянии 1
20:37:51.780 { A0: в состоянии 0 запущен с событием e0
20:37:51.800 > x61 - лифт на 1 этаже - вернул true
20:37:51.810 > x62 - лифт на 2 этаже - вернул false
20:37:51.820 > x63 - лифт на 3 этаже - вернул false
20:37:51.840 * z700 - остановить лифт
20:37:51.850 * z501 - включить лампу в кабине
20:37:51.870 T A0: перешел из состояния 0 в состояние 3
20:37:51.880 { A2: в состоянии 0 запущен с событием e2
20:37:51.900 > x51 - в лифте есть груз - вернул false
20:37:51.920 } A2: завершил обработку события e2 в состоянии 0
20:37:51.930 { A11: в состоянии 1 запущен с событием e3
20:37:51.950 > x61 - лифт на 1 этаже - вернул true
20:37:51.960 T A11: перешел из состояния 1 в состояние 2
20:37:51.980 * z110 - выключить лампу в кнопке "Вверх" на первом этаже
20:37:52.000 } A11: завершил обработку события e3 в состоянии 2
20:37:52.010 { A21: в состоянии 0 запущен с событием e3
20:37:52.030 > x62 - лифт на 2 этаже - вернул false
20:37:52.040 } A21: завершил обработку события e3 в состоянии 0
20:37:52.060 { A22: в состоянии 0 запущен с событием e3
20:37:52.080 > x62 - лифт на 2 этаже - вернул false
20:37:52.090 } A22: завершил обработку события e3 в состоянии 0
20:37:52.110 { A32: в состоянии 0 запущен с событием e3
20:37:52.130 > x63 - лифт на 3 этаже - вернул false
20:37:52.140 } A32: завершил обработку события e3 в состоянии 0
20:37:52.160 * z801 - открывать двери
20:37:52.180 } A0: завершил обработку события e0 в состоянии 3
20:37:52.240 { A0: в состоянии 3 запущен с событием e0
20:37:52.250 } A0: завершил обработку события e0 в состоянии 3
...
20:37:53.382 { A0: в состоянии 3 запущен с событием e0
20:37:53.402 } A0: завершил обработку события e0 в состоянии 3
20:37:53.472 { A0: в состоянии 3 запущен с событием e80
20:37:53.482 * z800 - автоматически остановить двери
20:37:53.502 T A0: перешел из состояния 3 в состояние 4
20:37:53.532 * z901 - запустить таймер закрытия дверей
20:37:53.552 } A0: завершил обработку события e80 в состоянии 4
20:37:53.572 { A0: в состоянии 4 запущен с событием e0
20:37:53.592 > x51 - в лифте есть груз - вернул true
20:37:53.612 * z900 - выключить таймер закрытия дверей
20:37:53.632 T A0: перешел из состояния 4 в состояние 6
20:37:53.662 } A0: завершил обработку события e0 в состоянии 6
20:37:53.742 { A0: в состоянии 6 запущен с событием e0
20:37:53.752 > x51 - в лифте есть груз - вернул true
```

```
20:37:53.783 } A0: завершил обработку события e0 в состоянии 6
...
20:37:54.223 { A0: в состоянии 6 запущен с событием e0
20:37:54.233 > x51 - в лифте есть груз - вернул true
20:37:54.263 } A0: завершил обработку события e0 в состоянии 6
20:37:54.343 { A2: в состоянии 0 запущен с событием e43
20:37:54.353 > x51 - в лифте есть груз - вернул true
20:37:54.383 > x63 - лифт на 3 этаже - вернул false
20:37:54.413 T A2: перешел из состояния 0 в состояние 3
20:37:54.433 * z431 - включить лампу в кнопке "3"
20:37:54.454 } A2: завершил обработку события e43 в состоянии 3
20:37:54.484 { A0: в состоянии 6 запущен с событием e0
20:37:54.504 T A0: перешел из состояния 6 в состояние 7
20:37:54.544 * z802 - закрывать двери
20:37:54.584 } A0: завершил обработку события e0 в состоянии 7
20:37:54.674 { A0: в состоянии 7 запущен с событием e0
20:37:54.694 > x51 - в лифте есть груз - вернул true
20:37:54.724 } A0: завершил обработку события e0 в состоянии 7
...
20:37:56.296 { A0: в состоянии 7 запущен с событием e0
20:37:56.316 > x51 - в лифте есть груз - вернул true
20:37:56.346 } A0: завершил обработку события e0 в состоянии 7
20:37:56.436 { A22: в состоянии 0 запущен с событием e22
20:37:56.456 T A22: перешел из состояния 0 в состояние 1
20:37:56.486 * z221 - включить лампу в кнопке "Вниз" на втором этаже
20:37:56.516 } A22: завершил обработку события e22 в состоянии 1
20:37:56.547 { A0: в состоянии 7 запущен с событием e0
20:37:56.577 > x51 - в лифте есть груз - вернул true
20:37:56.607 } A0: завершил обработку события e0 в состоянии 7
20:37:56.697 { A0: в состоянии 7 запущен с событием e81
20:37:56.717 * z800 - автоматически остановить двери
20:37:56.747 T A0: перешел из состояния 7 в состояние 9
20:37:56.777 { A11: в состоянии 2 запущен с событием e4
20:37:56.817 T A11: перешел из состояния 2 в состояние 0
20:37:56.857 } A11: завершил обработку события e4 в состоянии 0
20:37:56.897 { A21: в состоянии 0 запущен с событием e4
20:37:56.937 } A21: завершил обработку события e4 в состоянии 0
20:37:56.977 { A22: в состоянии 1 запущен с событием e4
20:37:57.007 } A22: завершил обработку события e4 в состоянии 1
20:37:57.047 { A32: в состоянии 0 запущен с событием e4
20:37:57.077 } A32: завершил обработку события e4 в состоянии 0
20:37:57.107 } A0: завершил обработку события e81 в состоянии 9
20:37:57.137 { A0: в состоянии 9 запущен с событием e0
20:37:57.167 > x61 - лифт на 1 этаже - вернул true
20:37:57.197 > x62 - лифт на 2 этаже - вернул false
20:37:57.227 > x63 - лифт на 3 этаже - вернул false
20:37:57.258 T A0: перешел из состояния 9 в состояние 10
20:37:57.288 { A1: в состоянии 0 запущен с событием e9
20:37:57.318 T A1: перешел из состояния 0 в состояние 1
20:37:57.348 } A1: завершил обработку события e9 в состоянии 1
20:37:57.388 * z701 - поехать вверх
20:37:57.418 } A0: завершил обработку события e0 в состоянии 10
20:37:57.508 { A0: в состоянии 10 запущен с событием e0
20:37:57.528 > x61 - лифт на 1 этаже - вернул false
20:37:57.568 > x62 - лифт на 2 этаже - вернул false
20:37:57.608 > x63 - лифт на 3 этаже - вернул false
20:37:57.658 } A0: завершил обработку события e0 в состоянии 10
...
20:38:33.410 { A0: в состоянии 10 запущен с событием e0
20:38:33.490 > x61 - лифт на 1 этаже - вернул false
20:38:33.590 > x62 - лифт на 2 этаже - вернул false
20:38:33.680 > x63 - лифт на 3 этаже - вернул false
20:38:33.770 } A0: завершил обработку события e0 в состоянии 10
20:38:33.900 { A0: в состоянии 10 запущен с событием e60
```

```
20:38:33.980 > x61 - лифт на 1 этаже - вернул false
20:38:34.060 > x62 - лифт на 2 этаже - вернул false
20:38:34.151 > x63 - лифт на 3 этаже - вернул true
20:38:34.271 * z700 - остановить лифт
20:38:34.381 T A0: перешел из состояния 10 в состояние 8
20:38:34.461 { A2: в состоянии 1 запущен с событием e2
20:38:34.551 T A2: перешел из состояния 3 в состояние 0
20:38:34.641 * z400 - выключить лампу в кнопке
20:38:34.721 } A2: завершил обработку события e2 в состоянии 0
20:38:34.812 { A11: в состоянии 0 запущен с событием e3
20:38:34.912 > x61 - лифт на 1 этаже - вернул false
20:38:35.012 } A11: завершил обработку события e3 в состоянии 0
20:38:35.122 { A21: в состоянии 0 запущен с событием e3
20:38:35.212 > x62 - лифт на 2 этаже - вернул false
20:38:35.292 } A21: завершил обработку события e3 в состоянии 0
20:38:35.382 { A22: в состоянии 1 запущен с событием e3
20:38:35.472 > x62 - лифт на 2 этаже - вернул false
20:38:35.563 } A22: завершил обработку события e3 в состоянии 1
20:38:35.643 { A32: в состоянии 0 запущен с событием e3
20:38:35.773 > x63 - лифт на 3 этаже - вернул true
20:38:35.873 T A32: перешел из состояния 0 в состояние 2
20:38:35.963 * z320 - выключить лампу в кнопке "Вниз" на третьем этаже
20:38:36.053 } A32: завершил обработку события e3 в состоянии 2
20:38:36.143 * z801 - открывать двери
20:38:36.224 } A0: завершил обработку события e60 в состоянии 8
20:38:36.314 { A0: в состоянии 8 запущен с событием e0
20:38:36.404 > x51 - в лифте есть груз - вернул true
20:38:36.514 } A0: завершил обработку события e0 в состоянии 8
...
20:38:38.697 { A0: в состоянии 8 запущен с событием e0
20:38:38.787 > x51 - в лифте есть груз - вернул true
20:38:38.897 } A0: завершил обработку события e0 в состоянии 8
20:38:39.078 { A0: в состоянии 8 запущен с событием e0
20:38:39.168 > x51 - в лифте есть груз - вернул false
20:38:39.258 T A0: перешел из состояния 8 в состояние 3
20:38:39.348 { A2: в состоянии 1 запущен с событием e2
20:38:39.438 > x51 - в лифте есть груз - вернул false
20:38:39.528 } A2: завершил обработку события e2 в состоянии 0
20:38:39.648 { A11: в состоянии 0 запущен с событием e3
20:38:39.779 > x61 - лифт на 1 этаже - вернул false
20:38:39.869 } A11: завершил обработку события e3 в состоянии 0
20:38:39.959 { A21: в состоянии 0 запущен с событием e3
20:38:40.049 > x62 - лифт на 2 этаже - вернул false
20:38:40.149 } A21: завершил обработку события e3 в состоянии 0
20:38:40.239 { A22: в состоянии 1 запущен с событием e3
20:38:40.329 > x62 - лифт на 2 этаже - вернул false
20:38:40.470 } A22: завершил обработку события e3 в состоянии 1
20:38:40.580 { A32: в состоянии 2 запущен с событием e3
20:38:40.680 } A32: завершил обработку события e3 в состоянии 2
20:38:40.770 * z801 - открывать двери
20:38:40.870 } A0: завершил обработку события e0 в состоянии 3
20:38:41.020 { A0: в состоянии 3 запущен с событием e0
...
20:38:42.382 } A0: завершил обработку события e0 в состоянии 3
20:38:42.533 { A0: в состоянии 3 запущен с событием e80
20:38:42.623 * z800 - автоматически остановить двери
20:38:42.713 T A0: перешел из состояния 3 в состояние 4
20:38:42.823 * z901 - запустить таймер закрытия дверей
20:38:42.953 } A0: завершил обработку события e80 в состоянии 4
20:38:43.053 { A0: в состоянии 4 запущен с событием e0
20:38:43.144 > x51 - в лифте есть груз - вернул false
20:38:43.244 } A0: завершил обработку события e0 в состоянии 4
...
20:38:55.692 { A0: в состоянии 4 запущен с событием e0
```

```
20:38:55.822 > x51 - в лифте есть груз - вернул false
20:38:55.942 } A0: завершил обработку события e0 в состоянии 4
20:38:56.102 { A0: в состоянии 4 запущен с событием e90
20:38:56.212 T A0: перешел из состояния 4 в состояние 5
20:38:56.423 * z802 - закрывать двери
20:38:56.543 } A0: завершил обработку события e90 в состоянии 5
20:38:56.653 { A0: в состоянии 5 запущен с событием e0
20:38:56.773 > x51 - в лифте есть груз - вернул false
20:38:56.883 } A0: завершил обработку события e0 в состоянии 5
...
20:39:02.261 { A0: в состоянии 5 запущен с событием e0
20:39:02.371 > x51 - в лифте есть груз - вернул false
20:39:02.501 } A0: завершил обработку события e0 в состоянии 5
20:39:02.652 { A0: в состоянии 5 запущен с событием e81
20:39:02.762 * z800 - автоматически остановить двери
20:39:02.872 T A0: перешел из состояния 5 в состояние 0
20:39:02.982 { A11: в состоянии 0 запущен с событием e4
20:39:03.092 } A11: завершил обработку события e4 в состоянии 0
20:39:03.202 { A21: в состоянии 0 запущен с событием e4
20:39:03.323 } A21: завершил обработку события e4 в состоянии 0
20:39:03.443 { A22: в состоянии 1 запущен с событием e4
20:39:03.553 } A22: завершил обработку события e4 в состоянии 1
20:39:03.663 { A32: в состоянии 2 запущен с событием e4
20:39:03.783 T A32: перешел из состояния 2 в состояние 0
20:39:03.893 } A32: завершил обработку события e4 в состоянии 0
20:39:04.014 * z500 - выключить лампу в кабине
20:39:04.124 } A0: завершил обработку события e81 в состоянии 0
20:39:04.244 { A0: в состоянии 0 запущен с событием e0
20:39:04.354 > x61 - лифт на 1 этаже - вернул false
20:39:04.464 > x62 - лифт на 2 этаже - вернул false
20:39:04.574 > x63 - лифт на 3 этаже - вернул true
20:39:04.684 T A0: перешел из состояния 0 в состояние 2
20:39:04.795 { A1: в состоянии 1 запущен с событием e8
20:39:04.905 T A1: перешел из состояния 1 в состояние 0
20:39:05.015 } A1: завершил обработку события e8 в состоянии 0
20:39:05.125 * z702 - поехать вниз
20:39:05.235 } A0: завершил обработку события e0 в состоянии 2
20:39:05.396 { A0: в состоянии 2 запущен с событием e0
20:39:05.496 > x61 - лифт на 1 этаже - вернул false
20:39:05.606 > x62 - лифт на 2 этаже - вернул false
20:39:05.726 > x63 - лифт на 3 этаже - вернул false
20:39:05.836 } A0: завершил обработку события e0 в состоянии 2
...
20:39:38.303 { A0: в состоянии 2 запущен с событием e0
20:39:38.433 > x61 - лифт на 1 этаже - вернул false
20:39:38.573 > x62 - лифт на 2 этаже - вернул false
20:39:38.713 > x63 - лифт на 3 этаже - вернул false
20:39:38.854 } A0: завершил обработку события e0 в состоянии 2
20:39:39.044 { A0: в состоянии 2 запущен с событием e60
20:39:39.174 > x61 - лифт на 1 этаже - вернул false
20:39:39.314 > x62 - лифт на 2 этаже - вернул true
20:39:39.454 * z700 - остановить лифт
20:39:39.595 * z501 - включить лампу в кабине
20:39:39.735 T A0: перешел из состояния 2 в состояние 3
20:39:39.865 { A2: в состоянии 0 запущен с событием e2
20:39:40.005 > x51 - в лифте есть груз - вернул false
20:39:40.155 } A2: завершил обработку события e2 в состоянии 0
20:39:40.296 { A11: в состоянии 0 запущен с событием e3
20:39:40.436 > x61 - лифт на 1 этаже - вернул false
20:39:40.576 } A11: завершил обработку события e3 в состоянии 0
20:39:40.726 { A21: в состоянии 0 запущен с событием e3
20:39:40.877 > x62 - лифт на 2 этаже - вернул true
20:39:41.007 T A21: перешел из состояния 0 в состояние 2
20:39:41.157 * z210 - выключить лампу в кнопке "Вверх" на втором этаже
```

```
20:39:41.287 } A21: завершил обработку события e3 в состоянии 2
20:39:41.427 { A22: в состоянии 1 запущен с событием e3
20:39:41.568 > x62 - лифт на 2 этаже - вернул true
20:39:41.708 T A22: перешел из состояния 1 в состояние 2
20:39:41.848 * z220 - выключить лампу в кнопке "Вниз" на втором этаже
20:39:41.998 } A22: завершил обработку события e3 в состоянии 2
20:39:42.138 { A32: в состоянии 0 запущен с событием e3
20:39:42.279 > x63 - лифт на 3 этаже - вернул false
20:39:42.429 } A32: завершил обработку события e3 в состоянии 0
20:39:42.569 * z801 - открывать двери
20:39:42.719 } A0: завершил обработку события e60 в состоянии 3
20:39:42.859 { A0: в состоянии 3 запущен с событием e0
...
20:39:47.716 { A0: в состоянии 3 запущен с событием e0
20:39:47.867 } A0: завершил обработку события e0 в состоянии 3
20:39:48.067 { A0: в состоянии 3 запущен с событием e80
20:39:48.217 * z800 - автоматически остановить двери
20:39:48.367 T A0: перешел из состояния 3 в состояние 4
20:39:48.518 * z901 - запустить таймер закрытия дверей
20:39:48.658 } A0: завершил обработку события e80 в состоянии 4
20:39:48.808 { A0: в состоянии 4 запущен с событием e0
20:39:48.958 > x51 - в лифте есть груз - вернул false
20:39:49.108 } A0: завершил обработку события e0 в состоянии 4
...
20:40:03.769 { A0: в состоянии 4 запущен с событием e0
20:40:03.930 > x51 - в лифте есть груз - вернул false
20:40:04.080 } A0: завершил обработку события e0 в состоянии 4
20:40:04.300 { A0: в состоянии 4 запущен с событием e90
20:40:04.450 T A0: перешел из состояния 4 в состояние 5
20:40:04.631 * z802 - закрывать двери
20:40:04.791 } A0: завершил обработку события e90 в состоянии 5
20:40:04.951 { A0: в состоянии 5 запущен с событием e0
20:40:05.111 > x51 - в лифте есть груз - вернул false
20:40:05.272 } A0: завершил обработку события e0 в состоянии 5
...
20:40:12.021 { A0: в состоянии 5 запущен с событием e0
20:40:12.172 > x51 - в лифте есть груз - вернул false
20:40:12.332 } A0: завершил обработку события e0 в состоянии 5
20:40:12.562 { A0: в состоянии 5 запущен с событием e81
20:40:12.722 * z800 - автоматически остановить двери
20:40:12.893 T A0: перешел из состояния 5 в состояние 0
20:40:13.063 { A11: в состоянии 0 запущен с событием e4
20:40:13.233 } A11: завершил обработку события e4 в состоянии 0
20:40:13.393 { A21: в состоянии 2 запущен с событием e4
20:40:13.554 T A21: перешел из состояния 2 в состояние 0
20:40:13.714 } A21: завершил обработку события e4 в состоянии 0
20:40:13.884 { A22: в состоянии 2 запущен с событием e4
20:40:14.044 T A22: перешел из состояния 2 в состояние 0
20:40:14.204 } A22: завершил обработку события e4 в состоянии 0
20:40:14.365 { A32: в состоянии 0 запущен с событием e4
20:40:14.525 } A32: завершил обработку события e4 в состоянии 0
20:40:14.685 * z500 - выключить лампу в кабине
20:40:14.855 } A0: завершил обработку события e81 в состоянии 0
20:40:15.016 { A0: в состоянии 0 запущен с событием e0
20:40:15.176 > x61 - лифт на 1 этаже - вернул false
20:40:15.336 > x62 - лифт на 2 этаже - вернул true
20:40:15.506 > x63 - лифт на 3 этаже - вернул false
20:40:15.667 } A0: завершил обработку события e0 в состоянии 0
20:40:15.877 { A0: в состоянии 0 запущен с событием e0
20:40:16.027 > x61 - лифт на 1 этаже - вернул false
20:40:16.197 > x62 - лифт на 2 этаже - вернул true
20:40:16.368 > x63 - лифт на 3 этаже - вернул false
20:40:16.528 } A0: завершил обработку события e0 в состоянии 0
```