Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

Д.С. Белешко

# Система управления моделью аудиопроигрывателя

## Программирование с явным выделением состояний

Проектная документация

Проект создан в рамках «Движение за открытую проектную документацию»
http://is.ifmo.ru

Санкт-Петербург 2006

## Оглавление

## Введение

В данной работе, также как и в работе [3], с помощью *SWITCH*-технологии [1,2] реализована система управления моделью аудиопроигрывателя.

Отличия от работы [3] состоят в следующем:

- введены дополнительные функции аудиопроигрывателя – возможность удаления, настройки повтора проигрывания треков, которые осуществляются с помощью дополнительных меню.
- новый метод взаимодействия и полностью переработанная структура автоматов, реализующих аудиопроигрыватель.

Программа написана на языке *Java* [4].

## 1. Постановка задачи

Целью данной работы является демонстрация принципов автоматного программирования на примере создания модели аудиопроигрывателя.

На рис. 1 приведен внешний вид модели.

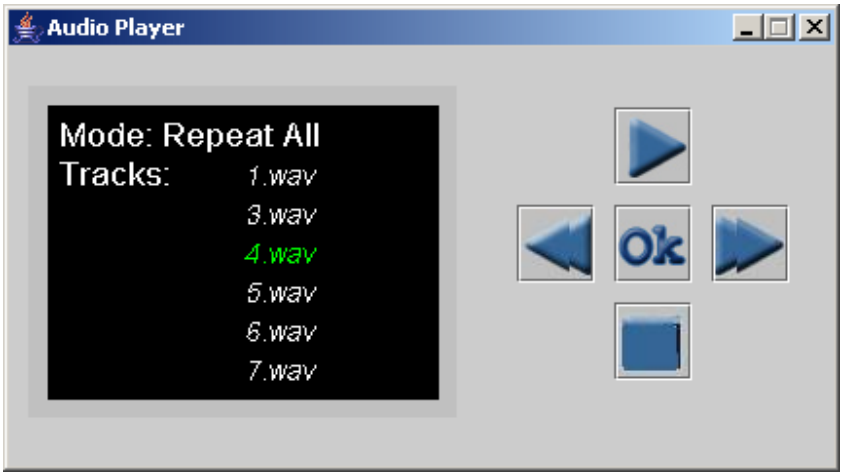


Рис. 1.Внешний вид модели

Интерфейс проигрывателя состоит из:

- дисплея, на котором отображается текущее состояние проигрывателя (список треков, проигрываемый трек или меню).
- кнопки «Играть/Пауза» ( ►), которая обеспечивает начало проигрывание трека, а при повторном нажатии – временную остановку.
- кнопки «Стоп» ( ■ ), обеспечивающей остановку проигрывания трека.
- кнопки «Вперед» ( ►►), осуществляющей переход к следующему треку.
- кнопки «Назад» ( ◄◄), которая обеспечивает переход к предыдущему треку.
- кнопки «Меню» ( *OK* ), обеспечивающей вывод вспомогательного меню на экран дисплея.

В зависимости от текущего режима проигрывателя, дисплей отображает различную информацию. В режиме готовности к проигрыванию на дисплее показаны имеющиеся

треки, а также выделен текущий. В режиме проигрывания трека – информация о текущем треке. В режиме меню – пункты меню.

Меню состоит из двух частей – главное и вложенное меню. При нажатии кнопки «Меню» появляется главное меню, позволяющее выбрать одно из вложенных меню: «удалить» или «повтор».

## 2. Структура программы

Классы могут быть разделены на две разновидности: вспомогательные и автоматные.

Перечислим вспомогательные классы:

- `AudioPlayer` (создает окно программы, инициализирует все автоматы и другие классы);
- `ClientArea` (рисует основную клиентскую область);
- `ControlPanel` (рисует навигационные кнопки правого меню);
- `ImageMap` (обеспечивает работу с графическими файлами);

При построении автоматных классов используется наследование. Класс `Automat` является базовым для всех остальных автоматов и содержит их общие поля и методы:

- `Y0` (номер состояния автомата);
- `GetState` (возвращает номер состояния автомата);
- `SetState` (устанавливает номер состояния автомата);
- `Name` (имя автомата);
- `GetName` (возвращает имя автомата);
- `SetName` (устанавливает номер состояния автомата);
- `AutomatType` (тип автомата: A0, A1, …);
- `GetType` (возвращает тип автомата);
- `SetType` (устанавливает тип автомата).

Программа содержит пять классов, наследуемых от базового:

- `AutomatScreen` (A0);
- `AutomatPlayMode` (A1);
- `AutomatMainMenu` (A2);
- `AutomatMenuDelete` (A3);
- `AutomatMenuRepeat` (A4).
- AutomatBackGr (A5);

## 3. Класс Screen

### *3.1. Описание класса*

Класс `Screen` предназначен для реализации экрана, управляемого автоматом `AutomatScreen`.

### *3.2. Автомат AutomatScreen(A0)*

#### 3.2.1. Описнаие событий

e1 – Нажатие кнопки «Вперед».
e2 – Нажатие кнопки «Назад».
e3 – Нажатие кнопки «Играть/Пауза».
e4 – Нажатие кнопки «Стоп».
e5 – Нажатие кнопки «Меню».
e5_1 – Отказ от меню.
e6 – Выбор меню «Удаление»
e7 – Выбор меню «Повтор»

#### 3.2.2. Описание выходных воздействий

z0_0 – перерисовать экран
z0_01 – послать соответствующее сообщение A1
z0_1 – перерисовать экран
z0_11 – послать соответствующее сообщение A1
z0_12 – послать соответствующее сообщение A2
z0_2 – перерисовать экран
z0_23 – послать соответствующее сообщение A3
z0_3 – перерисовать экран
z0_34 – послать соответствующее сообщение A4
z0_4 – перерисовать экран
z0_45 – послать соответствующее сообщение A5

#### 3.2.3. Схема связей автомата

Схема связей автомата A0 приведена на рис. 2.

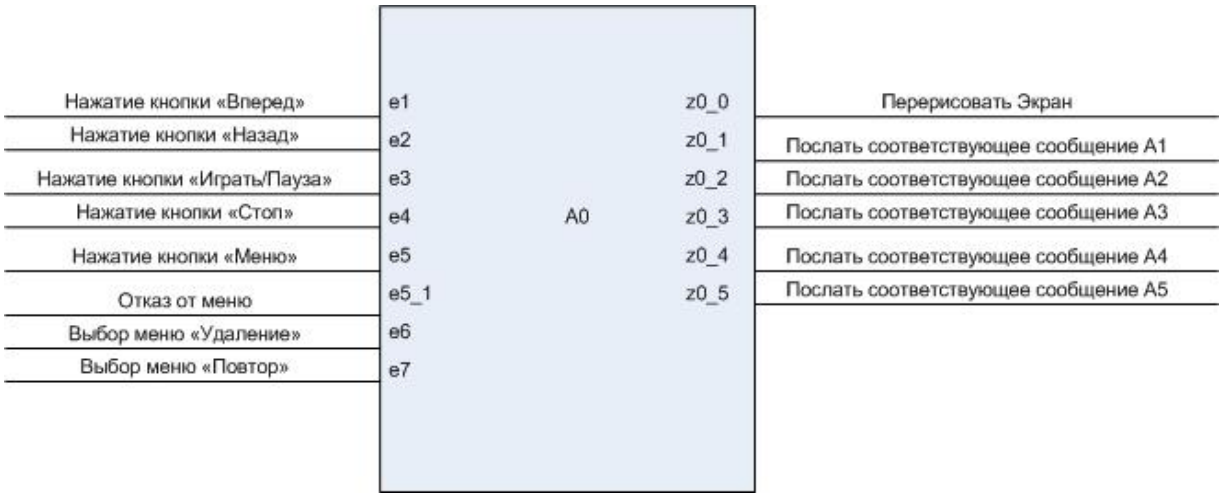


Рис. 2. Схема связей автомата A0

### 3.2.4. Граф переходов

Граф переходов автомата `AutomatScreen` изображен на рис. 3.

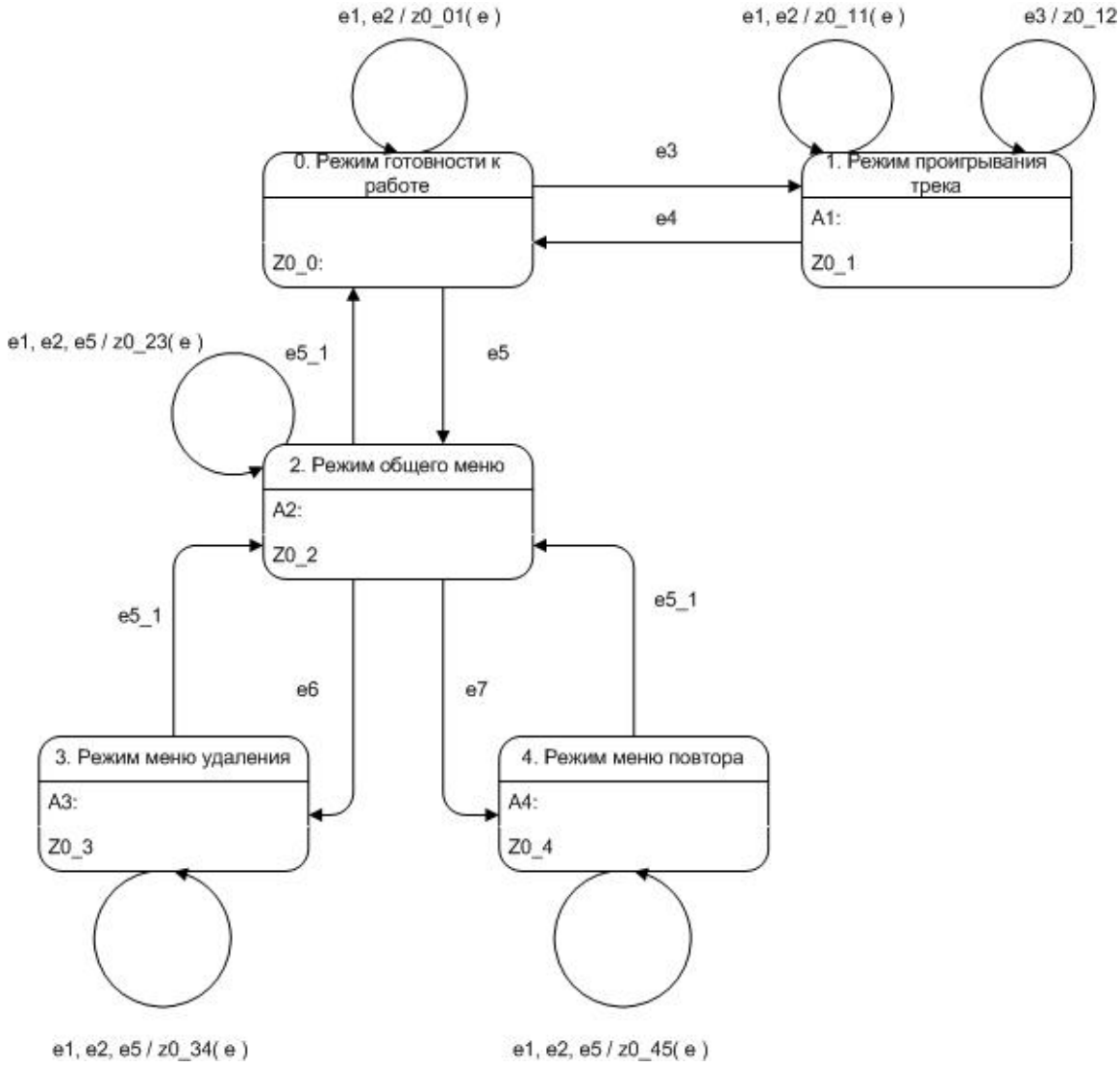


Рис.3. Граф переходов автомата `AutomatScreen`

## 4. Класс BackGR

### *4.1. Описание класса*

Данный класс отвечает за перерисовку содержимого дисплея.

### *4.2. Автомат AutomatBackGR(A5)*

### 4.2.1 Описание событий

e1 – Нажатие кнопки «Вперед». Поступает от автомата A0.
e2 – Нажатие кнопки «Назад». Поступает от автомата A0.

### 4.2.2. Описание входных переменных

x1 – Идет проигрывание трека.
x2 – Номер текущего трека.

### 4.2.3. Описание выходных воздействий

Z5_1 – увеличить значение перменной x2.
Z5_2 – уменьшить значение переменной x2.

### 4.2.4. Схема связей автомата

Схема связей автомата A5 приведена на рис. 4.

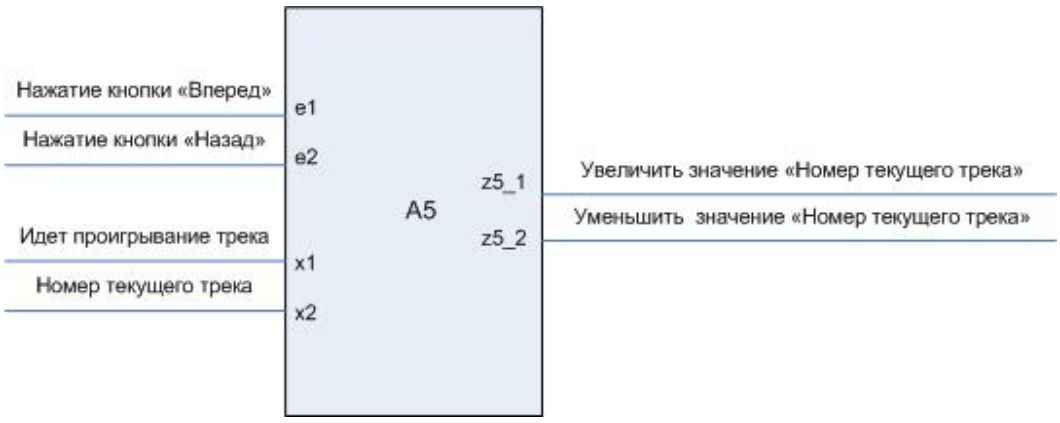


Рис. 4. Схема связей автомата A5

### 4.2.5. Граф переходов

Граф переходов автомата `AutomatBackGR` изображен на рис. 5.

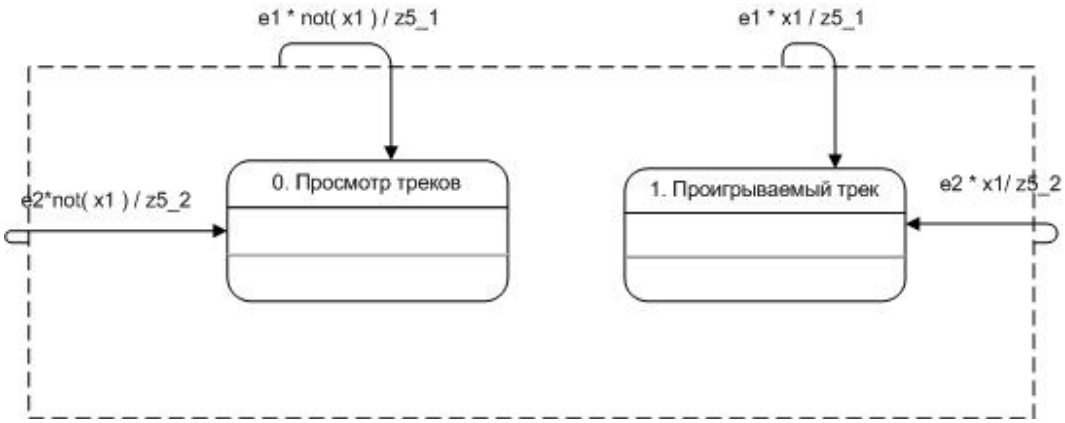


Рис.5. Граф переходов автомата `AutomatBackGR`

# 5. Класс Play

## *5.1. Описание класса*

Отвечает за режим проигрывания треков

## *5.2. Автомат AutomatPlay(A1)*

### 5.2.1 Описание событий

e1 – Нажатие кнопки «Играть/Пауза». Поступает от автомата A0.
e4 – Нажатие кнопки «Стоп».

### 5.2.2. Описание входных переменных

x1 – Идет проигрывание трека.
x2 – Наличие треков.

### 5.2.3. Описание выходных воздействий

Z1_2 – установить значение x1 в 1.
Z1_3 – обнулить значение x1.
Z1_4 – послать автомату A0 сообщение e4.

### 5.2.4. Схема связей автомата

Схема связей автомата A1 приведена на рис. 6.

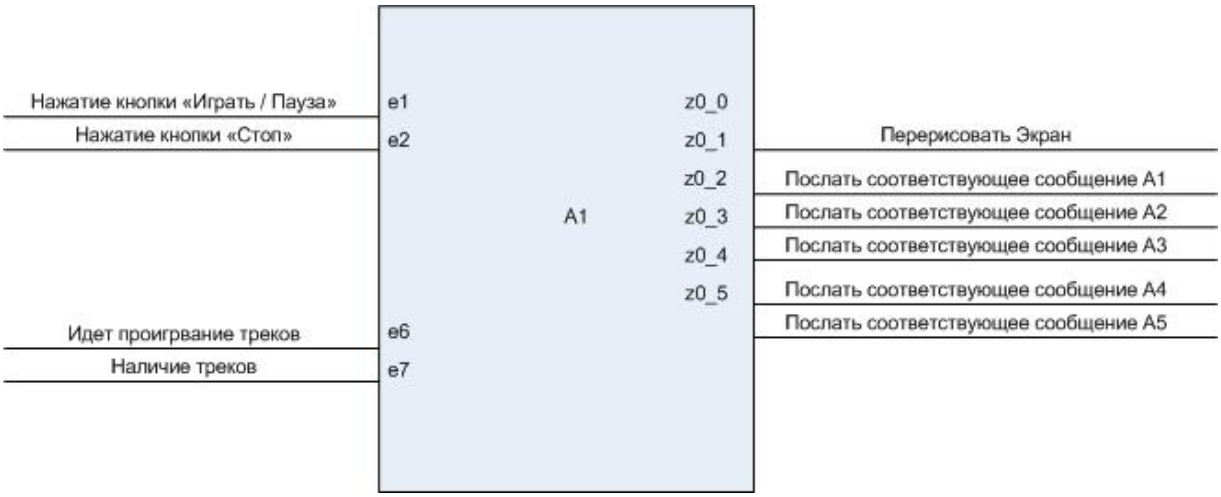


Рис. 6. Схема связей автомата A1

### 5.2.5. Граф переходов

Граф переходов автомата `AutomatPlay` изображен на рис. 7.

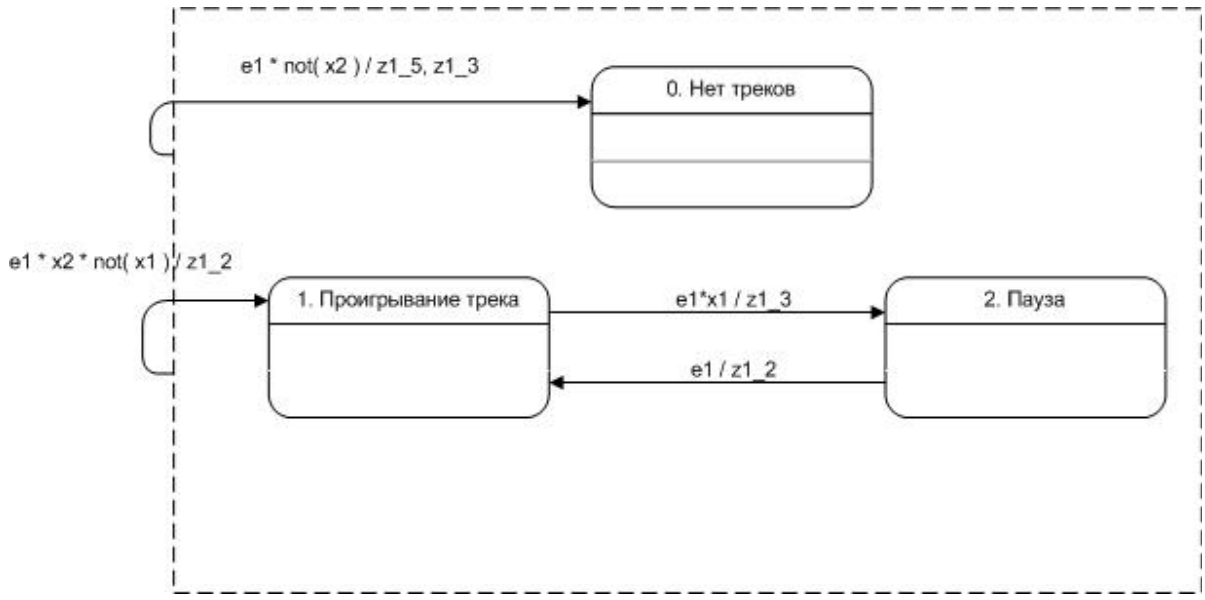


Рис.7. Граф переходов автомата `AutomatPlay`

# 6. Класс MainMenu

## *6.1. Описание класса*

Отвечает за режим главного меню.

## *6.2. Автомат AutomatMainMenu(A2)*

### 6.2.1 Описание событий

e1 – Нажатие кнопки «Вперед». Поступает от автомата A0.
e2 – Нажатие кнопки «Назад». Поступает от автомата A0.
e3 – Нажатие кнопки «Меню». Поступает от автомата A0.

### 6.2.2. Описание выходных воздействий

Z2_5 – послать автомату A0 сообщение e5_1.
Z2_6 – послать автомату A0 сообщение e6.
Z2_7 – послать автомату A0 сообщение e7.

### 6.2.3. Схема связей автомата

Схема связей автомата A2 приведена на рис. 8.

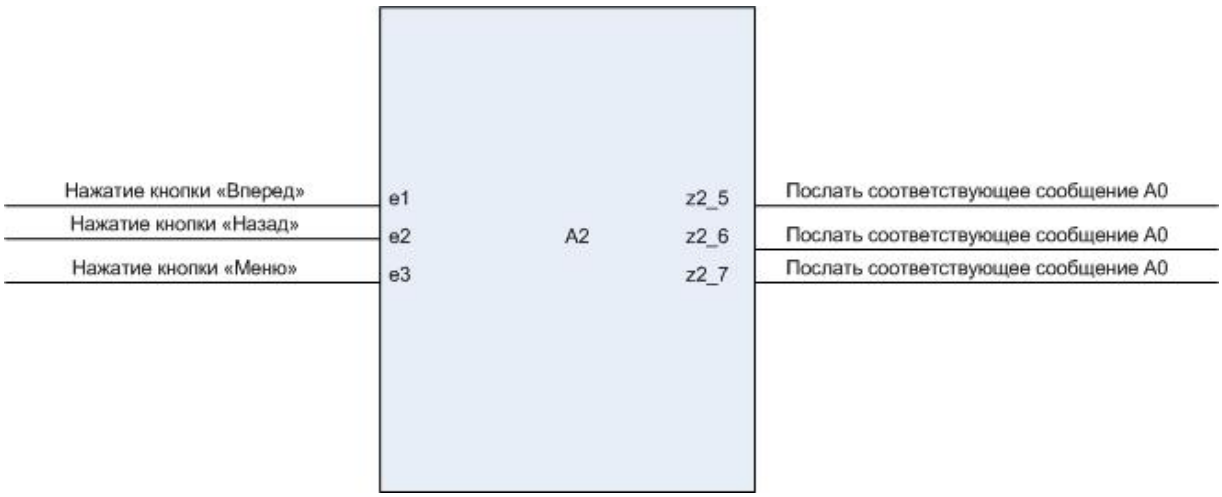


Рис. 8. Схема связей автомата A2

### 6.2.4. Граф переходов

Граф переходов автомата `AutomatMainMenu` изображен на рис. 9.

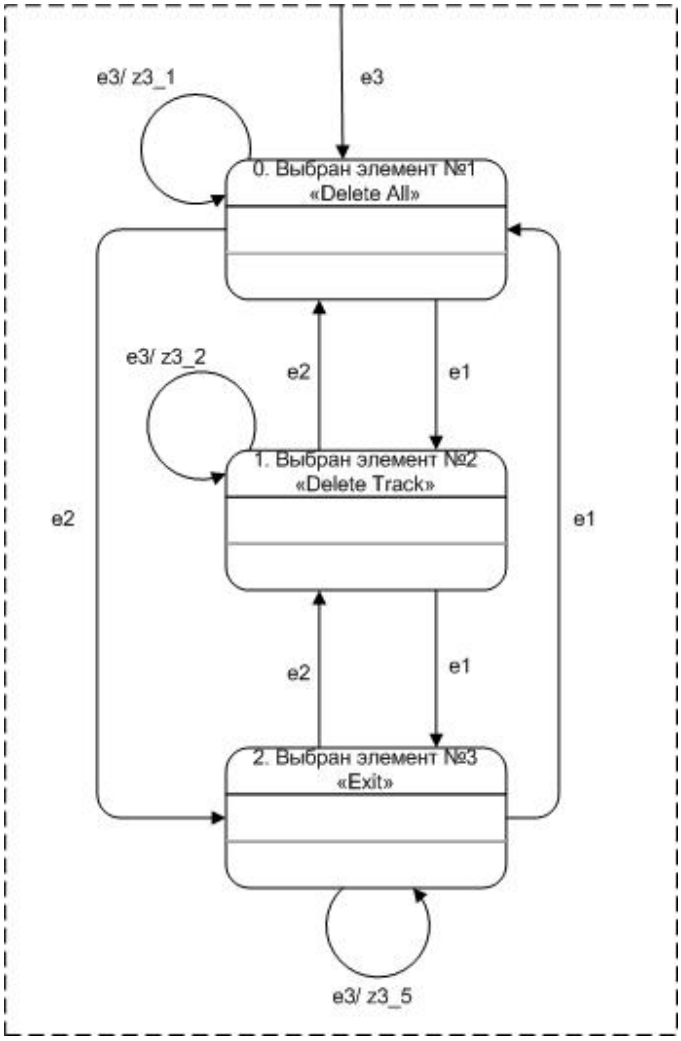


Рис.9. Граф переходов автомата `AutomatMainMenu`

## 7. Класс RepeatMenu

### *7.1. Описание класса*

Отвечает за меню выбора режима повтора.

### *7.2. Автомат AutomatRepeatMenu(A4)*

#### 7.2.1. Описание событий

e1 – Нажатие кнопки «Вперед». Поступает от автомата A0.
e2 – Нажатие кнопки «Назад». Поступает от автомата A0.

e3 – Нажатие кнопки «Меню». Поступает от автомата A0.

### 7.2.2. Описание входных переменных

x1 – Номер текущего трека.

### 7.2.3. Описание выходных воздействий

z4_1 – изменить значение параметра «номер выбранного элемента»
z4_2 – изменить значение параметра «номер выбранного элемента»
z4_5 – послать автомату A0 сообщение e5_1.

### 7.2.4. Схема связей автомата

Схема связей автомата A4 приведена на рис. 10.

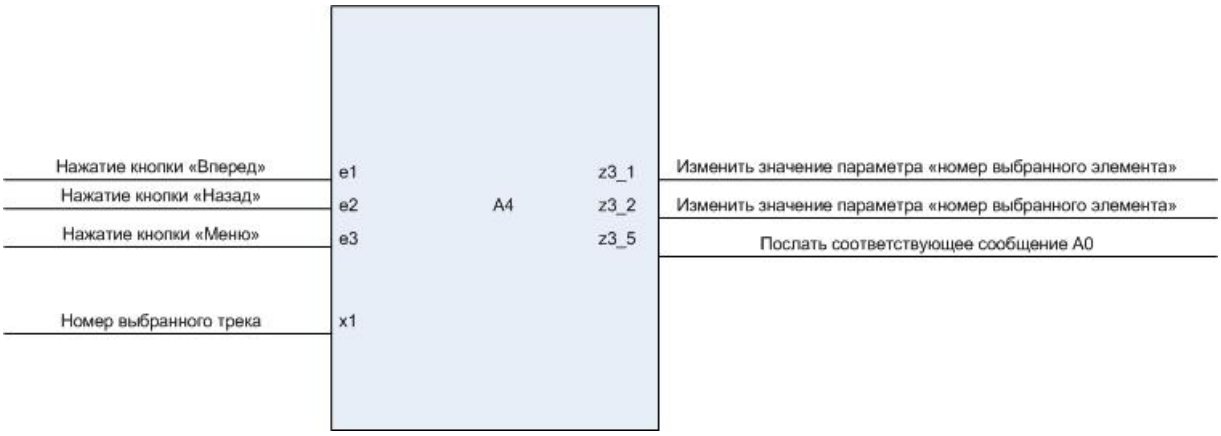


Рис. 10. Схема связей автомата A4

### 7.2.5. Граф переходов

Граф переходов автомата `AutomatRepeatMenu` изображен на рис. 11.

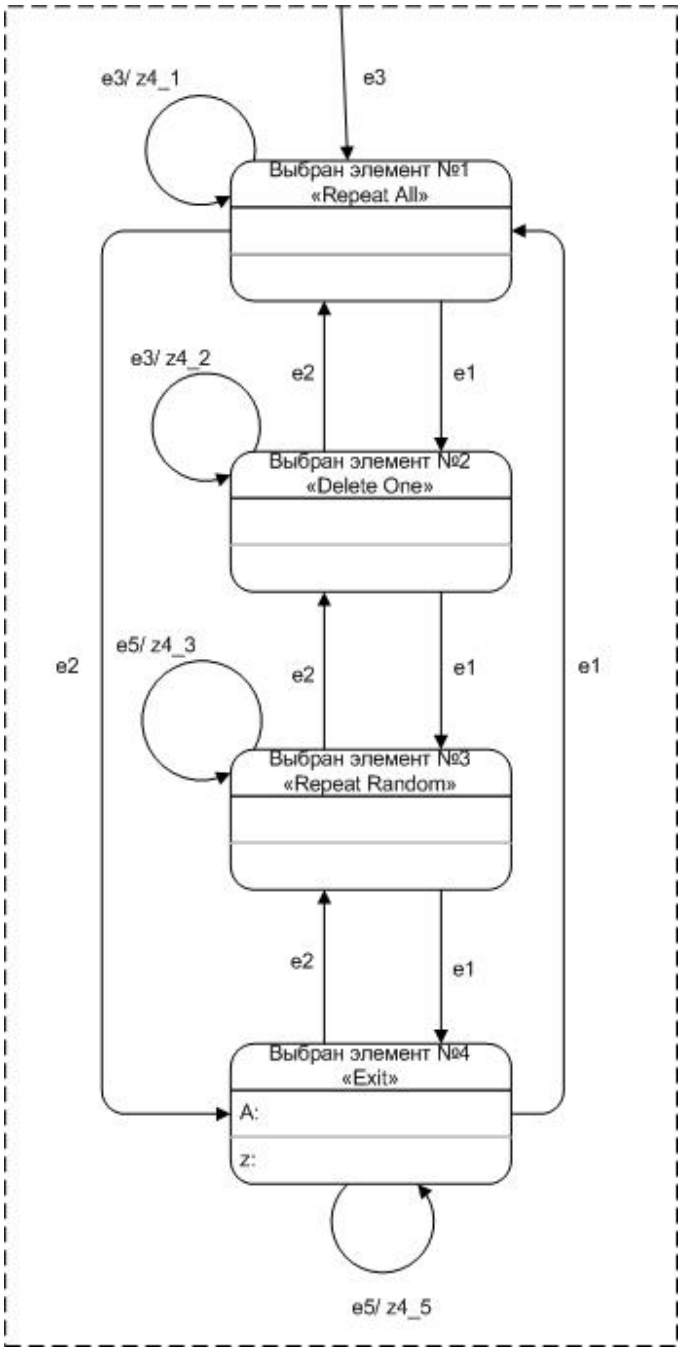


Рис.11. Граф переходов автомата `AutomatRepeatMenu`

## 8. Класс DeleteMenu

### *8.1. Описание класса*

Отвечает за меню выбора режима удаления.

### *8.2. Автомат AutomatDeleteMenu(A3)*

#### 8.2.1 Описание событий

e1 – Нажатие кнопки «Вперед». Поступает от автомата A0.
e2 – Нажатие кнопки «Назад». Поступает от автомата A0.
e3 – Нажатие кнопки «Меню». Поступает от автомата A0.

#### 8.2.2 Описание входных переменных

x1 – номер выбранного трека.

#### 8.2.3. Описание выходных воздействий

Z3_1 – изменить значение параметра «номер выбранного элемента»
Z3_2 – изменить значение параметра «номер выбранного элемента»
Z3_5 – послать автомату A0 сообщение e5_1.

#### 8.2.4. Схема связей автомата

Схема связей автомата A3 приведена на рис. 12.

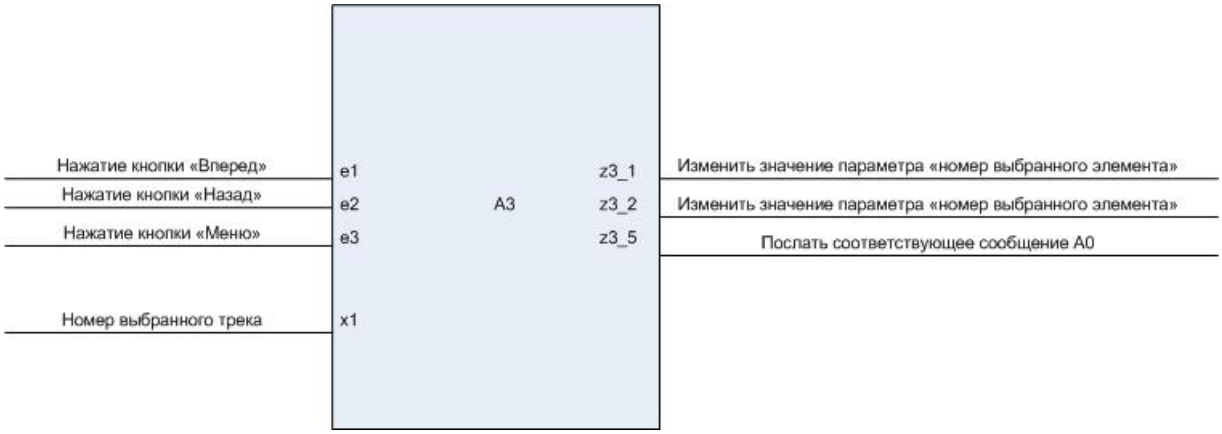


Рис. 12. Схема связей автомата A3

### 8.2.5. Граф переходов

Граф переходов автомата `AutomatDeleteMenu` изображен на рис. 13.

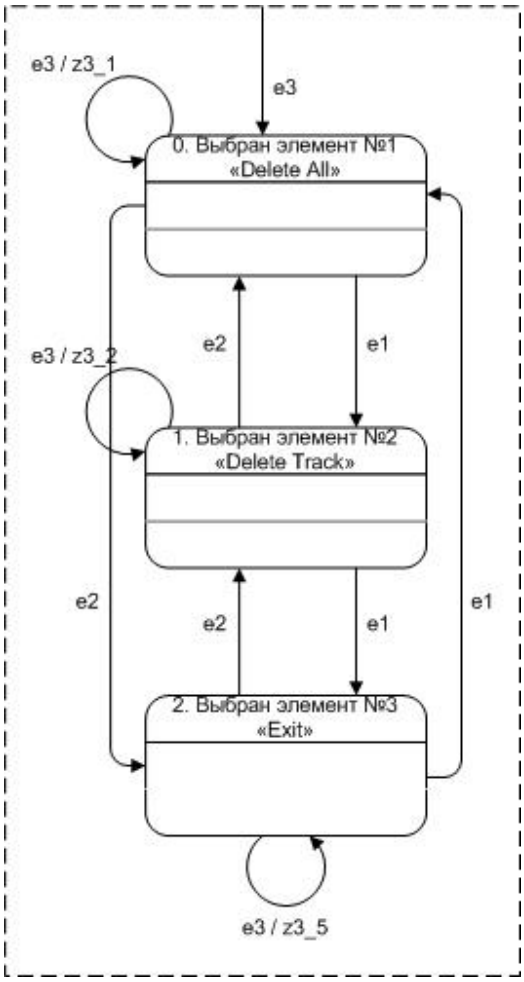


Рис.13. Граф переходов автомата `AutomatDeleteMenu`

## Заключение

Данная работа показывает, что применение автоматного подхода в программировании может быть удобным для решения целого класса задач. Особенно удобным является контроль над функциональной частью программы и изменение самой структуры работы приложения.

## Литература


1. *Шалыто А.А.* Switch-технология. Алгоритмизация и программирование задач логического управления . СПб.: Наука, 1998.
2. *Шалыто А.А., Туккель Н.И.* *Switch*-технология — автоматный подход к созданию программного обеспечения «реактивных» систем //Программирование. 2001. №5. http://is.ifmo.ru. Раздел «Статьи».
3. *Добровольский В.А., Степук А.В.* Простой аудиопроигрыватель. СПБГУИТМО, 2004, http://is.ifmo.ru. Раздел «Проекты».
4. *Эккель Б.* Философия *Java*. Библиотека программиста. – СПб: Питер, 2001.

Приложение. Текст программы.

**AudioPlayer.java**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

/**
 * AudioPlayer- это Класс, описывающий приложение.
 */
public final class AudioPlayer extends JFrame
        implements ActionListener, MouseMotionListener,
MouseListener, ComponentListener {

    public Screen screen;
    public BackGr backGr;
    public MenuAuto menuAuto;
    public MenuChooseDeleteMode menuChooseDeleteMode;
    public MenuChooseRepeatMode menuChooseRepeatMode;
    public PlayMode playMode;
    public PlayAudio playAudio;

    public ClientArea clientArea = new ClientArea(this);
    public ControlPanel controlPanel = new ControlPanel();


    public static void main(String[] args) {
        Frame f;
        f = new AudioPlayer("Audio Player");
        f.setResizable(false);
  }

    public void paintClient(Graphics g) {
        screen.draw(g);
    }

    public AudioPlayer(String s) {
        super(s);
        int i;

        screen = new Screen(this);
        backGr = new BackGr(screen);
        menuAuto = new MenuAuto(screen);
        playMode = new PlayMode(this);
        playAudio = new PlayAudio(this);

        menuChooseDeleteMode = new
MenuChooseDeleteMode(screen);
```

```
        menuChooseRepeatMode = new
MenuChooseRepeatMode(screen);
        //getContentPane().setLayout(new BorderLayout());
        getContentPane().setLayout(null);

        clientArea.setBounds(10, 20, 220, 170);
        clientArea.setBackground(Color.black);
        getContentPane().add(clientArea);

        controlPanel.addActionListener(this);
        controlPanel.setBounds(250, 20, 400, 250);
        getContentPane().add(controlPanel);//,
BorderLayout.WEST);

        clientArea.addMouseMotionListener(this);
        clientArea.addMouseListener(this);

        addComponentListener(this);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });

        setBounds(100, 100, 400, 200);
        setSize(430, 240);
        setVisible(true);
        repaint();

    }

    public void actionPerformed(ActionEvent e) {
        Object source = e.getSource();

        if (source == controlPanel.up) {
            screen.pressPlay();
        } else if (source == controlPanel.down) {
            screen.pressStop();
        } else if (source == controlPanel.right) {
            screen.pressRight();
        } else if (source == controlPanel.left) {
            screen.pressLeft();
        } else if (source == controlPanel.menubut) {
            screen.pressMenubut();
        }
        clientArea.repaint();
    }

    public void mouseDragged(MouseEvent e) {
    }
```

```
    public void mouseMoved(MouseEvent e) {
    }

    public void mouseClicked(MouseEvent e) {
    }

    public void mouseEntered(MouseEvent e) {
    }

    public void mouseExited(MouseEvent e) {
    }

    public void mousePressed(MouseEvent e) {
    }

    public void mouseReleased(MouseEvent e) {
    }

    public void componentResized(ComponentEvent e) {
    }

    public void componentMoved(ComponentEvent e) {
    }

    public void componentShown(ComponentEvent e) {
    }

    public void componentHidden(ComponentEvent e) {
    }
}
```

**Screen.java**

```
import java.awt.*;
import java.util.*;
import java.lang.reflect.Method;

/**
 * Класс "Экран"
 */
public class Screen {

    /**
     * Класс "Автомат для управления экраном"
     */
    class AutomatScreen extends Automat {

        /**
         * Конструктор
         */
```

```
        public AutomatScreen() {
            setState(0);
            setName("Automat A0");
        }

        public boolean x1() {
            return (mode == 0);
        }

        public boolean x2() {
            return (mode == 1);
        }
        /*
        * Функции-выходы
        */
        public void z0_0() {
            if (owner.playMode.getIsPlaying()) {
                owner.playMode.setIsPlaying(false);
                owner.playMode.a1.A(4);
            }
            owner.backGr.A5.A(4);
        }

        public void z0_1() {

owner.playMode.pressMenubut(owner.backGr.getTrackNum());
            owner.backGr.A5.A(3);
        }

        public void z0_2() {
            owner.menuAuto.pressMenubut();
        }

        public void z0_3() {
            owner.menuChooseDeleteMode.pressMenubut();
        }

        public void z0_31() {
            switch
(owner.menuChooseDeleteMode.getSelectedItemIndex()){
                case 0:
                    owner.playMode.deleteTracks();
                    break;
                case 1:

owner.playMode.deleteTrack(owner.backGr.getTrackNum() - 1);
                    break;
                case 2:
                    owner.playMode.deleteRandomTrack();
                    break;
            }
```

```
owner.backGr.setTrackTotal(owner.playMode.getTracks().size());
        }

        public void z0_4() {
            owner.menuChooseRepeatMode.pressMenubut();
        }

         public void z0_41() {
        }

        public void z0_5() {
        }

        /**
         * Реализация автомата A5
         */
        public void A(int e) {
            int y0 = getState();
            int yold = y0;

            Method method = null;
            Vector methodParametr = new Vector();
            Object object = null;

            Class classes[] = new Class[1];
            classes[0] = Integer.class;

            try {
                switch (y0) {
                    case 0:
                        if (e == 1) {
                            y0 = 0;
                            owner.backGr.A5.A(1);
                        }
                        if (e == 2) {
                            y0 = 0;
                            method =
BackGr.AutomatBackGr.class.getMethod("A", classes);
                            methodParametr.add(new
Integer(2));
                            object = owner.backGr.A5;
                        }
                        if (e == 3) {
                            y0 = 1;

                        }
                        if (e == 5) {
                            y0 = 2;
                        }
                        break;
```

```
                    case 1:
                        if (e == 1) {
                            y0 = 1;
                            owner.playMode.a1.A(1);
                        }
                        if (e == 2) {
                            y0 = 1;
                            owner.playMode.a1.A(2);
                        }
                        if (e == 3) {
                            y0 = 1;
                            owner.playMode.a1.A(3);
                        }
                        if (e == 4) {
                            y0 = 0;
                        }
                        break;
                    case 2:
                        if (e == 1) {
                            y0 = 2;
                            owner.menuAuto.A2.A(3);
                            //call A3
                        }
                        if (e == 2) {
                            y0 = 2;
                            owner.menuAuto.A2.A(4);
                            //call A3
                        }

                        if (e == 5) {
                            // y0 = 0;
                            method =
MenuAuto.AutomatMenuAuto.class.getMethod("A", classes);
                            methodParametr.add(new
Integer(11));
                            object = owner.menuAuto.A2;
                            //z0_2();
                        }
                        if (e == 51) {
                            y0 = 0;

                        }
                        if (e == 7) {
                            y0 = 3;

                        }
                        if (e == 6) {
                            y0 = 4;
                        }
                        break;
                    case 3:
```

```
                        if (e == 1) {
                            y0 = 3;

owner.menuChooseDeleteMode.A3.A(1);
                            //call A4
                        }
                        if (e == 2) {
                            y0 = 3;

owner.menuChooseDeleteMode.A3.A(2);
                            //call A4
                        }
                        if (e == 5) {

                            method =
MenuChooseDeleteMode.AutomatMenuChooseElement.class.getMethod(
"A", classes);
                            methodParametr.add(new
Integer(11));
                            object =
owner.menuChooseDeleteMode.A3;
                            //call A4
                        }
                        if (e == 51) {
                            y0 = 0;
                            z0_31();
                        }
                        break;
                    case 4:
                        if (e == 1) {
                            y0 = 4;

owner.menuChooseRepeatMode.A3.A(1);
                            //call A5
                        }
                        if (e == 2) {
                            y0 = 4;

owner.menuChooseRepeatMode.A3.A(2);
                            //call A5
                        }
                        if (e == 5) {

                            method =
MenuChooseRepeatMode.AutomatMenuChooseElement.class.getMethod(
"A", classes);
                            methodParametr.add(new
Integer(11));
                            object =
owner.menuChooseRepeatMode.A3;
                            //call A5
```

```
                    }
                    if (e == 51) {
                        y0 = 0;
                        z0_41();
                    }
                    break;
                }

            } catch (java.lang.NoSuchMethodException exp) {
            }

            setState(y0);

            if (y0 != yold) {
                switch (y0) {
                    case 0:
                        z0_0();
                        break;
                    case 1:
                        z0_1();
                        break;
                    case 2:
                        z0_2();
                        break;
                    case 3:
                        z0_3();
                        break;
                    case 4:
                        z0_4();
                        break;
                }
            }


            if (method != null) {
                java.lang.Object parametrs[] =
methodParametr.toArray();

                try {
                    method.invoke(object, parametrs);
                } catch (java.lang.IllegalAccessException exp)
{
                } catch
(java.lang.reflect.InvocationTargetException exp) {
                }
            }

            owner.clientArea.repaint();


        }
```

```
    ;


}

;

public java.util.Timer timer = new java.util.Timer();

public class ShootTimerTask extends java.util.TimerTask {
    public void run() {
        A0.A(17);
    }

    ;
}

;
private ShootTimerTask shootTimerTask;

/**
 * Автомат, управляющий экраном
 */
private AutomatScreen A0;

private int x; //координаты экрана на панели
private int y;

private int width; //ширина и высота экрана
private int height;

public int getX() {
    return x;
}

;

public int getY() {
    return y;
}

;

public int getWidth() {
    return width;
}

;

public int getHeight() {
```

```
        return height;
    }

    ;


    private int mode;
    private boolean onoff; //true -включен, false - выключен


    public AudioPlayer owner; //приложение-владелец

    /**
     * Конструктор с парамером o - приложение-владелец
     */
    public Screen(AudioPlayer o) {
        owner = o;
        //cannonName = o.getName() + " cannon";
        A0 = new AutomatScreen();
        onoff = true;
        mode = 0;
        x = 10;
        y = 10;
        width = 200;
        height = 150;
    }

    ;

    /**
     * Метод для перерисовки экрана
     */
    public void draw(Graphics g) {
        BackGr backGr = owner.backGr; //сначала перерисовываем
фон,

        if (onoff) {
            backGr.draw(g);
            owner.menuAuto.draw(g);
            owner.menuChooseDeleteMode.draw(g);
            owner.menuChooseRepeatMode.draw(g);
        }   //если, конечно, включен

        g.drawRect(x, y, width, height); //дальше просто
рисуем рамку
        Color oldc = g.getColor();
        g.setColor(Color.lightGray);

        for (int i = 1; i <= x + 1; i++) {
            g.drawRect(x - i, y - i, width + 2 * i, height + 2
* i);
```

```
        }
        g.setColor(oldc);
    }

    /**
     * Методы,посылающие сообщения автомату, управляющему
экраном
     */

    public void pressPlay() {
        A0.A(3);
    }

    public void pressStop() {
        A0.A(4);
    }

    public void pressRight() {
        A0.A(1);
    }

    public void pressLeft() {
        A0.A(2);
    }


    public void pressMenubut() {
        A0.A(5);
    }


    public void pressEscapeMenubut() {
        A0.A(51);
    }


    public void pressSetupMenubut() {
        A0.A(110);
    }

    public void pressMenuSizeChoosen() {
        A0.A(7);
    }

    public void pressMenuFlashChoosen() {
        A0.A(6);
    }

    public void shoot() {
        A0.A(16);
```

```
    }

}
```

**BackGr.java**

```
import java.awt.*;
import java.util.*;

/**
 * класс "Фон"
 */
public class BackGr {

    /**
     * Автомат, управляющий фоном.
     */
    class AutomatBackGr extends Automat {

        public AutomatBackGr() {
            setState(0);
            setName("Automat A1");
        }

        public boolean x1() {
            return (playing);
        }
        /*
        * Функции-выходы
        */
        public void z5_1() {
            trackNum++;
            if (trackNum == trackTotal + 1) trackNum = 1;
        }

        public void z5_2() {
            trackNum--;
            if (trackNum == 0) trackNum = trackTotal;
        }

        public void z5_3() {
            owner.pressPlay();
        }

        public void z5_4() {
            owner.pressStop();
        }

        /**
         * Функция автомата
```

```
     */
    public void A(Integer event) {
        A(event.intValue());
    }

    public void A(int e) {
        int y0 = getState();
        int yold = y0;
        /*
        e1-right
        e2-left
        e3-play
        e4-stop
        */
        try {
            switch (y0) {
                case 0:
                    if (e == 1) {
                        y0 = 0;
                        z5_1();
                    }
                    if (e == 2) {
                        y0 = 0;
                        z5_2();
                    }
                    if (e == 3) {
                        y0 = 1;
                        z5_3();
                    }
                    break;
                case 1:
                    if (e == 4) {
                        y0 = 0;
                        z5_4();
                    }

                    break;

                default:
                    break;
            }
        } catch (Exception ex) {
        }

        setState(y0);


    }

}
```

```
    ;

    /**
     * Автомат, управляющий фоном.
     */
    public AutomatBackGr A5;

    /**
     * Экран-владелец
     */
    private Screen owner;
    private boolean playing;


    /**
     * Координаты верхнего левого угла фона на панели
     */
    private int x;
    private int y;

    /**
     * Просто изображение (кучу раз используется в
перерисовке)
     */
    private Image img;

    /*Физические параметры*/
    /**
     * Номер просматриваемого снимка
     */
    private int trackNum;

    public int getTrackNum() {
        return trackNum;
    }

    ;

    public void setTrackNum(int n) {
        trackNum = n;
    }


    /**
     * Всего количество снимков
     */
    private int trackTotal;

    public int getTrackTotal() {
        return trackTotal;
    }
```

```
public void setTrackTotal(int n) {
    trackTotal = n;
}

/**
 * Текущая величина зума
 */
private int zoom;

public int getZoom() {
    return zoom;
}

private Font labelFont;
private Font trackFont;
private Font playingFont;

/**
 * Конструктор
 */
public BackGr(Screen s) {
    //owner = o;
    //cannonName = o.getName() + " cannon";
    A5 = new AutomatBackGr();
    owner = s;
    x = s.getX() + 1;
    y = s.getY() + 1;

    trackNum = 1;
    labelFont = new Font("Courier", Font.BOLD , 16);
    trackFont = new Font("Courier", Font.ITALIC,14);
    playingFont = new Font("Courier", Font.BOLD,22);

    /*trackTotal = 7;*/


    trackTotal = 0;

    zoom = 1;
}

/**
 * Метод для перерисовки фона
 */
public void draw(Graphics g) {
    int state = A5.getState();
    ArrayList al = owner.owner.playMode.getTracks();
    Color old = g.getColor();

    if (state == 0) {
```

```
            int num;
            int limit = al.size() > 6 ? 6 : al.size();
            g.setColor(Color.white);
            g.setFont(labelFont);
            g.drawString("Mode: " +
owner.owner.menuChooseRepeatMode.getSelectedItem(), x + 5, y +
20);
            g.drawString("Tracks: ", x + 5, y + 40);


            for (int i = 1; i <= limit; i++) {
                num = i + (trackNum / 6) * (trackNum % 6);


                g.setFont(trackFont);

                if (num == trackNum) {
                    g.setColor(Color.green);
                } else {
                    g.setColor(Color.white);
                }
                g.drawString(al.get(num - 1).toString(), x +
100, y + i * 20 + 20);

            }
        } else if (state == 1) {
            g.setFont(playingFont);
            g.setColor(Color.white);
            g.drawString("Playing: " + al.get(trackNum - 1), x
+ 20 , y + 80);
        }

        g.setColor(old);


    }
}
```

**MenuAuto.java**

```
import java.awt.*;
import java.util.*;
import java.lang.reflect.Method;

/**
 * Класс "Меню настройки режима авто"
 */
public class MenuAuto {

    /**
     * Класс "Автомат для управления меню"
```

```
 */
class AutomatMenuAuto extends Automat {

    /**
     * Конструктор
     */
    public AutomatMenuAuto() {
        setState(0);
        setName("Automat A2");
    }

    /**
     * Функция-флажок. Просто говорит, верно ли, что меню
отображается на экране
     */
    public boolean x1() {
        return (onoff);
    };

    /**
     * Функции-выходы
     */
    public void z2_0() {
        owner.pressEscapeMenubut();
        onoff = false;

    };

    public void z2_01() {
        choosenBut = 0;
        onoff = true;
    };


    public void z2_11() {
        //owner.pressMenuFlashChoosen();
        onoff = true;
        choosenBut = 2;

    };
    public void z2_12() {
        //owner.pressMenuSizeChoosen();
        onoff = true;
        choosenBut = 1;
    };

    public void z2_10() {
        onoff = false;
        if (choosenBut == 2) {
            owner.pressMenuFlashChoosen();
        }else{
```

```
                owner.pressMenuSizeChoosen();
            }
        };

        public void A(Integer event) {
            A(event.intValue());
        }
        /**
         * Непосредственно функция автомата
         */
        public void A(int e) {
            int y0 = getState();
            int yold = y0;

            Method method = null;
            Vector methodParametr = new Vector();
            Object object = null;
            Class classes[] = new Class[0];

            try {
                switch (y0) {
                    case 0:
                        if ((e == 11) && (!x1())) {
                            y0 = 0;
                            method =
MenuAuto.AutomatMenuAuto.class.getMethod("z2_01", classes);
                            object = this;
                        } else if ((e == 11) && (x1())) {
                            y0 = 0;
                            method =
MenuAuto.AutomatMenuAuto.class.getMethod("z2_0", classes);
                            object = this;
                        } else if (e == 3) {
                            y0 = 1;
                            method =
MenuAuto.AutomatMenuAuto.class.getMethod("z2_11", classes);
                            object = this;
                        } else if (e == 4) {
                            y0 = 1;
                            method =
MenuAuto.AutomatMenuAuto.class.getMethod("z2_12", classes);
                            object = this;
                        }
                        break;

                    case 1:
                        if ((e == 11) && (!x1())) {
                            y0 = 0;
                            method =
MenuAuto.AutomatMenuAuto.class.getMethod("z2_01", classes);
                            object = this;
```

```
                        } else if ((e == 11) && (x1())) {
                            y0 = 0;
                            method =
MenuAuto.AutomatMenuAuto.class.getMethod("z2_10", classes);
                            object = this;
                        } else if (e == 3) {
                            y0 = 1;
                            method =
MenuAuto.AutomatMenuAuto.class.getMethod("z2_11", classes);
                            object = this;
                        } else if (e == 4) {
                            y0 = 1;
                            method =
MenuAuto.AutomatMenuAuto.class.getMethod("z2_12", classes);
                            object = this;
                        }
                        break;

                    default:
                        break;
                }

            } catch (java.lang.NoSuchMethodException exp) {
            }
            setState(y0);
            logTrans(y0, yold, e);


            if (method != null) {
                java.lang.Object parametrs[] =
methodParametr.toArray();

                try {
                    method.invoke(object, parametrs);
                } catch (java.lang.IllegalAccessException exp)
{
                } catch
(java.lang.reflect.InvocationTargetException exp) {}
            }
        };


    };

    /**
     * Автомат, управляющий меню
     */
    public AutomatMenuAuto A2;

    private boolean onoff; //true -включен, false - выключен
    private int choosenBut; //номер выбранной кнопки
```

```
    /**
     * Экран-владелец
     */
    private Screen owner;

    /**
     * Конструктор с парамером o - приложение-владелец
     */
    public MenuAuto(Screen o) {
        owner = o;
        A2 = new AutomatMenuAuto();

        onoff = false;
        choosenBut = 0;
    };

    private void drawMenuItem(int n, boolean ifSelected,
Graphics g) {

        int width = 70;
        int height = 30;
        String caption;
        if (n == 1) {
            caption = "Delete";
        } else {
            caption = "Repeat";
        }
        ;
        int x = 30 + (n - 1) * (width + 20);
        int y = 60;

        Color oldc = g.getColor();
        g.setColor(Color.darkGray);
        g.fillRect(x, y, width, height);

        g.setColor(Color.white);
        if (ifSelected) g.setColor(Color.yellow);
        g.drawRect(x, y, width, height);

        int textWidth =
g.getFontMetrics().stringWidth(caption);
        g.drawString(caption, x + (width - textWidth)/2, y +
(int) (height / 2) + 5);

        g.setColor(oldc);
    };

    /**
     * Метод для перерисовки экрана
     */
```

```
    public void draw(Graphics g) {
       if (onoff) {
            Color oldc = g.getColor();
            g.setColor(Color.black);

            drawMenuItem(1, (choosenBut == 1), g);
            drawMenuItem(2, (choosenBut == 2), g);
            g.setColor(oldc);
        }
    };

    /**
     * Методы,посылающие сообщения автомату, управляющему
экраном
     */
    public void pressMenubut() {
        A2.A(11);
    };

    public void pressRight() {
        A2.A(3);
    };

    public void pressLeft() {
        A2.A(4);
    };


}
```

**MenuChooseDeleteMode.java**

```
public class MenuChooseDeleteMode extends MenuChooseElement{
    public MenuChooseDeleteMode(Screen screen) {
        super(screen);
        setItem(0, "Delete All");
        setItem(1, "Delete One");
        setItem(2, "Delete Random");
    }
}
```

**MenuChooseRepeatMode.java**

```
public class MenuChooseRepeatMode extends MenuChooseElement{
    public MenuChooseRepeatMode(Screen screen) {
        super(screen);
        setItem(0, "Repeat All");
        setItem(1, "Repeat One");
        setItem(2, "Repeat Random");
    }
}
```

**Automat.java**

```
/**
 * <code>Automat</code>
 * Базовый класс для любого автомата. Содержит объявление
необходимых методов.
 */
public abstract class Automat{

    /**
    * Номер состояния автомата
    */
    private int y0;

    /**
    * Возвращает номер состояния автомата
    */
    protected int getState() {return y0;};

    /**
    * Устанавливает новое состояния автомата
    */
    protected void setState(int y) {y0 = y;};

    /**
    * Имя автомата
    */
    private String name;

    /**
    * Получить имя автомата
    */
    protected String getName() {return name;};

    /**
    * Установить имя автомата
    */
    protected void setName(String n) {name = n;};

    /**
    * Тип автомата (A0, A1, A2 и.т.д...)
    */
    private String automatType;

    /**
    * Получить тип автомата
    */
    protected String getType() {return automatType;};

    /**
```

```
    * Установить тип автомата
    */
    protected void setType(String t) {automatType = t;};

    /**
    * Выводит лог о том, что автомат перешел
    * в состояние to из состояния from, получив событие e
    */
    protected void logTrans(int to, int from, int e) {
        java.util.Date time = new java.util.Date();
        time =
java.util.Calendar.getInstance(java.util.Locale.getDefault()).
getTime();
        Log.writeln(time.toString() + "| " + "T " +
automatType +
          " <" + name + ">: has changed state from " + from +
" to " + to + " | event: " + e);
    };

    /**
    * Функция автомата. Должна осуществлять переход автомата,
получившего событие e
    * в другое состояние
    */
    public abstract void A(int e);
};
```

**ClientArea.java**

```
import java.awt.*;

class ClientArea extends Panel {
         public Image buffer;
         private Image image;
         private Graphics bg;

         public AudioPlayer owner;

         public ClientArea(AudioPlayer o) {
             super();
             owner = o;
         }
         public void paint(Graphics g) {
                int clientWidth  = getSize().width;
                int clientHeight = getSize().height;

                if ((buffer ==
null)/*||(owner.flagChangeSize)*/) {
                    buffer = this.createImage(clientWidth,
clientHeight);
```

```
                }else
                {
                    buffer.flush();
                }

                if ((image ==
null)/*||(owner.flagChangeSize)*/) {
                    image = this.createImage(clientWidth,
clientHeight);
                }

                bg = buffer.getGraphics();

                Graphics bbg = image.getGraphics();

                bbg.drawImage(buffer, 0, 0, null);

                owner.paintClient(bbg);

                g.drawImage(image, 0, 0, null);
                bg.dispose();
            }
            public final void update(Graphics g) {
                paint(g);
            }

    };
```

**ControlPanel.java**

```
import java.awt.event.*;
import javax.swing.*;


/**
* Класс для работы с панелью управления
*/
public class ControlPanel extends JPanel implements
ActionListener{

    public JButton up;
    public JButton down;
    public JButton left;
    public JButton right;

    public JButton menubut;

    public ControlPanel() {
        super();
        int butSize = 40;
```

```
        setBounds(245, 80, 10 * 4 + 3 * butSize, 10 * 4 + 3 *
butSize);

        up = new
JButton(ImageMap.getImageIcon("buttons/play.jpg"));
        up.setBounds(10 + butSize + 10, 10, butSize, butSize);

        down = new
JButton(ImageMap.getImageIcon("buttons/stop.jpg"));
        down.setBounds(10 + butSize + 10, 10 + 2*(butSize +
10), butSize, butSize);

        left = new
JButton(ImageMap.getImageIcon("buttons/left.jpg"));
        left.setBounds(10,  10 + butSize + 10, butSize,
butSize);
        right = new
JButton(ImageMap.getImageIcon("buttons/right.jpg"));
        right.setBounds(10 + 2*(butSize + 10), 10 + butSize +
10, butSize, butSize);

        menubut = new
JButton(ImageMap.getImageIcon("buttons/ok.jpg"));
        menubut.setBounds(10 + butSize + 10, 10 + butSize +
10, butSize, butSize);

        addActionListener(this);

        setLayout(null);

        add(up);
        add(down);
        add(right);
        add(left);
        add(menubut);
    };

    public void addActionListener(ActionListener listener) {
        up.addActionListener(listener);
        down.addActionListener(listener);
        right.addActionListener(listener);
        left.addActionListener(listener);
        menubut.addActionListener(listener);
    }

    public void addKeyListener(KeyListener listener) {
        super.addKeyListener(listener);
    }

    public void actionPerformed(ActionEvent e)     {
        Object source = e.getSource();
```

```
    };

}
```

**MenuChooseElement.java**

```
import java.lang.reflect.Method;
import java.util.Vector;
import java.awt.*;

public class MenuChooseElement {
    /**
     * Класс "Автомат для управления меню"
     */
    public class AutomatMenuChooseElement extends Automat {

        /**
         * Конструктор
         */
        public AutomatMenuChooseElement() {
            setState(0);
            setName("Automat A3");
        };


        /**
         * Функция-флажок. Просто говорит, верно ли, что меню
отображается на экране
         */
        public boolean x1() {
            return (onoff);
        };

        /**
         * Функции-выходы
         */
        public void z3_0() {
            owner.pressEscapeMenubut();
            onoff = false;
        };

        public void z3_00() {
            //owner.pressEscapeMenubut();
            onoff = true;
        };

        public void z3_01() {
            onoff = false;
            owner.pressEscapeMenubut();
```

```
        };

        public void z3_11() {
            onoff = true;
            choosenBut = 1;
        };

        public void z3_21() {
            onoff = true;
            choosenBut = 2;
        };

        public void z3_31() {
            onoff = true;
            choosenBut = 3;
        };

        public void A(Integer event) {
            A(event.intValue());
        }
        /**
         * Непосредственно функция автомата
         */
        public void A(int e) {
            int y0 = getState();
            int yold = y0;

            Method method = null;
            Vector methodParametr = new Vector();
            Object object = null;
            Class classes[] = new Class[0];

            try {
                switch (y0) {
                    case 0:
                        if ((e == 2) || (e == 1)) {
                            y0 = 1;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_11", classes);
                            object = this;
                        } else if ((e == 11) && (x1())) {
                            y0 = 0;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_01", classes);
                            object = this;
                        } else if ((e == 11) && (!x1())) {
                            y0 = 0;
```

```
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_00", classes);
                            object = this;
                        } else if (e == 4) {
                            y0 = 0;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_0", classes);
                            object = this;
                        }
                        break;

                    case 1:
                        if (e == 1) {
                            y0 = 3;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_31", classes);
                            object = this;
                        } else if (e == 2) {
                            y0 = 2;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_21", classes);
                            object = this;
                        } else if ((e == 11) && (x1())) {
                            y0 = 0;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_01", classes);
                            object = this;
                        } else if ((e == 11) && (!x1())) {
                            y0 = 0;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_00", classes);
                            object = this;
                        } else if (e == 4) {
                            y0 = 0;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_0", classes);
                            object = this;
                        }
                        break;

                    case 2:
                        if (e == 1) {
                            y0 = 1;
```

```
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_11", classes);
                            object = this;
                        } else if (e == 2) {
                            y0 = 3;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_31", classes);
                            object = this;
                        } else if ((e == 11) && (x1())) {
                            y0 = 0;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_01", classes);
                            object = this;
                        } else if ((e == 11) && (!x1())) {
                            y0 = 0;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_00", classes);
                            object = this;
                        } else if (e == 4) {
                            y0 = 0;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_0", classes);
                            object = this;
                        }
                        break;

                    case 3:
                        if (e == 1) {
                            y0 = 2;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_21", classes);
                            object = this;
                        } else if (e == 2) {
                            y0 = 1;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_11", classes);
                            object = this;
                        } else if ((e == 11) && (x1())) {
                            y0 = 0;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_01", classes);
                            object = this;
                        } else if ((e == 11) && (!x1())) {
```

```
                            y0 = 0;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_00", classes);
                            object = this;
                        } else if (e == 4) {
                            y0 = 0;
                            method =
MenuChooseElement.AutomatMenuChooseElement.class.getMethod("z3
_0", classes);
                            object = this;
                        }
                        break;


                    default:
                        break;
                }

            } catch (java.lang.NoSuchMethodException exp) {
            }
            setState(y0);
            logTrans(y0, yold, e);

            if (method != null) {
                java.lang.Object parametrs[] =
methodParametr.toArray();

                try {
                    method.invoke(object, parametrs);
                } catch (java.lang.IllegalAccessException exp)
{
                } catch
(java.lang.reflect.InvocationTargetException exp) {}
            }
        };
    };

    private Screen owner;

    private boolean onoff; //true -включен, false - выключен
    private int choosenBut; //номер выбранной кнопки

    private int itemsNum;
    public int getItemsNum() {return itemsNum;};

    private String[] items;
    public void setItem(int index, String value) {
        items[index] = value;
    }
    public String getItem(int index) {return items[index]; };
```

```
    public int getSelectedItemIndex() {
         return choosenBut - 1;
    };
    public String getSelectedItem() {
        if (choosenBut > 0) {
            return items[choosenBut - 1];
        };
        return "";
    };

    /**
     * Автомат, управляющий меню
     */
    public MenuChooseElement.AutomatMenuChooseElement A3;

    public MenuChooseElement(Screen screen) {
        owner = screen;
        A3 = new MenuChooseElement.AutomatMenuChooseElement();


        onoff = false;
        choosenBut = 1;
        itemsNum = 3;
        items = new String[itemsNum];
    };

    private void drawMenuItem(int n, boolean ifSelected,
Graphics g) {
        int width = 100;
        int height = 20;
        int sizeBetweenItems = 10;
        String caption = items[n];

        int x = (owner.getWidth() - width) / 2 + owner.getX();
        int y = (owner.getHeight() - (sizeBetweenItems *
(itemsNum - 1) + height * itemsNum)) / 2 + owner.getY() +
                (n) * (height + sizeBetweenItems);

        Color oldColor = g.getColor();
        g.setColor(Color.darkGray);
        g.fillRect(x, y, width, height);

        g.setColor(Color.white);
        if (ifSelected) g.setColor(Color.yellow);
        g.drawRect(x, y, width, height);

        int textWidth =
g.getFontMetrics().stringWidth(caption);
        g.drawString(caption, x + (width - textWidth)/2, y +
(int) (height / 2) + 5);
```

```
        g.setColor(oldColor);
    };

    /**
     * Метод для перерисовки экрана
     */
    public void draw(Graphics g) {
        if (onoff) {
            Color oldc = g.getColor();
            g.setColor(Color.black);

            for (int i = 0; i <= itemsNum - 1; i++) {
                drawMenuItem(i, (choosenBut == i +
1)&&(A3.getState() != 0), g);
            }
            g.setColor(oldc);
        }
    };

    public void pressUp() {
        A3.A(1);
    };

    public void pressDown() {
        A3.A(2);
    };

    public void pressRight() {
        A3.A(3);
    };

    public void pressLeft() {
        A3.A(4);
    };


    public void pressMenubut() {
        A3.A(11); /*В документации, это событие e11*/
    };
}
```

**Log.java**

```
import java.io.*;
/**
 * Класс поддержки ведения логфайла
 */
public class Log {
     public static FileOutputStream fileOutputStream;

     public synchronized static void write(String l) {
```

```
        try{
            for (int i = 0; i < l.length(); i++) {
                fileOutputStream.write((byte)(l.charAt(i)));
            };
        }catch (IOException e) { };
    };

    public synchronized static void writeln(String l) {
        try{
            for (int i = 0; i < l.length(); i++) {
                fileOutputStream.write((byte)(l.charAt(i)));
            };
            fileOutputStream.write(13);
        }catch (IOException e) { };
    };

    static {
        try{
           fileOutputStream = new FileOutputStream(new
File("log.txt"));
        }catch (FileNotFoundException e) { };
    }
}
```

**PlayAudio.java**

```
import javax.sound.sampled.*;
import java.io.*;
import java.util.Random;

class PlayAudio implements LineListener {

    private Clip line = null;

    private AudioPlayer owner;

    PlayAudio(AudioPlayer ap) {
        owner = ap;
    }

    public void openFile(String file) {

        try {
            // Создаем объект, представляющий файл
            File f = new File(file);
            // Получаем информацию о способе записи файла
            AudioFileFormat aff =
AudioSystem.getAudioFileFormat(f);
            // Получаем информацию о способе записи звука
            AudioFormat af = aff.getFormat();
            // Собираем всю информацию вместе,
```

```
            // добавляя сведения о классе
            DataLine.Info info = new DataLine.Info(Clip.class,
af);
            // Проверяем, можно ли проигрывать такой формат
            if (!AudioSystem.isLineSupported(info)) {
                System.err.println("Line is not supported");
                System.exit(0);
            }
            // Получаем линию связи с файлом
            line = (Clip) AudioSystem.getLine(info);
            // Создаем поток байтов из файла
            AudioInputStream ais =
AudioSystem.getAudioInputStream(f);
            // Открываем линию

            line.addLineListener(this);

            line.open(ais);
        } catch (Exception e) {
            System.err.println(e);
        }

    }

    public void start() {
        line.start();
    }

    public void stop() {
        line.stop();
    }

    public void close() {
        line.close();
    }

    /*
    * Informs the listener that a line's state has changed.
The listener can then invoke
    * <code>LineEvent</code> methods to obtain information
about the event.
    * @param event a line event that describes the change
    */
    public void update(LineEvent event) {
        if (event.getType().equals(LineEvent.Type.STOP)) {
            if (owner.playMode.getIsPlaying()) {
                switch
(owner.menuChooseRepeatMode.getSelectedItemIndex()) {
                    case 0:
                        int tn = owner.backGr.getTrackNum();
                        tn++;
```

```
                        if (tn == owner.backGr.getTrackTotal()
+ 1) tn = 1;
                        owner.backGr.setTrackNum(tn);
                        //owner.playAudio.openFile("c:\\mus\\"
+ tn + ".wav");

openFile(System.getProperty("user.dir") + "/mus/" +
owner.playMode.getTracks().get(tn - 1));
                        break;
                    case 1:

openFile(System.getProperty("user.dir") + "/mus/" +
owner.playMode.getTracks().get(owner.backGr.getTrackNum() -
1));
                        break;
                    case 2:
                        Random rand = new Random();
                        int tn2 = Math.abs(rand.nextInt()) %
owner.playMode.getTracks().size();
                        owner.backGr.setTrackNum(tn2+1);

openFile(System.getProperty("user.dir") + "/mus/" +
owner.playMode.getTracks().get(tn2));
                        break;
                }
                start();
                owner.clientArea.repaint();
            }

        } else if
(event.getType().equals(LineEvent.Type.CLOSE)) {
            /*
             *There is a bug in the jdk1.3/1.4.
             *It prevents correct termination of the VM.
             *So we have to exit ourselves.
             */
            System.exit(0);
        }


    }
}
```

**PlayMode.java**

```
import java.lang.reflect.Method;
import java.util.Vector;
import java.util.ArrayList;
import java.util.Random;
import java.awt.*;
import java.io.File;
```

```
import java.io.FilenameFilter;



public class PlayMode {
/*
*Класс PlayMode
*/
    class AutomatPlayMode extends Automat {

        public AutomatPlayMode() {
            setState(0);
            setName("Automat A1");
        }

        ;

        public boolean x1() {
            return (isPlaying);
        }

        ;

        public boolean x2() {
            return (isTracks);
        }

        ;
        /*
        * Функции-выходы
        */
        public void z1_2() {
            isPlaying = true;
            //owner.pressPlay();
           // owner.playAudio.openFile("c:\\mus\\" +
owner.backGr.getTrackNum() + ".wav");

owner.playAudio.openFile(System.getProperty("user.dir") +
"/mus/" + tracks.get(owner.backGr.getTrackNum() - 1));
            owner.playAudio.start();

        }

        public void z1_11() {
            isPlaying = true;
            owner.playAudio.stop();
            int tn = owner.backGr.getTrackNum();
            tn++;
            if (tn == owner.backGr.getTrackTotal() + 1) tn =
1;
```

```
            owner.backGr.setTrackNum(tn);
            //owner.playAudio.openFile("c:\\mus\\" + tn +
".wav");

owner.playAudio.openFile(System.getProperty("user.dir") +
"/mus/" + tracks.get(tn - 1));
            owner.playAudio.start();
        }

        public void z1_12() {
            isPlaying = true;
            owner.playAudio.stop();
            int tn = owner.backGr.getTrackNum();
            tn--;
             if (tn == 0) tn = owner.backGr.getTrackTotal();
            owner.backGr.setTrackNum(tn);
            //owner.playAudio.openFile("c:\\mus\\" + tn +
".wav");

owner.playAudio.openFile(System.getProperty("user.dir") +
"/mus/" + tracks.get(tn - 1));
            owner.playAudio.start();
        }

        ;

        public void z1_3() {
            isPlaying = false;
            owner.playAudio.stop();
        }

        ;

        public void z1_4() {
            isPlaying = false;
            owner.playAudio.stop();
        }

        ;

        /*
        * Реализация автомата
        */
        public void A(int e) {
            int y0 = getState();
            int yold = y0;

            Method method = null;
            Vector methodParametr = new Vector();
            Object object = null;
            Class classes[] = new Class[0];
```

```
            try {
                switch (y0) {
                    case 0:
                        if ((e == 1) && !(x2())) {
                            y0 = 0;
                            z1_4();
                            z1_3();
                        }
                    case 1:
                        /*if ((e == 1) && (x2()) && (!x1())) {
                            y0 = 1;
                            z1_2();
                        } else if ((e == 1) && (x1())) {
                            y0 = 1;
                            z1_3();
                        } else if (e == 4) {
                            z1_4();
                        }  */
                        if ((e == 1) && (x1())) {
                            y0 = 1;
                            z1_11();
                        }
                        if ((e == 2) && (x1())) {
                            y0 = 1;
                            z1_12();
                        } else if (e == 4) {
                            z1_4();
                        }
                        break;
                    case 2:
                        if (e == 1) {
                            y0 = 1;
                            z1_2();
                        } else if (e == 4) {
                            z1_4();
                        }
                        break;
                    default:
                        break;
                }

            } catch (Exception ex) {

            }
            setState(y0);


            if (method != null) {
                java.lang.Object parametrs[] =
methodParametr.toArray();
```

```
                try {
                    method.invoke(object, parametrs);
                } catch (java.lang.IllegalAccessException exp)
{
                } catch
(java.lang.reflect.InvocationTargetException exp) {
                }
            }


        }

        ;


    }

    ;


    public AutomatPlayMode a1;

    private boolean isPlaying;
    private boolean isTracks;

    private AudioPlayer owner;

    private ArrayList tracks;

     public ArrayList getTracks() {
        return tracks;
    }

    public void setTracks(ArrayList t){
        tracks = t;
    }

    public void deleteTrack(int i){
        tracks.remove(i);
    }

    public void deleteTracks(){
        tracks.clear();
    }

    public void deleteRandomTrack(){
        Random rand = new Random();
        tracks.remove(Math.abs(rand.nextInt())%tracks.size());
    }
```

```
     public String getTrack(int i){
        return tracks.get(i).toString();
    }

    public boolean getIsPlaying() {
        return isPlaying;
    }

    public void setIsPlaying(boolean cond) {
        isPlaying = cond;
    }

    public PlayMode(AudioPlayer o) {
        owner = o;
        a1 = new AutomatPlayMode();

        isPlaying = false;
        isTracks = true;

        tracks = new ArrayList();

        File file = new File(System.getProperty("user.dir") +
"/mus");

        FilenameFilter filter = new FilenameFilter() {
            public boolean accept(File dir, String name) {
                File file = new File(dir.toString() + "/" +
name);
                if (file.exists() && file.isFile())
                    tracks.add(file.getName());
                return (file.exists() && file.isFile());
            }

            ;
        };

        owner.backGr.setTrackTotal(file.list(filter).length);
    }

    ;


    public void draw(Graphics g) {

    }

    ;


    private boolean HaveTracks() {
        return !tracks.isEmpty();
```

```
    }

    public void pressMenubut(int n) {
        if (HaveTracks()) {
            isTracks = true;
            a1.setState(1);
        } else {
            isTracks = false;
            a1.setState(0);
        }
        isPlaying = true;
        a1.z1_2();
    }

    ;

    public void pressRight() {
        a1.A(3);
    }

    ;

    public void pressLeft() {
        a1.A(4);
    }

}
```

**ImageMap.java**

```
import java.awt.*;
import javax.swing.*;
import java.util.*;


abstract public class ImageMap {
   protected static Map imageMap = new HashMap(); // хэш
картинок
   protected static Map cursorMap = new HashMap(); // хэш
курсоров

   protected static class CursorInfo {
        public String path;
        public Point center;
        public String name;
        public CursorInfo(String tpath, Point tcenter, String
tname) {
             path = tpath;
             center = tcenter;
             name = tname;
        }
```

```
    };

     /**
      * Берем ImageIcon с путем path
      */
     public static ImageIcon getImageIcon(String path) {
         if (imageMap.containsKey(path))
              return (ImageIcon)imageMap.get(path);
         ImageIcon img = new ImageIcon(path);
         imageMap.put(path, img);
         return img;
     }

     /**
      * Берем Image с путем path
      */
     public static Image getImage(String path) {
         return getImageIcon(path).getImage();
     }


   public static Cursor getCursor(String path, Point center,
String name) {
         CursorInfo ci = new CursorInfo(path, center, name);
         if (cursorMap.containsKey(ci))
              return (Cursor)cursorMap.get(ci);
         Cursor cursor =
Toolkit.getDefaultToolkit().createCustomCursor(
                                        getImage(path),
                                  center,
                                  name);
          cursorMap.put(ci, cursor);
          return cursor;
   }

}
```