Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

В.Р. Данилов, И.О. Варвалюк

Автоматизированная система оплаты мобильного телефона

Программирование с явным выделением состояний

Проектная документация

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

Санкт-Петербург
2006

Оглавление

Введение

В данной работе приведен пример применения автоматного подхода для проектирования устройства для оплаты мобильного телефона с использованием среды разработки *Unimod*.

Для алгоритмизации и программирования задач управления техническими объектами удобен автоматный подход, поскольку графы переходов автоматов наглядно представляют работу объекта. Это приводит к тому, что можно легко увидеть возможные ошибки в проектировании, такие как отсутствие некоторого перехода, недоступность состояния и т.д.

Использование среды разработки *Unimod* позволяет изначально исходить из автоматного подхода. При ее использовании сначала строится автомат, а затем на языке *Java* программируются по отдельности используемые в автомате объекты. Таким образом, программа разделяется на отдельные независимые блоки, в результате чего облегчается ее написание, и снижается вероятность возникновения ошибок.

1. Постановка задачи

Спроектировать и реализовать работу устройства для оплаты мобильного телефона.

Это устройство состоит из клавиатуры для ввода номера телефона, панели вывода информации, набора купюр, и кнопки управления устройством.

На рис.1 изображена лицевая панель устройства.

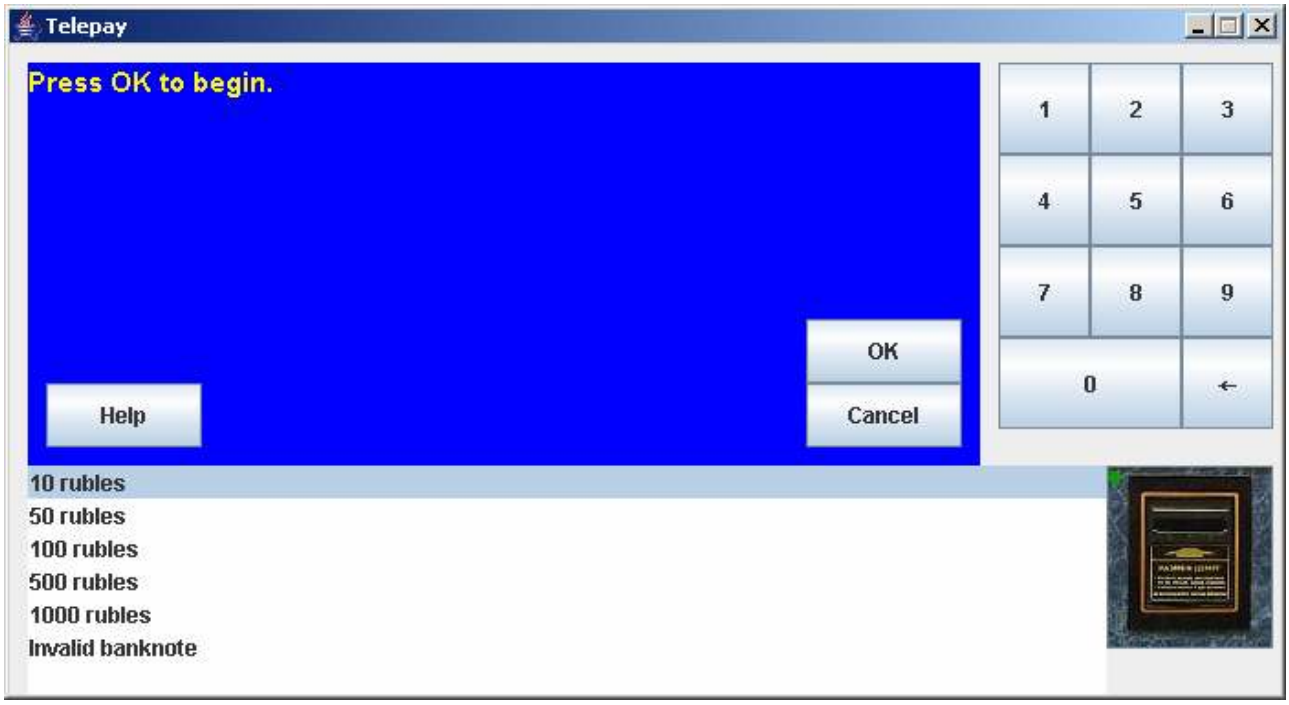

Рис. 1. Лицевая панель устройства

В правой стороне панели находится клавиатура для ввода номера телефона.

Под ней располагается мини-панель, содержащая кнопку, нажатие на которую сигнализирует о приеме или возврате денег, светодиод, сигнализирующий готовность к приему купюр и приемник купюр. При этом:

- если светодиод светится зеленым цветом, то нажатие на кнопку приводит к переводу банкноты в приемник для последующего зачисления на счет;
- если светодиод светится красным цветом, то необходимо вынуть «порченую купюру» из устройства, просто нажав ещё раз на кнопке.

Список номиналов банкнот приведен на панели слева.

Также в левом верхнем углу находится текстовый терминал с инструкциями пользователю устройства. На терминале присутствуют две кнопки управления: «OK» и «Cancel»:

- «OK» - переход к следующему шагу;
- «Отмена» - отмена текущего действия и возврат к предыдущему шагу.

Кнопка «Справка», находящаяся на терминале, служит для вызова окна помощи.

2. Автоматная реализация

Описанную систему логично разделить следующим образом: клавиатуру, кнопки управления, кнопку помощи и приемник банкнот представить как генераторы событий, а текстовый терминал, информацию о платеже и приемник банкнот – как объекты управления.

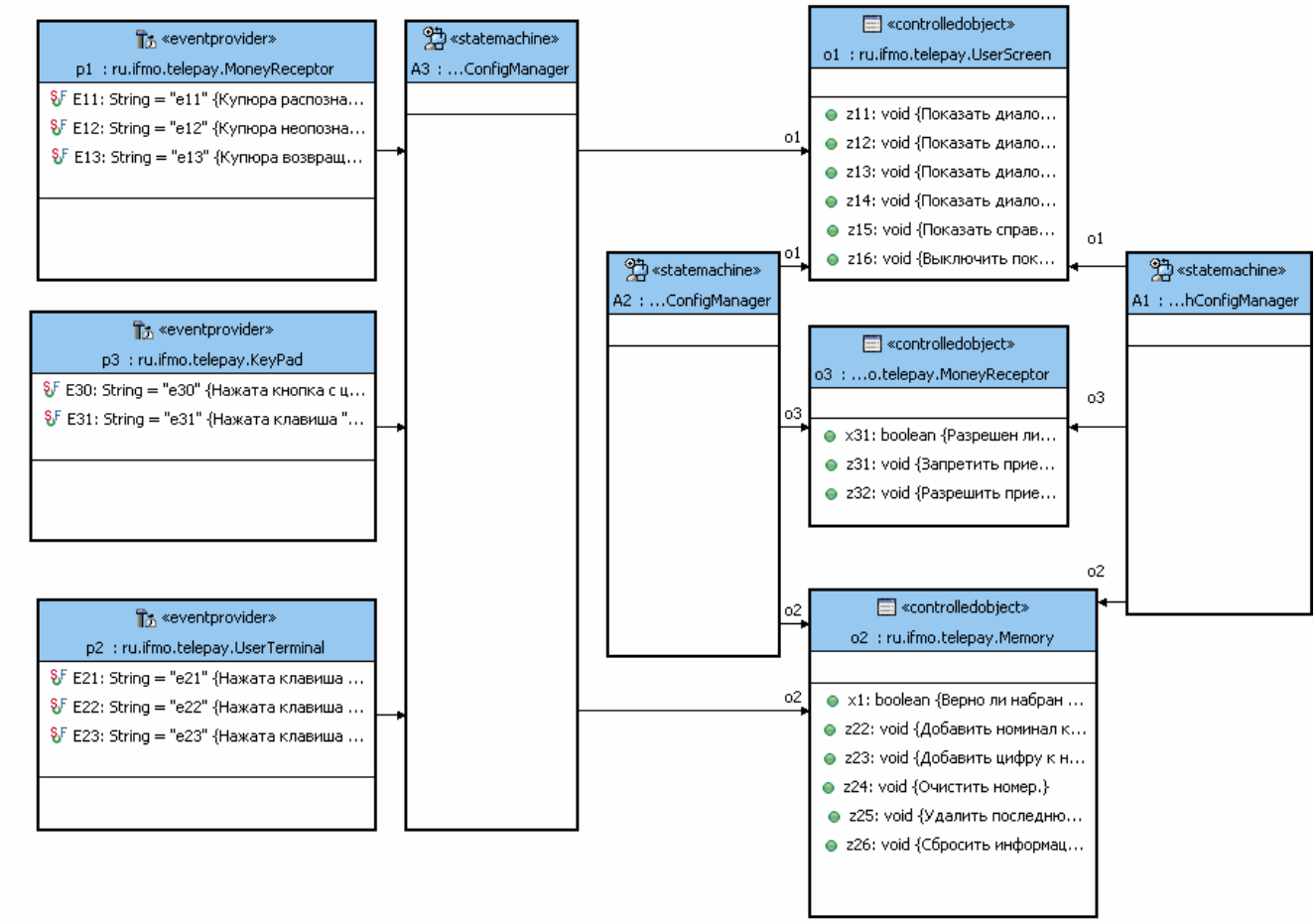

Рис. 2. Схема связей

2.1. Автоматы

Как видно из схемы связей, в системе имеется три автомата: *A1*, *A2 и A3*. Первый автомат эмулирует работу устройства, второй контролирует работу приемника банкнот. Третий автомат управляет переключением между режимами «справки» и «работы».

При этом первый автомат вкладывается в одно из состояний третьего, а второй в одно из состояний первого. Использование вложенных автоматов оправдано тем, что это сильно упрощает схему системы и уменьшает количество переходов.

На Рис. 3 изображен автомат *A1*. Его схема отражает последовательность действий, которые необходимо выполнить пользователю.

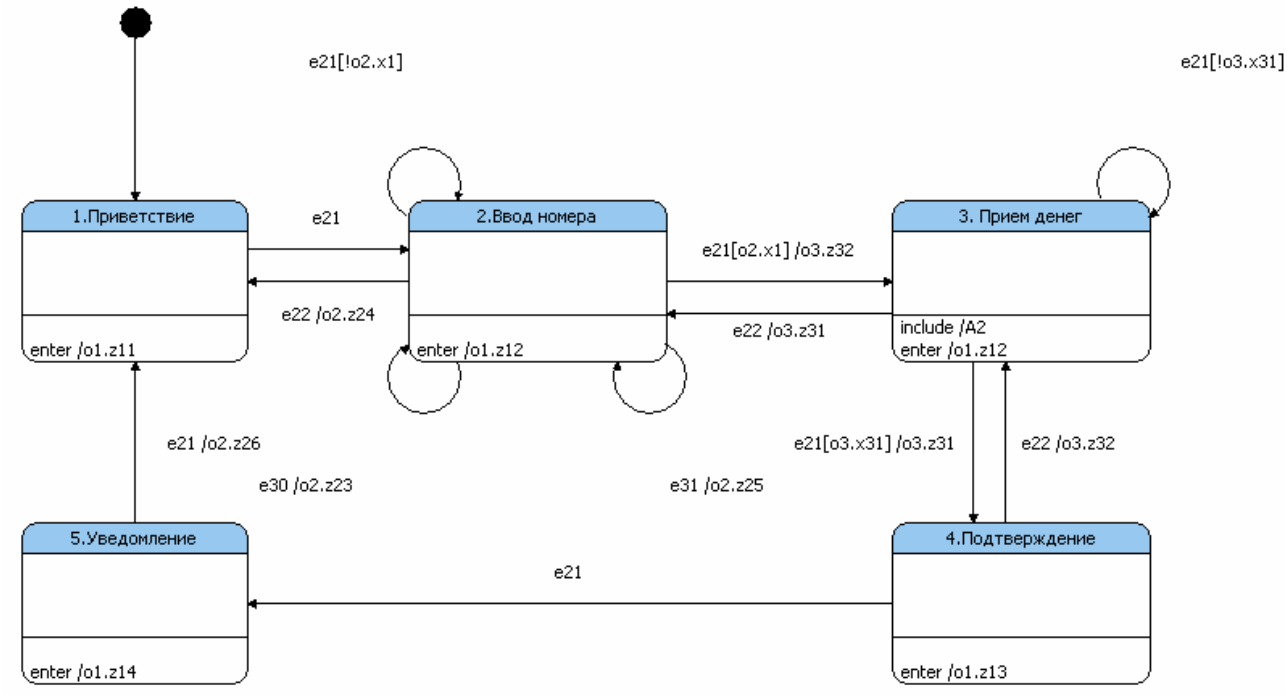

Рис. 3. Автомат *A1*

Как видно, автомат состоит из пяти состояний:

«1. Приветствие» — начальное состояние, нахождение автомата в котором сигнализирует о готовности устройства к работе;

«2. Ввод номера» — состояние, при нахождении в котором происходит ввод номера;

«*3. Прием денег*» — состояние отвечает за ввод денег, т.е. когда автомат находится в этом состоянии, устройство принимает банкноты. (Для упрощения архитектуры, в этом состоянии подключается автомат *A2*, отвечающий за правильность ввода банкнот);

«4. Подтверждение» — в этом состоянии, пользователю необходимо подтвердить правильность введенных данных, или же, в противном случае, вернуться в предыдущие состояния для их корректирования;

«5. Уведомление» — состояние автомата, сигнализирующее об успешном выполнении платежа на номер абонента.

Автомат *A2* контролирует процесс ввода банкнот. Схема автомата изображена на Рис. 4.

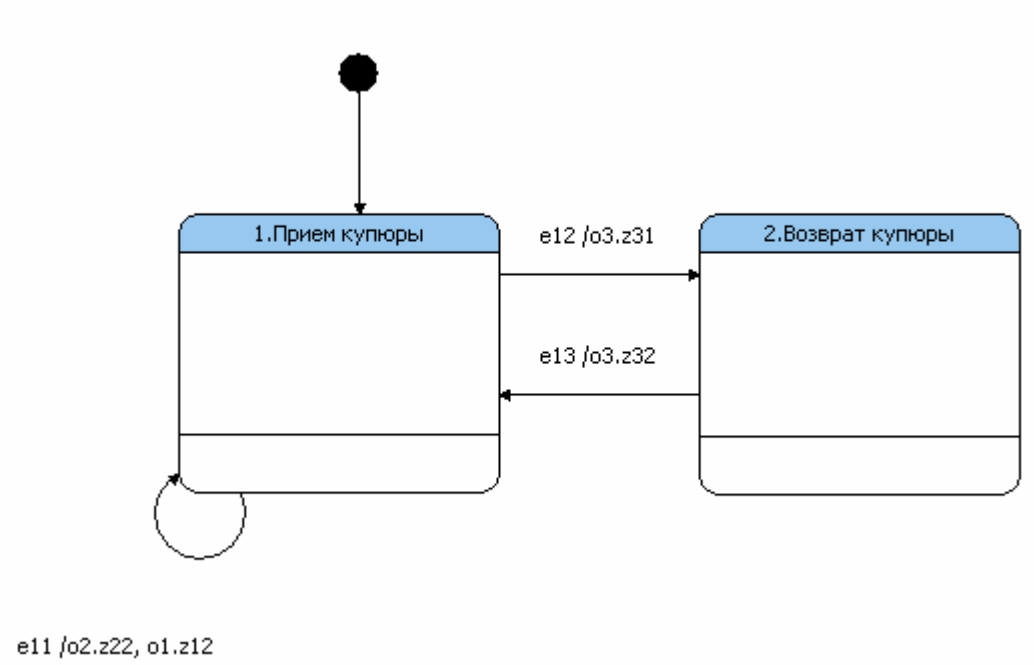

Рис. 4 Автомат *A2*

«*1. Прием купюры*» — состояние, при нахождении в котором осуществляется прием купюр от пользователя;

«*2. Возврат купюры*» — состояние, в котором осуществляется возврат купюр.

Из схемы видно, что при подаче «порченой купюры» автомат переходит в состояние возврата купюры, выйти из которого невозможно иначе, как забрав купюру из приемника.

Автомат *A3* управляет переключением между режимами работы и помощи. Переход происходит при нажатии кнопки «Help».

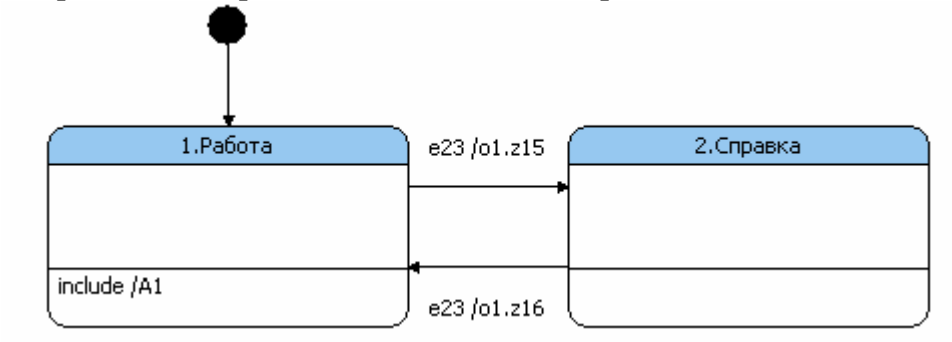

Рис. 5 Автомат *A3*

«*1. Работа*» — состояние, отвечающее режиму работы пользователя с устройством. Заметим, что автомат *A1*, отражающий логику взаимодействия, вложен в это состояние;

«*2. Справка*» — состояние, отвечающее показу справки по использованию устройства.

2.2. Генераторы событий

Далее будет описан физический смысл и описание генераторов событий данной системы.

Генератор событий *p1*

Этот объект описывает события, производимые устройством по приему денег.

- **e11** – прием банкноты;
- **e12** – «порченая» банкнота;
- **e13** – абонент забрал «порченую» банкноту.

Генератор событий *p2*

Данный объект соответствует кнопкам управления состоянием устройства.

- **e21** – нажата кнопка «ОК»;
- **e22** – нажата кнопка «Cancel».

Генератор событий *p3*

Этот объект соответствует клавиатуре для ввода номера:

- **e31** – нажата цифра;
- **e32** – нажата клавиша удаления последней цифры.

2.3. Объекты управления

Объект управления *o1*

Этот объект соответствует экрану терминала устройства.

- **z11** – показать диалог приветствия;
- **z12** – показать диалог для ввода информации о номере;
- **z13** – показать диалог подтверждения корректности введенных данных;
- **z14** – показать диалог об успешно проведенном платеже;
- **z15 –** показать экран помощи**.**

Объект управления *o2*

Этот объект используется для хранения информации о проводимом платеже.

- **x1** – логическая переменная, отражающая корректность введенного номера;
- **z22** – добавляет к текущей сумме платежа, номинал выбранной купюры;
- **z23** – добавляет к текущему номеру цифры;
- **z24** – очищает номер;
- **z25** – удаляет последнюю цифру.

Объект управления *о3*

Данный объект управления отражает работу светодиода на приёмнике купюр.

- **x31** – логическая переменная, отвечающая за готовность устройства к приему купюр;
- **z31** – изменяет цвет светодиода на красный, что означает, что устройству была подана «порченая» купюра;
- **z32** – изменяет цвет светодиода на зеленый, что означает, что устройство готово к приему купюр.

3. Реализация программы

Программа реализуется с помощью плагина *Unimod* к среде разработки *Java*-приложений *Eclipse*. Указанный плагин позволяет построить схему связей, а также автоматы, изображенные на Рис. 2-4. Далее реализуются объекты управления, генераторы событий и пользовательский интерфейс программы на языке *Java*. Вся логика работы приложения описывается в автомате.

Существуют два подхода к созданию исполняемого приложения: *интерпретационный* и *компилятивный*.

3.1. Интерпретационный подход

Данный подход требует значительного использования библиотек *Unimod* для работы приложения.

Построенный автомат сохраняется в формате *XML*, затем компилируются классы, реализующие работу приложения. Теперь можно запускать приложение, используя набор библиотек *Unimod*, *XML*-файл и откомпилированные классы. При работе в командную консоль будут выводиться комментарии *Unimod* о происходящих событиях и совершаемых переходах в автомате. Данный подход невыгоден тем, что требует подключения большого количества библиотек, которые для работы данного конкретного приложения не являются необходимыми.

3.2. Компилятивный подход

При использовании данного подхода с помощью средств *Unimod* из *XML*-файла, описывающего автомат, генерируется *Java*-файл. В нем явно описываются все объекты системы, события, автоматы, их состояния и переходы. Для его работы также необходимы некоторые классы библиотек Unimod, однако гораздо меньшее их количество, чем при использовании интерпретационного подхода.

Сгенерированный файл находится в *default package* и называется `ModelEventProcessor.java`.

3.3. Запуск программы

Как запустить программу? Необходимо скачать и распаковать архив с исполняемой программой. После этого запустить файл *run.bat.* Как открыть проект в *Eclipse*? Необходимо в среде разработки *Eclipse* выбрать пункт меню "*File/Import:*" ("*Файл/Импорт:*"), а в открывшемся диалоге мастера - пункт "*Existing Project into Workspace*" ("*Существующий проект в рабочую область*") и нажать кнопку "*Next >*" ("*Далее >*"). В окне выбора папки с файлами проекта необходимо указать ту папку, в которую распакован архив с проектом.

Выводы

Автоматный подход идеально подходит для проектирования такого рода технической системы, как устройство для приема платажей. Ведь наглядная диаграмма автомата позволяет легко видеть схему работы и возможные ошибки, достаточно весомые в этой области.

Инструментальное средство *Unimod* сильно упрощает реализацию спроектированной системы. Заметным преимуществом является невозможность

генерирования программ по некорректным или неполным спецификациям. Это ведет к раннему выявлению ошибок проектирования.

Данная модель аппарата для приема платежей на мобильный телефон была построена не для использования в реальных условиях. Целью работы было продемонстрировать удобство автоматного подхода и его применения с помощью *Unimod*.

По нашему мнению, проектирование программ с помощью *Unimod* значительно корректнее, чем с помощью традиционного подхода, так как не требуется думать о реализации программы в целом, а только об ее архитектуре.Это в свою очередь вынуждает разделять программу на слои, один из которых отвечает только за архитектуру программы, а другой за непосредственно работу каждого из модулей, не привязывая их сильно к архитектуре программы.

Приложение 1. Пример протокола работы программы

Протокол приводится для следующей последовательности действий. Положить 200 рублей на номер 88123883636, в процессе вызвать помощь.

```
04:29:15,500 INFO  [Run] Start event [e22] processing. In state [/A3:Top]
04:29:15,500 INFO  [Run] Transition to go found [s1#1.Работа##true]
04:29:15,500 INFO  [Run] Start event [e22] processing. In state [/A3:1.Работа/A1:Top]
04:29:15,500 INFO  [Run] Transition to go found [s1#1.Приветствие##true]
04:29:15,500 INFO  [Run] Start on-enter action [o1.z11] execution
04:29:15,515 INFO  [Run] Finish on-enter action [o1.z11] execution
04:29:15,515 INFO  [Run] Finish event [e22] processing. In state
[/A3:1.Работа/A1:1.Приветствие]
04:29:15,515 INFO  [Run] Finish event [e22] processing. In state [/A3:1.Работа]
04:29:16,359 INFO  [Run] Start event [e21] processing. In state [/A3:1.Работа]
04:29:16,359 INFO  [Run] Start event [e21] processing. In state
[/A3:1.Работа/A1:1.Приветствие]
04:29:16,359 DEBUG [Run] Try transition [1.Приветствие#2.Ввод номера#e21#true]
04:29:16,359 INFO  [Run] Transition to go found [1.Приветствие#2.Ввод номера#e21#true]
04:29:16,359 INFO  [Run] Start on-enter action [o1.z12] execution
04:29:16,375 INFO  [Run] Finish on-enter action [o1.z12] execution
04:29:16,375 INFO  [Run] Finish event [e21] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:16,375 INFO  [Run] Finish event [e21] processing. In state [/A3:1.Работа]
04:29:17,125 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа]
04:29:17,125 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:17,125 DEBUG [Run] Try transition [2.Ввод номера#2.Ввод номера#e30#true]
04:29:17,125 INFO  [Run] Transition to go found [2.Ввод номера#2.Ввод номера#e30#true]
04:29:17,125 INFO  [Run] Start output action [o2.z23] execution
04:29:17,125 INFO  [Run] Finish output action [o2.z23] execution
04:29:17,125 INFO  [Run] Start on-enter action [o1.z12] execution
04:29:17,125 INFO  [Run] Finish on-enter action [o1.z12] execution
04:29:17,125 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:17,125 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа]
04:29:17,531 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа]
04:29:17,531 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:17,531 DEBUG [Run] Try transition [2.Ввод номера#2.Ввод номера#e30#true]
04:29:17,531 INFO  [Run] Transition to go found [2.Ввод номера#2.Ввод номера#e30#true]
04:29:17,531 INFO  [Run] Start output action [o2.z23] execution
04:29:17,531 INFO  [Run] Finish output action [o2.z23] execution
04:29:17,531 INFO  [Run] Start on-enter action [o1.z12] execution
04:29:17,531 INFO  [Run] Finish on-enter action [o1.z12] execution
04:29:17,546 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:17,546 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа]
04:29:18,218 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа]
04:29:18,218 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:18,218 DEBUG [Run] Try transition [2.Ввод номера#2.Ввод номера#e30#true]
04:29:18,218 INFO  [Run] Transition to go found [2.Ввод номера#2.Ввод номера#e30#true]
04:29:18,218 INFO  [Run] Start output action [o2.z23] execution
04:29:18,218 INFO  [Run] Finish output action [o2.z23] execution
04:29:18,218 INFO  [Run] Start on-enter action [o1.z12] execution
04:29:18,218 INFO  [Run] Finish on-enter action [o1.z12] execution
04:29:18,218 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:18,218 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа]
04:29:18,687 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа]
04:29:18,687 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:18,687 DEBUG [Run] Try transition [2.Ввод номера#2.Ввод номера#e30#true]
04:29:18,687 INFO  [Run] Transition to go found [2.Ввод номера#2.Ввод номера#e30#true]
04:29:18,687 INFO  [Run] Start output action [o2.z23] execution
04:29:18,687 INFO  [Run] Finish output action [o2.z23] execution
```

```
04:29:18,687 INFO  [Run] Start on-enter action [o1.z12] execution
04:29:18,687 INFO  [Run] Finish on-enter action [o1.z12] execution
04:29:18,687 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:18,687 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа]
04:29:19,343 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа]
04:29:19,343 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:19,343 DEBUG [Run] Try transition [2.Ввод номера#2.Ввод номера#e30#true]
04:29:19,343 INFO  [Run] Transition to go found [2.Ввод номера#2.Ввод номера#e30#true]
04:29:19,359 INFO  [Run] Start output action [o2.z23] execution
04:29:19,359 INFO  [Run] Finish output action [o2.z23] execution
04:29:19,359 INFO  [Run] Start on-enter action [o1.z12] execution
04:29:19,359 INFO  [Run] Finish on-enter action [o1.z12] execution
04:29:19,359 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:19,359 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа]
04:29:20,046 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа]
04:29:20,046 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:20,046 DEBUG [Run] Try transition [2.Ввод номера#2.Ввод номера#e30#true]
04:29:20,046 INFO  [Run] Transition to go found [2.Ввод номера#2.Ввод номера#e30#true]
04:29:20,046 INFO  [Run] Start output action [o2.z23] execution
04:29:20,046 INFO  [Run] Finish output action [o2.z23] execution
04:29:20,046 INFO  [Run] Start on-enter action [o1.z12] execution
04:29:20,046 INFO  [Run] Finish on-enter action [o1.z12] execution
04:29:20,046 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:20,046 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа]
04:29:20,234 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа]
04:29:20,234 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:20,234 DEBUG [Run] Try transition [2.Ввод номера#2.Ввод номера#e30#true]
04:29:20,234 INFO  [Run] Transition to go found [2.Ввод номера#2.Ввод номера#e30#true]
04:29:20,234 INFO  [Run] Start output action [o2.z23] execution
04:29:20,234 INFO  [Run] Finish output action [o2.z23] execution
04:29:20,234 INFO  [Run] Start on-enter action [o1.z12] execution
04:29:20,234 INFO  [Run] Finish on-enter action [o1.z12] execution
04:29:20,234 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:20,234 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа]
04:29:20,937 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа]
04:29:20,953 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:20,953 DEBUG [Run] Try transition [2.Ввод номера#2.Ввод номера#e30#true]
04:29:20,953 INFO  [Run] Transition to go found [2.Ввод номера#2.Ввод номера#e30#true]
04:29:20,953 INFO  [Run] Start output action [o2.z23] execution
04:29:20,953 INFO  [Run] Finish output action [o2.z23] execution
04:29:20,953 INFO  [Run] Start on-enter action [o1.z12] execution
04:29:20,953 INFO  [Run] Finish on-enter action [o1.z12] execution
04:29:20,953 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:20,953 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа]
04:29:21,218 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа]
04:29:21,218 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:21,218 DEBUG [Run] Try transition [2.Ввод номера#2.Ввод номера#e30#true]
04:29:21,218 INFO  [Run] Transition to go found [2.Ввод номера#2.Ввод номера#e30#true]
04:29:21,218 INFO  [Run] Start output action [o2.z23] execution
04:29:21,218 INFO  [Run] Finish output action [o2.z23] execution
04:29:21,218 INFO  [Run] Start on-enter action [o1.z12] execution
04:29:21,218 INFO  [Run] Finish on-enter action [o1.z12] execution
04:29:21,218 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:21,218 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа]
04:29:21,562 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа]
04:29:21,562 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:21,562 DEBUG [Run] Try transition [2.Ввод номера#2.Ввод номера#e30#true]
04:29:21,562 INFO  [Run] Transition to go found [2.Ввод номера#2.Ввод номера#e30#true]
04:29:21,562 INFO  [Run] Start output action [o2.z23] execution
```

```
04:29:21,562 INFO  [Run] Finish output action [o2.z23] execution
04:29:21,562 INFO  [Run] Start on-enter action [o1.z12] execution
04:29:21,562 INFO  [Run] Finish on-enter action [o1.z12] execution
04:29:21,562 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:21,562 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа]
04:29:21,796 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа]
04:29:21,796 INFO  [Run] Start event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:21,796 DEBUG [Run] Try transition [2.Ввод номера#2.Ввод номера#e30#true]
04:29:21,796 INFO  [Run] Transition to go found [2.Ввод номера#2.Ввод номера#e30#true]
04:29:21,796 INFO  [Run] Start output action [o2.z23] execution
04:29:21,796 INFO  [Run] Finish output action [o2.z23] execution
04:29:21,796 INFO  [Run] Start on-enter action [o1.z12] execution
04:29:21,796 INFO  [Run] Finish on-enter action [o1.z12] execution
04:29:21,796 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:21,796 INFO  [Run] Finish event [e30] processing. In state [/A3:1.Работа]
04:29:22,484 INFO  [Run] Start event [e21] processing. In state [/A3:1.Работа]
04:29:22,484 INFO  [Run] Start event [e21] processing. In state [/A3:1.Работа/A1:2.Ввод
номера]
04:29:22,484 DEBUG [Run] Try transition [2.Ввод номера#2.Ввод номера#e21#!o2.x1]
04:29:22,484 INFO  [Run] Start input action [o2.x1] calculation
04:29:22,484 INFO  [Run] Finish input action [o2.x1] calculation. Its value is [true]
04:29:22,484 DEBUG [Run] Try transition [2.Ввод номера#3. Прием денег#e21#o2.x1]
04:29:22,484 INFO  [Run] Transition to go found [2.Ввод номера#3. Прием денег#e21#o2.x1]
04:29:22,484 INFO  [Run] Start output action [o3.z32] execution
04:29:22,500 INFO  [Run] Finish output action [o3.z32] execution
04:29:22,500 INFO  [Run] Start on-enter action [o1.z12] execution
04:29:22,500 INFO  [Run] Finish on-enter action [o1.z12] execution
04:29:22,500 INFO  [Run] Start event [e21] processing. In state [/A3:1.Работа/A1:3. Прием
денег/A2:Top]
04:29:22,500 INFO  [Run] Transition to go found [s4#1.Прием купюры##true]
04:29:22,500 INFO  [Run] Finish event [e21] processing. In state [/A3:1.Работа/A1:3.
Прием денег/A2:1.Прием купюры]
04:29:22,500 INFO  [Run] Finish event [e21] processing. In state [/A3:1.Работа/A1:3.
Прием денег]
04:29:22,500 INFO  [Run] Finish event [e21] processing. In state [/A3:1.Работа]
04:29:24,359 INFO  [Run] Start event [e11] processing. In state [/A3:1.Работа]
04:29:24,359 INFO  [Run] Start event [e11] processing. In state [/A3:1.Работа/A1:3. Прием
денег]
04:29:24,359 INFO  [Run] Start event [e11] processing. In state [/A3:1.Работа/A1:3. Прием
денег/A2:1.Прием купюры]
04:29:24,359 DEBUG [Run] Try transition [1.Прием купюры#1.Прием купюры#e11#true]
04:29:24,359 INFO  [Run] Transition to go found [1.Прием купюры#1.Прием купюры#e11#true]
04:29:24,359 INFO  [Run] Start output action [o2.z22] execution
04:29:24,359 INFO  [Run] Finish output action [o2.z22] execution
04:29:24,359 INFO  [Run] Start output action [o1.z12] execution
04:29:24,359 INFO  [Run] Finish output action [o1.z12] execution
04:29:24,359 INFO  [Run] Finish event [e11] processing. In state [/A3:1.Работа/A1:3.
Прием денег/A2:1.Прием купюры]
04:29:24,359 INFO  [Run] Finish event [e11] processing. In state [/A3:1.Работа/A1:3.
Прием денег]
04:29:24,359 INFO  [Run] Finish event [e11] processing. In state [/A3:1.Работа]
04:29:24,734 INFO  [Run] Start event [e11] processing. In state [/A3:1.Работа]
04:29:24,734 INFO  [Run] Start event [e11] processing. In state [/A3:1.Работа/A1:3. Прием
денег]
04:29:24,734 INFO  [Run] Start event [e11] processing. In state [/A3:1.Работа/A1:3. Прием
денег/A2:1.Прием купюры]
04:29:24,734 DEBUG [Run] Try transition [1.Прием купюры#1.Прием купюры#e11#true]
04:29:24,734 INFO  [Run] Transition to go found [1.Прием купюры#1.Прием купюры#e11#true]
04:29:24,734 INFO  [Run] Start output action [o2.z22] execution
04:29:24,734 INFO  [Run] Finish output action [o2.z22] execution
04:29:24,734 INFO  [Run] Start output action [o1.z12] execution
04:29:24,734 INFO  [Run] Finish output action [o1.z12] execution
04:29:24,734 INFO  [Run] Finish event [e11] processing. In state [/A3:1.Работа/A1:3.
Прием денег/A2:1.Прием купюры]
04:29:24,734 INFO  [Run] Finish event [e11] processing. In state [/A3:1.Работа/A1:3.
Прием денег]
04:29:24,734 INFO  [Run] Finish event [e11] processing. In state [/A3:1.Работа]
04:29:25,687 INFO  [Run] Start event [e21] processing. In state [/A3:1.Работа]
```

```
04:29:25,687 INFO  [Run] Start event [e21] processing. In state [/A3:1.Работа/A1:3. Прием
денег]
04:29:25,687 DEBUG [Run] Try transition [3. Прием денег#3. Прием денег#e21#!o3.x31]
04:29:25,687 INFO  [Run] Start input action [o3.x31] calculation
04:29:25,687 INFO  [Run] Finish input action [o3.x31] calculation. Its value is [true]
04:29:25,687 DEBUG [Run] Try transition [3. Прием денег#4.Подтверждение#e21#o3.x31]
04:29:25,687 INFO  [Run] Transition to go found [3. Прием
денег#4.Подтверждение#e21#o3.x31]
04:29:25,687 INFO  [Run] Start output action [o3.z31] execution
04:29:25,687 INFO  [Run] Finish output action [o3.z31] execution
04:29:25,687 INFO  [Run] Start on-enter action [o1.z13] execution
04:29:25,703 INFO  [Run] Finish on-enter action [o1.z13] execution
04:29:25,703 INFO  [Run] Finish event [e21] processing. In state
[/A3:1.Работа/A1:4.Подтверждение]
04:29:25,703 INFO  [Run] Finish event [e21] processing. In state [/A3:1.Работа]
04:29:26,546 INFO  [Run] Start event [e23] processing. In state [/A3:1.Работа]
04:29:26,546 DEBUG [Run] Try transition [1.Работа#2.Справка#e23#true]
04:29:26,546 INFO  [Run] Transition to go found [1.Работа#2.Справка#e23#true]
04:29:26,546 INFO  [Run] Start output action [o1.z15] execution
04:29:26,546 INFO  [Run] Finish output action [o1.z15] execution
04:29:26,546 INFO  [Run] Finish event [e23] processing. In state [/A3:2.Справка]
04:29:26,921 INFO  [Run] Start event [e23] processing. In state [/A3:2.Справка]
04:29:26,921 DEBUG [Run] Try transition [2.Справка#1.Работа#e23#true]
04:29:26,921 INFO  [Run] Transition to go found [2.Справка#1.Работа#e23#true]
04:29:26,921 INFO  [Run] Start output action [o1.z16] execution
04:29:26,921 INFO  [Run] Finish output action [o1.z16] execution
04:29:26,921 INFO  [Run] Start event [e23] processing. In state
[/A3:1.Работа/A1:4.Подтверждение]
04:29:26,921 INFO  [Run] Finish event [e23] processing. In state
[/A3:1.Работа/A1:4.Подтверждение]
04:29:26,921 INFO  [Run] Finish event [e23] processing. In state [/A3:1.Работа]
04:29:27,765 INFO  [Run] Start event [e21] processing. In state [/A3:1.Работа]
04:29:27,765 INFO  [Run] Start event [e21] processing. In state
[/A3:1.Работа/A1:4.Подтверждение]
04:29:27,765 DEBUG [Run] Try transition [4.Подтверждение#5.Уведомление#e21#true]
04:29:27,765 INFO  [Run] Transition to go found [4.Подтверждение#5.Уведомление#e21#true]
04:29:27,765 INFO  [Run] Start on-enter action [o1.z14] execution
04:29:27,765 INFO  [Run] Finish on-enter action [o1.z14] execution
04:29:27,765 INFO  [Run] Finish event [e21] processing. In state
[/A3:1.Работа/A1:5.Уведомление]
04:29:27,765 INFO  [Run] Finish event [e21] processing. In state [/A3:1.Работа]
04:29:28,687 INFO  [Run] Start event [e21] processing. In state [/A3:1.Работа]
04:29:28,687 INFO  [Run] Start event [e21] processing. In state
[/A3:1.Работа/A1:5.Уведомление]
04:29:28,687 DEBUG [Run] Try transition [5.Уведомление#1.Приветствие#e21#true]
04:29:28,687 INFO  [Run] Transition to go found [5.Уведомление#1.Приветствие#e21#true]
04:29:28,687 INFO  [Run] Start output action [o2.z26] execution
04:29:28,687 INFO  [Run] Finish output action [o2.z26] execution
04:29:28,687 INFO  [Run] Start on-enter action [o1.z11] execution
04:29:28,687 INFO  [Run] Finish on-enter action [o1.z11] execution
04:29:28,687 INFO  [Run] Finish event [e21] processing. In state
[/A3:1.Работа/A1:1.Приветствие]
04:29:28,687 INFO  [Run] Finish event [e21] processing. In state [/A3:1.Работа]
```

Приложение 2. Сгенерированное XML-описание

Telepay.unimod

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE model PUBLIC "-//eVelopers Corp.//DTD State machine graphical model V1.0//EN" "http://www.evelopers.com/dtd/unimod/gmodel.dtd">
<model version="1.0">
 <property name="name" value="Model1"/>
 <elements>
  <node id="id0" type="com.evelopers.unimod.plugin.eclipse.model.GControlledObjectHandler">
   <property name="name" value="o1"/>
   <property name="implName" value="ru.ifmo.telepay.UserScreen"/>
  </node>
  <node id="id2" type="com.evelopers.unimod.plugin.eclipse.model.GControlledObjectHandler">
   <property name="name" value="o3"/>
   <property name="implName" value="ru.ifmo.telepay.MoneyReceptor"/>
  </node>
  <node id="id1" type="com.evelopers.unimod.plugin.eclipse.model.GControlledObjectHandler">
   <property name="name" value="o2"/>
   <property name="implName" value="ru.ifmo.telepay.Memory"/>
  </node>
  <node id="id3" type="com.evelopers.unimod.plugin.eclipse.model.GEventProviderHandler">
   <property name="name" value="p1"/>
   <property name="implName" value="ru.ifmo.telepay.MoneyReceptor"/>
  </node>
  <node id="id5" type="com.evelopers.unimod.plugin.eclipse.model.GEventProviderHandler">
   <property name="name" value="p3"/>
   <property name="implName" value="ru.ifmo.telepay.KeyPad"/>
  </node>
  <node id="id4" type="com.evelopers.unimod.plugin.eclipse.model.GEventProviderHandler">
   <property name="name" value="p2"/>
   <property name="implName" value="ru.ifmo.telepay.UserTerminal"/>
  </node>
  <node id="id19" type="com.evelopers.unimod.plugin.eclipse.model.GStateMachine">
   <property name="name" value="A3"/>
   <property name="configManagerClassName"
value="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
   <composited role="top">
    <node id="id20" type="com.evelopers.unimod.plugin.eclipse.model.GTopState">
     <property name="name" value="Top"/>
     <composited role="substates[0]">
      <node id="id22" type="com.evelopers.unimod.plugin.eclipse.model.GInitialState">
       <property name="name" value="s1"/>
      </node>
     </composited>
     <composited role="substates[1]">
      <node id="id23" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
       <property name="name" value="1.Работа"/>
       <property name="actions" value=""/>
       <property name="subMachineHandles" value="A1"/>
      </node>
     </composited>
     <composited role="substates[2]">
      <node id="id21" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
       <property name="name" value="2.Справка"/>
       <property name="actions" value=""/>
       <property name="subMachineHandles" value=""/>
      </node>
     </composited>
    </node>
   </composited>
```

```
    </node>
    <node id="id14" type="com.evelopers.unimod.plugin.eclipse.model.GStateMachine">
     <property name="name" value="A2"/>
     <property name="configManagerClassName"
value="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
     <composited role="top">
      <node id="id15" type="com.evelopers.unimod.plugin.eclipse.model.GTopState">
       <property name="name" value="Top"/>
       <composited role="substates[0]">
        <node id="id17" type="com.evelopers.unimod.plugin.eclipse.model.GInitialState">
         <property name="name" value="s4"/>
        </node>
       </composited>
       <composited role="substates[1]">
        <node id="id18" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
         <property name="name" value="1.Прием купюры"/>
         <property name="actions" value=""/>
         <property name="subMachineHandles" value=""/>
        </node>
       </composited>
       <composited role="substates[2]">
        <node id="id16" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
         <property name="name" value="2.Возврат купюры"/>
         <property name="actions" value=""/>
         <property name="subMachineHandles" value=""/>
        </node>
       </composited>
      </node>
     </composited>
    </node>
    <node id="id6" type="com.evelopers.unimod.plugin.eclipse.model.GStateMachine">
     <property name="name" value="A1"/>
     <property name="configManagerClassName"
value="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
     <composited role="top">
      <node id="id7" type="com.evelopers.unimod.plugin.eclipse.model.GTopState">
       <property name="name" value="Top"/>
       <composited role="substates[0]">
        <node id="id10" type="com.evelopers.unimod.plugin.eclipse.model.GInitialState">
         <property name="name" value="s1"/>
        </node>
       </composited>
       <composited role="substates[1]">
        <node id="id9" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
         <property name="name" value="1.Приветствие"/>
         <property name="actions" value="o1.z11"/>
         <property name="subMachineHandles" value=""/>
        </node>
       </composited>
       <composited role="substates[2]">
        <node id="id11" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
         <property name="name" value="2.Ввод номера"/>
         <property name="actions" value="o1.z12"/>
         <property name="subMachineHandles" value=""/>
        </node>
       </composited>
       <composited role="substates[3]">
        <node id="id8" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
         <property name="name" value="3. Прием денег"/>
         <property name="actions" value="o1.z12"/>
         <property name="subMachineHandles" value="A2"/>
        </node>
       </composited>
```

```
        <composited role="substates[4]">
          <node id="id12" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
            <property name="name" value="5.Уведомление"/>
            <property name="actions" value="o1.z14"/>
            <property name="subMachineHandles" value=""/>
          </node>
        </composited>
        <composited role="substates[5]">
          <node id="id13" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
            <property name="name" value="4.Подтверждение"/>
            <property name="actions" value="o1.z13"/>
            <property name="subMachineHandles" value=""/>
          </node>
        </composited>
      </node>
    </composited>
  </node>
  <edge client="id3" id="euid1160567883343-1819818" supplier="id19"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
    <property name="name" value=""/>
    <property name="supplierRole" value=""/>
  </edge>
  <edge client="id5" id="euid1160567883343-20479628" supplier="id19"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
    <property name="name" value=""/>
    <property name="supplierRole" value=""/>
  </edge>
  <edge client="id4" id="euid1160567883343-4576850" supplier="id19"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
    <property name="name" value=""/>
    <property name="supplierRole" value=""/>
  </edge>
  <edge client="id19" id="id29" supplier="id0"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
    <property name="name" value=""/>
    <property name="supplierRole" value="o1"/>
  </edge>
  <edge client="id19" id="id34" supplier="id1"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
    <property name="name" value=""/>
    <property name="supplierRole" value="o2"/>
  </edge>
  <edge client="id22" id="id35" supplier="id23"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value=""/>
  </edge>
  <edge client="id23" id="id49" supplier="id21"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e23 /o1.z15"/>
  </edge>
  <edge client="id21" id="id31" supplier="id23"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e23 /o1.z16"/>
  </edge>
  <edge client="id14" id="id47" supplier="id0"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
    <property name="name" value=""/>
    <property name="supplierRole" value="o1"/>
  </edge>
  <edge client="id14" id="id30" supplier="id2"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
    <property name="name" value=""/>
    <property name="supplierRole" value="o3"/>
```

```
    </edge>
    <edge client="id14" id="id24" supplier="id1"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
      <property name="name" value=""/>
      <property name="supplierRole" value="o2"/>
    </edge>
    <edge client="id17" id="id28" supplier="id18"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
      <property name="labelText" value=""/>
    </edge>
    <edge client="id18" id="id40" supplier="id18"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
      <property name="labelText" value="e11 /o2.z22, o1.z12"/>
    </edge>
    <edge client="id18" id="id26" supplier="id16"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
      <property name="labelText" value="e12 /o3.z31"/>
    </edge>
    <edge client="id16" id="id38" supplier="id18"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
      <property name="labelText" value="e13 /o3.z32"/>
    </edge>
    <edge client="id6" id="id46" supplier="id0"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
      <property name="name" value=""/>
      <property name="supplierRole" value="o1"/>
    </edge>
    <edge client="id6" id="id36" supplier="id2"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
      <property name="name" value=""/>
      <property name="supplierRole" value="o3"/>
    </edge>
    <edge client="id6" id="id32" supplier="id1"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
      <property name="name" value=""/>
      <property name="supplierRole" value="o2"/>
    </edge>
    <edge client="id10" id="id25" supplier="id9"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
      <property name="labelText" value=""/>
    </edge>
    <edge client="id9" id="id45" supplier="id11"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
      <property name="labelText" value="e21"/>
    </edge>
    <edge client="id11" id="id50" supplier="id9"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
      <property name="labelText" value="e22 /o2.z24"/>
    </edge>
    <edge client="id11" id="id42" supplier="id11"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
      <property name="labelText" value="e31 /o2.z25"/>
    </edge>
    <edge client="id11" id="id51" supplier="id11"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
      <property name="labelText" value="e21[!o2.x1]"/>
    </edge>
    <edge client="id11" id="id41" supplier="id11"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
      <property name="labelText" value="e30 /o2.z23"/>
    </edge>
    <edge client="id11" id="id33" supplier="id8"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
```

```
    <property name="labelText" value="e21[o2.x1] /o3.z32"/>
   </edge>
   <edge client="id8" id="id44" supplier="id11"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e22 /o3.z31"/>
   </edge>
   <edge client="id8" id="id39" supplier="id8"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e21[!o3.x31]"/>
   </edge>
   <edge client="id8" id="id43" supplier="id13"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e21[o3.x31] /o3.z31"/>
   </edge>
   <edge client="id12" id="id48" supplier="id9"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e21 /o2.z26"/>
   </edge>
   <edge client="id13" id="id37" supplier="id8"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e22 /o3.z32"/>
   </edge>
   <edge client="id13" id="id27" supplier="id12"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e21"/>
   </edge>
  </elements>
  <diagrams>
   <diagram>
    <node-ref node="id0">
     <property name="location" value="(560, 10)"/>
     <property name="size" value="(181, 191)"/>
    </node-ref>
    <node-ref node="id2">
     <property name="location" value="(560, 231)"/>
     <property name="size" value="(181, 140)"/>
    </node-ref>
    <node-ref node="id1">
     <property name="location" value="(560, 413)"/>
     <property name="size" value="(201, 228)"/>
    </node-ref>
    <node-ref node="id3">
     <property name="location" value="(25, 20)"/>
     <property name="size" value="(236, 181)"/>
    </node-ref>
    <node-ref node="id5">
     <property name="location" value="(20, 221)"/>
     <property name="size" value="(241, 160)"/>
    </node-ref>
    <node-ref node="id4">
     <property name="location" value="(25, 420)"/>
     <property name="size" value="(236, 161)"/>
    </node-ref>
    <node-ref node="id19">
     <property name="location" value="(280, 20)"/>
     <property name="size" value="(121, 561)"/>
    </node-ref>
    <node-ref node="id14">
     <property name="location" value="(420, 180)"/>
     <property name="size" value="(121, 281)"/>
    </node-ref>
    <node-ref node="id6">
     <property name="location" value="(780, 180)"/>
```

```
    <property name="size" value="(130, 252)"/>
   </node-ref>
   <edge-ref edge="euid1160567883343-1819818">
    <property name="labelPosition" value="(100%) -&gt; (-10, -10)"/>
   </edge-ref>
   <edge-ref edge="euid1160567883343-20479628">
    <property name="labelPosition" value="(100%) -&gt; (-10, -10)"/>
   </edge-ref>
   <edge-ref edge="euid1160567883343-4576850">
    <property name="labelPosition" value="(100%) -&gt; (-10, -10)"/>
   </edge-ref>
   <edge-ref edge="id29">
    <property name="labelPosition" value="(100%) -&gt; (-10, -10)"/>
   </edge-ref>
   <edge-ref edge="id34">
    <property name="labelPosition" value="(100%) -&gt; (-10, -10)"/>
   </edge-ref>
   <edge-ref edge="id47">
    <property name="labelPosition" value="(100%) -&gt; (-10, -10)"/>
   </edge-ref>
   <edge-ref edge="id30">
    <property name="labelPosition" value="(100%) -&gt; (-10, -10)"/>
   </edge-ref>
   <edge-ref edge="id24">
    <property name="labelPosition" value="(100%) -&gt; (-10, -10)"/>
   </edge-ref>
   <edge-ref edge="id46">
    <property name="labelPosition" value="(100%) -&gt; (19, -20)"/>
   </edge-ref>
   <edge-ref edge="id36">
    <property name="labelPosition" value="(100%) -&gt; (19, -21)"/>
   </edge-ref>
   <edge-ref edge="id32">
    <property name="labelPosition" value="(100%) -&gt; (-1, -22)"/>
   </edge-ref>
  </diagram>
  <diagram>
   <node-ref node="id20"/>
   <node-ref node="id22">
    <property name="location" value="(280, 140)"/>
   </node-ref>
   <node-ref node="id23">
    <property name="location" value="(200, 260)"/>
    <property name="size" value="(181, 101)"/>
   </node-ref>
   <node-ref node="id21">
    <property name="location" value="(460, 260)"/>
    <property name="size" value="(181, 101)"/>
   </node-ref>
   <edge-ref edge="id35">
    <property name="sourceAnchor" value=""/>
    <property name="targetAnchor" value="NORTH:5"/>
    <property name="bendpoints" value=""/>
    <property name="labelPosition" value="(50%) -&gt; (20, -10)"/>
   </edge-ref>
   <edge-ref edge="id49">
    <property name="sourceAnchor" value="EAST:2"/>
    <property name="targetAnchor" value="WEST:2"/>
    <property name="bendpoints" value=""/>
    <property name="labelPosition" value="(50%) -&gt; (0, -20)"/>
   </edge-ref>
   <edge-ref edge="id31">
    <property name="sourceAnchor" value="WEST:4"/>
```

```
    <property name="targetAnchor" value="EAST:4"/>
    <property name="bendpoints" value=""/>
    <property name="labelPosition" value="(50%) -&amp;gt; (0, 20)"/>
   </edge-ref>
  </diagram>
  <diagram>
   <node-ref node="id15"/>
   <node-ref node="id17">
    <property name="location" value="(280, 160)"/>
   </node-ref>
   <node-ref node="id18">
    <property name="location" value="(200, 240)"/>
    <property name="size" value="(161, 140)"/>
   </node-ref>
   <node-ref node="id16">
    <property name="location" value="(460, 240)"/>
    <property name="size" value="(161, 141)"/>
   </node-ref>
   <edge-ref edge="id28">
    <property name="sourceAnchor" value=""/>
    <property name="targetAnchor" value="NORTH:5"/>
    <property name="bendpoints" value=""/>
    <property name="labelPosition" value="(50%) -&amp;gt; (0, -20)"/>
   </edge-ref>
   <edge-ref edge="id40">
    <property name="sourceAnchor" value="SOUTH:2"/>
    <property name="targetAnchor" value="WEST:7"/>
    <property name="bendpoints" value=""/>
    <property name="labelPosition" value="(50%) -&amp;gt; (-35, 45)"/>
   </edge-ref>
   <edge-ref edge="id26">
    <property name="sourceAnchor" value="EAST:2"/>
    <property name="targetAnchor" value="WEST:2"/>
    <property name="bendpoints" value=""/>
    <property name="labelPosition" value="(50%) -&amp;gt; (0, -20)"/>
   </edge-ref>
   <edge-ref edge="id38">
    <property name="sourceAnchor" value="WEST:5"/>
    <property name="targetAnchor" value="EAST:5"/>
    <property name="bendpoints" value=""/>
    <property name="labelPosition" value="(50%) -&amp;gt; (0, -20)"/>
   </edge-ref>
  </diagram>
  <diagram>
   <node-ref node="id7"/>
   <node-ref node="id10">
    <property name="location" value="(101, 60)"/>
   </node-ref>
   <node-ref node="id9">
    <property name="location" value="(40, 180)"/>
    <property name="size" value="(141, 101)"/>
   </node-ref>
   <node-ref node="id11">
    <property name="location" value="(280, 180)"/>
    <property name="size" value="(161, 101)"/>
   </node-ref>
   <node-ref node="id8">
    <property name="location" value="(580, 180)"/>
    <property name="size" value="(161, 101)"/>
   </node-ref>
   <node-ref node="id12">
    <property name="location" value="(40, 380)"/>
    <property name="size" value="(141, 101)"/>
```

```
</node-ref>
<node-ref node="id13">
  <property name="location" value="(580, 380)"/>
  <property name="size" value="(161, 101)"/>
</node-ref>
<edge-ref edge="id25">
  <property name="sourceAnchor" value=""/>
  <property name="targetAnchor" value="NORTH:4"/>
  <property name="bendpoints" value=""/>
  <property name="labelPosition" value="(50%) -&gt; (26, -20)"/>
</edge-ref>
<edge-ref edge="id45">
  <property name="sourceAnchor" value="EAST:2"/>
  <property name="targetAnchor" value="WEST:2"/>
  <property name="bendpoints" value=""/>
  <property name="labelPosition" value="(50%) -&gt; (0, -20)"/>
</edge-ref>
<edge-ref edge="id50">
  <property name="sourceAnchor" value="WEST:3"/>
  <property name="targetAnchor" value="EAST:3"/>
  <property name="bendpoints" value=""/>
  <property name="labelPosition" value="(50%) -&gt; (1, 20)"/>
</edge-ref>
<edge-ref edge="id42">
  <property name="sourceAnchor" value="EAST:5"/>
  <property name="targetAnchor" value="SOUTH:7"/>
  <property name="bendpoints" value=""/>
  <property name="labelPosition" value="(50%) -&gt; (35, 45)"/>
</edge-ref>
<edge-ref edge="id51">
  <property name="sourceAnchor" value="WEST:1"/>
  <property name="targetAnchor" value="NORTH:2"/>
  <property name="bendpoints" value=""/>
  <property name="labelPosition" value="(50%) -&gt; (-34, -55)"/>
</edge-ref>
<edge-ref edge="id41">
  <property name="sourceAnchor" value="SOUTH:2"/>
  <property name="targetAnchor" value="WEST:5"/>
  <property name="bendpoints" value=""/>
  <property name="labelPosition" value="(50%) -&gt; (-65, 45)"/>
</edge-ref>
<edge-ref edge="id33">
  <property name="sourceAnchor" value="EAST:3"/>
  <property name="targetAnchor" value="WEST:3"/>
  <property name="bendpoints" value=""/>
  <property name="labelPosition" value="(50%) -&gt; (0, -20)"/>
</edge-ref>
<edge-ref edge="id44">
  <property name="sourceAnchor" value="WEST:4"/>
  <property name="targetAnchor" value="EAST:4"/>
  <property name="bendpoints" value=""/>
  <property name="labelPosition" value="(50%) -&gt; (1, 10)"/>
</edge-ref>
<edge-ref edge="id39">
  <property name="sourceAnchor" value="NORTH:7"/>
  <property name="targetAnchor" value="EAST:1"/>
  <property name="bendpoints" value=""/>
  <property name="labelPosition" value="(50%) -&gt; (55, -55)"/>
</edge-ref>
<edge-ref edge="id43">
  <property name="sourceAnchor" value="SOUTH:3"/>
  <property name="targetAnchor" value="NORTH:3"/>
  <property name="bendpoints" value=""/>
```

```
      <property name="labelPosition" value="(50%) -&gt; (-60, 0)"/>
    </edge-ref>
    <edge-ref edge="id48">
      <property name="sourceAnchor" value="NORTH:4"/>
      <property name="targetAnchor" value="SOUTH:4"/>
      <property name="bendpoints" value=""/>
      <property name="labelPosition" value="(50%) -&gt; (50, 0)"/>
    </edge-ref>
    <edge-ref edge="id37">
      <property name="sourceAnchor" value="NORTH:4"/>
      <property name="targetAnchor" value="SOUTH:4"/>
      <property name="bendpoints" value=""/>
      <property name="labelPosition" value="(50%) -&gt; (40, 0)"/>
    </edge-ref>
    <edge-ref edge="id27">
      <property name="sourceAnchor" value="WEST:3"/>
      <property name="targetAnchor" value="EAST:3"/>
      <property name="bendpoints" value=""/>
      <property name="labelPosition" value="(50%) -&gt; (0, -20)"/>
    </edge-ref>
  </diagram>
 </diagrams>
</model>
```

Приложение 3. Исходные коды программы

KeyPad.java

```
package ru.ifmo.telepay;

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

import java.awt.event.*;

/**
 * This class implements the EventProvider of keypad abstraction used in the model.
 */
public class KeyPad implements EventProvider {
   /**
    * Model engine.
    */
   private ModelEngine engine;

   /**
    * @unimod.event.descr Нажата кнопка с цифрой.
    */
   public static final String E30 = "e30";

   /**
    * @unimod.event.descr Нажата клавиша "Backspace".
    */
   public static final String E31 = "e31";

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

      KeyPadPanel keyPadPanel = TelepayScreen.getInstance().getKeyPadPanel();

      for (int i = 0; i < KeyPadPanel.BUTTONS_COUNT; i++) {
         final int dig = i;
         keyPadPanel.addButtonListener(dig, new ActionListener() {
            public void actionPerformed(ActionEvent event) {
               Event fire = new Event(E30);
               StateMachineContext context = StateMachineContextImpl.create();
               Parameter digit = new Parameter(Memory.DIGIT, new Integer(dig));
               digit.addToContext(context.getEventContext());
               KeyPad.this.engine.getEventManager().handle(fire, context);
            }
         });
      }

      keyPadPanel.addBackspaceListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            KeyPad.this.engine.getEventManager().handle(new Event(E31),
                  StateMachineContextImpl.create());
         }
      });
   }

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

```
        TelepayScreen.getInstance().dispose();
    }
}
```

KeyPadPanel.java

```
package ru.ifmo.telepay;

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

/**
 * This class implements user interface of KeyPad abstraction used in system model.
 */
public class KeyPadPanel extends JPanel {
    static final long serialVersionUID = 0;

    /**
     * Number of buttons.
     */
    public static final int BUTTONS_COUNT = 10;

    /**
     * Buttons of key pad.
     */
    private JButton[] buttons;

    /**
     * Backspace button of key pad.
     */
    private JButton backspace;

    /**
     * Creates new KeyPadPanel.
     */
    public KeyPadPanel() {
        this.setLayout(null);
        this.setSize(150, 200);

        buttons = new JButton[10];

        for (int i = 0; i < 10; i++) {
            buttons[i] = new JButton("" + i);
            this.add(buttons[i]);
        }

        final int btnWidth = this.getWidth() / 3;
        final int btnHeight = this.getHeight() / 4;

        buttons[0].setBounds(new Rectangle(this.getBounds().x, this.getBounds().y + 3 * btnHeight,
                2 * btnWidth, btnHeight));

        for (int i = 1; i < 10; i++) {
            buttons[i].setBounds(this.getBounds().x + btnWidth * ((i - 1) % 3),
                    this.getBounds().y + btnHeight * ((i - 1) / 3),
                    btnWidth, btnHeight);
        }

        backspace = new JButton("\u2190");
        this.add(backspace);
        backspace.setBounds(new Rectangle(this.getBounds().x + 2 * btnWidth,
                this.getBounds().y + 3 * btnHeight,
                btnWidth, btnHeight));
```

```
    }

    /**
     * Adds listener to button representing digit.
     *
     * @param digit Listener will be added to button containing digit.
     *
     * @param listener Listener to be added.
     */
    public void addButtonListener(int digit, ActionListener listener) {
        buttons[digit].addActionListener(listener);
    }

    /**
     * Adds listener to backspace button.
     *
     * @param listener Listener to be added.
     */
    public void addBackspaceListener(ActionListener listener) {
        backspace.addActionListener(listener);
    }

}
```

Memory.java

```
package ru.ifmo.telepay;

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

/**
 * Implements memory controlled object abstraction.
 */
public class Memory implements ControlledObject {
    /**
     * Name for money receiver parameter for events.
     */
    public static final String MONEY = "MONEY";

    /**
     * Name for pressed digit paremeter for events.
     */
    public static final String DIGIT = "DIGIT";

    /**
     * Money received from user.
     */
    private static int money;

    /**
     * Number entered by user.
     */
    private static String number = "";

    /**
     * Returns if the number entered is valid.
     * In this model number is considered valid if consists of 11 digits.
     *
     * @return Returns true if provided number is valid.
     */
    public static boolean isNumberValid() {
        return (number.length() == 11);
```

```
}

/**
 * Returns number entered by user.
 *
 * @return Number entered by user.
 */
public static String getNumber() {
   return number;
}

/**
 * Returns amount of money on user's account.
 *
 * @return Amount of money on user's account.
 */
public static int getMoney() {
   return money;
}

/**
 * @unimod.action.descr Добавить номинал к счету пользователя.
 */
public void z22(StateMachineContext context) {
   /*TODO: automatically generated by UniMod method*/
   money += ((Integer)context.getEventContext().getParameter(Memory.MONEY)).intValue();
}

/**
 * @unimod.action.descr Добавить цифру к номеру.
 */
public void z23(StateMachineContext context) {
   number += context.getEventContext().getParameter(Memory.DIGIT).toString();
}

/**
 * @unimod.action.descr Очистить номер.
 */
public void z24(StateMachineContext context) {
   number = "";
}

/**
 * @unimod.action.descr Удалить последнюю цифру.
 */
public void z25(StateMachineContext context) {
   if (number.length() != 0) {
      number = number.substring(0, number.length() - 1);
   }
}

/**
 * @unimod.action.descr Верно ли набран номер.
 */
public boolean x1(StateMachineContext context) {
   /*TODO: automatically generated by UniMod method*/
   return (isNumberValid());
}

/**
 * @unimod.action.descr Сбросить информацию.
 */
public void z26(StateMachineContext context) {
```

```
        /*TODO: automatically generated by UniMod method*/
        number = "";
        money = 0;
    }
}
```

MoneyReceiverPanel.java

```
package ru.ifmo.telepay;

import javax.swing.*;
import java.net.*;
import java.util.*;
import java.io.*;

/**
 * Implements UI component representing MoneyReceiver.
 */
public class MoneyReceiverPanel extends JPanel {
    static final long serialVersionUID = 0;

    /**
     * Format to display banknotes in string representation.
     */
    public static final String BANKNOTE_NAME_FORMAT = "%d рублей";

    /**
     * Invalid banknote's string representation.
     */
    public static String INVALID_BANKNOTE = "Неопознаваемая купюра";

    /**
     * Button of receiver.
     */
    private JButton receiverButton;

    /**
     * List of banknotes.
     */
    private JList banknotesList;

    /**
     * Path to search images.
     */
    private static final String IMG_PATH = "/ru/ifmo/telepay/img/";

    /**
     * Filename of "receiver on" image.
     */
    private static final String RECEIVER_IMG_NAME_ON = "receiver_on.jpg";

    /**
     * Filename of "receiver off" image.
     */
    private static final String RECEIVER_IMG_NAME_OFF = "receiver_off.jpg";

    /**
     * Image of receiver turned off.
     */
    private ImageIcon iconOff;

    /**
     * Image of receiver turned on.
```

```
 */
private ImageIcon iconOn;

/**
 * Creates new instance of MoneyReceiverPanel.
 */
public MoneyReceiverPanel() {
    this.setSize(680, 170);
    this.setLayout(null);

    URL url = MoneyReceiverPanel.class.getResource(IMG_PATH + RECEIVER_IMG_NAME_ON);
    iconOn = new ImageIcon(url);
    url = MoneyReceiverPanel.class.getResource(IMG_PATH + RECEIVER_IMG_NAME_OFF);
    iconOff = new ImageIcon(url);

    receiverButton = new JButton();
    this.add(receiverButton);
    receiverButton.setBounds(getBounds().width - iconOn.getIconWidth(), 0,
            iconOn.getIconWidth(), iconOn.getIconHeight());
    setReceiverValid(true);

    ArrayList banknotes = new ArrayList();
    for (int i = 0; i < MoneyReceptor.BANKNOTE_VALUES.length; i++) {
        final int banknoteValue = MoneyReceptor.BANKNOTE_VALUES[i];
        StringWriter writer = new StringWriter();
        PrintWriter printer = new PrintWriter(writer);
        printer.printf(BANKNOTE_NAME_FORMAT, new Object[] {new Integer(banknoteValue)});
        banknotes.add(writer.getBuffer().toString());
    }
    banknotes.add(INVALID_BANKNOTE);

    banknotesList = new JList(banknotes.toArray());
    this.add(banknotesList);
    banknotesList.setBounds(0, 0, receiverButton.getBounds().x, getBounds().height);
    banknotesList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    banknotesList.setSelectedIndex(0);

    setReceiverValid(false);
}

/**
 * Returns receiver button.
 *
 * @return receiver button.
 */
public JButton getReceiverButton() {
    return receiverButton;
}

/**
 * Returns banknotes list component.
 *
 * @return banknotes list component.
 */
public JList getBanknotesList() {
    return banknotesList;
}

/**
 * Sets receiver to valid or invalid state.
 *
 * @param valid If valid is true then receiver is set to valid state.
 */
```

```
    public void setReceiverValid(boolean valid) {
        receiverButton.setIcon(valid ? iconOn : iconOff);
    }

    /**
     * Returns whether receiver is valid.
     *
     * @return Returns true if receiver is valid and false if it is not.
     */
    public boolean isReceiverValid() {
        return receiverButton.getIcon() == iconOn;
    }
}
```

MoneyReceptor.java

```
package ru.ifmo.telepay;

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

import javax.swing.*;
import java.awt.event.*;
import com.evelopers.unimod.runtime.ControlledObject;

/**
 * Implements MoneyReceptor abstraction of model.
 */
public class MoneyReceptor implements EventProvider, ControlledObject {
    /**
     * Values of banknotes.
     */
    public static final int[] BANKNOTE_VALUES = {10, 50, 100, 500, 1000};

    /**
     * Model engine.
     */
    private ModelEngine engine;

    /**
     * @unimod.event.descr Купюра распознана.
     */
    public static final String E11 = "e11";

    /**
     * @unimod.event.descr Купюра неопознана.
     */
    public static final String E12 = "e12";

    /**
     * @unimod.event.descr Купюра возвращена.
     */
    public static final String E13 = "e13";

    public void init(ModelEngine engine) throws CommonException {
        // TODO Auto-generated method stub
        this.engine = engine;
```

```
        JButton receiverButton = TelepayScreen.getInstance().getMoneyReceiverPanel().getReceiverButton();
        receiverButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                Event automataEvent = null;
                StateMachineContext context = StateMachineContextImpl.create();
                JButton button = (JButton)(event.getSource());
                MoneyReceiverPanel panel = (MoneyReceiverPanel)button.getParent();
                JList list = panel.getBanknotesList();
                int selIndex = list.getSelectedIndex();
                if (selIndex == MoneyReceptor.BANKNOTE_VALUES.length) {
                    automataEvent = new Event(panel.isReceiverValid() ? E12 : E13);
                }
                else {
                    automataEvent = new Event(E11);
                    Parameter moneyCount = new Parameter(Memory.MONEY,
                            new Integer(MoneyReceptor.BANKNOTE_VALUES[selIndex]));
                    moneyCount.addToContext(context.getEventContext());
                }
                MoneyReceptor.this.engine.getEventManager().handle(automataEvent, context);
            }
        });
    }

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

    }

    /**
     * @unimod.action.descr Запретить прием денег от пользователя.
     */
    public void z31(StateMachineContext context) {
        /*TODO: automatically generated by UniMod method*/
        MoneyReceiverPanel panel = TelepayScreen.getInstance().getMoneyReceiverPanel();
        panel.setReceiverValid(false);
    }

    /**
     * @unimod.action.descr Разрешить прием денег от пользователя.
     */
    public void z32(StateMachineContext context) {
        /*TODO: automatically generated by UniMod method*/
        MoneyReceiverPanel panel = TelepayScreen.getInstance().getMoneyReceiverPanel();
        panel.setReceiverValid(true);
    }

    /**
     * @unimod.action.descr Разрешен ли прием денег.
     */
    public boolean x31(StateMachineContext context) {
        /*TODO: automatically generated by UniMod method*/
        MoneyReceiverPanel panel = TelepayScreen.getInstance().getMoneyReceiverPanel();
        return panel.isReceiverValid();
    }
}
```

TelepayScreen.java

```
package ru.ifmo.telepay;

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

/**
 * Frame of application.
 */
public class TelepayScreen extends JFrame {
    static final long serialVersionUID = 0;

    /**
     * Instance of application frame. It is created only once.
     */
    private static TelepayScreen instance = null;

    /**
     * Content of frame.
     */
    private JPanel content;

    /**
     * User screen UI component.
     */
    private UserScreenPanel userScreenPanel;

    /**
     * Keypad UI component.
     */
    private KeyPadPanel keyPadPanel;

    /**
     * Money receiver UI component.
     */
    private MoneyReceiverPanel moneyReceiverPanel;

    /**
     * Returns instance of frame.
     * Note that there is a single instance of TelepayScreen in lifetime of application.
     *
     * @return Instance of frame.
     */
    public static TelepayScreen getInstance() {
        if (instance == null) {
            instance = new TelepayScreen();
        }
        return instance;
    }

    /**
     * Creates new instance of TelepayScreen.
     * Is made private to disable ability to create more then one instance.
     */
    private TelepayScreen() {
        super("Telepay");

        this.setBounds(0, 0, 700, 380);
        content = new JPanel();
        content.setBounds(0, 0, this.getWidth(), this.getHeight());
        content.setLayout(null);
```

```
        userScreenPanel = new UserScreenPanel();
        content.add(userScreenPanel);
        userScreenPanel.setBounds(10, 10, userScreenPanel.getWidth() + 10,
                userScreenPanel.getHeight() + 10);

        keyPadPanel = new KeyPadPanel();
        content.add(keyPadPanel);
        keyPadPanel.setBounds(new Rectangle(getBounds().width - keyPadPanel.getBounds().width - 10,
                userScreenPanel.getBounds().y, keyPadPanel.getWidth(), keyPadPanel.getHeight()));

        moneyReceiverPanel = new MoneyReceiverPanel();
        content.add(moneyReceiverPanel);
        moneyReceiverPanel.setBounds(userScreenPanel.getBounds().x,
                userScreenPanel.getBounds().y + userScreenPanel.getBounds().height,
                moneyReceiverPanel.getWidth(), moneyReceiverPanel.getHeight());

        this.getContentPane().add(content);

        this.setResizable(false);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setVisible(true);
    }

    /**
     * Returns user screen pane.
     *
     * @return User screen UI component pane.
     */
    public UserScreenPanel getUserScreenPanel() {
        return userScreenPanel;
    }

    /**
     * Returns key pad pane.
     *
     * @return Key pad UI component pane.
     */
    public KeyPadPanel getKeyPadPanel() {
        return keyPadPanel;
    }

    /**
     * Returns money receiver pane.
     *
     * @return Money receiver UI component pane.
     */
    public MoneyReceiverPanel getMoneyReceiverPanel() {
        return moneyReceiverPanel;
    }
}
```

UserScreen.java

```
package ru.ifmo.telepay;

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

/**
 * Implements user screen abstraction of model.
 */
public class UserScreen implements ControlledObject {
```

```
    /**
     * Text to be shown as greeting.
     */
    public static final String GREETING = "Нажмите OK для начала работы.";

    /**
     * Text to be shown at text numering.
     */
    public static final String ENTER_NUMBER = "Пожалуйста, введите номер: %s \n";

    /**
     * Text to be shown at money receiving.
     */
    public static final String ENTER_MONEY = ENTER_NUMBER
        + "Полученная сумма: %d рублей. \n";

    /**
     * Text to be shown at confirmation.
     */
    public static final String CONFIRM_INFORMATION = "Пожалуйста, подтвердите предоставленную
информацию: \n"
        + "%d рублей будет зачислено на номер %s. \n";

    /**
     * Text to be shown at the end of work.
     */
    public static final String GOODBYE = "Ваш платеж доставлен. \n" +
        "Спасибо за использование Telepay.";

    /**
     * Text to be shown as help.
     */
    public static final String HELP = "1) Введите номер для оплаты.\n" +
        "2) Заплатите деньги с помощью приемника купюр.\n" +
        "3) Подтвердите информацию.\n" +
        "4) Дождитесь сообщения о доставке платежа.\n" +
        "\n" +
        "Для перехода к следущему шагу используйте клавишу \"OK\" или \n"+
        "возвратитесь для изменения информации, нажав \"Отмена\".";


    /**
     * @unimod.action.descr Показать диалог приветствия.
     */
    public void z11(StateMachineContext context) {
        /*TODO: automatically generated by UniMod method*/
        UserScreenPanel screen = TelepayScreen.getInstance().getUserScreenPanel();
        screen.setTerminalText(GREETING);
        screen.enableOK(true);
    }

    /**
     * @unimod.action.descr Показать диалог ввода номера и приема денег.
     */
    public void z12(StateMachineContext context) {
        /*TODO: automatically generated by UniMod method*/
        UserScreenPanel screen = TelepayScreen.getInstance().getUserScreenPanel();
        screen.setTerminalText(ENTER_MONEY, new Object[] {Memory.getNumber(), new
Integer(Memory.getMoney())});
        screen.enableOK(Memory.isNumberValid());
    }

    /**
```

```
    * @unimod.action.descr Показать диалог подтверждения информации.
    */
   public void z13(StateMachineContext context) {
      /*TODO: automatically generated by UniMod method*/
      UserScreenPanel screen = TelepayScreen.getInstance().getUserScreenPanel();
      screen.setTerminalText(CONFIRM_INFORMATION, new Object[] {new Integer(Memory.getMoney()),
Memory.getNumber()});
   }

   /**
    * @unimod.action.descr Показать диалог окончания работы.
    */
   public void z14(StateMachineContext context) {
      /*TODO: automatically generated by UniMod method*/
      UserScreenPanel screen = TelepayScreen.getInstance().getUserScreenPanel();
      screen.setTerminalText(GOODBYE);
   }

   /**
    * @unimod.action.descr Показать справку.
    */
   public void z15(StateMachineContext context) {
      /*TODO: automatically generated by UniMod method*/
      UserScreenPanel screen = TelepayScreen.getInstance().getUserScreenPanel();
      screen.showHelp(true);
      screen.setHelpText(HELP);
   }

   /**
    * @unimod.action.descr Выключить показ справки.
    */
   public void z16(StateMachineContext context) {
      /*TODO: automatically generated by UniMod method*/
      UserScreenPanel screen = TelepayScreen.getInstance().getUserScreenPanel();
      screen.showHelp(false);
   }
}
```

UserScreenPanel.java

```
package ru.ifmo.telepay;

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

import java.io.*;

/**
 * Implements user screen UI component.
 */
public class UserScreenPanel extends JPanel {
   static final long serialVersionUID = 0;

   /**
    * Text to be written on OK button.
    */
   private static final String OK_TEXT = "OK";

   /**
    * Text to be written on Cancel button.
    */
   private static final String CANCEL_TEXT = "Отмена";
```

```
/**
 * Text to be written on Help button.
 */
private static final String HELP_TEXT = "Справка";

/**
 * Text to be written on Back button.
 */
private static final String BACK_TEXT = "Назад";

/**
 * OK control touch key.
 */
private JButton ok;

/**
 * Cancel cotrol touch key.
 */
private JButton cancel;

/**
 * Help touch key.
 */
private JButton help;

/**
 * Text area displaying user screen text.
 */
private JTextArea textLabel;

/**
 * Text area displaying help.
 */
private JTextArea helpArea;

/**
 * Creates new instance of UserScreenPanel.
 */
public UserScreenPanel() {
   this.setLayout(null);
   this.setSize(510, 210);
   this.setBackground(Color.BLUE);

   final int btnWidth = this.getBounds().width / 6;
   final int btnHeight = this.getBounds().height / 6;
   final int space = 10;

   ok = new JButton(OK_TEXT);
   this.add(ok);
   ok.setBounds(this.getBounds().x + this.getBounds().width - btnWidth,
            this.getBounds().y + this.getBounds().height - 2 * btnHeight,
            btnWidth, btnHeight);

   cancel = new JButton(CANCEL_TEXT);
   this.add(cancel);
   cancel.setBounds(this.getBounds().x + this.getBounds().width - btnWidth,
         this.getBounds().y + this.getBounds().height - btnHeight,
         btnWidth, btnHeight);

   help = new JButton(HELP_TEXT);
   help.setVisible(true);
   this.add(help);
   help.setBounds(this.getBounds().x + space,
```

```
            this.getBounds().y + this.getBounds().height - btnHeight,
            btnWidth, btnHeight);


        textLabel = new JTextArea();
        this.add(textLabel);
        textLabel.setForeground(Color.YELLOW.brighter().brighter());
        textLabel.setBackground(this.getBackground());
        textLabel.setEditable(false);
        textLabel.setFont(new Font("Monospace", Font.BOLD, 15));
        textLabel.setBounds(new Rectangle(0, 0, getBounds().width, ok.getBounds().y));
        textLabel.setText(UserScreen.GREETING);

        helpArea = new JTextArea();
        this.add(helpArea);
        helpArea.setForeground(textLabel.getForeground());
        helpArea.setBackground(this.getBackground());
        helpArea.setEditable(false);
        helpArea.setFont(textLabel.getFont());
        helpArea.setBounds(textLabel.getBounds());
        helpArea.setWrapStyleWord(true);
        showHelp(false);
    }

    /**
     * Returns OK button.
     * Used to add listeners for event handling.
     *
     * @return OK button of component.
     */
    public JButton getOkButton() {
        return ok;
    }

    /**
     * Returns Cancel button.
     * Used to add listeners for event handling.
     *
     * @return Cancel button of component.
     */
    public JButton getCancelButton() {
        return cancel;
    }

    /**
     * Sets text to user terminal screen.
     *
     * @param text Text to show user.
     */
    public void setTerminalText(String text) {
        textLabel.setText(text);
    }

    /**
     * Returns text displayed at user terminal.
     *
     * @return Text displayed on terminal.
     */
    public String getTerminalText() {
        return textLabel.getText();
    }

    /**
```

```
     * Sets text to terminal.
     * Text is printf-like formatted depending on format and args.
     *
     * @param format Pritf-like format of text.
     * @param args Arguments to use in instantiating format.
     */
    public void setTerminalText(String format, Object[] args) {
        StringWriter writer = new StringWriter();
        PrintWriter printer = new PrintWriter(writer);
        printer.printf(format, args);
        this.setTerminalText(writer.getBuffer().toString());
    }

    /**
     * Enables or disables OK button.
     *
     * @param enable Whether to enable button.
     */
    public void enableOK(boolean enable) {
        ok.setEnabled(enable);
    }

    /**
     * Returns Help button of component.
     * Used to add listeners.
     *
     * @return Help button.
     */
    public JButton getHelpButton() {
        return help;
    }

    /**
     * Shows or hides help.
     * @param show If show is true help is shown else it is hidden.
     */
    public void showHelp(boolean show) {
        helpArea.setVisible(show);
        textLabel.setVisible(!show);
        help.setText(show ? BACK_TEXT : HELP_TEXT);
    }

    /**
     * Sets text for help.
     *
     * @param helpText
     */
    public void setHelpText(String helpText) {
        helpArea.setText(helpText);
    }
}
```

UserTerminal.java

```
package ru.ifmo.telepay;

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

import javax.swing.*;
```

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

/**
 * Implements user terminal event provider abstraction.
 */
public class UserTerminal implements EventProvider {
   /**
    * Model engine.
    */
   private ModelEngine engine;

   /**
    * @unimod.event.descr Нажата клавиша "OK".
    */
   public static final String E21 = "e21";
   /**
    * @unimod.event.descr Нажата клавиша "Отмена".
    */
   public static final String E22 = "e22";

   /**
    * @unimod.event.descr Нажата клавиша "Справка".
    */
   public static final String E23 = "e23";

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

      UserScreenPanel userScreenPanel = TelepayScreen.getInstance().getUserScreenPanel();
      JButton ok = userScreenPanel.getOkButton();
      JButton cancel = userScreenPanel.getCancelButton();
      JButton help = userScreenPanel.getHelpButton();

      ok.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            Event fire = new Event(E21);
            UserTerminal.this.engine.getEventManager().handle(fire,
                  StateMachineContextImpl.create());
         }
      });

      cancel.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            Event fire = new Event(E22);
            UserTerminal.this.engine.getEventManager().handle(fire,
                  StateMachineContextImpl.create());
         }
      });

      help.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            Event fire = new Event(E23);
            UserTerminal.this.engine.getEventManager().handle(fire,
                  StateMachineContextImpl.create());
         }
      });
   }

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

Приложение 4. Сгенерированный код на *Java*

ModelEventProcessor.java

```
/**
 * This file was generated from model [Model1] on [Sat Oct 14 03:19:33 MSD 2006].
 * Do not change content of this file.
 */

import java.io.IOException;
import java.util.*;

import org.apache.commons.lang.BooleanUtils;
import org.apache.commons.lang.math.NumberUtils;
import org.apache.commons.lang.StringUtils;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import com.evelopers.common.exception.*;
import com.evelopers.unimod.core.stateworks.*;
import com.evelopers.unimod.debug.app.AppDebugger;
import com.evelopers.unimod.debug.protocol.JavaSpecificMessageCoder;
import com.evelopers.unimod.runtime.*;
import com.evelopers.unimod.runtime.context.*;
import com.evelopers.unimod.runtime.logger.SimpleLogger;


public class ModelEventProcessor extends AbstractEventProcessor {

   private ModelStructure modelStructure;

   private static final int A3 = 1;
   private static final int A2 = 2;
   private static final int A1 = 3;

   private int decodeStateMachine(String sm) {

      if ("A3".equals(sm)) {
         return A3;
      } else

      if ("A2".equals(sm)) {
         return A2;
      } else

      if ("A1".equals(sm)) {
         return A1;
      }

      return -1;
   }

   private A3EventProcessor _A3;
   private A2EventProcessor _A2;
   private A1EventProcessor _A1;

   public ModelEventProcessor() {
      modelStructure = new Model1ModelStructure();

      _A3 = new A3EventProcessor();
      _A2 = new A2EventProcessor();
```

```
    _A1 = new A1EventProcessor();
}

public static void run(int debuggerPort, boolean debuggerSuspend) throws
        InterruptedException, EventProcessorException, CommonException,
        IOException {

    /* Create runtime engine */
    ModelEngine engine = createModelEngine(true);

    /* Setup logger */
    final Log log = LogFactory.getLog(ModelEventProcessor.class);
    engine.getEventProcessor().addEventProcessorListener(new SimpleLogger(log));

    /* Setup exception handler */
    engine.getEventProcessor().addExceptionHandler(new ExceptionHandler() {
        public void handleException(StateMachineContext context, SystemException e) {
            log.fatal(e.getChainedMessage(), e.getRootException());
        }
    });

    if (debuggerPort > 0) {
        AppDebugger d = new AppDebugger(
                debuggerPort, debuggerSuspend,
                new JavaSpecificMessageCoder(), engine);
        d.start();
    }
    engine.start();
}

public static void main(String[] args) throws Exception {
    int debuggerPort = NumberUtils.stringToInt(System.getProperty("debugger.port"), -1);
    boolean debuggerSuspend = BooleanUtils.toBoolean(System.getProperty("debugger.suspend"));
    ModelEventProcessor.run(debuggerPort, debuggerSuspend);
}

public static ModelEngine createModelEngine(boolean useEventQueue) throws CommonException {
    ObjectsManager objectsManager = new ObjectsManager();
    return ModelEngine.createStandAlone(
            useEventQueue ? (EventManager) new QueuedHandler() : (EventManager) new StrictHandler(),
            new ModelEventProcessor(),
            objectsManager.getControlledObjectsManager(),
            objectsManager.getEventProvidersManager());
}

public static class ObjectsManager {
    private ru.ifmo.telepay.UserScreen o1 = null;
    private ru.ifmo.telepay.MoneyReceptor o3 = null;
    private ru.ifmo.telepay.Memory o2 = null;
    private ru.ifmo.telepay.MoneyReceptor p1 = null;
    private ru.ifmo.telepay.KeyPad p3 = null;
    private ru.ifmo.telepay.UserTerminal p2 = null;

    private ControlledObjectsManager controlledObjectsManager = new ControlledObjectsManagerImpl();
    private EventProvidersManager eventProvidersManager = new EventProvidersManagerImpl();

    public ControlledObjectsManager getControlledObjectsManager() {
        return controlledObjectsManager;
    }

    public EventProvidersManager getEventProvidersManager() {
        return eventProvidersManager;
    }
```

```
private class ControlledObjectsManagerImpl implements ControlledObjectsManager {
   public void init(ModelEngine engine) throws CommonException {
   }

   public void dispose() {
   }

   public ControlledObject getControlledObject(String coName) {
      if (StringUtils.equals(coName, "o1")) {
         if (o1 == null) {
            o1 = new ru.ifmo.telepay.UserScreen();
         }
         return o1;
      }
      if (StringUtils.equals(coName, "o3")) {
         if (o3 == null) {
            o3 = new ru.ifmo.telepay.MoneyReceptor();
         }
         return o3;
      }
      if (StringUtils.equals(coName, "o2")) {
         if (o2 == null) {
            o2 = new ru.ifmo.telepay.Memory();
         }
         return o2;
      }
      throw new IllegalArgumentException("Controlled object with name [" + coName + "] wasn't found");
   }
}

private class EventProvidersManagerImpl implements EventProvidersManager {
   private List nonameEventProviders = new ArrayList();

   public void init(ModelEngine engine) throws CommonException {
      EventProvider ep;
      ep = getEventProvider("p1");
      ep.init(engine);
      ep = getEventProvider("p3");
      ep.init(engine);
      ep = getEventProvider("p2");
      ep.init(engine);
   }

   public void dispose() {
      EventProvider ep;
      ep = getEventProvider("p1");
      ep.dispose();
      ep = getEventProvider("p3");
      ep.dispose();
      ep = getEventProvider("p2");
      ep.dispose();
      for (Iterator i = nonameEventProviders.iterator(); i.hasNext();) {
         ep = (EventProvider) i.next();
         ep.dispose();
      }
   }

   public EventProvider getEventProvider(String epName) {
      if (StringUtils.equals(epName, "p1")) {
         if (p1 == null) {
            p1 = new ru.ifmo.telepay.MoneyReceptor();
         }
```

```
            return p1;
        }
        if (StringUtils.equals(epName, "p3")) {
            if (p3 == null) {
                p3 = new ru.ifmo.telepay.KeyPad();
            }
            return p3;
        }
        if (StringUtils.equals(epName, "p2")) {
            if (p2 == null) {
                p2 = new ru.ifmo.telepay.UserTerminal();
            }
            return p2;
        }
        throw new IllegalArgumentException("Event provider with name [" + epName + "] wasn't found");
    }
  }
}

public ModelStructure getModelStructure() {
  return modelStructure;
}

public void setControlledObjectsMap(ControlledObjectsMap controlledObjectsMap) {
  super.setControlledObjectsMap(controlledObjectsMap);

  _A3.init(controlledObjectsMap);
  _A2.init(controlledObjectsMap);
  _A1.init(controlledObjectsMap);
}

protected StateMachineConfig process(
      Event event, StateMachineContext context,
      StateMachinePath path, StateMachineConfig config) throws SystemException {

  // get state machine from path
  int sm = decodeStateMachine(path.getStateMachine());

  try {
    switch (sm) {
      case A3:
        return _A3.process(event, context, path, config);
      case A2:
        return _A2.process(event, context, path, config);
      case A1:
        return _A1.process(event, context, path, config);
      default:
        throw new EventProcessorException("Unknown state machine [" + path.getStateMachine() + "]");
    }
  } catch (Exception e) {
    if (e instanceof SystemException) {
      throw (SystemException) e;
    } else {
      throw new SystemException(e);
    }
  }
}

protected StateMachineConfig transiteToStableState(
      StateMachineContext context,
      StateMachinePath path, StateMachineConfig config) throws SystemException {

  // get state machine from path
```

```
    int sm = decodeStateMachine(path.getStateMachine());

    try {
      switch (sm) {
        case A3:
          return _A3.transiteToStableState(context, path, config);
        case A2:
          return _A2.transiteToStableState(context, path, config);
        case A1:
          return _A1.transiteToStableState(context, path, config);
        default:
          throw new EventProcessorException("Unknown state machine [" + path.getStateMachine() + "]");
      }
    } catch (Exception e) {
      if (e instanceof SystemException) {
        throw (SystemException) e;
      } else {
        throw new SystemException(e);
      }
    }
  }


  private class Model1ModelStructure implements ModelStructure {
    private Map configManagers = new HashMap();

    private Model1ModelStructure() {
      configManagers.put("A3", new com.evelopers.unimod.runtime.config.DistinguishConfigManager());
      configManagers.put("A2", new com.evelopers.unimod.runtime.config.DistinguishConfigManager());
      configManagers.put("A1", new com.evelopers.unimod.runtime.config.DistinguishConfigManager());
    }

    public StateMachinePath getRootPath()
        throws EventProcessorException {
      return new StateMachinePath("A3");
    }

    public StateMachineConfigManager getConfigManager(String stateMachine)
        throws EventProcessorException {
      return (StateMachineConfigManager) configManagers.get(stateMachine);
    }

    public StateMachineConfig getTopConfig(String stateMachine)
        throws EventProcessorException {
      int sm = decodeStateMachine(stateMachine);

      switch (sm) {
        case A3:
          return new StateMachineConfig("Top");
        case A2:
          return new StateMachineConfig("Top");
        case A1:
          return new StateMachineConfig("Top");
        default:
          throw new EventProcessorException("Unknown state machine [" + stateMachine + "]");
      }
    }

    public boolean isFinal(String stateMachine, StateMachineConfig config)
        throws EventProcessorException {
      /* Get state machine from path */
      int sm = decodeStateMachine(stateMachine);
      int state;
```

```
            switch (sm) {
               case A3:
                  state = _A3.decodeState(config.getActiveState());
                  switch (state) {
                     default:
                        return false;
                  }
               case A2:
                  state = _A2.decodeState(config.getActiveState());
                  switch (state) {
                     default:
                        return false;
                  }
               case A1:
                  state = _A1.decodeState(config.getActiveState());
                  switch (state) {
                     default:
                        return false;
                  }
               default:
                  throw new EventProcessorException("Unknown state machine [" + stateMachine + "]");
            }
         }
      }


      private class A3EventProcessor {

         // states
         private static final int Top = 1;
         private static final int s1 = 2;
         private static final int _1_Работа = 3;
         private static final int _2_Справка = 4;

         private int decodeState(String state) {

            if ("Top".equals(state)) {
               return Top;
            } else

            if ("s1".equals(state)) {
               return s1;
            } else

            if ("1.Работа".equals(state)) {
               return _1_Работа;
            } else

            if ("2.Справка".equals(state)) {
               return _2_Справка;
            }

            return -1;
         }

         // events
         private static final int e23 = 1;

         private int decodeEvent(String event) {

            if ("e23".equals(event)) {
               return e23;
            }
```

```
        return -1;
    }

    private ru.ifmo.telepay.UserScreen o1;
    private ru.ifmo.telepay.Memory o2;

    private void init(ControlledObjectsMap controlledObjectsMap) {
        o1 = (ru.ifmo.telepay.UserScreen) controlledObjectsMap.getControlledObject("o1");
        o2 = (ru.ifmo.telepay.Memory) controlledObjectsMap.getControlledObject("o2");
    }

    private StateMachineConfig process(Event event, StateMachineContext context, StateMachinePath path,
StateMachineConfig config) throws Exception {
        config = lookForTransition(event, context, path, config);

        config = transiteToStableState(context, path, config);

        // execute included state machines
        executeSubmachines(event, context, path, config);

        return config;
    }

    private void executeSubmachines(Event event, StateMachineContext context, StateMachinePath path,
StateMachineConfig config) throws Exception {
        int state = decodeState(config.getActiveState());

        while (true) {
            switch (state) {
                case s1:

                    return;
                case _1_Работа:
                    // 1.Работа includes A1

                    fireBeforeSubmachineExecution(context, event, path, "1.Работа", "A1");

                    ModelEventProcessor.this.process(event, context, new StateMachinePath(path,
                            "1.Работа", "A1"));

                    fireAfterSubmachineExecution(context, event, path, "1.Работа", "A1");

                    return;
                case _2_Справка:

                    return;
                default:
                    throw new EventProcessorException("State with name [" + config.getActiveState() + "] is unknown
for state machine [A3]");
            }
        }
    }

    private StateMachineConfig transiteToStableState(StateMachineContext context, StateMachinePath path,
StateMachineConfig config) throws Exception {

        int s = decodeState(config.getActiveState());
        Event event;

        switch (s) {
            case Top:
```

```
            fireComeToState(context, path, "s1");

            // s1->1.Работа [true]/
            event = Event.NO_EVENT;
            fireTransitionFound(context, path, "s1", event, "s1#1.Работа##true");


            fireComeToState(context, path, "1.Работа");

            // 1.Работа []

            return new StateMachineConfig("1.Работа");
        }

        return config;
    }

    private StateMachineConfig lookForTransition(Event event, StateMachineContext context, StateMachinePath
path, StateMachineConfig config) throws Exception {


        BitSet calculatedInputActions = new BitSet(0);

        int s = decodeState(config.getActiveState());
        int e = decodeEvent(event.getName());

        while (true) {
            switch (s) {
                case _1_Работа:


                    switch (e) {
                        case e23:

                            // 1.Работа->2.Справка e23[true]/o1.z15

                            fireTransitionCandidate(context, path, "1.Работа", event, "1.Работа#2.Справка#e23#true");


                            fireTransitionFound(context, path, "1.Работа", event, "1.Работа#2.Справка#e23#true");

                            fireBeforeOutputActionExecution(context, path, "1.Работа#2.Справка#e23#true", "o1.z15");

                            o1.z15(context);

                            fireAfterOutputActionExecution(context, path, "1.Работа#2.Справка#e23#true", "o1.z15");

                            fireComeToState(context, path, "2.Справка");

                            // 2.Справка []
                            return new StateMachineConfig("2.Справка");


                        default:

                            // transition not found
                            return config;
                    }

                case _2_Справка:
```

```
                switch (e) {
                  case e23:

                    // 2.Справка->1.Работа e23[true]/o1.z16

                    fireTransitionCandidate(context, path, "2.Справка", event, "2.Справка#1.Работа#e23#true");


                    fireTransitionFound(context, path, "2.Справка", event, "2.Справка#1.Работа#e23#true");

                    fireBeforeOutputActionExecution(context, path, "2.Справка#1.Работа#e23#true", "o1.z16");

                    o1.z16(context);

                    fireAfterOutputActionExecution(context, path, "2.Справка#1.Работа#e23#true", "o1.z16");

                    fireComeToState(context, path, "1.Работа");

                    // 1.Работа []
                    return new StateMachineConfig("1.Работа");


                  default:

                    // transition not found
                    return config;
                }

              default:
                throw new EventProcessorException("Incorrect stable state [" + config.getActiveState() + "] in state
machine [A3]");
            }
        }
    }


}


  private class A2EventProcessor {

    // states
    private static final int Top = 1;
    private static final int s4 = 2;
    private static final int _1_Прием_купюры = 3;
    private static final int _2_Возврат_купюры = 4;

    private int decodeState(String state) {

      if ("Top".equals(state)) {
        return Top;
      } else

      if ("s4".equals(state)) {
        return s4;
      } else

      if ("1.Прием купюры".equals(state)) {
        return _1_Прием_купюры;
      } else

      if ("2.Возврат купюры".equals(state)) {
        return _2_Возврат_купюры;
```

```
        }

        return -1;
    }

    // events
    private static final int e13 = 1;
    private static final int e11 = 2;
    private static final int e12 = 3;

    private int decodeEvent(String event) {

        if ("e13".equals(event)) {
            return e13;
        } else

        if ("e11".equals(event)) {
            return e11;
        } else

        if ("e12".equals(event)) {
            return e12;
        }

        return -1;
    }

    private ru.ifmo.telepay.UserScreen o1;
    private ru.ifmo.telepay.MoneyReceptor o3;
    private ru.ifmo.telepay.Memory o2;

    private void init(ControlledObjectsMap controlledObjectsMap) {
        o1 = (ru.ifmo.telepay.UserScreen) controlledObjectsMap.getControlledObject("o1");
        o3 = (ru.ifmo.telepay.MoneyReceptor) controlledObjectsMap.getControlledObject("o3");
        o2 = (ru.ifmo.telepay.Memory) controlledObjectsMap.getControlledObject("o2");
    }

    private StateMachineConfig process(Event event, StateMachineContext context, StateMachinePath path,
StateMachineConfig config) throws Exception {
        config = lookForTransition(event, context, path, config);

        config = transiteToStableState(context, path, config);

        // execute included state machines
        executeSubmachines(event, context, path, config);

        return config;
    }

    private void executeSubmachines(Event event, StateMachineContext context, StateMachinePath path,
StateMachineConfig config) throws Exception {
        int state = decodeState(config.getActiveState());

        while (true) {
            switch (state) {
                case s4:

                    return;
                case _1_Прием_купюры:

                    return;
                case _2_Возврат_купюры:
```

```
                return;
            default:
                throw new EventProcessorException("State with name [" + config.getActiveState() + "] is unknown
for state machine [A2]");
            }
        }
    }

    private StateMachineConfig transiteToStableState(StateMachineContext context, StateMachinePath path,
StateMachineConfig config) throws Exception {

        int s = decodeState(config.getActiveState());
        Event event;

        switch (s) {
            case Top:


                fireComeToState(context, path, "s4");

                // s4->1.Прием купюры [true]/
                event = Event.NO_EVENT;
                fireTransitionFound(context, path, "s4", event, "s4#1.Прием купюры##true");


                fireComeToState(context, path, "1.Прием купюры");

                // 1.Прием купюры []

                return new StateMachineConfig("1.Прием купюры");
        }

        return config;
    }

    private StateMachineConfig lookForTransition(Event event, StateMachineContext context, StateMachinePath
path, StateMachineConfig config) throws Exception {


        BitSet calculatedInputActions = new BitSet(0);

        int s = decodeState(config.getActiveState());
        int e = decodeEvent(event.getName());

        while (true) {
            switch (s) {
                case _1_Прием_купюры:


                    switch (e) {
                        case e11:

                            // 1.Прием купюры->1.Прием купюры e11[true]/o2.z22,o1.z12

                            fireTransitionCandidate(context, path, "1.Прием купюры", event, "1.Прием
купюры#1.Прием купюры#e11#true");


                            fireTransitionFound(context, path, "1.Прием купюры", event, "1.Прием купюры#1.Прием
купюры#e11#true");

                            fireBeforeOutputActionExecution(context, path, "1.Прием купюры#1.Прием
купюры#e11#true", "o2.z22");
```

```
                    o2.z22(context);

                    fireAfterOutputActionExecution(context, path, "1.Прием купюры#1.Прием
купюры#e11#true", "o2.z22");
                    fireBeforeOutputActionExecution(context, path, "1.Прием купюры#1.Прием
купюры#e11#true", "o1.z12");

                    o1.z12(context);

                    fireAfterOutputActionExecution(context, path, "1.Прием купюры#1.Прием
купюры#e11#true", "o1.z12");

                    fireComeToState(context, path, "1.Прием купюры");

                    // 1.Прием купюры []
                    return new StateMachineConfig("1.Прием купюры");


                case e12:

                    // 1.Прием купюры->2.Возврат купюры e12[true]/o3.z31

                    fireTransitionCandidate(context, path, "1.Прием купюры", event, "1.Прием
купюры#2.Возврат купюры#e12#true");


                    fireTransitionFound(context, path, "1.Прием купюры", event, "1.Прием купюры#2.Возврат
купюры#e12#true");

                    fireBeforeOutputActionExecution(context, path, "1.Прием купюры#2.Возврат
купюры#e12#true", "o3.z31");

                    o3.z31(context);

                    fireAfterOutputActionExecution(context, path, "1.Прием купюры#2.Возврат
купюры#e12#true", "o3.z31");

                    fireComeToState(context, path, "2.Возврат купюры");

                    // 2.Возврат купюры []
                    return new StateMachineConfig("2.Возврат купюры");


                default:

                    // transition not found
                    return config;
            }

        case _2_Возврат_купюры:


            switch (e) {
              case e13:

                    // 2.Возврат купюры->1.Прием купюры e13[true]/o3.z32

                    fireTransitionCandidate(context, path, "2.Возврат купюры", event, "2.Возврат
купюры#1.Прием купюры#e13#true");
```

```
                    fireTransitionFound(context, path, "2.Возврат купюры", event, "2.Возврат купюры#1.Прием
купюры#e13#true");

                    fireBeforeOutputActionExecution(context, path, "2.Возврат купюры#1.Прием
купюры#e13#true", "o3.z32");

                    o3.z32(context);

                    fireAfterOutputActionExecution(context, path, "2.Возврат купюры#1.Прием
купюры#e13#true", "o3.z32");

                    fireComeToState(context, path, "1.Прием купюры");

                    // 1.Прием купюры []
                    return new StateMachineConfig("1.Прием купюры");


                default:

                    // transition not found
                    return config;
                }

            default:
              throw new EventProcessorException("Incorrect stable state [" + config.getActiveState() + "] in state
machine [A2]");
            }
        }
    }


}


  private class A1EventProcessor {

    // states
    private static final int Top = 1;
    private static final int s1 = 2;
    private static final int _1_Приветствие = 3;
    private static final int _2_Ввод_номера = 4;
    private static final int _3__Прием_денег = 5;
    private static final int _5_Уведомление = 6;
    private static final int _4_Подтверждение = 7;

    private int decodeState(String state) {

      if ("Top".equals(state)) {
        return Top;
      } else

      if ("s1".equals(state)) {
        return s1;
      } else

      if ("1.Приветствие".equals(state)) {
        return _1_Приветствие;
      } else

      if ("2.Ввод номера".equals(state)) {
        return _2_Ввод_номера;
      } else
```

```
            if ("3. Прием денег".equals(state)) {
                return _3__Прием_денег;
            } else

            if ("5.Уведомление".equals(state)) {
                return _5_Уведомление;
            } else

            if ("4.Подтверждение".equals(state)) {
                return _4_Подтверждение;
            }

            return -1;
        }

        // events
        private static final int e31 = 1;
        private static final int e22 = 2;
        private static final int e30 = 3;
        private static final int e21 = 4;

        private int decodeEvent(String event) {

            if ("e31".equals(event)) {
                return e31;
            } else

            if ("e22".equals(event)) {
                return e22;
            } else

            if ("e30".equals(event)) {
                return e30;
            } else

            if ("e21".equals(event)) {
                return e21;
            }

            return -1;
        }

        private ru.ifmo.telepay.UserScreen o1;
        private ru.ifmo.telepay.MoneyReceptor o3;
        private ru.ifmo.telepay.Memory o2;

        private void init(ControlledObjectsMap controlledObjectsMap) {
            o1 = (ru.ifmo.telepay.UserScreen) controlledObjectsMap.getControlledObject("o1");
            o3 = (ru.ifmo.telepay.MoneyReceptor) controlledObjectsMap.getControlledObject("o3");
            o2 = (ru.ifmo.telepay.Memory) controlledObjectsMap.getControlledObject("o2");
        }

        private StateMachineConfig process(Event event, StateMachineContext context, StateMachinePath path,
StateMachineConfig config) throws Exception {
            config = lookForTransition(event, context, path, config);

            config = transiteToStableState(context, path, config);

            // execute included state machines
            executeSubmachines(event, context, path, config);

            return config;
        }
```

```
    private void executeSubmachines(Event event, StateMachineContext context, StateMachinePath path,
StateMachineConfig config) throws Exception {
        int state = decodeState(config.getActiveState());

        while (true) {
          switch (state) {
            case s1:

              return;
            case _1_Приветствие:

              return;
            case _2_Ввод_номера:

              return;
            case _3__Прием_денег:
              // 3. Прием денег includes A2

              fireBeforeSubmachineExecution(context, event, path, "3. Прием денег", "A2");

              ModelEventProcessor.this.process(event, context, new StateMachinePath(path,
                  "3. Прием денег", "A2"));

              fireAfterSubmachineExecution(context, event, path, "3. Прием денег", "A2");

              return;
            case _5_Уведомление:

              return;
            case _4_Подтверждение:

              return;
            default:
              throw new EventProcessorException("State with name [" + config.getActiveState() + "] is unknown
for state machine [A1]");
            }
          }
      }

    private StateMachineConfig transiteToStableState(StateMachineContext context, StateMachinePath path,
StateMachineConfig config) throws Exception {

        int s = decodeState(config.getActiveState());
        Event event;

        switch (s) {
          case Top:


            fireComeToState(context, path, "s1");

            // s1->1.Приветствие [true]/
            event = Event.NO_EVENT;
            fireTransitionFound(context, path, "s1", event, "s1#1.Приветствие##true");


            fireComeToState(context, path, "1.Приветствие");

            // 1.Приветствие [o1.z11]
            fireBeforeOutputActionExecution(context, path, "s1#1.Приветствие##true", "o1.z11");

            o1.z11(context);
```

```
            fireAfterOutputActionExecution(context, path, "s1#1.Приветствие##true", "o1.z11");

            return new StateMachineConfig("1.Приветствие");
        }

        return config;
    }

    private StateMachineConfig lookForTransition(Event event, StateMachineContext context, StateMachinePath
path, StateMachineConfig config) throws Exception {


        boolean

            o2_x1 = false,

            o3_x31 = false;


        BitSet calculatedInputActions = new BitSet(2);

        int s = decodeState(config.getActiveState());
        int e = decodeEvent(event.getName());

        while (true) {
          switch (s) {
            case _1_Приветствие:


              switch (e) {
                case e21:

                  // 1.Приветствие->2.Ввод номера e21[true]//

                  fireTransitionCandidate(context, path, "1.Приветствие", event, "1.Приветствие#2.Ввод
номера#e21#true");


                  fireTransitionFound(context, path, "1.Приветствие", event, "1.Приветствие#2.Ввод
номера#e21#true");


                  fireComeToState(context, path, "2.Ввод номера");

                  // 2.Ввод номера [o1.z12]
                  fireBeforeOutputActionExecution(context, path, "1.Приветствие#2.Ввод номера#e21#true",
"o1.z12");

                  o1.z12(context);

                  fireAfterOutputActionExecution(context, path, "1.Приветствие#2.Ввод номера#e21#true",
"o1.z12");
                  return new StateMachineConfig("2.Ввод номера");


                default:

                  // transition not found
                  return config;
              }

            case _2_Ввод_номера:
```

```
                switch (e) {
                  case e31:

                    // 2.Ввод номера->2.Ввод номера e31[true]/o2.z25

                    fireTransitionCandidate(context, path, "2.Ввод номера", event, "2.Ввод номера#2.Ввод
номера#e31#true");


                    fireTransitionFound(context, path, "2.Ввод номера", event, "2.Ввод номера#2.Ввод
номера#e31#true");

                    fireBeforeOutputActionExecution(context, path, "2.Ввод номера#2.Ввод номера#e31#true",
"o2.z25");

                    o2.z25(context);

                    fireAfterOutputActionExecution(context, path, "2.Ввод номера#2.Ввод номера#e31#true",
"o2.z25");

                    fireComeToState(context, path, "2.Ввод номера");

                    // 2.Ввод номера [o1.z12]
                    fireBeforeOutputActionExecution(context, path, "2.Ввод номера#2.Ввод номера#e31#true",
"o1.z12");

                    o1.z12(context);

                    fireAfterOutputActionExecution(context, path, "2.Ввод номера#2.Ввод номера#e31#true",
"o1.z12");
                    return new StateMachineConfig("2.Ввод номера");


                case e22:

                    // 2.Ввод номера->1.Приветствие e22[true]/o2.z24

                    fireTransitionCandidate(context, path, "2.Ввод номера", event, "2.Ввод
номера#1.Приветствие#e22#true");


                    fireTransitionFound(context, path, "2.Ввод номера", event, "2.Ввод
номера#1.Приветствие#e22#true");

                    fireBeforeOutputActionExecution(context, path, "2.Ввод номера#1.Приветствие#e22#true",
"o2.z24");

                    o2.z24(context);

                    fireAfterOutputActionExecution(context, path, "2.Ввод номера#1.Приветствие#e22#true",
"o2.z24");

                    fireComeToState(context, path, "1.Приветствие");

                    // 1.Приветствие [o1.z11]
                    fireBeforeOutputActionExecution(context, path, "2.Ввод номера#1.Приветствие#e22#true",
"o1.z11");

                    o1.z11(context);
```

```
                fireAfterOutputActionExecution(context, path, "2.Ввод номера#1.Приветствие#e22#true",
"o1.z11");
                return new StateMachineConfig("1.Приветствие");

            case e30:

                // 2.Ввод номера->2.Ввод номера e30[true]/o2.z23

                fireTransitionCandidate(context, path, "2.Ввод номера", event, "2.Ввод номера#2.Ввод
номера#e30#true");


                fireTransitionFound(context, path, "2.Ввод номера", event, "2.Ввод номера#2.Ввод
номера#e30#true");

                fireBeforeOutputActionExecution(context, path, "2.Ввод номера#2.Ввод номера#e30#true",
"o2.z23");

                o2.z23(context);

                fireAfterOutputActionExecution(context, path, "2.Ввод номера#2.Ввод номера#e30#true",
"o2.z23");

                fireComeToState(context, path, "2.Ввод номера");

                // 2.Ввод номера [o1.z12]
                fireBeforeOutputActionExecution(context, path, "2.Ввод номера#2.Ввод номера#e30#true",
"o1.z12");

                o1.z12(context);

                fireAfterOutputActionExecution(context, path, "2.Ввод номера#2.Ввод номера#e30#true",
"o1.z12");
                return new StateMachineConfig("2.Ввод номера");

            case e21:

                // 2.Ввод номера->2.Ввод номера e21[!o2.x1]/

                fireTransitionCandidate(context, path, "2.Ввод номера", event, "2.Ввод номера#2.Ввод
номера#e21#!o2.x1");

                if (!isInputActionCalculated(calculatedInputActions, _o2_x1)) {

                    fireBeforeInputActionExecution(context, path, "2.Ввод номера#2.Ввод номера#e21#!o2.x1",
"o2.x1");

                    o2_x1 = o2.x1(context);

                    fireAfterInputActionExecution(context, path, "2.Ввод номера#2.Ввод номера#e21#!o2.x1",
"o2.x1", new Boolean(o2_x1));
                }

                if (!o2_x1) {

                    fireTransitionFound(context, path, "2.Ввод номера", event, "2.Ввод номера#2.Ввод
номера#e21#!o2.x1");


                    fireComeToState(context, path, "2.Ввод номера");
```

```
                        // 2.Ввод номера [o1.z12]
                        fireBeforeOutputActionExecution(context, path, "2.Ввод номера#2.Ввод
номера#e21#!o2.x1", "o1.z12");

                        o1.z12(context);

                        fireAfterOutputActionExecution(context, path, "2.Ввод номера#2.Ввод номера#e21#!o2.x1",
"o1.z12");
                        return new StateMachineConfig("2.Ввод номера");

                    }
                    // 2.Ввод номера->3. Прием денег e21[o2.x1]/o3.z32

                    fireTransitionCandidate(context, path, "2.Ввод номера", event, "2.Ввод номера#3. Прием
денег#e21#o2.x1");


                    if (o2_x1) {

                        fireTransitionFound(context, path, "2.Ввод номера", event, "2.Ввод номера#3. Прием
денег#e21#o2.x1");

                        fireBeforeOutputActionExecution(context, path, "2.Ввод номера#3. Прием
денег#e21#o2.x1", "o3.z32");

                        o3.z32(context);

                        fireAfterOutputActionExecution(context, path, "2.Ввод номера#3. Прием денег#e21#o2.x1",
"o3.z32");

                        fireComeToState(context, path, "3. Прием денег");

                        // 3. Прием денег [o1.z12]
                        fireBeforeOutputActionExecution(context, path, "2.Ввод номера#3. Прием
денег#e21#o2.x1", "o1.z12");

                        o1.z12(context);

                        fireAfterOutputActionExecution(context, path, "2.Ввод номера#3. Прием денег#e21#o2.x1",
"o1.z12");
                        return new StateMachineConfig("3. Прием денег");

                    }

                    // transition not found
                    return config;
                default:

                    // transition not found
                    return config;
            }

        case _3__Прием_денег:


            switch (e) {
                case e22:

                    // 3. Прием денег->2.Ввод номера e22[true]/o3.z31

                    fireTransitionCandidate(context, path, "3. Прием денег", event, "3. Прием денег#2.Ввод
номера#e22#true");
```

```
                fireTransitionFound(context, path, "3. Прием денег", event, "3. Прием денег#2.Ввод
номера#e22#true");

                fireBeforeOutputActionExecution(context, path, "3. Прием денег#2.Ввод номера#e22#true",
"o3.z31");

                o3.z31(context);

                fireAfterOutputActionExecution(context, path, "3. Прием денег#2.Ввод номера#e22#true",
"o3.z31");

                fireComeToState(context, path, "2.Ввод номера");

                // 2.Ввод номера [o1.z12]
                fireBeforeOutputActionExecution(context, path, "3. Прием денег#2.Ввод номера#e22#true",
"o1.z12");

                o1.z12(context);

                fireAfterOutputActionExecution(context, path, "3. Прием денег#2.Ввод номера#e22#true",
"o1.z12");
                return new StateMachineConfig("2.Ввод номера");


            case e21:

                // 3. Прием денег->3. Прием денег e21[!o3.x31]/

                fireTransitionCandidate(context, path, "3. Прием денег", event, "3. Прием денег#3. Прием
денег#e21#!o3.x31");

                if (!isInputActionCalculated(calculatedInputActions, _o3_x31)) {

                    fireBeforeInputActionExecution(context, path, "3. Прием денег#3. Прием
денег#e21#!o3.x31", "o3.x31");

                    o3_x31 = o3.x31(context);

                    fireAfterInputActionExecution(context, path, "3. Прием денег#3. Прием
денег#e21#!o3.x31", "o3.x31", new Boolean(o3_x31));
                }


                if (!o3_x31) {

                    fireTransitionFound(context, path, "3. Прием денег", event, "3. Прием денег#3. Прием
денег#e21#!o3.x31");


                    fireComeToState(context, path, "3. Прием денег");

                    // 3. Прием денег [o1.z12]
                    fireBeforeOutputActionExecution(context, path, "3. Прием денег#3. Прием
денег#e21#!o3.x31", "o1.z12");

                    o1.z12(context);

                    fireAfterOutputActionExecution(context, path, "3. Прием денег#3. Прием
денег#e21#!o3.x31", "o1.z12");
                    return new StateMachineConfig("3. Прием денег");
```

```
                    }
                    // 3. Прием денег->4.Подтверждение e21[o3.x31]/o3.z31

                    fireTransitionCandidate(context, path, "3. Прием денег", event, "3. Прием
денег#4.Подтверждение#e21#o3.x31");


                    if (o3_x31) {

                        fireTransitionFound(context, path, "3. Прием денег", event, "3. Прием
денег#4.Подтверждение#e21#o3.x31");

                        fireBeforeOutputActionExecution(context, path, "3. Прием
денег#4.Подтверждение#e21#o3.x31", "o3.z31");

                        o3.z31(context);

                        fireAfterOutputActionExecution(context, path, "3. Прием
денег#4.Подтверждение#e21#o3.x31", "o3.z31");

                        fireComeToState(context, path, "4.Подтверждение");

                        // 4.Подтверждение [o1.z13]
                        fireBeforeOutputActionExecution(context, path, "3. Прием
денег#4.Подтверждение#e21#o3.x31", "o1.z13");

                        o1.z13(context);

                        fireAfterOutputActionExecution(context, path, "3. Прием
денег#4.Подтверждение#e21#o3.x31", "o1.z13");
                        return new StateMachineConfig("4.Подтверждение");

                    }

                    // transition not found
                    return config;
                default:

                    // transition not found
                    return config;
            }

        case _5_Уведомление:


            switch (e) {
                case e21:

                    // 5.Уведомление->1.Приветствие e21[true]/o2.z26

                    fireTransitionCandidate(context, path, "5.Уведомление", event,
"5.Уведомление#1.Приветствие#e21#true");


                    fireTransitionFound(context, path, "5.Уведомление", event,
"5.Уведомление#1.Приветствие#e21#true");

                    fireBeforeOutputActionExecution(context, path, "5.Уведомление#1.Приветствие#e21#true",
"o2.z26");

                    o2.z26(context);
```

```
                    fireAfterOutputActionExecution(context, path, "5.Уведомление#1.Приветствие#e21#true",
"o2.z26");

                    fireComeToState(context, path, "1.Приветствие");

                    // 1.Приветствие [o1.z11]
                    fireBeforeOutputActionExecution(context, path, "5.Уведомление#1.Приветствие#e21#true",
"o1.z11");

                    o1.z11(context);

                    fireAfterOutputActionExecution(context, path, "5.Уведомление#1.Приветствие#e21#true",
"o1.z11");
                    return new StateMachineConfig("1.Приветствие");


                default:

                    // transition not found
                    return config;
                }

            case _4_Подтверждение:


                switch (e) {
                  case e22:

                    // 4.Подтверждение->3. Прием денег e22[true]/o3.z32

                    fireTransitionCandidate(context, path, "4.Подтверждение", event, "4.Подтверждение#3.
Прием денег#e22#true");


                    fireTransitionFound(context, path, "4.Подтверждение", event, "4.Подтверждение#3. Прием
денег#e22#true");

                    fireBeforeOutputActionExecution(context, path, "4.Подтверждение#3. Прием
денег#e22#true", "o3.z32");

                    o3.z32(context);

                    fireAfterOutputActionExecution(context, path, "4.Подтверждение#3. Прием денег#e22#true",
"o3.z32");

                    fireComeToState(context, path, "3. Прием денег");

                    // 3. Прием денег [o1.z12]
                    fireBeforeOutputActionExecution(context, path, "4.Подтверждение#3. Прием
денег#e22#true", "o1.z12");

                    o1.z12(context);

                    fireAfterOutputActionExecution(context, path, "4.Подтверждение#3. Прием денег#e22#true",
"o1.z12");
                    return new StateMachineConfig("3. Прием денег");


                case e21:

                    // 4.Подтверждение->5.Уведомление e21[true]/
```

```
                    fireTransitionCandidate(context, path, "4.Подтверждение", event,
"4.Подтверждение#5.Уведомление#e21#true");


                    fireTransitionFound(context, path, "4.Подтверждение", event,
"4.Подтверждение#5.Уведомление#e21#true");


                    fireComeToState(context, path, "5.Уведомление");

                    // 5.Уведомление [o1.z14]
                    fireBeforeOutputActionExecution(context, path,
"4.Подтверждение#5.Уведомление#e21#true", "o1.z14");

                    o1.z14(context);

                    fireAfterOutputActionExecution(context, path, "4.Подтверждение#5.Уведомление#e21#true",
"o1.z14");
                    return new StateMachineConfig("5.Уведомление");


                default:

                    // transition not found
                    return config;
                }

            default:
              throw new EventProcessorException("Incorrect stable state [" + config.getActiveState() + "] in state
machine [A1]");
            }
        }
    }

    //o2.x1
    private static final int _o2_x1 = 0;
    //o3.x31
    private static final int _o3_x31 = 1;

  }

  private static boolean isInputActionCalculated(BitSet calculatedInputActions, int k) {
    boolean b = calculatedInputActions.get(k);

    if (!b) {
      calculatedInputActions.set(k);
    }

    return b;
  }

}
```