САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ПРОГРАММИРОВАНИЯ

КАФЕДРА «КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ»

Д.И. Кулагин, А.В.Суслов

Моделирование устройства для продажи проездных билетов с помощью инструментальной среды *UniMod*

Проектная документация

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

Санкт-Петербург
2007

Оглавление

Введение

В наши дни все большее распространение получают электронные устройства, позволяющие переложить рутинную работу «на плечи» машин. Например, в Европейских странах практически повсеместно используются автоматы по продаже билетов на поезда и на городской транспорт. Одни из главных требований, предъявляемых к таким типам машин, это надежность и отказоустойчивость.

Традиционный подход к написанию программ, когда постепенно добавляется все новый и новый код, далеко не совершенен. Основная проблема состоит в том, что стадия проектирования программы фактически отсутствует. В то же время, автоматная технология программирования [1] позволяет разработчику основной акцент сделать именно на проектировании программы. Подробнее об автоматном подходе можно прочитать на сайте http://is.ifmo.ru.

Инструментальная среда *UniMod*, встраиваемая в интегрированную среду разработки программного обеспечения *Eclipse*, способствует логичному ходу создания программы: первоначально строится схема связей, состоящая из источников событий, системы управления и объектов управления, в которых реализованы вызываемые из автоматов выходные воздействия и опрашиваемые автоматами входные переменные.

Сайт инструментальной среды *UniMod* – http://unimod.sf.net.

1. Описание устройства

Как сказано выше, в настоящее время все чаще рутинный труд человека выполняют машины. Сфера продажи проездных билетов – не исключение. Нами была разработана модель устройства для продажи билетов. Основные составляющие части данного устройства:

- консоль (сенсорный экран);
- механизм для ввода купюр и монет;
- печатающий механизм (термопринтер);
- механизм выдачи билетов, а также денег (сдачи или введенной суммы при возврате).

Именно эти составляющие и были реализованы с помощью инструментальной среды *UniMod*.

Важно отметить, что данное устройство в своем обслуживании практически не требует вмешательства человека: требуется лишь периодическое изъятие денег и вставка новых бланков проездных билетов.

1.1. Описание основных узлов

Опишем подробнее принципы работы основных узлов устройства.

На сенсорный экран выводится информация о том, какие действия выполняет автомат или какие действия может выполнить пользователь. Информация выводится в виде текста, изображений и кнопок. Кнопки могут быть как активными, так и неактивными.

Механизм для ввода купюр и монет, с точки зрения пользователя, может находиться в двух состояниях:

- ввод купюр или монет разрешен (купюро- и монетоприемник подсвечены зеленым светом);
- ввод купюр или монет невозможен (купюро- и монетоприемник подсвечены красным светом).

Также этот механизм отвечает за проверку денег на допустимость к вводу в автомат. Если вводимая купюра или монета не является допустимой (все допустимые к вводу купюры и монеты отображены на сенсорном экране), то механизм возвращает их обратно.

Печатающий механизм (термопринтер) отвечает за печать данных на специальных бланках билетов. В случае если после очередной печати билета бланки закончились, печатающий механизм сигнализирует об этом и устройство для продажи билетов переходит в неактивный режим (выводится сообщение «Устройство не работает»).

1.2. Описание принципа работы

В начале, если устройство находится в рабочем состоянии, пользователю демонстрируется экран со списком возможных типов билетов (рис. 1).

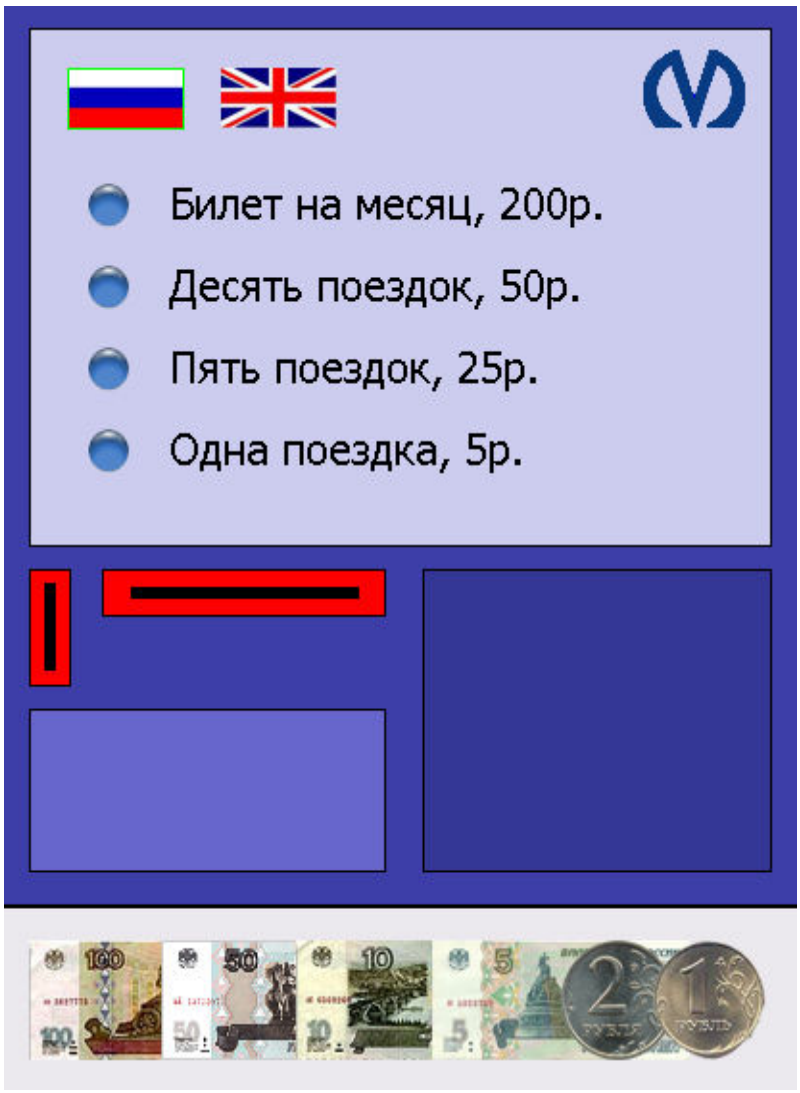

Рис. 1. Внешний вид приложения на начальном этапе

С помощью сенсорного экрана (верхняя часть приложения) пользователь может выбрать подходящий ему тип билета.

На примере рис. 1 рассмотрим внешний вид приложения. Окно программы поделено на две части: верхняя – собственно устройство, нижняя – наличные деньги пользователя, которые он может вводить в устройство.

При выборе типа билета начинается этап оплаты билета (рис. 2): пользователю демонстрируется тип билета, его стоимость, введенная в устройство сумма, список допустимых к вводу купюр, активная или неактивная кнопка «Оплатить».

Рис. 2. Внешний вид приложения. Этап оплаты билета

На этом этапе купюро- и монетоприемник переходят в состоянии готовности к приему денег, что демонстрируется зеленой подсветкой соответствующих механизмов. Пользователь может вернуться к начальному этапу нажатием на кнопку «Назад» или может приступить к оплате.

Как только введенная пользователем сумма станет равной или превысит стоимость выбранного билета, купюро- и монетоприемник переходят в неактивное состояние, а кнопка «Оплатить» переходит в активное состояние. В этот момент пользователь может либо снова вернуться в начальное состояние (при этом введенные деньги будут возвращены), либо произвести операцию оплаты.

При нажатии на кнопку «Оплатить» на экране будет отображена надпись «Заберите билет» или надпись «Заберите билет и сдачу» (если введенная пользователем сумма превышает стоимость билета), а механизм выдачи выдаст соответственно билет и сдачу (рис.3).

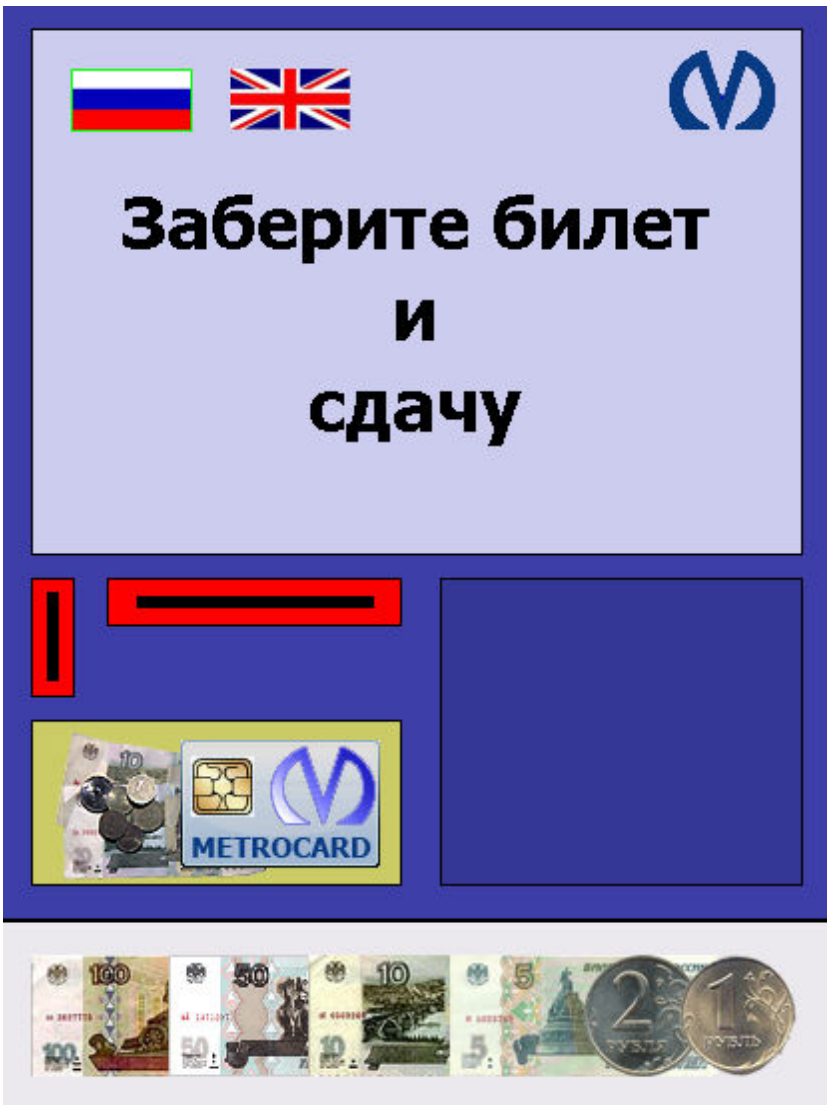

Рис. 3. Внешний вид приложения. Этап выдачи билета и сдачи

На данном этапе устройство остается до тех пор, пока билет и, возможно, сдача не будут извлечены из механизма выдачи. В данной модели извлечению билета и сдачи соответствует нажатие на их изображения. После извлечения билета и сдачи устройство возвращается на начальный этап или переходит в неактивное состояние, если печатающий механизм сообщил об отсутствии бланков билетов.

2. Реализация с использованием автоматов

2.1. Схема связей автомата

Управление устройством для продажи проездных билетов выполняет автомат *A1*. На схеме связей автомата *A1* (рис. 4) каждому событию соответствует обозначение вида `e#`, где вместо символа **#** находится число. Входные переменные объектов управления обозначаются `x1, x2, ….` Выходные воздействия объектов управления – `z0, z1, ….` Все они являются методами и реализуются вручную на языке *Java*.

На схеме связей автомата каждая составляющая системы изображена в виде прямоугольника, в верхней части которого приведено краткое обозначение и название *Java*-класса, реализующего функциональность данного объекта. При этом для автомата *A1* название *Java*-класса не указано, так как *UniMod* автоматически генерирует его *XML*-описание.

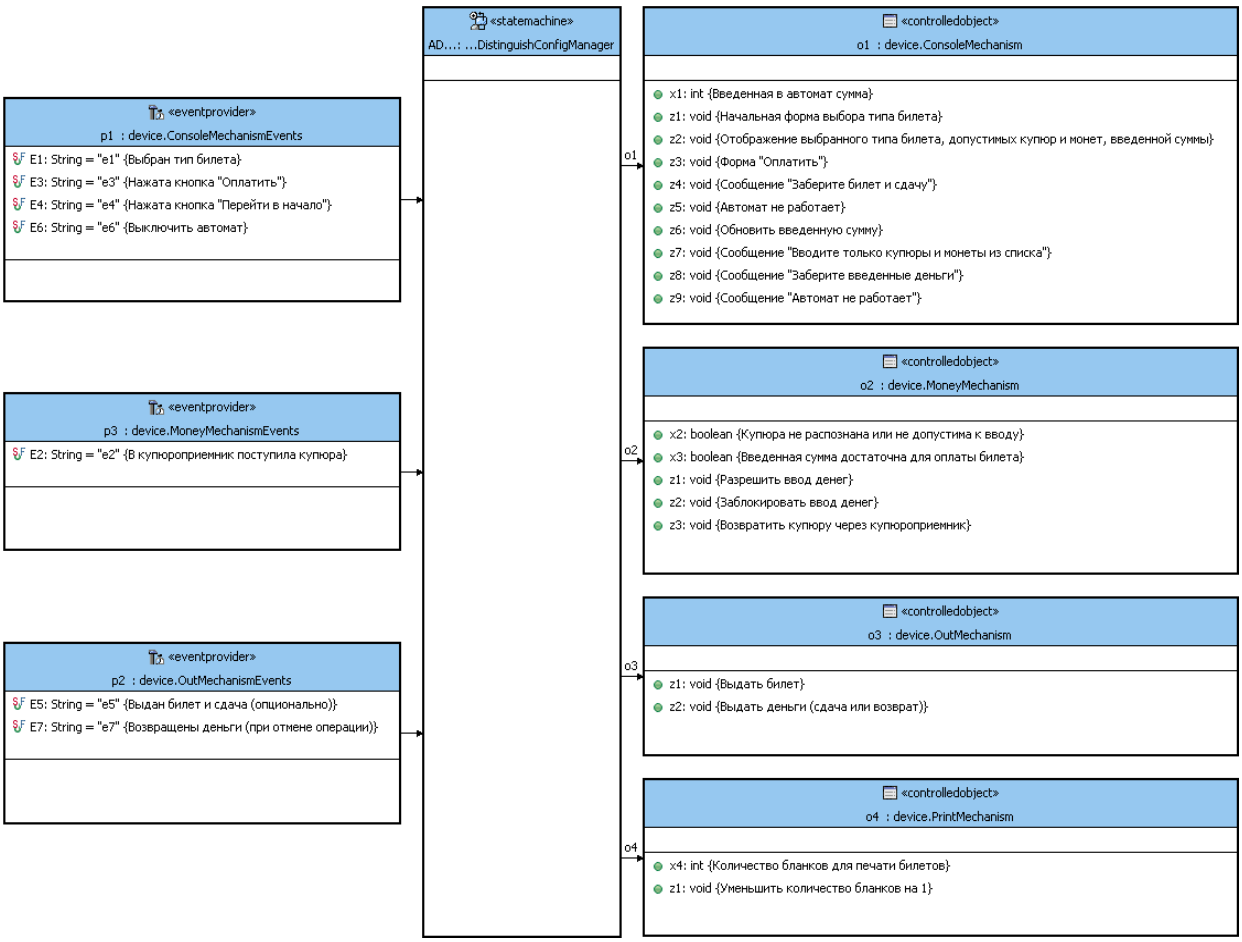

Рис. 4. Схема связей автомата *A1*

Краткие обозначения обеспечивают возможность компактного построения автоматов в инструментальной среде *UniMod*. При этом рядом с краткими обозначениями на схеме связей приводятся комментарии, поясняющие назначение конкретной входной переменной или функциональность конкретного выходного воздействия.

Инструментальная среда *UniMod* обладает удобным интерфейсом: при построении графов переходов автоматов всплывающие подсказки (рис. 5), которые обеспечивают простоту моделирования автомата.

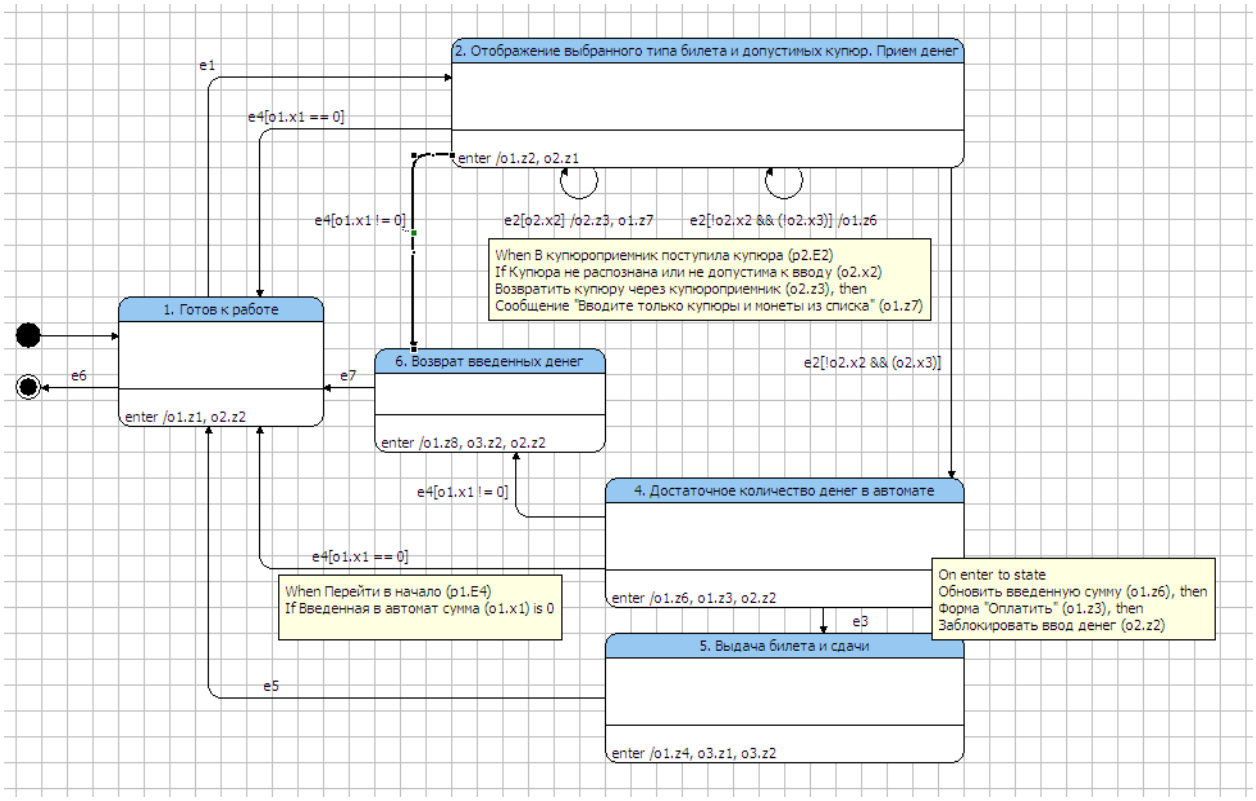

Рис. 5. Всплывающие подсказки на графе переходов автомата *A1* в среде *UniMod* (снимок экрана из среды разработки)

2.2. События

Опишем механизм создания и обработки событий. Каждому источнику событий соответствует *Java*-класс, в котором описаны действия автомата. Приведем список событий (табл. 1).

Таблица 1. События и соответствующие им поставщики

Условное обозначение	**Короткое описание источника событий в *UniMod***	**Соответствующий поставщик событий**
`e1`	Выбран тип билета	`p1: device.ConsoleMechanismEvents`
`e2`	В купюроприемник поступила купюра	`p3: device.MoneyMechanismEvents`
`e3`	Нажата кнопка «Оплатить»	`p1: device.ConsoleMechanismEvents`
`e4`	Нажата кнопка «Перейти в начало»	`p1: device.ConsoleMechanismEvents`
`e5`	Выдан билет и сдача (опционально)	`p2: device.OutMechanismEvents`
`e6`	Выключить автомат	`p1: device.ConsoleMechanismEvents`
`e7`	Возвращены деньги (при отмене операции)	`p2: device.OutMechanismEvents`

2.3. Граф переходов автомата

Граф переходов (рис. 6) задает логику работы автомата *A1*.

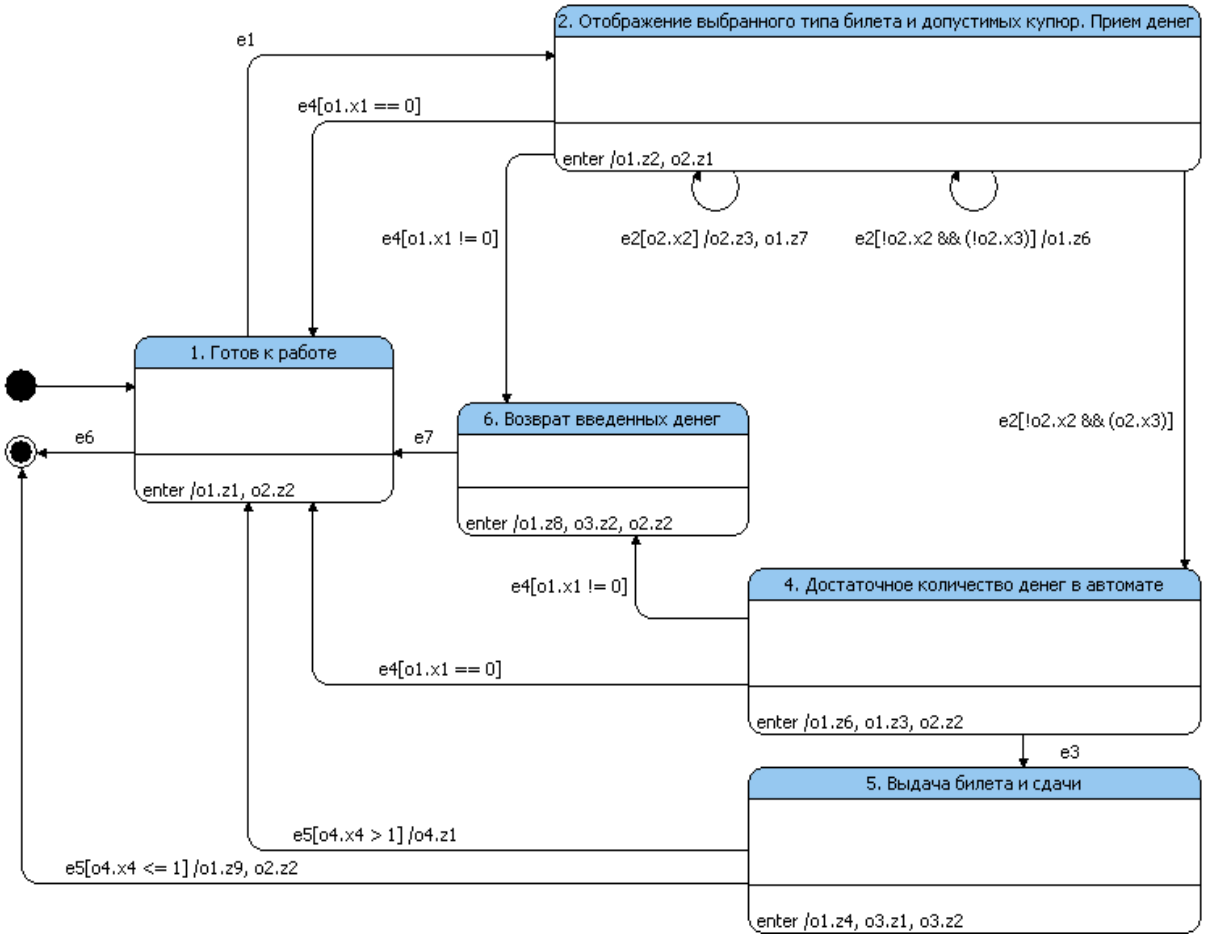

Рис. 6. Граф переходов автомата *A1*

2.4. Полнота и непротиворечивость логических условий

Изучая граф переходов и схему связей, можно понять, что происходит в ответ на то или иное событие. При проектировании графа переходов с помощью инструментальной среды *UniMod* выполняется автоматическая проверка условий переходов на полноту и непротиворечивость. Покажем эту возможность на примере автомата *A1*.

Рассмотрим состояние автомата «2. Отображение выбранного типа билета и допустимых купюр. Прием денег». Из этого состояния есть пять переходов: из них два по событию `e4`, а остальные три по событию `e5`. Перечислим переходы по событию `e4`:

- `e2[o2.x2] /o2.z3, o1.z7;`
- `e2[!o2.x2 && (!o2.x3)] /o1.z6;`
- `e2[!o2.x2 && (o2.x3)].`

Эта система логических условий полнота и непротиворечива. Если изменить третий переход `e2[!o2.x2 && (o2.x3)]`, например, на `e2[!o2.x2]` инструментальная среда *UniMod* подсветит все переходы по соответствующему событию, сигнализируя о неполноте системы логических условий (рис. 7). Подробнее проверка системы логических условий инструментальной средой UniMod описана в работе [3].

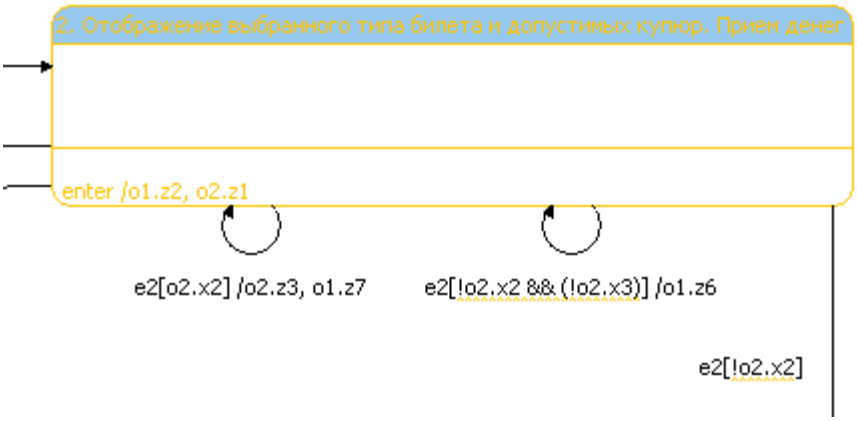

Рис. 7. Подсветка неполной системы логических условий в среде *UniMod* (снимок экрана из среды разработки)

3. Подходы к проектированию

Одна из мощных возможностей, которые предоставляет пакет *UniMod*, это визуальное конструирование программ. Довольно распространенным является подход, когда *UML*-диаграммы и вспомогательные рисунки создаются для улучшения документации. В среде *UniMod* диаграммы и классы, написанные вручную, позволяют полностью сформировать работающее приложение.

3.1. Интерпретационный подход

Интерпретационный подход применяется при разработке приложения в среде *Eclipse*. Этот подход дает возможность эффективно тестировать и отлаживать создаваемую программу.

Для получения готового приложения *UML*-диаграммы переводятся в специальный формат описания, созданный на базе технологии *XML*.

При интерпретационном подходе на стадии исполнения программы виртуальная машина *Java* создает в памяти экземпляры классов источников событий и объектов управления.

Рассмотрим процесс обработки события:

- интерпретатор получает информацию о возможных переходах в новые состояния;
- для каждого из переходов вычисляются логические условия перехода;
- если одно из логических условий является истинным, происходит следующее:
    1. интерпретатор вызывает выходные воздействия, заданные для выбранного перехода;
    2. система переходит в новое состояние, при этом исполняются выходные воздействия, заданные для выполнения при входе в новое состояние;
    3. управление передается вложенным автоматам.

Интерпретационный подход удобен в ходе разработки приложения, но он обладает несколькими недостатками:

- для запуска приложения требуются не только скомпилированные *Java*-классы, но также *XML*-описание и интерпретатор *UniMod*;
- интерпретационный подход уступает компиляционному по производительности.

Для устранения этих недостатков был разработан компиляционный подход, который будет рассмотрен ниже.

3.2. Компиляционный подход

В рамках данного подхода *UML*-диаграммы транслируются в готовый код на языке *Java* с помощью инструмента *Velocity* (http://jakarta.apache.org/velocity). *Java*-класс, реализующий систему автоматов, автоматически генерируется по *XML*-описанию. Затем он и классы, которые реализуют функциональность источников событий и объектов управления, компилируются. Приложение, собранное таким способом, не требует наличия *XML*-описания и интерпретатора *UniMod*. Для запуска программы требуется только библиотека *UniMod* времени выполнения в скомпилированном виде.

Заметим, что функционально приложения построенные на базе обоих подходов будут идентичны.

4. Реализация

4.1. Диаграмма классов

Для лучшего понимания принципов взаимодействия классов составлена диаграмма классов проекта (рис. 8).

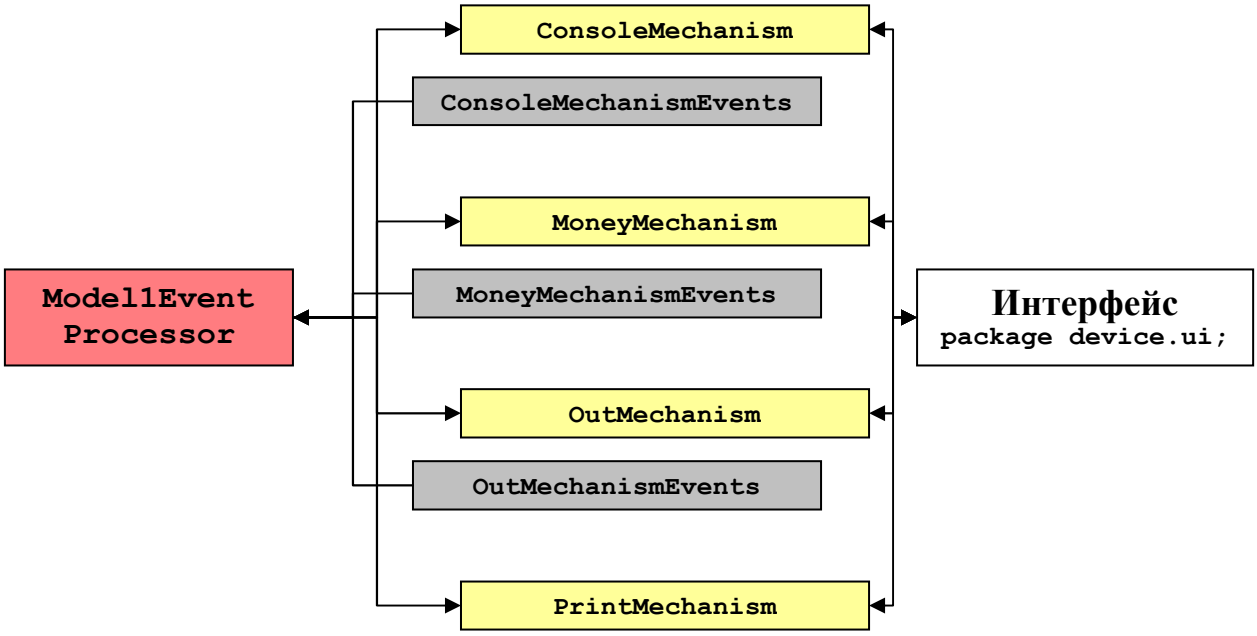

Рис. 8. Диаграмма классов проекта

Все классы, отвечающие за интерфейс программы (Приложение 2), показаны на данной диаграмме в виде одного блока «Интерфейс». Остальные классы выделены следующими цветами:

- желтым цветом, если это объект управления;
- серым цветом, если это поставщик событий;
- розовым цветом, если это класс *UniMod*-модели.

Отметим, что только классы, отвечающие за интерфейс, создавались вручную, остальные файлы были автоматически сгенерированы инструментальной средой *UniMod*. Приведем статистику:

- 885 строк кода написано вручную;
- 2189 строк кода сгенерировано втоматически.

Статистика показывает, что использованный подход позволяет проектировать и разрабатывать приложение на более высоком уровне абстракции, нежели чем при традиционном подходе.

4.2. Запуск программы

Программа может быть запущена как с помощью интерпретационного, так и компиляционного подхода. Если головной автомат запустить с помощью интерпретатора *UniMod*, то ход работы приложения можно наблюдать в виде протокола работы, выводящегося на консоль (Приложение 1).

Для демонстрации приложения обычно используют компилятивный подход, поэтому на сайте http://is.ifmo.ru в разделе «*UniMod*-проекты» программа опубликована в виде *Java*-апплета.

Заключение

Данная работа показала, что автоматный подход целесообразно использовать для проектирования и программирования автоматических устройств, обслуживающих человека в повседневной жизни. Явное выделение состояний делает логику программы более простой для восприятия, а, следовательно, и минимизирует работы по отладке приложения.

В свою очередь, инструментальная среда *UniMod* [4] значительно упрощает реализацию спроектированной системы с помощью автоматного подхода. Наглядное построение схем связей и графов переходов, а также возможность *UniMod* интерпретировать или транслировать полученные диаграммы в *Java*-код позволяет разработчику не тратить время на реализацию самих конечных автоматов.

Литература

1. *Шалыто А.А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
2. *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
3. *Колыхматов И.И., Рыбак О.О., Шалыто А.А.* Моделирование устройства для продажи газированной воды на инструментальном средстве *UniMod*. http://is.ifmo.ru/unimod-projects/vending/
4. *Инструментальная среда UniMod*. http://unimod.sf.net.

Приложение 1. Пример протокола работы приложения

```
  19:04:49,093 INFO  [Run] Start event [] processing. In state [/ADevice:Top]
  19:04:49,093 INFO  [Run] Transition to go found [s1#1. Готов к
работе##true]
  19:04:49,093 INFO  [Run] Start on-enter action [o1.z1] execution
  19:04:49,093 INFO  [Run] Finish on-enter action [o1.z1] execution
  19:04:49,093 INFO  [Run] Start on-enter action [o2.z2] execution
  19:04:49,125 INFO  [Run] Finish on-enter action [o2.z2] execution
  19:04:49,125 INFO  [Run] Finish event [] processing. In state [/ADevice:1.
Готов к работе]
  19:05:19,984 INFO  [Run] Start event [e1] processing. In state [/ADevice:1.
Готов к работе]
  19:05:20,000 DEBUG [Run] Try transition [1. Готов к работе#2. Отображение
выбранного типа билета и допустимых купюр. Прием денег#e1#true]
  19:05:20,000 INFO  [Run] Transition to go found [1. Готов к работе#2.
Отображение выбранного типа билета и допустимых купюр. Прием денег#e1#true]
  19:05:20,000 INFO  [Run] Start on-enter action [o1.z2] execution
  19:05:20,000 INFO  [Run] Finish on-enter action [o1.z2] execution
  19:05:20,000 INFO  [Run] Start on-enter action [o2.z1] execution
  19:05:20,000 INFO  [Run] Finish on-enter action [o2.z1] execution
  19:05:20,000 INFO  [Run] Finish event [e1] processing. In state
[/ADevice:2. Отображение выбранного типа билета и допустимых купюр. Прием
денег]
  19:05:21,437 INFO  [Run] Start event [e2] processing. In state [/ADevice:2.
Отображение выбранного типа билета и допустимых купюр. Прием денег]
  19:05:21,437 DEBUG [Run] Try transition [2. Отображение выбранного типа
билета и допустимых купюр. Прием денег#2. Отображение выбранного типа билета
и допустимых купюр. Прием денег#e2#!o2.x2 && (!o2.x3)]
  19:05:21,437 INFO  [Run] Start input action [o2.x2] calculation
  19:05:21,437 INFO  [Run] Finish input action [o2.x2] calculation. Its value
is [false]
  19:05:21,468 INFO  [Run] Start input action [o2.x3] calculation
  19:05:21,468 INFO  [Run] Finish input action [o2.x3] calculation. Its value
is [false]
  19:05:21,468 INFO  [Run] Transition to go found [2. Отображение выбранного
типа билета и допустимых купюр. Прием денег#2. Отображение выбранного типа
билета и допустимых купюр. Прием денег#e2#!o2.x2 && (!o2.x3)]
  19:05:21,468 INFO  [Run] Start output action [o1.z6] execution
  19:05:21,468 INFO  [Run] Finish output action [o1.z6] execution
  19:05:21,468 INFO  [Run] Start on-enter action [o1.z2] execution
  19:05:21,468 INFO  [Run] Finish on-enter action [o1.z2] execution
  19:05:21,468 INFO  [Run] Start on-enter action [o2.z1] execution
  19:05:21,468 INFO  [Run] Finish on-enter action [o2.z1] execution
  19:05:21,468 INFO  [Run] Finish event [e2] processing. In state
[/ADevice:2. Отображение выбранного типа билета и допустимых купюр. Прием
денег]
  19:05:22,484 INFO  [Run] Start event [e2] processing. In state [/ADevice:2.
Отображение выбранного типа билета и допустимых купюр. Прием денег]
  19:05:22,484 DEBUG [Run] Try transition [2. Отображение выбранного типа
билета и допустимых купюр. Прием денег#2. Отображение выбранного типа билета
и допустимых купюр. Прием денег#e2#!o2.x2 && (!o2.x3)]
  19:05:22,484 INFO  [Run] Start input action [o2.x2] calculation
  19:05:22,484 INFO  [Run] Finish input action [o2.x2] calculation. Its value
is [false]
  19:05:22,484 INFO  [Run] Start input action [o2.x3] calculation
  19:05:22,484 INFO  [Run] Finish input action [o2.x3] calculation. Its value
is [false]
  19:05:22,484 INFO  [Run] Transition to go found [2. Отображение выбранного
типа билета и допустимых купюр. Прием денег#2. Отображение выбранного типа
билета и допустимых купюр. Прием денег#e2#!o2.x2 && (!o2.x3)]
  19:05:22,484 INFO  [Run] Start output action [o1.z6] execution
  19:05:22,484 INFO  [Run] Finish output action [o1.z6] execution
```

```
  19:05:22,484 INFO  [Run] Start on-enter action [o1.z2] execution
  19:05:22,484 INFO  [Run] Finish on-enter action [o1.z2] execution
  19:05:22,484 INFO  [Run] Start on-enter action [o2.z1] execution
  19:05:22,484 INFO  [Run] Finish on-enter action [o2.z1] execution
  19:05:22,484 INFO  [Run] Finish event [e2] processing. In state
[/ADevice:2. Отображение выбранного типа билета и допустимых купюр. Прием
денег]
  19:05:24,531 INFO  [Run] Start event [e2] processing. In state [/ADevice:2.
Отображение выбранного типа билета и допустимых купюр. Прием денег]
  19:05:24,531 DEBUG [Run] Try transition [2. Отображение выбранного типа
билета и допустимых купюр. Прием денег#2. Отображение выбранного типа билета
и допустимых купюр. Прием денег#e2#!o2.x2 && (!o2.x3)]
  19:05:24,531 INFO  [Run] Start input action [o2.x2] calculation
  19:05:24,531 INFO  [Run] Finish input action [o2.x2] calculation. Its value
is [false]
  19:05:24,531 INFO  [Run] Start input action [o2.x3] calculation
  19:05:24,531 INFO  [Run] Finish input action [o2.x3] calculation. Its value
is [true]
  19:05:24,531 DEBUG [Run] Try transition [2. Отображение выбранного типа
билета и допустимых купюр. Прием денег#2. Отображение выбранного типа билета
и допустимых купюр. Прием денег#e2#o2.x2]
  19:05:24,531 DEBUG [Run] Try transition [2. Отображение выбранного типа
билета и допустимых купюр. Прием денег#4. Достаточное количество денег в
автомате#e2#!o2.x2 && (o2.x3)]
  19:05:24,531 INFO  [Run] Transition to go found [2. Отображение выбранного
типа билета и допустимых купюр. Прием денег#4. Достаточное количество денег в
автомате#e2#!o2.x2 && (o2.x3)]
  19:05:24,531 INFO  [Run] Start on-enter action [o1.z6] execution
  19:05:24,531 INFO  [Run] Finish on-enter action [o1.z6] execution
  19:05:24,531 INFO  [Run] Start on-enter action [o1.z3] execution
  19:05:24,546 INFO  [Run] Finish on-enter action [o1.z3] execution
  19:05:24,546 INFO  [Run] Start on-enter action [o2.z2] execution
  19:05:24,546 INFO  [Run] Finish on-enter action [o2.z2] execution
  19:05:24,546 INFO  [Run] Finish event [e2] processing. In state
[/ADevice:4. Достаточное количество денег в автомате]
  19:05:26,796 INFO  [Run] Start event [e3] processing. In state [/ADevice:4.
Достаточное количество денег в автомате]
  19:05:26,796 DEBUG [Run] Try transition [4. Достаточное количество денег в
автомате#5. Выдача билета и сдачи#e3#true]
  19:05:26,796 INFO  [Run] Transition to go found [4. Достаточное количество
денег в автомате#5. Выдача билета и сдачи#e3#true]
  19:05:26,796 INFO  [Run] Start on-enter action [o1.z4] execution
  19:05:26,796 INFO  [Run] Finish on-enter action [o1.z4] execution
  19:05:26,796 INFO  [Run] Start on-enter action [o3.z1] execution
  19:05:26,796 INFO  [Run] Finish on-enter action [o3.z1] execution
  19:05:26,796 INFO  [Run] Start on-enter action [o3.z2] execution
  19:05:26,796 INFO  [Run] Finish on-enter action [o3.z2] execution
  19:05:26,796 INFO  [Run] Finish event [e3] processing. In state
[/ADevice:5. Выдача билета и сдачи]
  19:05:28,593 INFO  [Run] Start event [e5] processing. In state [/ADevice:5.
Выдача билета и сдачи]
  19:05:28,593 DEBUG [Run] Try transition [5. Выдача билета и сдачи#1. Готов
к работе#e5#o4.x4 > 1]
  19:05:28,593 INFO  [Run] Start input action [o4.x4] calculation
  19:05:28,593 INFO  [Run] Finish input action [o4.x4] calculation. Its value
is [5]
  19:05:28,593 INFO  [Run] Transition to go found [5. Выдача билета и
сдачи#1. Готов к работе#e5#o4.x4 > 1]
  19:05:28,593 INFO  [Run] Start output action [o4.z1] execution
  19:05:28,593 INFO  [Run] Finish output action [o4.z1] execution
  19:05:28,593 INFO  [Run] Start on-enter action [o1.z1] execution
  19:05:28,625 INFO  [Run] Finish on-enter action [o1.z1] execution
  19:05:28,625 INFO  [Run] Start on-enter action [o2.z2] execution
  19:05:28,625 INFO  [Run] Finish on-enter action [o2.z2] execution
```

```
19:05:28,625 INFO  [Run] Finish event [e5] processing. In state
[/ADevice:1. Готов к работе]
```

Приложение 2. Исходные коды программы

Package `device.ui` (интерфейс программы)

`MainForm.java`

```
package device.ui;

import java.awt.*;
import java.awt.event.*;

import javax.swing.JFrame;

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

public class MainForm extends JFrame {
    static final long serialVersionUID = 2345723894503742L;

    InterfaceBuilder interfaceBuilder = new InterfaceBuilder();

    public EventHandler eventHandler;

    public ModelEngine engine = null;

    Image shadeBuffer;

    public MainForm(ModelEngine engine) {
          super("FahrTicketKasse");
          this.setResizable(false);
          setBounds(100, 50, interfaceBuilder.WIDTH,
          interfaceBuilder.HEIGHT);
          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          setVisible(true);

          this.engine = engine;

          interfaceBuilder.loadInterface(this);

          EventHandler.interfaceBuilder = interfaceBuilder;

          shadeBuffer = createImage(interfaceBuilder.WIDTH,
                      interfaceBuilder.HEIGHT);
    }

    public void paint(Graphics g) {
          if (shadeBuffer != null) {
                interfaceBuilder.render(shadeBuffer.getGraphics());
                g.drawImage(shadeBuffer, 0, 0, this);
          }
    }

    public void notify(String event) {
          engine.getEventManager().handle(new Event(event),
                      StateMachineContextImpl.create());
    }
}
```

EventHandler.java

```
package device.ui;

public class EventHandler {
    public final static int CHOISE_CONSOLE = 1;

    public final static int MONEY_CONSOLE = 2;

    public final static int CHANGE_TICKET_CONSOLE = 3;

    public final static int MONEYBACK_CONSOLE = 4;

    public final static int OUTOFPAPER_CONSOLE = 5;

    public final static int MM_BADMONEY = 1;

    public final static int MM_ALLOWED = 2;

    public final static int MM_DISALLOWED = 3;

    public final static int OM_EMPTY = 1;

    public final static int OM_FILLED = 1;

    public static InterfaceBuilder interfaceBuilder = null;

    public static void switchConsoleState(int state) {
          interfaceBuilder.screen = state;
          interfaceBuilder.mainForm.repaint();
    }

    public static void switchMoneyMechanismState(int state) {
          InterfaceState.moneyEnabled = (state != MM_DISALLOWED);
          interfaceBuilder.mainForm.repaint();
    }

    public static void switchOutMechanism(int state) {
          InterfaceState.outMechanismState = state;
          interfaceBuilder.mainForm.repaint();
    }

    public static void switchPayButtonState(boolean state) {
          InterfaceState.payButtonState = state;
          if (state) {
                interfaceBuilder.btnPay.load("pay.png");
          } else {
                interfaceBuilder.btnPay.load("pay_NA.png");
          }
          interfaceBuilder.mainForm.repaint();
    }

    public static void refreshMoneyAmount(int amount) {
          InterfaceState.coinsIn = amount;
    }
}
```

InterfaceBuilder.java

```
package device.ui;

import java.awt.Color;
import java.awt.Font;
```

```
import java.awt.Graphics;
import java.awt.Rectangle;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import device.MoneyMechanismEvents;
import device.ConsoleMechanismEvents;
import device.OutMechanismEvents;

public class InterfaceBuilder {
    public final static float TOP_MARGIN = 0.07f;

    public final static float LEFT_MARGIN = 0.04f;

    public final static float RIGHT_MARGIN = 0.04f;

    public final static float SCREEN_SIZE = 0.45f;

    public final static float FUNC_MARGIN = 0.02f;

    public final static float FUNC_VSIZE = 0.1f;

    public final static float FUNC_HSIZE = 0.05f;

    public final static float MONEY_MARGIN = 0.04f;

    public final static float MONEY_HSIZE = 0.35f;

    public final static float MONEY_VSIZE = 0.04f;

    public final static float MONEY_INSIDE_VSIZE = 0.8f;

    public final static float MONEY_INSIDE_HSIZE = 0.2f;

    public final static float OUTPUT_MARGIN = 0.02f;

    public final static float OUTPUT_VSIZE = 0.14f;

    public final static int LAMP_HCOUNT = 5;

    public final static int LAMP_VCOUNT = 3;

    public final static int LAMP_SIZE = 5;

    public final static int IMAGE_MARGIN = 20;

    public final static int WIDTH = 420;

    public final static int HEIGHT = 575;

    public final static int EDGE_COLOR = 0;

    public final static int SCREEN_COLOR = 0xCCCCEE;

    public final static int MONEY_ALLOWED = 3329330;

    public final static int MONEY_DISALLOWED = 0xFF0000;

    public int screen = 1;

    public static int chosenTicket = 0;

    public String[] ticketTypesRu = { "Билет на месяц", "Десять поездок",
                "Пять поездок", "Одна поездка" };
```

```
public String[] ticketTypesEng = { "Month unlimited ride", "Ten pass",
            "Five pass", "Single pass" };

public static int[] ticketPrice = { 200, 50, 25, 5 };

public boolean[] moneyAvailable = {true, true, true, true, true, true };

volatile boolean interfaceLoaded = false;

MyButton[] assignations;

MyButton[] pickableAssignations;

public int[] assignationNominals = { 100, 50, 10, 5, 2, 1 };

InterfaceState interfaceState = new InterfaceState();

String backString = "uniground.png";

public MyButton btnRu = new MyButton(), btnEng = new MyButton(),
            btnLogo = new MyButton(), btnBack = new MyButton(),
            btnPay = new MyButton(), btnChange = new MyButton(),
            btnTicket = new MyButton();

MyLabeledButton btnCh[];

MyRegion outputDevice = null;

MainForm mainForm;

void loadInterface(MainForm _mainForm) {
      mainForm = _mainForm;
      btnRu.load("rus_flag.png");
      btnRu.setActive();
      btnEng.load("uk_flag.png");
      btnLogo.load("metro.png");
      btnBack.load("back.png");

      btnCh = new MyLabeledButton[4];
      for (int i = 0; i < 4; i++) {
            btnCh[i] = new MyLabeledButton();
            if (i == 0) {
                  btnCh[i].load("button.png");
            } else {
                  btnCh[i].setTheSameImage(btnCh[0]);
            }
                  btnCh[i].textRu = ticketTypesRu[i] + ", " +
      ticketPrice[i] + "p.";
            btnCh[i].textEng = ticketTypesEng[i] + ", " + ticketPrice[i]
                        + "rub.";
      }

      btnPay.load("pay_NA.png");
      btnChange.load("change.png");
      btnTicket.load("smart_card.png");

      assignations = new MyButton[6];
      for (int i = 0; i < 6; i++) {
            assignations[i] = new MyButton();
      }
      assignations[0].load("rub100.png");
      assignations[1].load("rub50.png");
      assignations[2].load("rub10.png");
      assignations[3].load("rub5.png");
```

```
assignations[4].load("rub2.png");
assignations[5].load("rub1.png");

pickableAssignations = new MyButton[6];
for (int i = 0; i < 6; i++) {
      pickableAssignations[i] = new MyButton();
}

pickableAssignations[0].load("rub100.png");
pickableAssignations[1].load("rub50.png");
pickableAssignations[2].load("rub10.png");
pickableAssignations[3].load("rub5.png");
pickableAssignations[4].load("rub2.png");
pickableAssignations[5].load("rub1.png");

mainForm.addMouseListener(new MouseAdapter() {
      public void mouseClicked(MouseEvent ev) {
            int x = ev.getX();
            int y = ev.getY();
            Rectangle boundingRect = mainForm.getBounds();
            int boundingWidth, boundingHeight;
            boundingWidth = boundingRect.width;
            boundingHeight = boundingRect.height;
            if (btnRu.test(x, y) && !InterfaceState.language) {
                  InterfaceState.switchLanguage();
                  btnRu.setActive();
                  btnEng.setPassive();
                  for (int i = 0; i < 4; i++) {
                        btnCh[i].switchLanguage();
                  }
                  mainForm.repaint();
            }
            if (btnEng.test(x, y) && InterfaceState.language) {
                  InterfaceState.switchLanguage();
                  btnEng.setActive();
                  btnRu.setPassive();
                  for (int i = 0; i < 4; i++) {
                        btnCh[i].switchLanguage();
                  }
                  mainForm.repaint();
            }
            if (screen == 1) {
                  for (int i = 0; i < 4; i++) {
                        if (btnCh[i].test(x, y)) {
                              chosenTicket = i;
                  mainForm.notify(ConsoleMechanismEvents.E1);
                              mainForm.repaint();
                        }
                  }
            }
            if (screen > 1 && btnBack.test(x, y)) {
                  btnPay.load("pay_NA.png");
                  mainForm.notify(ConsoleMechanismEvents.E4);
                  mainForm.repaint();
            }
            if (screen == 2) {
                  for (int i = 5; i >= 0; i--) {
                        if (pickableAssignations[i].test(x, y)
                        && interfaceState.moneyEnabled == true) {
            InterfaceState.coinsIn += assignationNominals[i];
            mainForm.notify(MoneyMechanismEvents.E2);
                              mainForm.repaint();
                              break;
                        }
```

```
                        }
                    if (btnPay.test(x, y) && btnPay.fileName == "pay.png")
                    {
                        mainForm.notify(ConsoleMechanismEvents.E3);
                        mainForm.repaint();
                        }
                    }
                    if (outputDevice != null && outputDevice.test(x, y)) {
                        if (screen == 3) {
                            mainForm.notify(OutMechanismEvents.E5);
                        }
                        if (screen == 4) {
                            mainForm.notify(OutMechanismEvents.E7);
                        }
                    }
                }
            });
            interfaceLoaded = true;
        }

        void render(Graphics g) {
            Rectangle boundingRect = mainForm.getBounds();
            while (!interfaceLoaded)
                ;
            int boundingWidth, boundingHeight;
            boundingWidth = boundingRect.width;
            boundingHeight = boundingRect.height;
            // Refill screen
            g.setColor(new Color(0x3f3fa9));
            g.fillRect(0, 0, WIDTH, HEIGHT);
            // Screen
            MyRegion screenRegion = new MyRegion(g, Math.round(boundingWidth
                    * LEFT_MARGIN), Math.round(boundingHeight *
                    TOP_MARGIN), Math
                    .round(boundingHeight * SCREEN_SIZE),
                    Math.round(boundingWidth
                    * (1 - RIGHT_MARGIN - LEFT_MARGIN)));
            screenRegion.setBackColor(new Color(SCREEN_COLOR));
            screenRegion.setFrameColor(new Color(EDGE_COLOR));
            screenRegion.render();
            // Inside the screen
            // RUS & ENG language choice
            while (!btnRu.loaded || !btnEng.loaded)
                ;
            for (int i = 0; i < 6; i++) {
                while (!assignations[0].loaded)
                    ;
                while (!pickableAssignations[0].loaded)
                    ;
            }
            btnRu.applyLeftTopPosition(screenRegion.getShifted(IMAGE_MARGIN,
                    IMAGE_MARGIN));
            btnRu.draw(g);
            btnEng.applyLeftTopPosition(btnRu.getRegion().getShifted(
                    IMAGE_MARGIN + btnRu.getRegion().getWidth(), 0));
            btnEng.draw(g);
            if (screen <= 4) {
                btnBack.applyLeftTopPosition(btnEng.getRegion().getShifted(
                    IMAGE_MARGIN + btnEng.getRegion().getWidth(), 0));
            }
            if (screen == 2) {
                btnBack.draw(g);
            }
            btnLogo.applyLeftTopPosition(screenRegion.getShifted(screenRegion
```

```
                .getWidth()
                - 10 - btnLogo.getRegion().getWidth(), 10));
        btnLogo.draw(g);
        if (screen == 1) {
              btnCh[0].applyLeftTopPosition(screenRegion.getShifted(
              IMAGE_MARGIN + 10, Math.round(2.5f * btnEng.getRegion()
                                    .getHeight())));
              btnCh[0].draw(g);
              for (int i = 1; i < 4; i++) {
                    btnCh[i].applyLeftTopPosition(btnCh[i - 1].getRegion()
                    .getShifted(0,
                    Math.round(1.5f * btnCh[0].getRegion()
                                                      .getHeight())));
                    btnCh[i].draw(g);
              }
        }
        if (screen == 2) {
              MyRegion ticketTypeLabel = screenRegion.getShifted(
              IMAGE_MARGIN + 20, 3 * btnEng.getRegion().getHeight() + 10);
              Font font = new Font("Tahoma", Font.PLAIN, 22);
              g.setFont(font);
              if (InterfaceState.language) {
                    ticketTypeLabel.renderString("Тип билета: "
                                + ticketTypesRu[chosenTicket]);
              } else {
                    ticketTypeLabel.renderString("Ticket type: "
                                + ticketTypesEng[chosenTicket]);
              }
                          MyRegion priceLabel =
              ticketTypeLabel.getShifted(0, 10 +
              btnEng.getRegion().getHeight());
              if (InterfaceState.language) {
                    priceLabel.renderString("Цена билета: "
                                + ticketPrice[chosenTicket] + "р.");
              } else {
                    priceLabel.renderString("Ticket price: "
                                + ticketPrice[chosenTicket] + "rub.");
              }
              MyRegion assignationsLabel = screenRegion
                          .getShifted(IMAGE_MARGIN, -Math.round(1.7f *
              assignations[0].getRegion().getHeight()));
              if (InterfaceState.language) {
              assignationsLabel.renderString("Доступные купюры и
              монеты:");
              } else {
                    assignationsLabel.renderString("Notes and coins
              available:");
              }
              assignations[0].applyLeftTopPosition(screenRegion.getShifted
              (IMAGE_MARGIN, -Math.round(1.5f * assignations[0]
                                    .getRegion().getHeight()), false,
                                    true));
              assignations[0].draw(g);
              for (int i = 1; i < 5; i++) {
              assignations[i].applyLeftTopPosition(assignations[i - 1]
              .getRegion().getShifted(Math.round(assignations[i - 1]
              .getRegion().getWidth() * 0.2f), 0));
                    assignations[i].draw(g);
              }
              for (int i = 5; i < 6; i++) {
              assignations[i].applyLeftTopPosition(assignations[i - 1]
              .getRegion().getShifted(Math.round(assignations[i - 1]
              .getRegion().getWidth() * 0.6f), 0));
                    assignations[i].draw(g);
```

```
            }
            Font newFont = new Font("Tahoma", Font.BOLD, 36);
            g.setFont(newFont);
            String label = Integer.toString(InterfaceState.coinsIn);
            if (InterfaceState.language) {
                  label += "р.";
            } else {
                  label += "r.";
            }
            MyRegion coinsLabel = screenRegion.getShifted(-22 *
            label.length(),
                        -80, true, true);
            coinsLabel.renderString(label);
            btnPay.applyLeftTopPosition(screenRegion.getShifted(-80, -
            60, true, true));
            btnPay.draw(g);
      }
      if (screen == 3) {
            Font newFont = new Font("Tahoma", Font.BOLD, 36);
            g.setFont(newFont);
            String label;
            if (InterfaceState.language) {
                  label = "Заберите билет";
            } else {
                  label = "Take a ticket";
            }
            MyRegion takeTicketLabel = screenRegion.getShifted(0,
            screenRegion.getHeight() / 2 - 20);
            if (!(InterfaceState.coinsIn > ticketPrice[chosenTicket])) {
                  takeTicketLabel = takeTicketLabel.getShifted(0, 45);
            }
            takeTicketLabel.renderString(label, screenRegion);
            if (InterfaceState.coinsIn > ticketPrice[chosenTicket]) {
                  takeTicketLabel = takeTicketLabel.getShifted(0, 45);
                  if (InterfaceState.language) {
                        label = "и";
                  } else {
                        label = "and";
                  }
                  takeTicketLabel.renderString(label, screenRegion);
                  takeTicketLabel = takeTicketLabel.getShifted(0, 45);
                  if (InterfaceState.language) {
                        label = "сдачу";
                  } else {
                        label = "change";
                  }
                  takeTicketLabel.renderString(label, screenRegion);
            }
      }
      if (screen == 4) {
            Font newFont = new Font("Tahoma", Font.BOLD, 36);
            g.setFont(newFont);
            String label;
            if (InterfaceState.language) {
                  label = "Заберите деньги";
            } else {
                  label = "Take money back";
            }
            MyRegion moneyBackLabel = screenRegion.getShifted(0,
            screenRegion.getHeight() / 2 + 25);
            moneyBackLabel.renderString(label, screenRegion);
      }
      if (screen == 5) {
            Font newFont = new Font("Tahoma", Font.BOLD, 36);
```

```
		g.setFont(newFont);
		String label;
		if (InterfaceState.language) {
			label = "Устройство";
		} else {
			label = "Device";
		}
		MyRegion deviceOutOfService = screenRegion.getShifted(0,
				screenRegion.getHeight() / 2 - 10);
		deviceOutOfService.renderString(label, screenRegion);
		if (InterfaceState.language) {
			label = "не работает!";
		} else {
			label = "doesn't work!";
		}
		deviceOutOfService = deviceOutOfService.getShifted(0, 45);
		deviceOutOfService.renderString(label, screenRegion);
	}
	// Coin outside
	MyRegion coinOutside = screenRegion.getShifted(0, Math
			.round(boundingHeight * FUNC_MARGIN), false, true);
	coinOutside.setDimensions(Math.round(boundingWidth * FUNC_HSIZE),
	Math.round(boundingHeight * FUNC_VSIZE));
	if (InterfaceState.moneyEnabled) {
		coinOutside.setBackColor(new Color(MONEY_ALLOWED));
	} else {
		coinOutside.setBackColor(new Color(MONEY_DISALLOWED));
	}
	coinOutside.render();
	// Coin inside
	MyRegion coinInside = coinOutside
			.getShifted(3 * coinOutside.getWidth() / 8 + 1,
	coinOutside.getHeight() / 8);
	coinInside.setDimensions(coinOutside.getWidth() / 4, 3 *
	coinOutside.getHeight() / 4);
	coinInside.setBackColor(new Color(0));
	coinInside.render();
	// Money outside
	MyRegion moneyOutside =
	coinOutside.getShifted(Math.round(boundingWidth
			* MONEY_MARGIN), 0, true, false);
	moneyOutside.setDimensions(Math.round(boundingWidth *
	MONEY_HSIZE),
			Math.round(boundingHeight * MONEY_VSIZE));
	if (InterfaceState.moneyEnabled) {
		moneyOutside.setBackColor(new Color(MONEY_ALLOWED));
	} else {
		moneyOutside.setBackColor(new Color(MONEY_DISALLOWED));
	}
	moneyOutside.render();
	// Money inside
	MyRegion moneyInside =
		moneyOutside.getShifted(Math.round(moneyOutside
			.getWidth() * (1 - MONEY_INSIDE_VSIZE) / 2),
	Math.round(moneyOutside.getHeight()
			* (1 - MONEY_INSIDE_HSIZE) / 2));
	moneyInside.setBackColor(new Color(0));
	moneyInside.setDimensions(Math.round(moneyOutside.getWidth()
			* MONEY_INSIDE_VSIZE),
	Math.round(moneyOutside.getHeight()
			* MONEY_INSIDE_HSIZE));
	moneyInside.render();
	// Output
	outputDevice = coinOutside.getShifted(0, Math.round(boundingHeight
```

```
                        * OUTPUT_MARGIN), false, true);
        outputDevice.setDimensions(moneyOutside.getWidth()
                        + moneyOutside.getLeft() - coinOutside.getLeft(), Math
                        .round(boundingHeight * OUTPUT_VSIZE));
        outputDevice.setBackColor(new Color(0x6666cc));
        outputDevice.render();
        if (screen == 3 && (InterfaceState.coinsIn >
              ticketPrice[chosenTicket]) || screen == 4) {
              outputDevice.setBackColor(new Color(0xcccc66));
              outputDevice.render();
              btnChange.applyLeftTopPosition(outputDevice.getShifted(10,
                    5));
              btnChange.draw(g);
        }
        if (screen == 3) {
              btnTicket.applyLeftTopPosition(outputDevice.getShifted(75,
                    10));
              btnTicket.draw(g);
        }
        // Auxilary
        MyRegion auxScreen = moneyOutside.getShifted(20, 0, true, false);
        auxScreen.setWidth(Math.round((1 - RIGHT_MARGIN) * boundingWidth)
                        - auxScreen.getLeft());
        auxScreen.setHeight(outputDevice.getTop() +
              outputDevice.getHeight() - auxScreen.getTop());
        auxScreen.setBackColor(new Color(0x353795));
        auxScreen.render();
        // Pickable money
        pickableAssignations[0].getRegion().setLeft(
                        Math.round(boundingWidth * LEFT_MARGIN));
        pickableAssignations[0].getRegion().setTop(
                        boundingHeight - Math.round(1.3f *
        pickableAssignations[0].getRegion() .getHeight()));
        // Divizor
        MyRegion divizor = new MyRegion(g, 0, (outputDevice.getTop()
                        + outputDevice.getHeight() + pickableAssignations[0]
                        .getRegion().getTop()) / 2, 1, boundingWidth);
        divizor.setBackColor(new Color(0));
        // Background refill
        MyRegion bottomRefill = divizor.getShifted(0, 0);
        bottomRefill.setBackColor(new Color(0xebe9ed));
        bottomRefill.setHeight(boundingHeight - bottomRefill.getTop());
        bottomRefill.render();
        // Divizor rendering
        divizor.render();
        // <-- Pickable money
        pickableAssignations[0].draw(g);
        for (int i = 1; i < 5; i++) {
              pickableAssignations[i]
              .applyLeftTopPosition(pickableAssignations[i - 1]
              .getRegion().getShifted(
              Math.round(pickableAssignations[i - 1]
                        .getRegion().getWidth() * 0.5f), 0));
              pickableAssignations[i].draw(g);
        }
        for (int i = 5; i < 6; i++) {
              pickableAssignations[i]
                        .applyLeftTopPosition(pickableAssignations[i - 1]
              .getRegion()
              .getShifted(Math.round(pickableAssignations[i - 1]
        .getRegion().getWidth() * 0.8f), 0));
              pickableAssignations[i].draw(g);
        }
}
```

```
}
```

InterfaceState.java

```
package device.ui;

public class InterfaceState {
    public static boolean moneyEnabled = false;

    public static int outMechanismState;

    /**
     * true - russian language
     * false - english language
     */
    public static boolean language = true;

    public static boolean payButtonState;

    public static int coinsIn;

    public static int blanksCount = 5;

    static void switchLanguage(){
          language = !language;
    }

    static void switchMoneyAccess(){
          moneyEnabled = !moneyEnabled;
    }
}
```

MyButton.java

```
package device.ui;

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

import javax.swing.JFrame;

public class MyButton {

    public final static int ACTIVE_COLOR = 65280;

    public String fileName;

    protected MyRegion btnRegion = null;

    Image buttonImage;

    boolean isActive = false;

    volatile boolean loaded = false;

    void setActive() {
          isActive = true;
    }

    void setPassive() {
          isActive = false;
```

```
    }

    public MyButton() {
          btnRegion = new MyRegion();
    }

    void load(String fileName) {
          MediaTracker tracker = new MediaTracker(new Container());
          if (fileName != "") {
                File readfile = new File(imageDirectory + fileName);
                try {
                      buttonImage = javax.imageio.ImageIO.read(readfile);
                      tracker.addImage(buttonImage, 0);
                      try {
                            tracker.waitForID(0);
                      } catch (InterruptedException ex1) {
                      }
                      this.fileName = fileName;
                } catch (IOException ex) {
                      System.out.print("Image files haven't been found!");
                      System.exit(0);
                }
                loaded = true;
                btnRegion.setWidth(buttonImage.getWidth(null));
                btnRegion.setHeight(buttonImage.getHeight(null));
          }
    }

    void setTheSameImage(MyButton btn){
          buttonImage = btn.buttonImage;
          btnRegion.setWidth(buttonImage.getWidth(null));
          btnRegion.setHeight(buttonImage.getHeight(null));
    }

    boolean test(int x, int y) {
          return (x >= btnRegion.getLeft()
                      && x <= btnRegion.getLeft() + btnRegion.getWidth()
                      && y >= btnRegion.getTop() && y <= btnRegion.getTop()
                      + btnRegion.getHeight());
    }

    void draw(Graphics g) {
          g.drawImage(buttonImage, btnRegion.getLeft(), btnRegion.getTop(),
                null);
          if (isActive) {
                g.setColor(new Color(ACTIVE_COLOR));
                g.drawRect(btnRegion.getLeft(), btnRegion.getTop(),
                btnRegion.getWidth(), btnRegion.getHeight());
                g.setColor(new Color(0));
          }
    }

    public MyRegion getRegion() {
          return btnRegion;
    }

    public void applyLeftTopPosition(MyRegion btnRegion) {
          this.btnRegion.setLeft(btnRegion.getLeft());
          this.btnRegion.setTop(btnRegion.getTop());
    }
}
```

MyLabeledButton.java

```
package device.ui;

import java.awt.Font;
import java.awt.Graphics;

class MyLabeledButton extends MyButton {
    public String textRu = "";

    public String textEng = "";

    /**
     * true - russian language false - english language
     */
    public boolean language = true;

    void draw(Graphics g) {
          super.draw(g);
          Font font = new Font("Tahoma", Font.PLAIN, 22);
          g.setFont(font);
          if (language) {
                g.drawString(textRu, btnRegion.getLeft() +
                btnRegion.getWidth() + 20, btnRegion.getTop() +
                btnRegion.getWidth() / 2 + 10);
          } else {
                g.drawString(textEng, btnRegion.getLeft() +
                btnRegion.getWidth() + 20, btnRegion.getTop() +
                btnRegion.getWidth() / 2 + 10);
          }
    }

    void switchLanguage() {
          language = !language;
    }
}
```

MyRegion.java

```
package device.ui;

import java.awt.Color;
import java.awt.FontMetrics;
import java.awt.Graphics;

public class MyRegion {
    private int left, top, height, width;

    private Graphics g;

    private Color backColor = null, frameColor = null;

    public MyRegion() {
    }

    public MyRegion(Graphics g, int left, int top, int height, int width) {
          this.g = g;
          this.left = left;
          this.top = top;
          this.height = height;
          this.width = width;
    }
```

```
public MyRegion getShifted(int rightShift, int downShift) {
      MyRegion newRegion = new MyRegion(g, left + rightShift,
                  top + downShift, height, width);
      return newRegion;
}

public MyRegion getShifted(int rightShift, int downShift,
            boolean fromRight, boolean fromBottom) {
      if (fromRight) {
            rightShift += width;
      }
      if (fromBottom) {
            downShift += height;
      }
      MyRegion newRegion = new MyRegion(g, left + rightShift,
                  top + downShift, height, width);
      return newRegion;
}

boolean test(int x, int y) {
      return (x >= left && x <= left + width && y >= top && y <= top +
            height);
}

public void render() {

      Color tmpColor = g.getColor();
      if (backColor != null) {
            g.setColor(backColor);
            g.fillRect(left, top, width, height);
      }
      if (frameColor != null) {
            g.setColor(frameColor);
      } else {
            g.setColor(new Color(0));
      }
      g.drawRect(left, top, width, height);
      g.setColor(tmpColor);
}

public void renderString(String text) {
      g.drawString(text, left, top);
}

public void renderString(String text, MyRegion centerBy) {
      FontMetrics fontMetrics = g.getFontMetrics();
      left = centerBy.left + (centerBy.getWidth() -
            fontMetrics.stringWidth(text)) / 2;
      g.drawString(text, left, top);
}

public void setBackColor(Color backColor) {
      this.backColor = backColor;
}

public void setFrameColor(Color frameColor) {
      this.frameColor = frameColor;
}

public int getHeight() {
      return height;
}

public int getLeft() {
```

```
        return left;
    }

    public int getTop() {
        return top;
    }

    public int getWidth() {
        return width;
    }

    public void setHeight(int height) {
        this.height = height;
    }

    public void setLeft(int left) {
        this.left = left;
    }

    public void setTop(int top) {
        this.top = top;
    }

    public void setWidth(int width) {
        this.width = width;
    }

    public void setDimensions(int width, int height) {
        this.width = width;
        this.height = height;
    }
}
```

Package `device` (объекты управления и поставщики событий)

`ConsoleMechanism.java`

```
package device;

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

import device.ui.*;

public class ConsoleMechanism implements ControlledObject {

    public static MainForm mainForm;

    public static void init(ModelEngine engine) {
        mainForm = new MainForm(engine);
    }

    /**
     * @unimod.action.descr Начальная форма выбора типа билета
     */
    public void z1(StateMachineContext context) {
        EventHandler.switchConsoleState(EventHandler.CHOISE_CONSOLE);
        EventHandler.switchOutMechanism(EventHandler.OM_EMPTY);
        EventHandler.switchPayButtonState(false);
        InterfaceState.coinsIn = 0;
    }
```

```
/**
 * @unimod.action.descr Отображение выбранного типа билета, допустимых
 * купюр и монет, введенной суммы
 */
public void z2(StateMachineContext context) {
	EventHandler.switchConsoleState(EventHandler.MONEY_CONSOLE);
}

/**
 * @unimod.action.descr Форма "Оплатить"
 */
public void z3(StateMachineContext context) {
	EventHandler.switchPayButtonState(true);
}

/**
 * @unimod.action.descr Сообщение "Заберите билет и сдачу"
 */
public void z4(StateMachineContext context) {

EventHandler.switchConsoleState(EventHandler.CHANGE_TICKET_CONSOLE);
}

/**
 * @unimod.action.descr Автомат не работает
 */
public void z5(StateMachineContext context) {
	mainForm.dispose();
}

/**
 * @unimod.action.descr Обновить введенную сумму
 */
public void z6(StateMachineContext context) {
	EventHandler.refreshMoneyAmount(x1(context));
}

/**
 * @unimod.action.descr Сообщение "Вводите только купюры и монеты
 * из списка"
 */
public void z7(StateMachineContext context) {
	/* TODO: automatically generated by UniMod method */
}

/**
 * @unimod.action.descr Введенная в автомат сумма
 */
public int x1(StateMachineContext context) {
	/* TODO: automatically generated by UniMod method */
	return InterfaceState.coinsIn;
}

/**
 * @unimod.action.descr Сообщение "Заберите введенные деньги"
 */
public void z8(StateMachineContext context) {
	EventHandler.switchConsoleState(EventHandler.MONEYBACK_CONSOLE);
}

/**
 * @unimod.action.descr Сообщение "Автомат не работает"
 */
public void z9(StateMachineContext context) {
```

```
            EventHandler.switchConsoleState(EventHandler.OUTOFPAPER_CONSOLE);
        }

    }
```

ConsoleMechanismEvents.java

```
    package device;

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

    import device.ui.*;

    public class ConsoleMechanismEvents implements EventProvider {
        /**
         * @unimod.event.descr Выбран тип билета
         */
        public static final String E1 = "e1";

        /**
         * @unimod.event.descr Нажата кнопка "Оплатить"
         */
        public static final String E3 = "e3";

        /**
         * @unimod.event.descr Нажата кнопка "Перейти в начало"
         */
        public static final String E4 = "e4";

        /**
         * @unimod.event.descr Выключить автомат
         */
        public static final String E6 = "e6";

        public void dispose() {

        }

        public static ModelEngine engine;

        public void init(ModelEngine engine) throws CommonException {
              ConsoleMechanismEvents.engine = engine;
              ConsoleMechanism.init(engine);
              ConsoleMechanism.mainForm.notify("");
        }

        public static void setTimeout(final String event, int time_delay) {
              java.util.Timer t = new java.util.Timer(false);
              t.schedule(new java.util.TimerTask() {
                    public void run() {
                          engine.getEventManager().handle(new Event(event),
                                      StateMachineContextImpl.create());
                    }
              }, time_delay);
        }

    }
```

MoneyMechanism.java

```
package device;

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

import device.ui.*;

public class ConsoleMechanismEvents implements EventProvider {
    /**
     * @unimod.event.descr Выбран тип билета
     */
    public static final String E1 = "e1";

    /**
     * @unimod.event.descr Нажата кнопка "Оплатить"
     */
    public static final String E3 = "e3";

    /**
     * @unimod.event.descr Нажата кнопка "Перейти в начало"
     */
    public static final String E4 = "e4";

    /**
     * @unimod.event.descr Выключить автомат
     */
    public static final String E6 = "e6";

    public void dispose() {
          // TODO Auto-generated method stub

    }

    public static ModelEngine engine;

    public void init(ModelEngine engine) throws CommonException {
          ConsoleMechanismEvents.engine = engine;
          ConsoleMechanism.init(engine);
          ConsoleMechanism.mainForm.notify("");
    }

    public static void setTimeout(final String event, int time_delay) {
          java.util.Timer t = new java.util.Timer(false);
          t.schedule(new java.util.TimerTask() {
                public void run() {
                      engine.getEventManager().handle(new Event(event),
                                  StateMachineContextImpl.create());
                }
          }, time_delay);
    }

}
```

MoneyMechanismEvents.java

```
package device;

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

public class MoneyMechanismEvents implements EventProvider {
    /**
     * @unimod.event.descr В купюроприемник поступила купюра
     */
    public static final String E2 = "e2";

    public void init(ModelEngine engine) throws CommonException {
          // TODO Auto-generated method stub

    }

    public void dispose() {
          // TODO Auto-generated method stub

    }

}
```

OutMechanism.java

```
package device;

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

import device.ui.EventHandler;

public class OutMechanism implements ControlledObject {

    /**
     * @unimod.action.descr Выдать билет
     */
    public void z1(StateMachineContext context) {
          EventHandler.switchOutMechanism(EventHandler.OM_FILLED);
    }

    /**
     * @unimod.action.descr Выдать деньги (сдача или возврат)
     */
    public void z2(StateMachineContext context) {
          EventHandler.switchOutMechanism(EventHandler.OM_FILLED);
    }

}
```

OutMechanismEvents.java

```
package device;

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

```
public class OutMechanismEvents implements EventProvider {
    /**
     * @unimod.event.descr Выдан билет и сдача (опционально)
     */
    public static final String E5 = "e5";

    /**
     * @unimod.event.descr Возвращены деньги (при отмене операции)
     */
    public static final String E7 = "e7";

    public void init(ModelEngine engine) throws CommonException {
          // TODO Auto-generated method stub

    }

    public void dispose() {
          // TODO Auto-generated method stub

    }

}
package device;

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

public class OutMechanismEvents implements EventProvider {
    /**
     * @unimod.event.descr Выдан билет и сдача (опционально)
     */
    public static final String E5 = "e5";

    /**
     * @unimod.event.descr Возвращены деньги (при отмене операции)
     */
    public static final String E7 = "e7";

    public void init(ModelEngine engine) throws CommonException {
          // TODO Auto-generated method stub

    }

    public void dispose() {
          // TODO Auto-generated method stub

    }

}
```

PrintMechanism.java

```
package device;

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

import device.ui.InterfaceState;

public class PrintMechanism implements ControlledObject {
    /**
```

```
     * @unimod.action.descr Количество бланков для печати билетов
     */
    public int x4(StateMachineContext context) {
          return InterfaceState.blanksCount;
    }

    /**
     * @unimod.action.descr Уменьшить количество бланков на 1
     */
    public void z1(StateMachineContext context) {
          InterfaceState.blanksCount--;
    }

}
```