Санкт-Петербургский государственный университет
информационных технологий, механики и оптики
Факультет информационных технологий и программирования
Кафедра «Компьютерных технологий»

Е. В. Геращенко, А. П. Чеботаев

Моделирование портативного аудиоплеера на основе автоматного программирования

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

Санкт-Петербург
2008

Оглавление

Введение

Данный проект демонстрирует возможности автоматного программирования [1] в сфере разработки управляющих систем для устройств бытовой электроники на примере портативного аудиоплеера.

Устройства бытовой электроники, как правило, имеют ограниченный набор элементов управления, которые могут выполнять разные функции в разных контекстах, что весьма удачно вписывается в модель автоматного программирования. В то же время автоматная модель позволяет создавать надежные системы, поведение которых будет предсказуемо, так как оно полностью отражается в проектной документации, содержащей диаграммы переходов автоматов и их схемы связей с поставщиками событий и объектами управления.

В данном проекте при помощи автоматного подхода реализована модель портативного аудиоплеера, созданная при помощи инструментальной среды *UniMod* [2], которая представляет из себя надстройку над средой разработки *Eclipse*, добавляющую в нее такие концепции, как «Исполняемый *UML*» и «Разработка на базе моделей».

1. Описание проекта

1.1. Функциональность

В работе моделируется портативный аудиоплеер (далее для краткости будем использовать слово *плеер*). При этом в качестве базового прототипа был взят плеер *MSI MegaStick 528*. При этом авторы не ставили целью повторить полностью функциональность последнего. Моделировались только те функции, которые являются наиболее типичными для большинства плееров.

Предлагаемая модель обладает следующей функциональностью:

- воспроизведение аудиофайлов формата *MPEG-1 Audio Layer 3* (расширение – *mp3*) и *Waveform* (расширение – *wav*), находящихся в одной директории (в представлении плеера они рассматриваются как *треки*);
- четыре режима воспроизведения:
    1. Последовательное (после окончания последнего трека воспроизведение прекращается) – режим по умолчанию.
    2. Повтор одного трека.
    3. Последовательное с повтором (после завершения последнего трека воспроизведение начинается сначала, с первого).
    4. В случайном порядке.
- приостановка и возобновление воспроизведения;
- переключение в начало трека, на предыдущий и на следующий треки;
- изменение громкости;
- блокировка кнопок;
- отображение информации о статусе на экране:
    - название трека;
    - длительность трека и прогресс его воспроизведения – насколько он прослушан (в минутах и секундах, а также в виде полоски);
    - статус воспроизведения (включено или приостановлено);
    - текущий режим воспроизведения;
    - включена ли блокировка кнопок;

- громкость воспроизведения;
- выключение плеера при нажатии и удерживании кнопки «выключить» в течение 3 с (во избежание случайного выключения).

1.2. Пользовательский интерфейс

Разработанная модель плеера является графическим приложением, которое состоит из одного окна. Это окно имитирует главную панель плеера (рис. 1).

Рис. 1. Вид окна приложения

Левая часть окна представляет собой ЖК-экран, который используется для отображения статуса воспроизведения. Правая часть окна содержит элементы управления плеером (кнопки) и кнопку вызова окна информации о проекте (знак «?»). Элементы управления описаны в табл. 1.

Таблица 1. Элементы управления

Кнопка	Назначение
⏯	Включение и приостановка воспроизведения
⏮	Переход в начало трека или на предыдущий (если текущий трек проигрывался не более 5 с)
⏭	Переход на следующий трек
+	Увеличение громкости на один шаг (всего 10 градаций)
−	Уменьшение громкости на один шаг (всего 10 градаций)

	Включение и выключение плеера. Для выключения необходимо нажать и удерживать кнопку в течение 3 с
	Включение и выключение блокировки кнопок. Данная кнопка, в отличие от остальных, является переключателем, имеющим два положения: «включен» (кнопка нажата) и «выключен» (кнопка отпущена)
	Выбор режима воспроизведения

2. Проектирование

Проектирование приложения было выполнено при помощи инструментального средства *UniMod*. Это средство позволило визуально построить схему связей между формальными объектами автоматной модели: поставщиками событий, автоматами и объектами управления.

2.1. Схема связей

Схема связей в виде диаграммы классов данного приложения приведена на рис. 2. На ней представлены два поставщика событий *p1*, *p2*, два автомата *A0*, *A1* и один объект управления *o*.

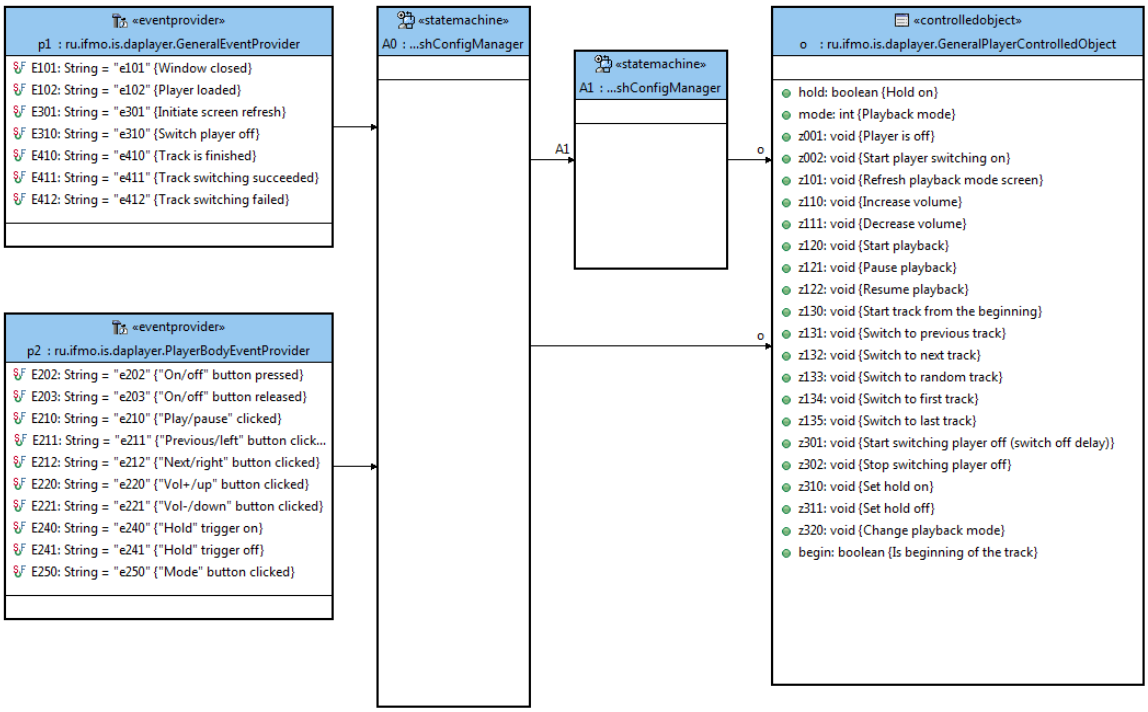

Рис. 2. Схема связей

2.2. Поставщики событий

2.2.1.Основной поставщик событий (*p1*)

События, относящиеся к основному поставщику, инициируются окном плеера, его внутренними процессами и частью системы, отвечающей за переключение между треками. Данные события описаны в табл. 2.

Таблица 2. События основного поставщика

Событие	**Описание**
`e101`	Закрытие окна приложения
`e102`	Плеер загружен
`e301`	Необходимо обновить информацию на экране (инициируется периодически из отдельного потока)
`e310`	Выключить плеер (инициируется после трехсекундного удерживания кнопки «включение/выключение»)
`e410`	Трек завершен (инициируется из модуля воспроизведения после того, как трек воспроизведен полностью)
`e411`	Переход на трек произошел (инициируется из модуля воспроизведения после попытки перейти на другой трек)
`e412`	Переход на трек не произошел (инициируется из модуля воспроизведения после попытки перейти на другой трек)

2.2.2. Поставщик событий элементов управления (*p2*)

События, относящиеся к данному поставщику, инициируется элементами управления плеером (кнопками). Данные события описаны в табл. 3.

Таблица 3. События элементов управления

Событие	**Описание**
`e202`	Нажата, но не отпущена кнопка «включение/выключение».
`e203`	Отпущена кнопка «включение/выключение»
`e210`	Нажата и отпущена кнопка «воспроизведение/пауза»
`e211`	Нажата и отпущена кнопка «предыдущий трек»
`e212`	Нажата и отпущена кнопка «следующий трек»
`e220`	Нажата и отпущена кнопка «увеличить громкость»
`e221`	Нажата и отпущена кнопка «уменьшить громкость»
`e240`	Переключатель «блокировка кнопок» переведен в состояние «включен»
`e241`	Переключатель «блокировка кнопок» переведен в состояние «выключен»
`e250`	Нажата и отпущена кнопка «переключение режима воспроизведения»

2.3. Объект управления (*o*)

Объект управления содержит все входные и выходные воздействия, относящиеся к модели плеера. Они описаны в табл. 4, 5 соответственно.

Таблица 4. Входные воздействия объекта управления

Воздействие	**Тип**	**Описание**
`hold`	Логический	Включена ли блокировка кнопок
`mode`	Целое число от нуля до трех	Текущий режим воспроизведения, описанный числом. Режимы перечислены в п. 1.1. Обращаем внимание, что нумерация режимов в программе ведется с нуля, а в документации – с единицы
`begin`	Логический	Находится ли текущий трек в начале (в пределах первых 5 с)

Таблица 5. Выходные воздействия объекта управления

Воздействие	**Описание**
`z001`	Плеер выключен. Все кнопки модели, кроме включения/выключения блокировки и включения/выключения плеера, становятся недоступными для нажатия
`z002`	Начать включение плеера. На реальном устройстве включение происходит несколько секунд, так как необходимо обойти все дерево директорий на носителе и найти все аудиофайлы. В модели данная задержка эмулируется программным образом, а после нее инициируется событие `e102`

z101	Произвести обновление экрана
z110	Увеличить громкость на один шаг
z111	Уменьшить громкость на один шаг
z120	Начать воспроизведение
z121	Приостановить воспроизведение
z122	Продолжить воспроизведение
z130	Перейти в начало трека
z131	Переключиться на предыдущий трек. При этом в зависимости от того, был ли успешен переход, инициируется событие e411 или e412. Переход может не удаться, если данный трек – первый
z132	Переключиться на следующий трек. При этом в зависимости от того, был ли успешен переход, инициируется событие e411 или e412. Переход может не удаться, если данный трек – последний
z133	Переключиться на случайный трек
z134	Переключиться на первый трек
z135	Переключиться на последний трек
z301	Начать выключение плеера. При этом инициируется задержка в 3 с, после которой плеер выключается. Если при этом пользователь успевает отпустить кнопку выключения –

	активизируется воздействие z302, то плеер не выключается
z302	Прервать выключение плеера (описание воздействия z301)
z310	Включить блокировку кнопок
z311	Выключить блокировку кнопок
z320	Изменить режим воспроизведения на следующий. Режимы переключаются последовательно циклически в порядке, указанном в п. 1.1

2.4. Автоматы

2.4.1.Основной автомат (*A0*)

Данный автомат является базовым в данном приложении. Он управляет основным циклом работы плеера, обеспечивая процессы его включения/выключения, изменения режима воспроизведения, включения/снятия блокировки кнопок и изменения громкости. Схема автомата представлена на рис. 3, описание его состояний – в табл. 6.

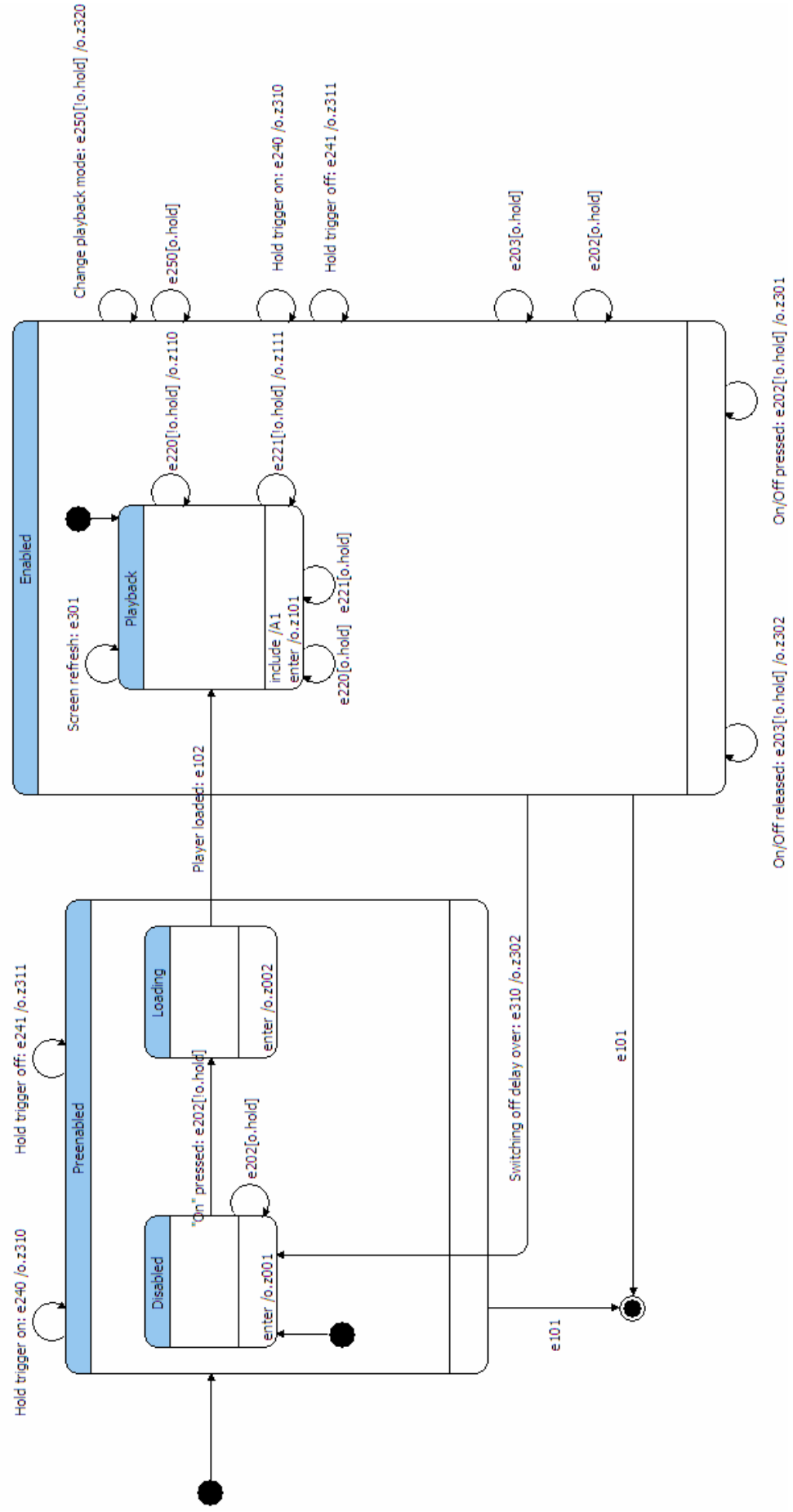

Рис. 3.Схема основного автомата

Таблица 6. Описание состояний основного автомата

Состояние	**Описание**
Disabled	В данном состоянии автомат находится сразу после запуска программы или после выключения плеера
Loading	Плеер загружается. В это состояние он переходит после нажатия кнопки включения
Playback	Состояние воспроизведения. В этом состоянии плеер полностью включен и реагирует на все нажатия кнопок в соответствии с их предназначением. В это состояние вложен автомат режима воспроизведения (*A1*)

2.4.2.Автомат режима воспроизведения (*A1*)

Этот автомат описывает логику режима воспроизведения. Он обеспечивает процессы воспроизведения, его приостановления, переходов между треками. Схема автомата представлена на рис. 4, описание его состояний – в табл. 7.

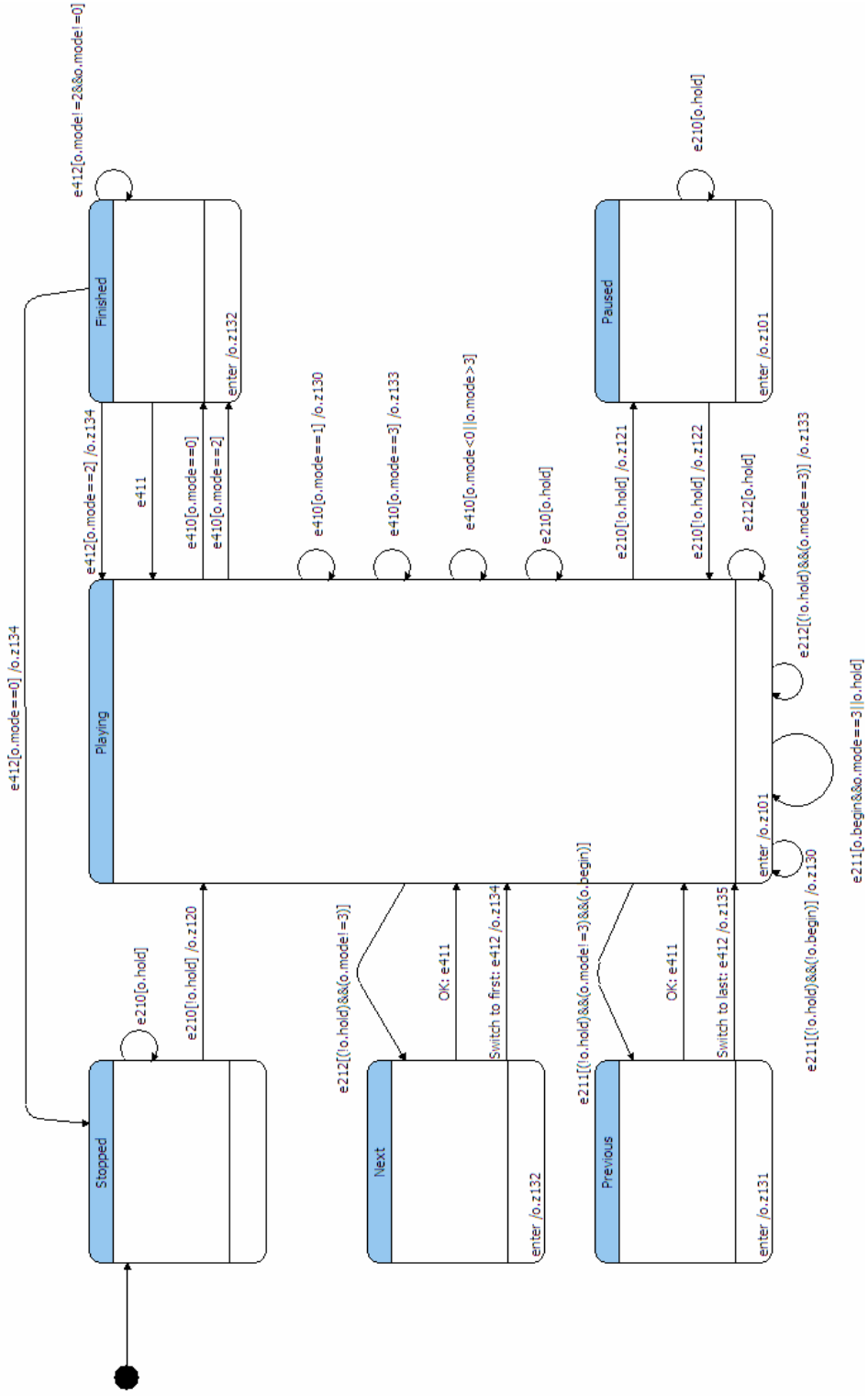

Рис. 4. Схема автомата режима воспроизведения

Таблица 7. Описание состояний режима воспроизведения

Состояние	**Описание**
Stopped	Воспроизведение остановлено. Данное состояние является для автомата режима воспроизведения начальным.
Playing	Воспроизведение. В этом состоянии автомат находится во время воспроизведения музыки.
Finished	Трек закончился. В это состояние автомат переходит из состояния *Playing*, когда воспроизводимый трек закончился, в случае если текущий режим воспроизведения – режим 1 или режим 3 (п. 1.1). При входе плеер переходит на следующий трек. В случае успеха автомат переходит обратно в состояние *Playing*. Если же переход не произошел, то автомат переходит в состояние *Stopped* – в случае режима 1, или в состояние *Playing* с одновременным переходом плеера на первый трек – в случае режима 3.
Paused	Воспроизведение приостановлено. В данное состояние автомат переходит из состояния *Playing* при нажатии кнопки приостановления, и обратно в состояние *Playing* – при повторном нажатии этой кнопки.
Next	Переход на следующий трек. В данное состояние переход производится из состояния *Playing* при нажатии кнопки «следующий трек», если текущий режим воспроизведения – не четвертый (случайный порядок). В данном состоянии плеер пытается перейти на следующий трек. Если это не получается, то есть текущий трек – последний, то плеер переходит на первый трек. В любом случае автомат переходит в состояние *Playing*.
Previous	Переход на предыдущий трек. В данное состояние переход производится из состояния *Playing* при нажатии кнопки

	«предыдущий трек», если текущий режим воспроизведения – не четверной (случайный порядок), а также плеер находится в начале текущего трека, то есть, в пределах первых 5 с. Если трек находится не в начале, то плеер переходит в начало текущего трека, не переходя в данное состояние. В данном состоянии плеер пытается перейти на предыдущий трек. Если это не получается, то есть текущий трек – первый, то плеер переходит на последний трек. В любом случае автомат переходит в состояние *Playing*.

3. Реализация

3.1. Используемые средства

В реализации проекта использовались следующие средства:

- *Java Development Kit* (*JDK*) 6;
- среда разработки *Eclipse*;
- средство проектирования *UniMod*;
- библиотека *Java Media Framework* для воспроизведения аудиофайлов.

3.2. Структура программы

Программа состоит из классов, созданных при помощи UniMod и нескольких других, отвечающих за графический интерфейс и воспроизведение аудиофайлов.

3.2.1. Поставщики событий и объекты управления

- `GeneralEventProvider` – основной поставщик событий;
- `PlayerBodyEventProvider` – поставщик событий элементов управления;

- `GeneralPlayerControlledObject` – объект управления.

3.2.2. Прочие классы

- `PlayerModel` – модель плеера. Этот класс является главным в проекте. Он отвечает за создание графического интерфейса, «движка» автоматной модели и модуля воспроизведения;

- `PlayerScreen` – экран плеера. Класс представляет собой экран плеера в виде компонента графического интерфейса;

- `PlaybackUnit` – модуль воспроизведения. Является промежуточным звеном между программой и *Java Media Framework*, отвечая за собственно воспроизведение аудиофайлов, переход от одного к другому, изменение громкости и обеспечение информацией для экрана плеера.

3.3. Запуск программы

Использованное в разработке программное средство *UniMod* поддерживает два подхода к исполнению программ: интерпретационный и компилятивный.

3.3.1.Интерпретационный подход

При интерпретационном подходе *UniMod* генерирует *XML*-файлы с описанием логики автоматов, которые при запуске программы интерпретируются *UniMod*-библиотекой. Для запуска требуется вышеуказанная библиотека. Схема работы программы при данном подходе изображена на рис. 5.

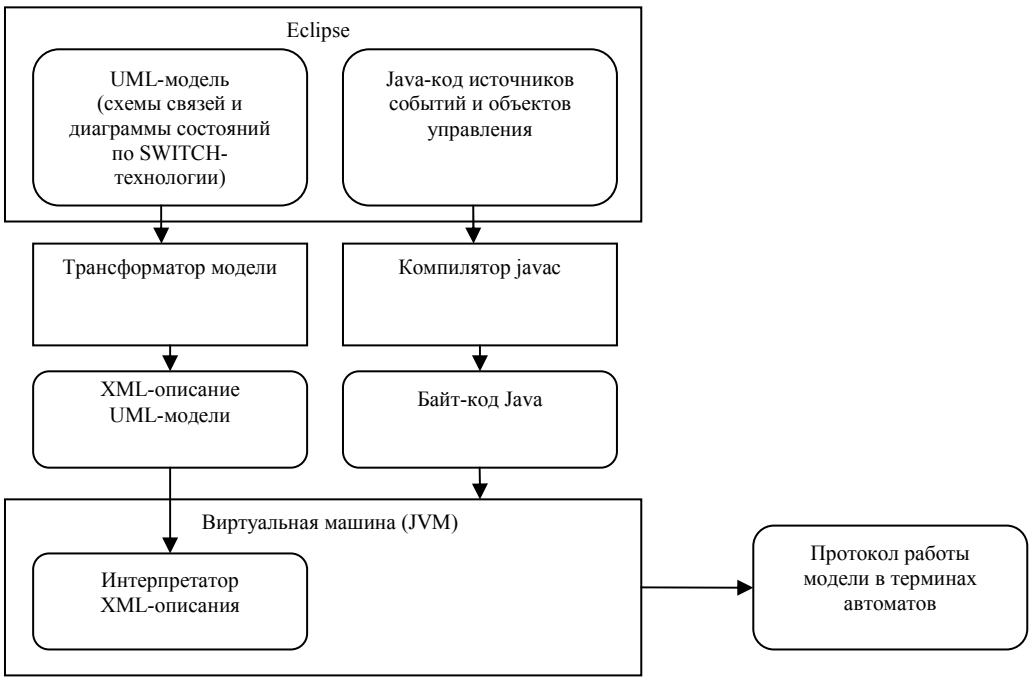

Рис. 5. Схема работы при интерпретационном подходе

3.3.2. Компилятивный подход

При компилятивном подходе автоматы, нарисованные в виде схемы *UniMod*, преобразовываются в исходный код на *Java*, который затем компилируется наряду с остальными исходными кодами. Для запуска требуется базовая *UniMod*-библиотека, имеющаяся в поставке этого инструментального средства. Схема работы при данном подходе изображена на рис. 6.

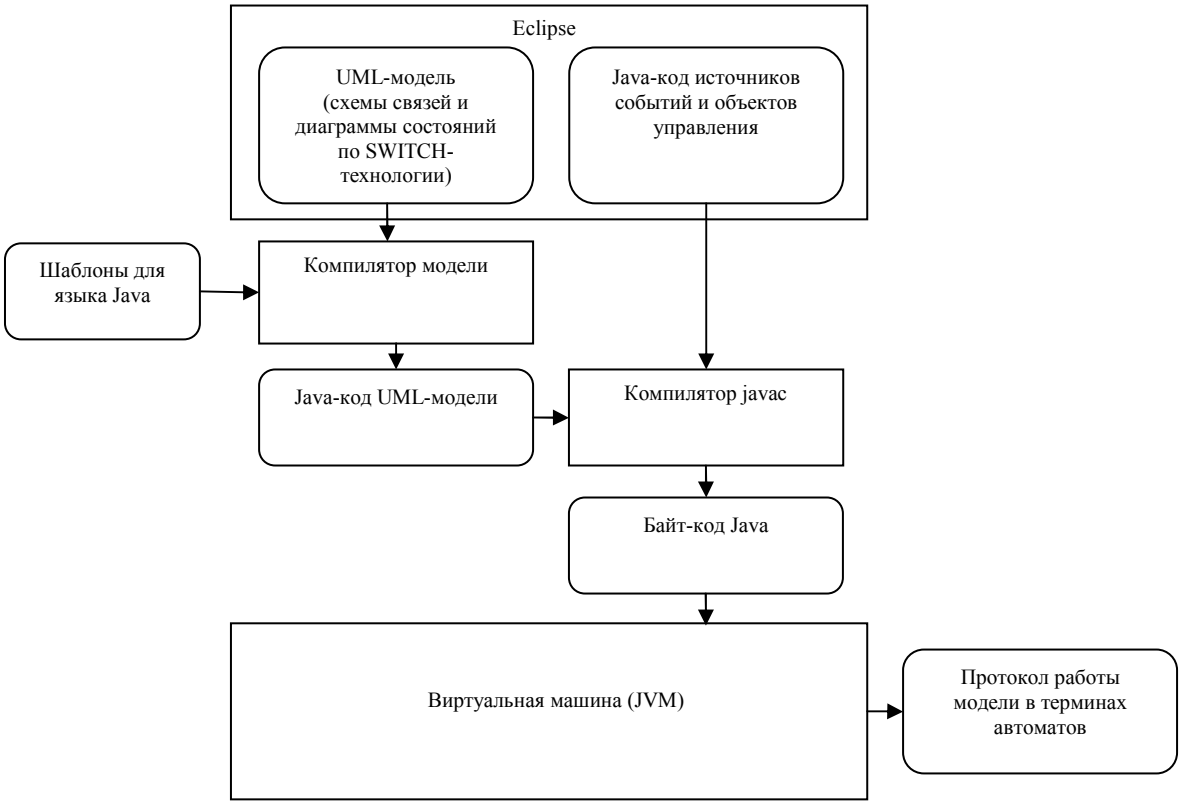

Рис. 6. Схема работы при компилятивном подходе

3.3.3. Инструкции для запуска

Для запуска проекта нужен установленный пакет *Java Runtime Environment (JRE) 6*. Все остальные необходимые библиотеки поставляются в комплекте.

Для запуска необходимо распаковать архив с проектом в пустую папку. Затем поместить в поддиректорию *music* аудиофайлы в формате *MP3* или *WAV*. После этого нужно запустить *JAR*-файл с проектом:

```
java –jar daplayer.jar
```

Заключение

Данный проект показал целесообразность и удобство автоматного подхода для разработки систем управления устройствами бытовой электроники. Данный подход позволяет легко добавлять новую функциональность путем добавления новых состояний, переходов, событий и внешних воздействий в автоматную модель системы. Наглядность и доступность этих изменений повышается при использовании средств визуального построения схемы связей и автоматов.

Источники

1. *Сайт по автоматному программированию и мотивации к творчеству*. http://is.ifmo.ru/
2. *Сайт проекта UniMod на SourceForge*. http://unimod.sourceforge.net/
3. *Сайт Java Media Framework*. http://java.sun.com/javase/technologies/desktop/media/jmf/
4. *Статья об автоматном программировании в Википедии*. http://ru.wikipedia.org/wiki/Автоматное_программирование
5. *Гуров В. С., Мазин М. А., Нарвский А. С., Шалыто А. А*. UML. SWITCH-технология. Eclipse // Информационно-управляющие системы. 2004. № 6, с. 12–17. http://is.ifmo.ru/works/uml-switch-eclipse/
6. *Шалыто А.А*. Новая инициатива в программировании. Движение за открытую проектную документацию // Мир ПК. 2003. № 9, с. 52–56. http://is.ifmo.ru/works/open_doc/

Приложения

Приложение А. *XML*-описание модели

A.1. `A0.xml`

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE model PUBLIC "-//eVelopers Corp.//DTD State machine model V1.0//EN"
"http://www.evelopers.com/dtd/unimod/statemachine.dtd">
<model name="ru.ifmo.is.daplayer.Player">
  <controlledObject class="ru.ifmo.is.daplayer.GeneralPlayerControlledObject"
name="o"/>
  <eventProvider class="ru.ifmo.is.daplayer.GeneralEventProvider" name="p1">
    <association clientRole="p1" targetRef="A0"/>
  </eventProvider>
  <eventProvider class="ru.ifmo.is.daplayer.PlayerBodyEventProvider"
name="p2">
    <association clientRole="p2" targetRef="A0"/>
  </eventProvider>
  <rootStateMachine>
    <stateMachineRef name="A0"/>
  </rootStateMachine>
  <stateMachine name="A0">
    <configStore
class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
    <association clientRole="A0" supplierRole="o" targetRef="o"/>
    <association clientRole="A0" supplierRole="A1" targetRef="A1"/>
    <state name="Top" type="NORMAL">
      <state name="Enabled" type="NORMAL">
        <state name="s4" type="INITIAL"/>
        <state name="Playback" type="NORMAL">
          <stateMachineRef name="A1"/>
          <outputAction ident="o.z101"/>
        </state>
      </state>
      <state name="Preenabled" type="NORMAL">
        <state name="Disabled" type="NORMAL">
          <outputAction ident="o.z001"/>
        </state>
        <state name="Loading" type="NORMAL">
          <outputAction ident="o.z002"/>
        </state>
        <state name="s3" type="INITIAL"/>
      </state>
      <state name="s1" type="INITIAL"/>
      <state name="s2" type="FINAL"/>
    </state>
    <transition event="e240" name="Hold trigger on" sourceRef="Enabled"
targetRef="Enabled">
      <outputAction ident="o.z310"/>
    </transition>
    <transition event="e241" name="Hold trigger off" sourceRef="Enabled"
targetRef="Enabled">
      <outputAction ident="o.z311"/>
    </transition>
    <transition event="e203" guard="!o.hold" name="On/Off released"
sourceRef="Enabled" targetRef="Enabled">
      <outputAction ident="o.z302"/>
    </transition>
```

```
    <transition event="e202" guard="!o.hold" name="On/Off pressed"
sourceRef="Enabled" targetRef="Enabled">
      <outputAction ident="o.z301"/>
    </transition>
    <transition event="e203" guard="o.hold" sourceRef="Enabled"
targetRef="Enabled"/>
    <transition event="e202" guard="o.hold" sourceRef="Enabled"
targetRef="Enabled"/>
    <transition event="e250" guard="!o.hold" name="Change playback mode"
sourceRef="Enabled" targetRef="Enabled">
      <outputAction ident="o.z320"/>
    </transition>
    <transition event="e250" guard="o.hold" sourceRef="Enabled"
targetRef="Enabled"/>
    <transition event="e310" name="Switching off delay over"
sourceRef="Enabled" targetRef="Disabled">
      <outputAction ident="o.z302"/>
    </transition>
    <transition event="e101" sourceRef="Enabled" targetRef="s2"/>
    <transition sourceRef="s4" targetRef="Playback"/>
    <transition event="e221" guard="!o.hold" sourceRef="Playback"
targetRef="Playback">
      <outputAction ident="o.z111"/>
    </transition>
    <transition event="e220" guard="!o.hold" sourceRef="Playback"
targetRef="Playback">
      <outputAction ident="o.z110"/>
    </transition>
    <transition event="e301" name="Screen refresh" sourceRef="Playback"
targetRef="Playback"/>
    <transition event="e220" guard="o.hold" sourceRef="Playback"
targetRef="Playback"/>
    <transition event="e221" guard="o.hold" sourceRef="Playback"
targetRef="Playback"/>
    <transition event="e240" name="Hold trigger on" sourceRef="Preenabled"
targetRef="Preenabled">
      <outputAction ident="o.z310"/>
    </transition>
    <transition event="e241" name="Hold trigger off" sourceRef="Preenabled"
targetRef="Preenabled">
      <outputAction ident="o.z311"/>
    </transition>
    <transition event="e101" sourceRef="Preenabled" targetRef="s2"/>
    <transition event="e202" guard="o.hold" sourceRef="Disabled"
targetRef="Disabled"/>
    <transition event="e202" guard="!o.hold" name="&amp;quot;On&amp;quot; pressed"
sourceRef="Disabled" targetRef="Loading"/>
    <transition event="e102" name="Player loaded" sourceRef="Loading"
targetRef="Playback"/>
    <transition sourceRef="s3" targetRef="Disabled"/>
    <transition sourceRef="s1" targetRef="Preenabled"/>
  </stateMachine>
  <stateMachine name="A1">
    <configStore
class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
    <association clientRole="A1" supplierRole="o" targetRef="o"/>
    <state name="Top" type="NORMAL">
      <state name="Stopped" type="NORMAL"/>
      <state name="Playing" type="NORMAL">
        <outputAction ident="o.z101"/>
      </state>
      <state name="Finished" type="NORMAL">
        <outputAction ident="o.z132"/>
```

```
      </state>
      <state name="s1" type="INITIAL"/>
      <state name="Next" type="NORMAL">
        <outputAction ident="o.z132"/>
      </state>
      <state name="Previous" type="NORMAL">
        <outputAction ident="o.z131"/>
      </state>
      <state name="Paused" type="NORMAL">
        <outputAction ident="o.z101"/>
      </state>
    </state>
    <transition event="e210" guard="o.hold" sourceRef="Stopped"
targetRef="Stopped"/>
    <transition event="e210" guard="!o.hold" sourceRef="Stopped"
targetRef="Playing">
      <outputAction ident="o.z120"/>
    </transition>
    <transition event="e212" guard="(!o.hold)&amp;&amp;(o.mode==3)"
sourceRef="Playing" targetRef="Playing">
      <outputAction ident="o.z133"/>
    </transition>
    <transition event="e410" guard="o.mode==1" sourceRef="Playing"
targetRef="Playing">
      <outputAction ident="o.z130"/>
    </transition>
    <transition event="e410" guard="o.mode==3" sourceRef="Playing"
targetRef="Playing">
      <outputAction ident="o.z133"/>
    </transition>
    <transition event="e210" guard="o.hold" sourceRef="Playing"
targetRef="Playing"/>
    <transition event="e212" guard="o.hold" sourceRef="Playing"
targetRef="Playing"/>
    <transition event="e211" guard="(!o.hold)&amp;&amp;(!o.begin)"
sourceRef="Playing" targetRef="Playing">
      <outputAction ident="o.z130"/>
    </transition>
    <transition event="e211" guard="o.begin&amp;&amp;o.mode==3||o.hold"
sourceRef="Playing" targetRef="Playing"/>
    <transition event="e410" guard="o.mode&lt;0||o.mode&gt;3"
sourceRef="Playing" targetRef="Playing"/>
    <transition event="e410" guard="o.mode==0" sourceRef="Playing"
targetRef="Finished"/>
    <transition event="e410" guard="o.mode==2" sourceRef="Playing"
targetRef="Finished"/>
    <transition event="e212" guard="(!o.hold)&amp;&amp;(o.mode!=3)"
sourceRef="Playing" targetRef="Next"/>
    <transition event="e211"
guard="(!o.hold)&amp;&amp;(o.mode!=3)&amp;&amp;(o.begin)" sourceRef="Playing"
targetRef="Previous"/>
    <transition event="e210" guard="!o.hold" sourceRef="Playing"
targetRef="Paused">
      <outputAction ident="o.z121"/>
    </transition>
    <transition event="e412" guard="o.mode==0" sourceRef="Finished"
targetRef="Stopped">
      <outputAction ident="o.z134"/>
    </transition>
    <transition event="e411" sourceRef="Finished" targetRef="Playing"/>
    <transition event="e412" guard="o.mode==2" sourceRef="Finished"
targetRef="Playing">
      <outputAction ident="o.z134"/>
```

```
    </transition>
    <transition event="e412" guard="o.mode!=2&amp;&amp;o.mode!=0"
sourceRef="Finished" targetRef="Finished"/>
    <transition sourceRef="s1" targetRef="Stopped"/>
    <transition event="e411" name="OK" sourceRef="Next" targetRef="Playing"/>
    <transition event="e412" name="Switch to first" sourceRef="Next"
targetRef="Playing">
      <outputAction ident="o.z134"/>
    </transition>
    <transition event="e411" name="OK" sourceRef="Previous"
targetRef="Playing"/>
    <transition event="e412" name="Switch to last" sourceRef="Previous"
targetRef="Playing">
      <outputAction ident="o.z135"/>
    </transition>
    <transition event="e210" guard="!o.hold" sourceRef="Paused"
targetRef="Playing">
      <outputAction ident="o.z122"/>
    </transition>
    <transition event="e210" guard="o.hold" sourceRef="Paused"
targetRef="Paused"/>
  </stateMachine>
</model>
```

A.2. A1.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE model PUBLIC "-//eVelopers Corp.//DTD State machine model V1.0//EN"
"http://www.evelopers.com/dtd/unimod/statemachine.dtd">
<model name="ru.ifmo.is.daplayer.Player">
  <controlledObject class="ru.ifmo.is.daplayer.GeneralPlayerControlledObject"
name="o"/>
  <eventProvider class="ru.ifmo.is.daplayer.GeneralEventProvider" name="p1">
    <association clientRole="p1" targetRef="A0"/>
  </eventProvider>
  <eventProvider class="ru.ifmo.is.daplayer.PlayerBodyEventProvider"
name="p2">
    <association clientRole="p2" targetRef="A0"/>
  </eventProvider>
  <rootStateMachine>
    <stateMachineRef name="A1"/>
  </rootStateMachine>
  <stateMachine name="A0">
    <configStore
class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
    <association clientRole="A0" supplierRole="o" targetRef="o"/>
    <association clientRole="A0" supplierRole="A1" targetRef="A1"/>
    <state name="Top" type="NORMAL">
      <state name="Enabled" type="NORMAL">
        <state name="s4" type="INITIAL"/>
        <state name="Playback" type="NORMAL">
          <stateMachineRef name="A1"/>
          <outputAction ident="o.z101"/>
        </state>
      </state>
      <state name="Preenabled" type="NORMAL">
        <state name="Disabled" type="NORMAL">
          <outputAction ident="o.z001"/>
        </state>
        <state name="Loading" type="NORMAL">
          <outputAction ident="o.z002"/>
        </state>
        <state name="s3" type="INITIAL"/>
```

```
      </state>
      <state name="s1" type="INITIAL"/>
      <state name="s2" type="FINAL"/>
    </state>
    <transition event="e240" name="Hold trigger on" sourceRef="Enabled"
targetRef="Enabled">
      <outputAction ident="o.z310"/>
    </transition>
    <transition event="e241" name="Hold trigger off" sourceRef="Enabled"
targetRef="Enabled">
      <outputAction ident="o.z311"/>
    </transition>
    <transition event="e203" guard="!o.hold" name="On/Off released"
sourceRef="Enabled" targetRef="Enabled">
      <outputAction ident="o.z302"/>
    </transition>
    <transition event="e202" guard="!o.hold" name="On/Off pressed"
sourceRef="Enabled" targetRef="Enabled">
      <outputAction ident="o.z301"/>
    </transition>
    <transition event="e203" guard="o.hold" sourceRef="Enabled"
targetRef="Enabled"/>
    <transition event="e202" guard="o.hold" sourceRef="Enabled"
targetRef="Enabled"/>
    <transition event="e250" guard="!o.hold" name="Change playback mode"
sourceRef="Enabled" targetRef="Enabled">
      <outputAction ident="o.z320"/>
    </transition>
    <transition event="e250" guard="o.hold" sourceRef="Enabled"
targetRef="Enabled"/>
    <transition event="e310" name="Switching off delay over"
sourceRef="Enabled" targetRef="Disabled">
      <outputAction ident="o.z302"/>
    </transition>
    <transition event="e101" sourceRef="Enabled" targetRef="s2"/>
    <transition sourceRef="s4" targetRef="Playback"/>
    <transition event="e221" guard="!o.hold" sourceRef="Playback"
targetRef="Playback">
      <outputAction ident="o.z111"/>
    </transition>
    <transition event="e220" guard="!o.hold" sourceRef="Playback"
targetRef="Playback">
      <outputAction ident="o.z110"/>
    </transition>
    <transition event="e301" name="Screen refresh" sourceRef="Playback"
targetRef="Playback"/>
    <transition event="e220" guard="o.hold" sourceRef="Playback"
targetRef="Playback"/>
    <transition event="e221" guard="o.hold" sourceRef="Playback"
targetRef="Playback"/>
    <transition event="e240" name="Hold trigger on" sourceRef="Preenabled"
targetRef="Preenabled">
      <outputAction ident="o.z310"/>
    </transition>
    <transition event="e241" name="Hold trigger off" sourceRef="Preenabled"
targetRef="Preenabled">
      <outputAction ident="o.z311"/>
    </transition>
    <transition event="e101" sourceRef="Preenabled" targetRef="s2"/>
    <transition event="e202" guard="o.hold" sourceRef="Disabled"
targetRef="Disabled"/>
    <transition event="e202" guard="!o.hold" name="&amp;quot;On&amp;quot; pressed"
sourceRef="Disabled" targetRef="Loading"/>
```

```
    <transition event="e102" name="Player loaded" sourceRef="Loading"
targetRef="Playback"/>
    <transition sourceRef="s3" targetRef="Disabled"/>
    <transition sourceRef="s1" targetRef="Preenabled"/>
  </stateMachine>
  <stateMachine name="A1">
    <configStore
class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
    <association clientRole="A1" supplierRole="o" targetRef="o"/>
    <state name="Top" type="NORMAL">
      <state name="Stopped" type="NORMAL"/>
      <state name="Playing" type="NORMAL">
        <outputAction ident="o.z101"/>
      </state>
      <state name="Finished" type="NORMAL">
        <outputAction ident="o.z132"/>
      </state>
      <state name="s1" type="INITIAL"/>
      <state name="Next" type="NORMAL">
        <outputAction ident="o.z132"/>
      </state>
      <state name="Previous" type="NORMAL">
        <outputAction ident="o.z131"/>
      </state>
      <state name="Paused" type="NORMAL">
        <outputAction ident="o.z101"/>
      </state>
    </state>
    <transition event="e210" guard="o.hold" sourceRef="Stopped"
targetRef="Stopped"/>
    <transition event="e210" guard="!o.hold" sourceRef="Stopped"
targetRef="Playing">
      <outputAction ident="o.z120"/>
    </transition>
    <transition event="e212" guard="(!o.hold)&amp;amp;&amp;amp;(o.mode==3)"
sourceRef="Playing" targetRef="Playing">
      <outputAction ident="o.z133"/>
    </transition>
    <transition event="e410" guard="o.mode==1" sourceRef="Playing"
targetRef="Playing">
      <outputAction ident="o.z130"/>
    </transition>
    <transition event="e410" guard="o.mode==3" sourceRef="Playing"
targetRef="Playing">
      <outputAction ident="o.z133"/>
    </transition>
    <transition event="e210" guard="o.hold" sourceRef="Playing"
targetRef="Playing"/>
    <transition event="e212" guard="o.hold" sourceRef="Playing"
targetRef="Playing"/>
    <transition event="e211" guard="(!o.hold)&amp;amp;&amp;amp;(!o.begin)"
sourceRef="Playing" targetRef="Playing">
      <outputAction ident="o.z130"/>
    </transition>
    <transition event="e211" guard="o.begin&amp;amp;&amp;amp;o.mode==3||o.hold"
sourceRef="Playing" targetRef="Playing"/>
    <transition event="e410" guard="o.mode&amp;lt;0||o.mode&amp;gt;3"
sourceRef="Playing" targetRef="Playing"/>
    <transition event="e410" guard="o.mode==0" sourceRef="Playing"
targetRef="Finished"/>
    <transition event="e410" guard="o.mode==2" sourceRef="Playing"
targetRef="Finished"/>
```

```
    <transition event="e212" guard="(!o.hold)&amp;amp;&amp;amp;(o.mode!=3)"
sourceRef="Playing" targetRef="Next"/>
    <transition event="e211"
guard="(!o.hold)&amp;amp;&amp;amp;(o.mode!=3)&amp;amp;&amp;amp;(o.begin)" sourceRef="Playing"
targetRef="Previous"/>
    <transition event="e210" guard="!o.hold" sourceRef="Playing"
targetRef="Paused">
      <outputAction ident="o.z121"/>
    </transition>
    <transition event="e412" guard="o.mode==0" sourceRef="Finished"
targetRef="Stopped">
      <outputAction ident="o.z134"/>
    </transition>
    <transition event="e411" sourceRef="Finished" targetRef="Playing"/>
    <transition event="e412" guard="o.mode==2" sourceRef="Finished"
targetRef="Playing">
      <outputAction ident="o.z134"/>
    </transition>
    <transition event="e412" guard="o.mode!=2&amp;amp;&amp;amp;o.mode!=0"
sourceRef="Finished" targetRef="Finished"/>
    <transition sourceRef="s1" targetRef="Stopped"/>
    <transition event="e411" name="OK" sourceRef="Next" targetRef="Playing"/>
    <transition event="e412" name="Switch to first" sourceRef="Next"
targetRef="Playing">
      <outputAction ident="o.z134"/>
    </transition>
    <transition event="e411" name="OK" sourceRef="Previous"
targetRef="Playing"/>
    <transition event="e412" name="Switch to last" sourceRef="Previous"
targetRef="Playing">
      <outputAction ident="o.z135"/>
    </transition>
    <transition event="e210" guard="!o.hold" sourceRef="Paused"
targetRef="Playing">
      <outputAction ident="o.z122"/>
    </transition>
    <transition event="e210" guard="o.hold" sourceRef="Paused"
targetRef="Paused"/>
  </stateMachine>
</model>
```

Приложение В. Исходные коды

B.1. GeneralEventProvider.java

```
package ru.ifmo.is.daplayer;

import com.evelopers.common.exception.CommonException;
import com.evelopers.unimod.runtime.EventProvider;
import com.evelopers.unimod.runtime.ModelEngine;

public class GeneralEventProvider implements EventProvider {

    /**
     * @unimod.event.descr Window closed
     */
    public static final String E101 = "e101";
    /**
     * @unimod.event.descr Player loaded
     */
    public static final String E102 = "e102";
    /**
```

```
     * @unimod.event.descr Initiate screen refresh
     */
    public static final String E301 = "e301";
    /**
     * @unimod.event.descr Switch player off
     */
    public static final String E310 = "e310";
    /**
     * @unimod.event.descr Track is finished
     */
    public static final String E410 = "e410";
    /**
     * @unimod.event.descr Track switching succeeded
     */
    public static final String E411 = "e411";
    /**
     * @unimod.event.descr Track switching failed
     */
    public static final String E412 = "e412";

    public void dispose() {}

    public void init(ModelEngine engine) throws CommonException {}
}
```

B.2. PlayerBodyEventProvider.java

```
package ru.ifmo.is.daplayer;

import com.evelopers.common.exception.CommonException;
import com.evelopers.unimod.runtime.EventProvider;
import com.evelopers.unimod.runtime.ModelEngine;

public class PlayerBodyEventProvider implements EventProvider {
    /**
     * @unimod.event.descr "On/off" button pressed
     */
    public static final String E202 = "e202";
    /**
     * @unimod.event.descr "On/off" button released
     */
    public static final String E203 = "e203";
    /**
     * @unimod.event.descr "Play/pause" clicked
     */
    public static final String E210 = "e210";
    /**
     * @unimod.event.descr "Previous/left" button clicked
     */
    public static final String E211 = "e211";
    /**
     * @unimod.event.descr "Next/right" button clicked
     */
    public static final String E212 = "e212";
    /**
     * @unimod.event.descr "Vol+/up" button clicked
     */
    public static final String E220 = "e220";
    /**
     * @unimod.event.descr "Vol-/down" button clicked
     */
    public static final String E221 = "e221";
    /**
```

```
     * @unimod.event.descr "Hold" trigger on
     */
    public static final String E240 = "e240";
    /**
     * @unimod.event.descr "Hold" trigger off
     */
    public static final String E241 = "e241";
    /**
     * @unimod.event.descr "Mode" button clicked
     */
    public static final String E250 = "e250";

    public void dispose() {}

    public void init(ModelEngine engine) throws CommonException {}
}
```

B.3. GeneralPlayerControlledObject.java

```
package ru.ifmo.is.daplayer;

import javax.swing.SwingUtilities;

import com.evelopers.unimod.runtime.ControlledObject;
import com.evelopers.unimod.runtime.context.StateMachineContext;

public class GeneralPlayerControlledObject implements ControlledObject {
    // Playback modes
    public static final int PLM_DEFAULT = 0;
    public static final int PLM_REP1 = 1;
    public static final int PLM_REPA = 2;
    public static final int PLM_RANDOM = 3;

    // Is hold mode enabled
    private boolean holdMode = false;
    // Current playback mode
    private int playbackMode = PLM_DEFAULT;

    // Thread for switch off delay
    private Thread switchOffDelay = null;

    /**
     * @unimod.action.descr Playback mode
     */
    public int mode(StateMachineContext context) {
        return playbackMode;
    }

    /**
     * @unimod.action.descr Hold on
     */
    public boolean hold(StateMachineContext context) {
        return holdMode;
    }

    /**
     * @unimod.action.descr Is beginning of the track
     */
    public boolean begin(StateMachineContext context) {
        return PlayerModel.getPlayerModel().getPlaybackUnit().isBeginning();
    }

    /**
```

```
 * @unimod.action.descr Player is off
 */
public void z001(StateMachineContext context) {
    System.out.println("Player is off");
    PlayerModel.getPlayerModel().setButtonsEnabled(false);
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            PlayerModel.getPlayerModel().getPlayerScreen().invalidate();
            PlayerModel.getPlayerModel().getPlayerScreen().repaint();
        }
    });
    PlayerModel.getPlayerModel().getPlaybackUnit().stop();
    PlayerModel.getPlayerModel().getPlayerScreen()
            .setModelEnabled(false);
}

/**
 * @unimod.action.descr Start player switching on
 */
public void z002(StateMachineContext context) {
    // delayed player starting
    new Thread(new Runnable() {
        public void run() {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {}
            PlayerModel.getPlayerModel().setButtonsEnabled(true);
            PlayerModel.getPlayerModel()
                    .raiseEvent(GeneralEventProvider.E102);
        }
    }).start();
}

/**
 * @unimod.action.descr Refresh playback mode screen
 */
public void z101(StateMachineContext context) {
    PlayerModel.getPlayerModel().getPlayerScreen().setModelEnabled(true);
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            PlayerModel.getPlayerModel().getPlayerScreen().invalidate();
            PlayerModel.getPlayerModel().getPlayerScreen().repaint();
        }
    });
}

/**
 * @unimod.action.descr Increase volume
 */
public void z110(StateMachineContext context) {
    PlayerModel.getPlayerModel().getPlaybackUnit().volUp();
}

/**
 * @unimod.action.descr Decrease volume
 */
public void z111(StateMachineContext context) {
    PlayerModel.getPlayerModel().getPlaybackUnit().volDown();
}

/**
 * @unimod.action.descr Start playback
 */
```

```
public void z120(StateMachineContext context) {
    PlayerModel.getPlayerModel().getPlaybackUnit().start();
}

/**
 * @unimod.action.descr Pause playback
 */
public void z121(StateMachineContext context) {
    PlayerModel.getPlayerModel().getPlaybackUnit().stop();
}

/**
 * @unimod.action.descr Resume playback
 */
public void z122(StateMachineContext context) {
    PlayerModel.getPlayerModel().getPlaybackUnit().start();
}

/**
 * @unimod.action.descr Start track from the beginning
 */
public void z130(StateMachineContext context) {
    PlayerModel.getPlayerModel().getPlaybackUnit().restart();
}

/**
 * @unimod.action.descr Switch to previous track
 */
public void z131(StateMachineContext context) {
    if (PlayerModel.getPlayerModel().getPlaybackUnit()
            .switchToTrack(PlaybackUnit.SWITCH_PREVIOUS)) {
        PlayerModel.getPlayerModel()
                .raiseEvent(GeneralEventProvider.E411);
    } else {
        PlayerModel.getPlayerModel()
                .raiseEvent(GeneralEventProvider.E412);
    }
}

/**
 * @unimod.action.descr Switch to next track
 */
public void z132(StateMachineContext context) {
    if (PlayerModel.getPlayerModel().getPlaybackUnit()
            .switchToTrack(PlaybackUnit.SWITCH_NEXT)) {
        PlayerModel.getPlayerModel()
                .raiseEvent(GeneralEventProvider.E411);
    } else {
        PlayerModel.getPlayerModel()
                .raiseEvent(GeneralEventProvider.E412);
    }
}

/**
 * @unimod.action.descr Switch to random track
 */
public void z133(StateMachineContext context) {
    PlayerModel.getPlayerModel().getPlaybackUnit()
            .switchToTrack(PlaybackUnit.SWITCH_RANDOM);
}

/**
 * @unimod.action.descr Switch to first track
```

```
     */
    public void z134(StateMachineContext context) {
        PlayerModel.getPlayerModel().getPlaybackUnit()
                .switchToTrack(PlaybackUnit.SWITCH_FIRST);
    }

    /**
     * @unimod.action.descr Switch to last track
     */
    public void z135(StateMachineContext context) {
        PlayerModel.getPlayerModel().getPlaybackUnit()
                .switchToTrack(PlaybackUnit.SWITCH_LAST);
    }

    /**
     * @unimod.action.descr Start switching player off (switch off delay)
     */
    public void z301(StateMachineContext context) {
        switchOffDelay = new Thread(new Runnable() {
            public void run() {
                try {
                    Thread.sleep(3000);
                } catch (InterruptedException e) {
                    return;
                }
                if (switchOffDelay != null) {
                    PlayerModel.getPlayerModel()
                            .raiseEvent(GeneralEventProvider.E310);
                }
            }
        });
        switchOffDelay.start();
    }

    /**
     * @unimod.action.descr Stop switching player off
     */
    public void z302(StateMachineContext context) {
        switchOffDelay = null;
    }

    /**
     * @unimod.action.descr Set hold on
     */
    public void z310(StateMachineContext context) {
        holdMode = true;
    }

    /**
     * @unimod.action.descr Set hold off
     */
    public void z311(StateMachineContext context) {
        holdMode = false;
    }

    /**
     * @unimod.action.descr Change playback mode
     */
    public void z320(StateMachineContext context) {
        playbackMode = (playbackMode + 1) % 4;
    }

}
```

B.4. PlayerModel.java

```
package ru.ifmo.is.daplayer;

import java.awt.*;
import java.awt.event.*;
import java.io.File;

import javax.swing.*;

import com.evelopers.common.exception.CommonException;
import com.evelopers.unimod.core.stateworks.Event;
import com.evelopers.unimod.runtime.ModelEngine;
import com.evelopers.unimod.runtime.context.StateMachineContextImpl;

public class PlayerModel extends JFrame {
    private static PlayerModel myPlayerModel;

    // Model engine
    private ModelEngine modelEngine;
    private PlaybackUnit playbackUnit;
    private PlayerScreen playerScreen;

    // Buttons
    private JButton upButton;
    private JButton downButton;
    private JButton leftButton;
    private JButton rightButton;
    private JButton playPauseButton;
    private JButton onOffButton;
    private JToggleButton holdButton;
    private JButton modeButton;
    private JButton aboutButton;

    public PlayerModel() {
        // Create window
        setSize(800, 120);
        setTitle("Portable Player Model");
        setResizable(false);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        addWindowListener(new WindowAdapter() {
            public void windowClosed(WindowEvent e) {
                raiseEvent(GeneralEventProvider.E101);
            }
        });
        setLayout(new BorderLayout());

        // Create buttons
        upButton = new JButton(loadIcon("plus"));
        downButton = new JButton(loadIcon("minus"));
        leftButton = new JButton(loadIcon("back"));
        rightButton = new JButton(loadIcon("forward"));
        playPauseButton = new JButton(loadIcon("play"));
        onOffButton = new JButton(loadIcon("onoff"));
        holdButton = new JToggleButton(loadIcon("hold"));
        modeButton = new JButton(loadIcon("mode"));
        aboutButton = new JButton("?");

        // Disable buttons
        setButtonsEnabled(false);
```

```
// Add action listeners for buttons
upButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        raiseEvent(PlayerBodyEventProvider.E220);
    }
});

downButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        raiseEvent(PlayerBodyEventProvider.E221);
    }
});

leftButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        raiseEvent(PlayerBodyEventProvider.E211);
    }
});

rightButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        raiseEvent(PlayerBodyEventProvider.E212);
    }
});

playPauseButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        raiseEvent(PlayerBodyEventProvider.E210);
    }
});

onOffButton.addMouseListener(new MouseAdapter() {
    public void mousePressed(MouseEvent e) {
        raiseEvent(PlayerBodyEventProvider.E202);
    }

    public void mouseReleased(MouseEvent e) {
        raiseEvent(PlayerBodyEventProvider.E203);
    }
});

holdButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (((JToggleButton) e.getSource()).isSelected()) {
            raiseEvent(PlayerBodyEventProvider.E240);
        } else {
            raiseEvent(PlayerBodyEventProvider.E241);
        }
    }
});

modeButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        raiseEvent(PlayerBodyEventProvider.E250);
    }
});

aboutButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(PlayerModel.this,
                "Digital Audio Player Model\n"
                + "Powered by UniMod\n"
                + "Copyright 2008 E. Gerashchenko, A. Chebotaev");
```

```
        }
    });

    // Put buttons on panel
    JPanel buttonsPanel = new JPanel(new GridLayout(3, 7));

    buttonsPanel.add(new JPanel());
    buttonsPanel.add(new JPanel());
    buttonsPanel.add(upButton);
    buttonsPanel.add(new JPanel());
    buttonsPanel.add(new JPanel());
    buttonsPanel.add(new JPanel());
    buttonsPanel.add(aboutButton);

    buttonsPanel.add(new JPanel());
    buttonsPanel.add(leftButton);
    buttonsPanel.add(playPauseButton);
    buttonsPanel.add(rightButton);
    buttonsPanel.add(new JPanel());
    buttonsPanel.add(new JPanel());
    buttonsPanel.add(new JPanel());

    buttonsPanel.add(new JPanel());
    buttonsPanel.add(new JPanel());
    buttonsPanel.add(downButton);
    buttonsPanel.add(new JPanel());
    buttonsPanel.add(onOffButton);
    buttonsPanel.add(holdButton);
    buttonsPanel.add(modeButton);

    add(buttonsPanel, BorderLayout.EAST);

    // Add player screen
    playerScreen = new PlayerScreen(this);
    add(playerScreen, BorderLayout.CENTER);

    setVisible(true);

    // Create playback unit
    playbackUnit = new PlaybackUnit(new File("music"));

    // Create model engine
    try {
        modelEngine = PlayerEventProcessor.createModelEngine(true);
        modelEngine.start();
    } catch (CommonException e) {
        throw new RuntimeException(e);
    }

    // Create screen refresh cycle
    new Thread(new Runnable() {
        public void run() {
            while (true) {
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {}
                raiseEvent(GeneralEventProvider.E301);
            }
        }
    }).start();
}

public static ImageIcon loadIcon(String name) {
```

```
        return new ImageIcon(PlayerModel.class.getResource("img/" + name
                                                            + ".gif"));
    }

    public void setButtonsEnabled(boolean value) {
        upButton.setEnabled(value);
        downButton.setEnabled(value);
        leftButton.setEnabled(value);
        rightButton.setEnabled(value);
        playPauseButton.setEnabled(value);
        modeButton.setEnabled(value);
    }


    public void raiseEvent(String event) {
        modelEngine.getEventManager().handle(new Event(event),
                StateMachineContextImpl.create());
    }

    public PlaybackUnit getPlaybackUnit() {
        return playbackUnit;
    }

    public static PlayerModel getPlayerModel() {
        return myPlayerModel;
    }

    public PlayerScreen getPlayerScreen() {
        return playerScreen;
    }

    public GeneralPlayerControlledObject getControlledObject() {
        return (GeneralPlayerControlledObject)
                modelEngine.getControlledObjectsManager()
                .getControlledObject("o");
    }

    public static void main(String[] args) {
        myPlayerModel = new PlayerModel();
    }
}
```

B.5. PlayerScreen.java

```
package ru.ifmo.is.daplayer;

import java.awt.*;
import java.util.Formatter;

import javax.swing.JPanel;

import com.evelopers.unimod.runtime.context.StateMachineContextImpl;

public class PlayerScreen extends JPanel {
    private PlayerModel playerModel;
    private static Image playing =
            PlayerModel.loadIcon("playing").getImage();
    private static Image paused = PlayerModel.loadIcon("paused").getImage();
    private static Image stopped =
            PlayerModel.loadIcon("stopped").getImage();
    private static Image rep1 = PlayerModel.loadIcon("rep1").getImage();
    private static Image repa = PlayerModel.loadIcon("repa").getImage();
    private static Image rand = PlayerModel.loadIcon("rand").getImage();
```

```
private static Image hold = PlayerModel.loadIcon("hold").getImage();

private String lastDrawnTrack = null;
private int trackTitleOffset = 0;

private boolean modelEnabled = false;

public PlayerScreen(PlayerModel playerModel) {
    this.playerModel = playerModel;
}

private String getTimeText(int a, int b) {
    if (b > 3600) {
        b = 0;
    }
    StringBuffer buf = new StringBuffer();
    new Formatter(buf).format("%d:%02d/%d:%02d", new Object[]{
            new Integer(a / 60), new Integer(a % 60),
            new Integer(b / 60), new Integer(b % 60)
    });
    return buf.toString();
}

public void setModelEnabled(boolean value) {
    modelEnabled = value;
}

public void paintComponent(Graphics g) {
    Rectangle rect = g.getClipBounds();

    // Fill background
    g.setColor(Color.GRAY);
    g.fillRect(rect.x, rect.y, rect.width, rect.height);

    if (!modelEnabled || playerModel == null
                      || playerModel.getPlaybackUnit() == null) {
        return;
    }

    PlaybackUnit playbackUnit = playerModel.getPlaybackUnit();

    // Draw main status icon
    switch (playerModel.getPlaybackUnit().getPlaybackState()) {
    case PlaybackUnit.PLAYBACK_STOPPED:
        g.drawImage(stopped, 5, 5, this);
        break;
    case PlaybackUnit.PLAYBACK_PLAYING:
        g.drawImage(playing, 5, 5, this);
        break;
    case PlaybackUnit.PLAYBACK_PAUSED:
        g.drawImage(paused, 5, 5, this);
        break;
    }

    // Draw playback progress time and line
    g.setColor(Color.BLACK);
    g.setFont(new Font("Courier New", Font.BOLD, 24));

    int playedTime = playbackUnit.getPlayedTime();
    int totalTime = playbackUnit.getTotalTime();

    String timeText = getTimeText(playedTime, totalTime);
```

```
int timeTextWidth = (int)g.getFontMetrics()
        .getStringBounds(timeText, g).getWidth();

// 18 for icon, 154 to align
g.drawString(timeText, 5 + 18 + 154 - timeTextWidth, 16 + 5);
// Draw playback progress line
if (totalTime != 0) {
    g.fillRect(5, rect.height - 8 - 5, (int)
            ((long) rect.width * playedTime / totalTime) - 10, 8);
}

// Draw playback mode icon
GeneralPlayerControlledObject o = playerModel.getControlledObject();
int playbackMode = o.mode(StateMachineContextImpl.create());
switch (playbackMode) {
case GeneralPlayerControlledObject.PLM_DEFAULT:
    break;
case GeneralPlayerControlledObject.PLM_REP1:
    g.drawImage(rep1, 200 + 5, 5, this);
    break;
case GeneralPlayerControlledObject.PLM_REPA:
    g.drawImage(repa, 200 + 5, 5, this);
    break;
case GeneralPlayerControlledObject.PLM_RANDOM:
    g.drawImage(rand, 200 + 5, 5, this);
    break;
}

// Draw hold icon
if (o.hold(StateMachineContextImpl.create())) {
    g.drawImage(hold, 230 + 5, 5, this);
}

// Draw file name
g.setFont(new Font("Courier New", Font.BOLD,
           (rect.height - 30) / 3));
String track = playbackUnit.getTrack();
if (track.equals(lastDrawnTrack)) {
    trackTitleOffset += 20;
} else {
    lastDrawnTrack = track;
    trackTitleOffset = -20;
}
int trackTextWidth = (int)g.getFontMetrics().getStringBounds(
        track, g).getWidth();
if (trackTitleOffset > trackTextWidth) {
    trackTitleOffset = -trackTextWidth;
}
g.drawString(playbackUnit.getTrack(), 10 - trackTitleOffset,
        rect.height - 24); // 18 for icon, 154 to align

// Draw volume indicator

float volume = playbackUnit.getVolume();
int volX = 250 + 5;
int volY = 1 + 5;
int volW = (int) (volume * 29);
int volH = (int) (volume * 14);
g.drawPolygon(new int[]{volX + 29, volX + 29, volX},
              new int[]{volY,      volY + 14, volY + 14}, 3);
g.fillPolygon(new int[]{volX + volW,      volX + volW, volX},
              new int[]{volY + 14 - volH, volY + 14,   volY + 14},
               3);
```

```
    }
}
```

B.6. PlaybackUnit.java

```
package ru.ifmo.is.daplayer;

import java.io.*;
import javax.media.*;


public class PlaybackUnit {
    private File[] files;
    private int currentIndex;
    private Player player;

    private int playbackState = PLAYBACK_STOPPED;

    public static final int SWITCH_PREVIOUS = 0;
    public static final int SWITCH_NEXT = 1;
    public static final int SWITCH_RANDOM = 2;
    public static final int SWITCH_FIRST = 3;
    public static final int SWITCH_LAST = 4;

    public static final int PLAYBACK_STOPPED = 0;
    public static final int PLAYBACK_PLAYING = 1;
    public static final int PLAYBACK_PAUSED = 2;

    public PlaybackUnit(File directory) {
        this.files = directory.listFiles(new FilenameFilter() {
            public boolean accept(File dir, String name) {
                return name.endsWith(".wav") || name.endsWith(".mp3");
            }
        });
        switchToFile(0);
    }

    private void switchToFile(int index) {
        currentIndex = index;
        File file = files[index];
        MediaLocator mrl = null;
        if ((mrl = new MediaLocator("file:///" + file.getAbsolutePath()
                                    .toString())) == null) {
            System.out.println("Bad");
            throw new RuntimeException("Couldn't open " + file
                                       + ". Have you installed JMF?");
        }
        try {
            player = Manager.createPlayer(mrl);
        } catch (NoPlayerException e) {
            throw new RuntimeException(e);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        player.addControllerListener(new ControllerListener() {
            public void controllerUpdate(ControllerEvent e) {
                if (e instanceof EndOfMediaEvent) {
                    PlayerModel.getPlayerModel()
                            .raiseEvent(GeneralEventProvider.E410);
                }
            }
        });
    }
```

```
public void start() {
    player.start();
    playbackState = PLAYBACK_PLAYING;
}

public void stop() {
    player.stop();
    playbackState = PLAYBACK_PAUSED;
}

public void volUp() {
    if (player.getState() == Player.Unrealized) {
        return;
    }
    float level = player.getGainControl().getLevel() + 0.1f;
    if (level > 1.0f) {
        level = 1.0f;
    }
    player.getGainControl().setLevel(level);
}

public void volDown() {
    if (player.getState() == Player.Unrealized) {
        return;
    }
    float level = player.getGainControl().getLevel() - 0.1f;
    if (level < 0.0f) {
        level = 0.0f;
    }
    player.getGainControl().setLevel(level);
}

public boolean isBeginning() {
    System.out.println(player.getMediaTime().getSeconds());
    return player.getMediaTime().getSeconds() < 5;
}

public void restart() {
    player.setMediaTime(new Time(0));
    player.start();
}

public boolean switchToTrack(int mode) {
    int newIndex = -1;
    switch (mode) {
    case SWITCH_PREVIOUS:
        if (currentIndex == 0) {
            return false;
        } else {
            newIndex = currentIndex - 1;
        }
        break;
    case SWITCH_NEXT:
        if (currentIndex == files.length - 1) {
            return false;
        } else {
            newIndex = currentIndex + 1;
        }
        break;
    case SWITCH_RANDOM:
        newIndex = (int) (Math.random() * files.length);
        break;
```

```
            case SWITCH_FIRST:
                newIndex = 0;
                break;
            case SWITCH_LAST:
                newIndex = files.length - 1;
                break;
            }
            currentIndex = newIndex;
            float volume = player.getGainControl().getLevel();
            player.stop();
            player.deallocate();
            switchToFile(newIndex);
            player.start();
            while (player.getState() != Player.Realized
                   && player.getState() != Player.Started
                   && player.getState() != Player.Prefetched) {
                //System.out.println(player.getState());
            }
            player.getGainControl().setLevel(volume);
            return true;
        }

        public int getPlaybackState() {
            return playbackState;
        }

        public int getPlayedTime() {
            return (int) player.getMediaTime().getSeconds();
        }

        public int getTotalTime() {
            return (int) player.getDuration().getSeconds();
        }

        public String getTrack() {
            return files[currentIndex].getName();
        }

        public float getVolume() {
            if (player.getState() == Player.Unrealized) {
                return 1;
            } else {
                while (player.getState() != Player.Realized
                       && player.getState() != Player.Started
                       && player.getState() != Player.Prefetched) {
                    //System.out.println(player.getState());
                    System.out.print("");
                }
                return player.getGainControl().getLevel();
            }
        }
    }
```