Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

С.Ю. Канжелев, А.А. Шалыто

Моделирование кнопочного телефона с использованием SWITCH-технологии Вариант 2

Проектная документация

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

Санкт-Петербург
2004

Оглавление

Введение

В данной работе, также как и в работе [1], с помощью SWITCH-технологии реализована модель кнопочного телефона и автоматической телефонной станции (АТС).

Отличия от работы [1] состоят в следующем:

- введены дополнительные функции телефона – способность принимать звонки, возможность донабора номера после поднятия трубки;
- новый метод взаимодействия автоматов, реализующих телефон и АТС.

Этот метод назван **«отложенный вызов автомата»**. Он, в отличие от подхода, изложенного в работе [2], позволяет:

- избавиться от *реентерабельности* автоматных функций (повторного вызова каждой из них до завершения ее работы) на уровне реализации вызова;
- осуществлять параллельную работу автоматов в разных потоках;
- обрабатывать внешние события, приходящие из параллельных процессов, после их предварительного запоминания в очереди, сформированной для каждого автомата.

При этом нотация графов переходов, предложенная в работе [3] и использующаяся в работе [1], сохраняется.

Отложенный вызов автомата характеризуется следующими особенностями:

- не разделяются внутренние и внешние события, как это сделано в работах [4,5];
- все события ставятся в соответствующую каждому автомату очередь и обрабатываются последовательно;
- допускается вызов автоматом самого себя. При этом реентерабельность отсутствует – автомат повторно не запускается до завершения своей работы.

Тестирование программы происходило с помощью автоматически получаемых логов, а также на основе анализа графов переходов автоматов. Тестирование на основе анализа графов переходов автоматов выполнялось, в том числе, и при помощи построенных по графам переходов диаграмм последовательности.

Отметим, что построение диаграмм последовательности по графам переходов отличает предлагаемый подход от классического [6], в котором эти два типа диаграмм обычно формально не связаны.

Проект реализован в виде апплета на языке *Java*.

1. Постановка задачи

Цель работы – создание модели кнопочного телефона, способного, взаимодействуя с АТС, устанавливать соединение с другим кнопочным телефоном, и, соответственно, принимать звонки.

Интерфейс кнопочного телефона (рис. 1) состоит из:

- номера данного телефона;
- дисплея, отображающего набранную к данному моменту часть номера;
- стандартной кнопочной клавиатуры (цифры от 0 до 9);

- кнопки «Сброс», обозначенной буквой «С». Она обеспечивает сброс набранного номера без опускания трубки;
- кнопки «Поднять трубку»;
- строки, отображающей текущее состояние кнопочного телефона.

АТС должна обеспечивать корректное «соединение» телефонов для разговора. Интерфейс АТС (рис. 2) состоит из строки состояния АТС (на рисунке указано состояние «Набор номера») и строки полученной к данному моменту от телефона части номера.

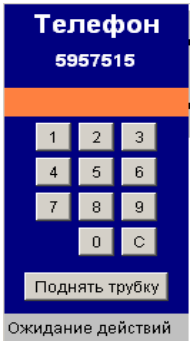

Рис. 1. Внешний вид кнопочного телефона

Рис. 2. Внешний вид АТС

2. Диаграмма классов

На рис. 3 приведена диаграмма классов для рассматриваемой задачи.

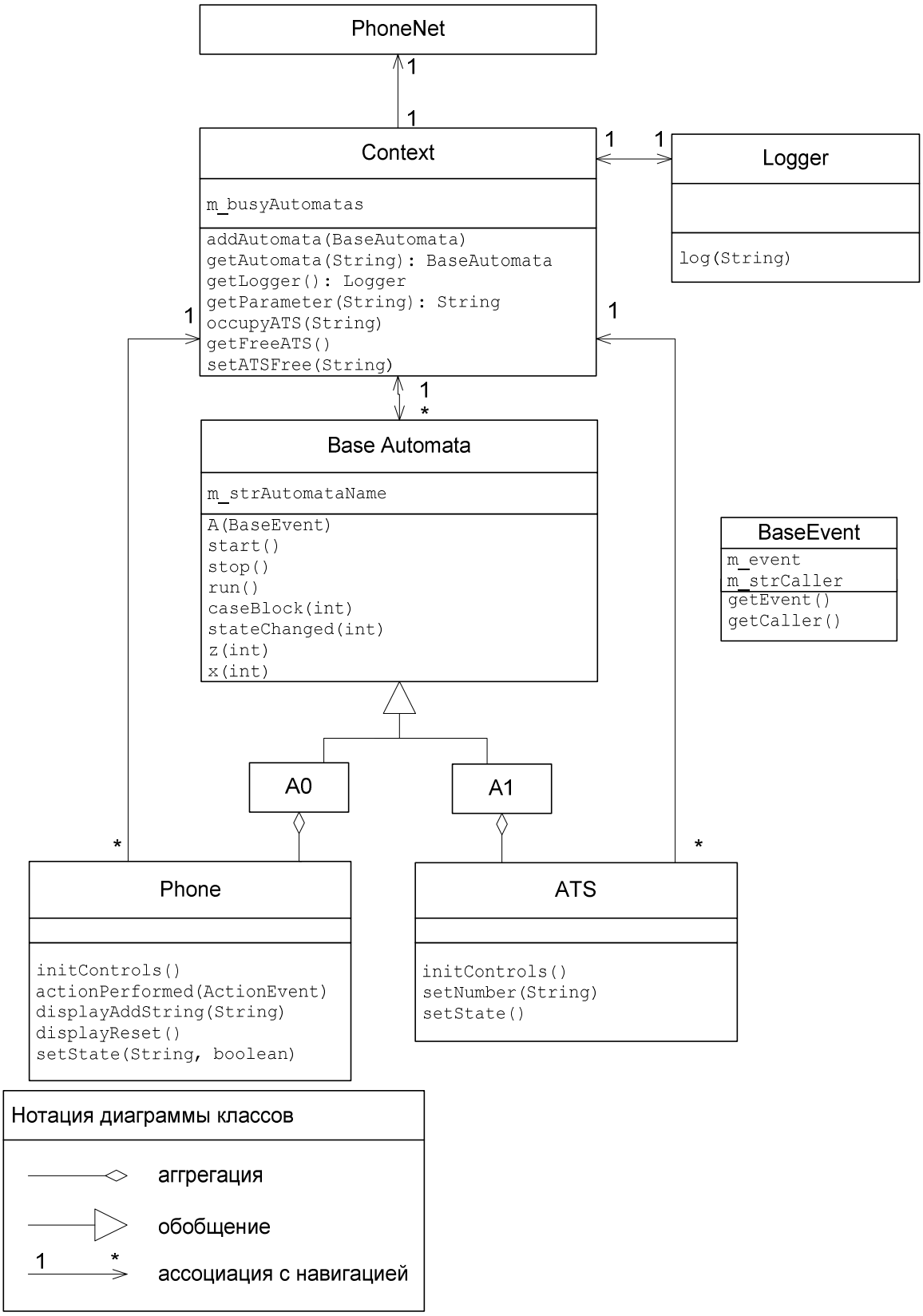

Рис. 3. Диаграмма классов

На рис. 3 используется, как и в книге [6], следующие обозначения:

- класс представляется прямоугольником. Если прямоугольник разделен на три части, то эти части означают название, атрибуты и операции класса. Иначе, указывается только название класса;
- стрелка с ромбом на конце означает агрегацию классов. Класс у ромба включает в себя класс с противоположной стороны стрелки;
- стрелка с большим треугольником на конце означает обобщение. Класс у треугольника является обобщением классов на другом конце стрелки;
- обычная стрелка означает ассоциацию с навигацией. Класс, находящийся с противоположной от стрелки стороны, должен иметь ссылку на класс, к которому направлена стрелка. Цифры показывают количество ассоциированных объектов, звездочка * означает любое число от нуля до бесконечности. Например, на рис. 3 с классом Context может быть ассоциировано несколько объектов `BaseAutomata`, в то время как класс `Context` один для всех классов `BaseAutomata`.

На диаграмме представлены только основные отношения между классами. Более подробную информацию о составе и отношениях между классами можно получить из исходного кода программы (Приложение 1) и документа, автоматически сформированного программой *JavaDoc* (Приложение 2). Также на диаграмме не представлены библиотечные классы, от которых происходит наследование.

Центральным классом программы является класс `Context`. Он обеспечивает доступ к автоматным классам проекта (`getAutomata`), логирование (`getLogger`) и извлечение параметров из HTML-файла (`getParameter`). Также он содержит информацию о занятых АТС (`getFreeATS`, `occupyATS`, `setATSFree`) и телефонах (`m_busyAutomatas`). Все классы, кроме `PhoneNet`, должны содержать ссылку на класс `Context`.

Автоматы в программе наследуются от класса `BaseAutomata`. Они однозначно задаются своим именем (`m_strAutomataName`). Для автоматов, реализующих телефон, в качестве имени берется номер телефона. Это имя используется при взаимодействии автоматов.

Автоматы взаимодействуют, передавая друг другу объекты класса `BaseEvent`. Этот класс содержит название вызвавшего абонента и номер посылаемого события.

Интерфейсы автоматов, классы ATS и Phone содержат исключительно методы управления интерфейсом (`displayReset`, `setState`, …). Кроме того, интерфейс телефона обеспечивает трансляцию автомату действий пользователя (`actionPerformed`).

Класс `PhoneNet` наследуется от системного класса `Applet`. Он инициализирует все объекты и является контейнером для классов, реализующих интерфейс автоматов.

3. Реализация

3.1. Базовый автоматный класс (`BaseAutomata`)

3.1.1. Наследование

Автоматные классы в проекте наследуются от класса `BaseAutomata` (рис. 3). Этот класс, в частности, обеспечивает логирование прихода событий, выполнение действий и проверку

переменных, используя класс `Logger`. Базовый класс также реализует **отложенный вызов автомата**.

Наследник класса `BaseAutomata` должен инициализировать массив названий состояний, событий, переменных и действий автомата. Также в классе-наследнике, должны быть реализованы абстрактные методы:

- `protected abstract void caseBlock(int p_Event);`
- `protected abstract void stateChanged(int p_intOldState);`

Метод `caseBlock` должен содержать логику работы автомата и строиться по методике, описанной в работе [2].

Метод `stateChanged` служит для удобства управления интерфейсом. Он вызывается автоматически при каждом изменении состояния автомата, что является удобным, если каждое состояние автомата, как в настоящей работе, имеет свое представление в интерфейсе. Например, в данной программе, удобно было использовать этот метод для изменения строки, указывающей текущее состояние автоматов.

В проекте все логирование вынесно в класс `BaseAutomata`. Это достигается размещением в этом классе метода `A()` и методов `z()` и `x()`. Метод `A()` будет описан в разд. 3.1.2 Рассмотрим методы `z` и `x.`

Эти методы в качестве параметров принимают номер выходной (z) или входной (x) переменных соответственно. Они логируют вычисление выходных и входных переменных, а затем вычисляют их.

Например, пусть в автомате `A0` реализована выходная переменная `z11()`. Для того, чтобы вычислить ее, автомат `A0` – наследник класса BaseAutomata – должен выполнить метод `z(11)`.

3.1.2. Реализация отложенного вызова

Класс `BaseAutomata` содержит метод `A(int e)`. При вызове этого метода происходит постановка в очередь пришедшего события `e`. При этом если класс находился в состоянии ожидания события, то в новом потоке начинается обработка событий из очереди, используя логику работы конкретного автомата, реализованную в функции `stateChanged.` Обработка событий из очереди заканчивается после обработки последнего события, а автомат переходит в состояние ожидания события.

Заметим, что при такой реализации невозможно утверждать, что вызванный автомат закончил обработку переданного события. Если в работе [2] использовался один поток и вызов автоматной функции останавливал выполнение действий до тех пор, пока событие не было обработано вызванным автоматом, то в данном случае вызов автомата не останавливает выполнение действий. При этом событие может быть обработано через неопределенное время, что зависит от свойств операционной системы в части организации переключений между потоками.

Также отметим, что каждый автомат при такой реализации выполняется в своем потоке. Постановка события в очередь и извлечение события из очереди синхронизирцются с помощью блокировки объекта очереди.

3.2. Класс Телефон (A0)

3.2.1. Словесное описание

Телефон реализуется на базе класса `BaseAutomata`. Служит для набора номера и приема звонков.

3.2.2. Схема связей

На рис. 4 приведена схема связей автомата A0.

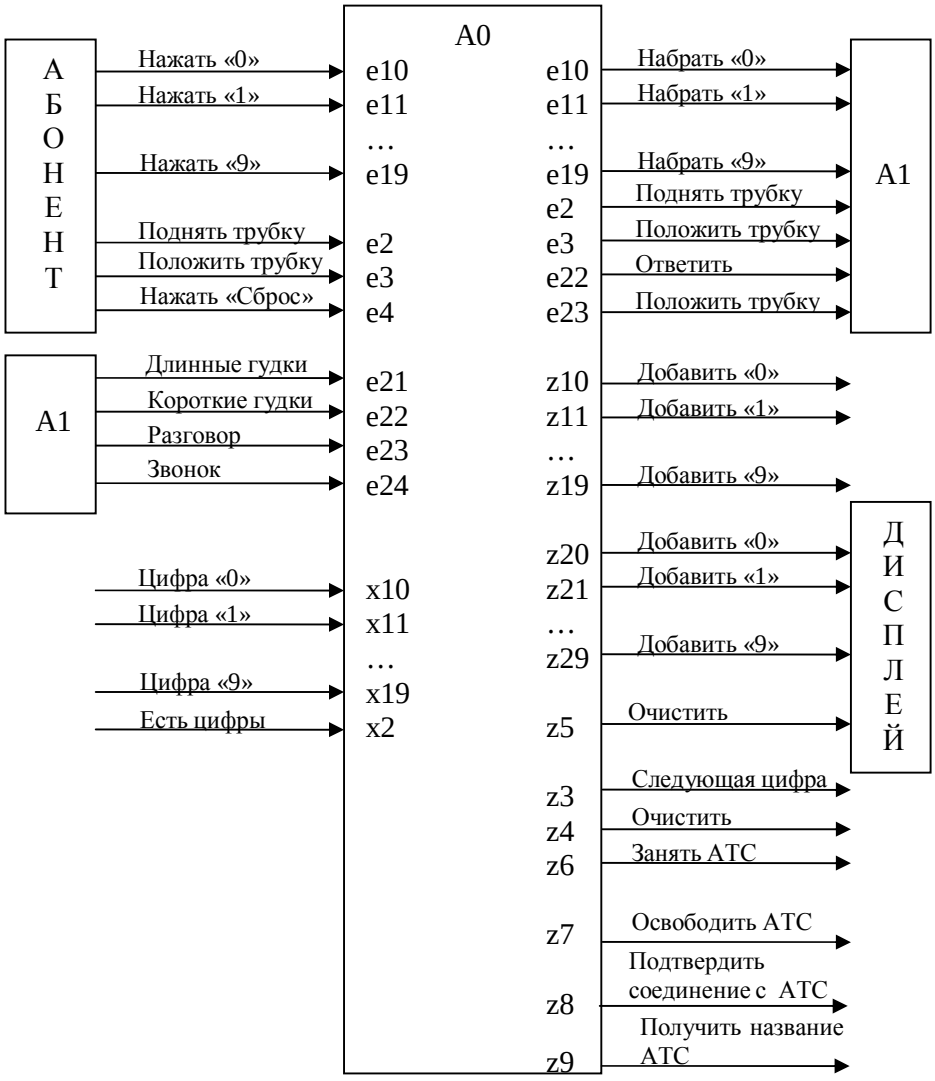

Рис. 4. Схема связей автомата A0

3.2.3. Граф переходов

На рис. 5 приведен граф переходов автомата A0.

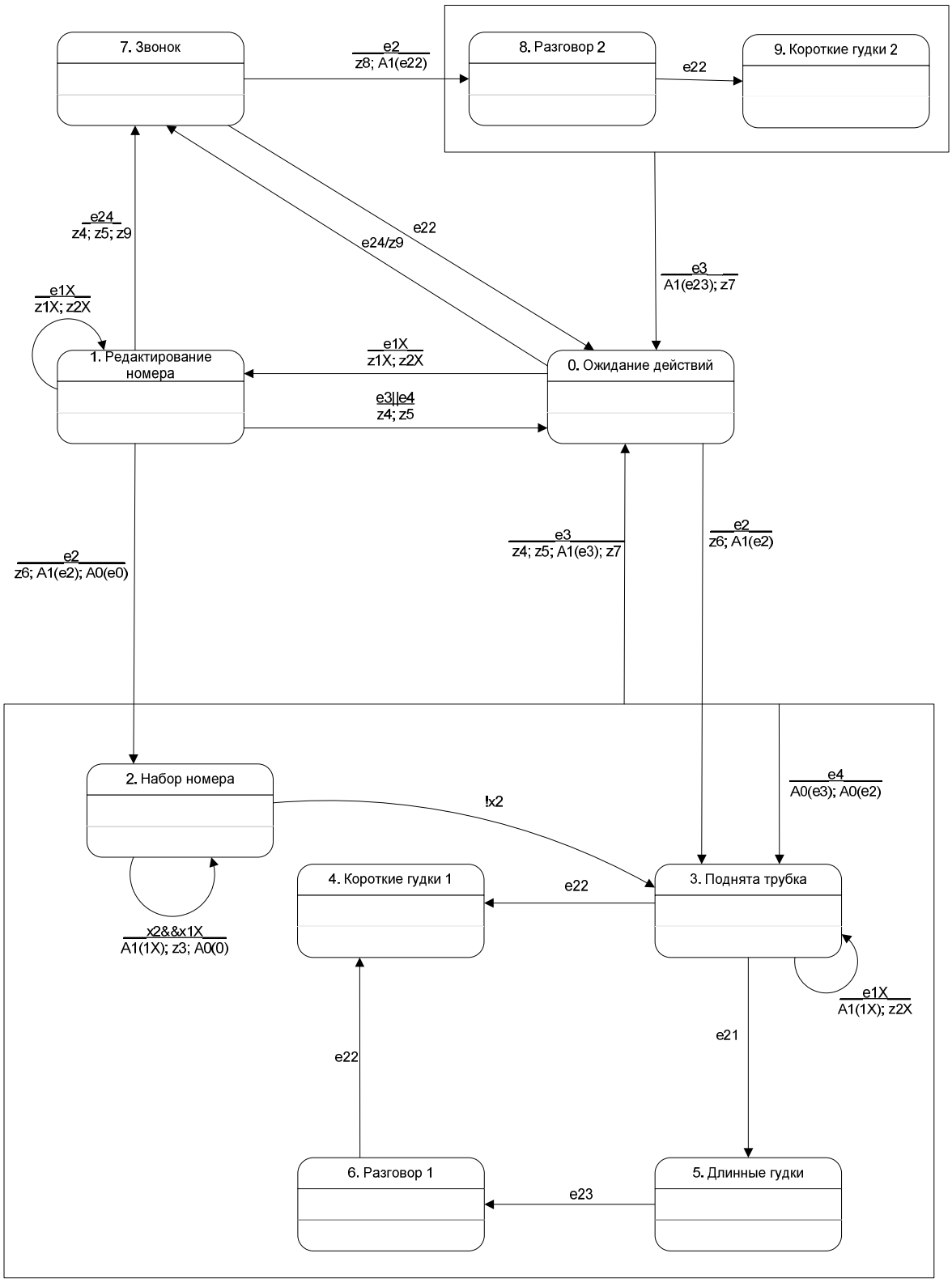

Рис. 5. Граф переходов автомата A0

На этом графе введены параметризованные дуги – дуги, у которых в номере события или действия содержится буква X. Такие дуги при реализации следует представлять как десять дуг, на каждой из которых буква X заменена цифрой от 0 до 9 соответственно.

Аналогичные обозначения встречаются и на графе переходов класса A1.

3.3. Класс АТС (A1)

3.3.1. Словесное описание

АТС реализуется на базе класса `BaseAutomata`. Служит для выделения требуемого абонента (по набираемому номеру) и соединению связи с ним.

Класс `A1` содержит статические переменные – списки свободных и занятых АТС и телефонов.

3.3.2. Схема связей

На рис. 6 приведена схема связей автомата A1.

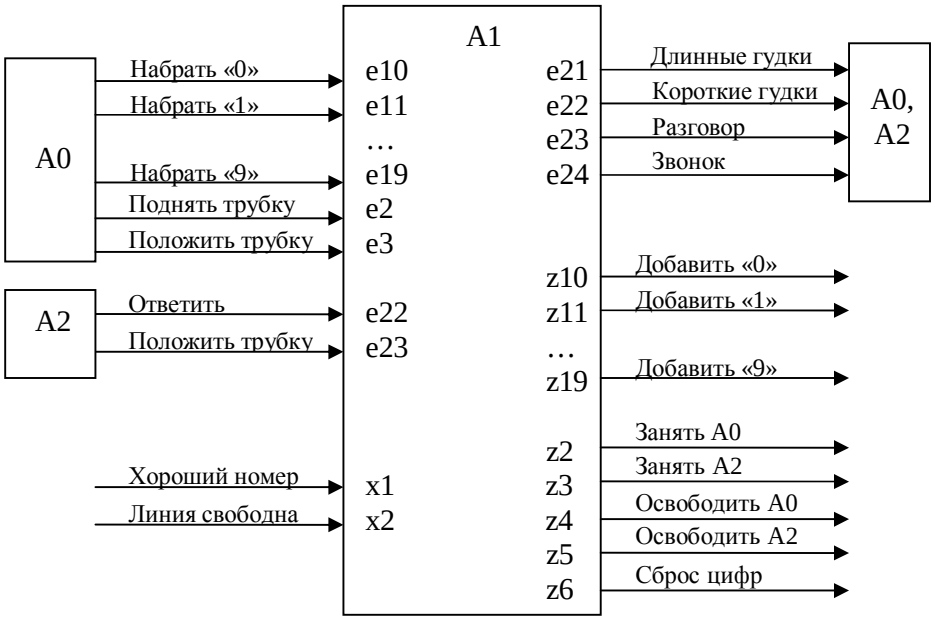

Рис. 6. Схема связей автомата A1

В эту схему введен автомат A2, являющийся объектом того же класса, что и автомат A0. Он предназначен для различения вызывающего и вызываемого абонента.

3.3.3. Граф переходов

На рис. 7 приведен граф переходов автомата A1.

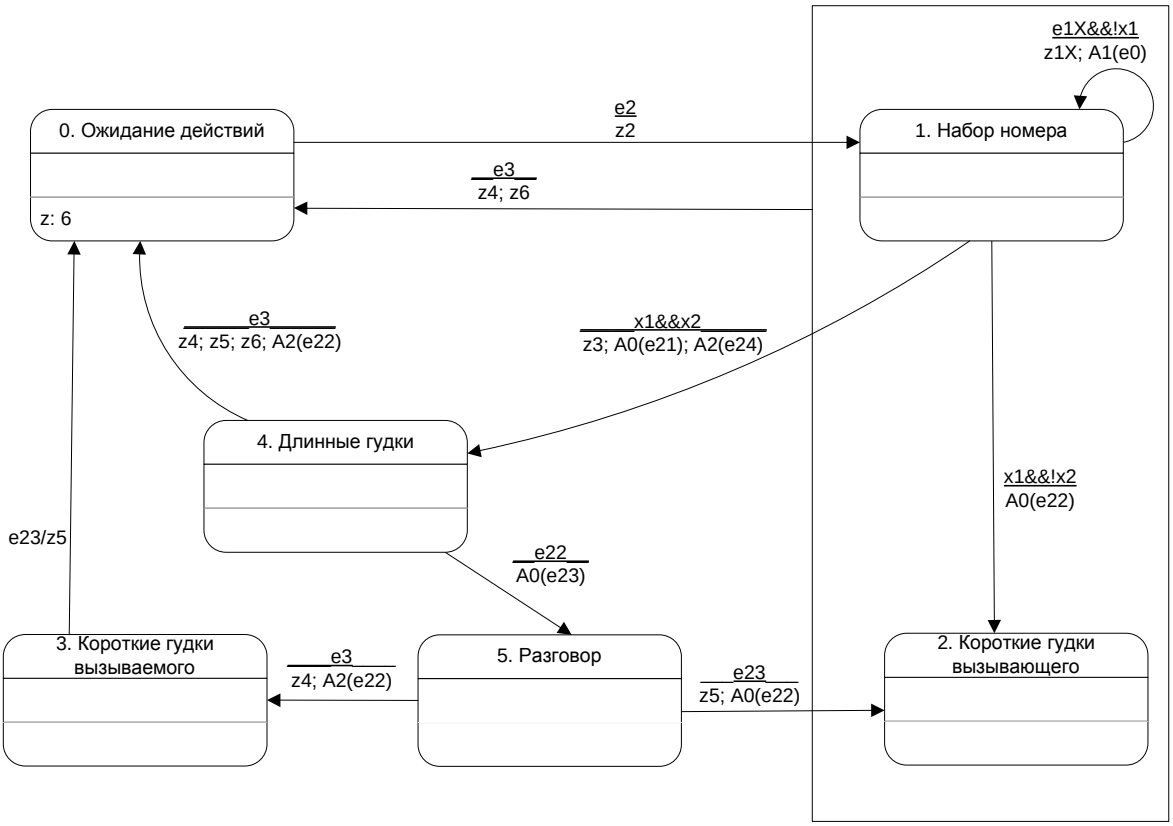

Рис. 7. Граф переходов автомата A1

На этом графе введены параметризованные дуги – дуги, у которых в номере события или действия содержится буква X. Значение таких дуг объясняется в разд. 3.2.3.

3.4. Класс `BaseEvent`

Класс используется при автоматном взаимодействии. Объект этого класса посылается автоматами при вызове одним автоматом другого. Класс содержит название вызывающего автомата и номер события.

3.5. Класс `Context`

Класс обеспечивает:

- логирование;
- извлечение параметров (например, названия состояний) из HTML-файла;
- доступ к автоматным классам.

Также класс `Context` содержит списки занятых телефонов и АТС. Причем, для занятых телефонов используется список занятых телефонов, а для АТС – состояние каждой АТС.

АТС может находиться в трех состояниях: *free* (АТС не занята), *busy* (АТС занята одним абонентом), *very busy* (АТС обеспечивает разговор двух абонентов). Для получения свободной АТС используется класс `getFreeATS()`. Данный метод занимает АТС и возвращает ее название. Также есть два метода для занятия и освобождения АТС. Это методы `occuupyATS()` и `freeATS()`. Они переводят АТС в следующее или

предыдущее состояние. Например, метод `occupyATS` переведет АТС, находившуюся в состоянии *free*, в сотояние *busy*. Повторный вызов метода переведет АТС в состояние *very busy*. Метод `freeATS` действует в обратном порядке.

3.6. Класс `Logger`

Класс обеспечивает логирование. Он содержит методы вывода сообщений в *log*-файл. В настоящей реализации информация выводится на консоль.

Пример *log*-файла приведен в приложении 3.

3.7. Класс `Phone`

Класс реализует интерфейс телефона. Содержит методы управления интерфейсом, а также регистрирует нажатия кнопок пользователем и сообщает об этих событиях классу A1.

3.8. Класс `ATS`

Класс реализует интерфейс АТС. Содержит методы управления интерфейсом.

3.9. Класс `PhoneNet`

Наследуется от системного класса `Applet`. Инициализирует все классы проекта.

4. Тестирование автоматов

4.1. Необходимость тестирования

В связи с тем, что автоматы чаще всего строятся эмпирчески, после их построения возникает вопрос об их корректной работе. Частично тестирование можно произвести с помощью автоматически созданных программой логов. Однако такое тестирование возможно только после реализации автоматов. Невозможно также утверждать, что после него весь код будет «покрыт» - произойдет тестирование всех возможных вариантов работы. В данной программе такой способ тестирования дополнительно осложнен тем, что автоматы, как отмечалось в разд. 3.1.2, выполняются каждый в своем потоке. Поэтому их логи будут перемешаны.

В данной работе дополнительно произведено тестирование:

- взаимодействия автоматов;
- корректной последовательности занятия – освобождения автоматов и АТС.

4.2. Способ тестирования

Работа автоматов в данной работе начинается с состояния «Ожидание действий». После каждого звонка автоматы возвращаются в то же состояние. Это соображение определило

способ тестирования – выделялись и анализировались циклы, проходящие через состояние «Ожидание действий».

При тестировании проверялось:

- занятие – освобождение АТС;
- занятие – освобождение телефонов;
- взаимодействие автоматов;

Каждая разновидность тестирования проводилась в три этапа:

- выбор значимых для этой разновидности события, выходные переменные и вызовы автоматов;
- построение всех возможных циклов, проходящих через состояние «Ожидание действий». При этом выделялись выбранные ранее события, выходные переменные и вызовы автоматов;
- анализ полученных данных.

Второй этап тестирования автоматизирован

4.3. Построение циклов для автоматов

Построение всевозможных циклов, проходящих через состояние «Ожидание действий», для автоматов осложняется тем, что в графе могут присутствовать внутренние циклы – циклы, не проходящие через указанное состояние. Наличие таких циклов делает невозможным обычный поиск в глубину. Такой поиск, в зависимоти от реализации, зациклится на внутреннем цикле, либо не учтет его. Возникла проблема поиска и выделения внутренних циклов.

Был реализован следующий алгоритм, решающий проблему выделения внутренних циклов:

- находятся все циклы, проходящие через нулевую вершину (нулевой вершиной называем вершину, отвечающую состоянию «Ожидание действий») и проходящие по каждой вершине не более одного раза;
- для каждой вершины из каждого пути находятся циклы, проходящие через данную вершину, проходящие по каждой вершине лишь по одному разу и не проходящие через нулевую вершину;
- все полученные на втором этапе вершины обрабатываются аналогичным образом. Только теперь для каждой вершины будет две «запрещенные» вершины: нулевая и та, в цикл через которую она входит.

4.4. Описание циклов, выделяемых в графах переходов

Для построения циклов была написана программа, использующая в качесте входных данных граф переходов автомата в формате *MS Visio*, и список значимых событий, действий и вызовов автоматов. На выходе программы получается список циклов в следующем формате:

- входные события обозначаются как «`->eXX`», где XX – номер события в графе переходов;
- автоматные вызовы обозначаются как «`<-AY(eXX)`», где Y – номер вызываемого автомата, а XX имеют тот же смысл, что и в обозначении входного события;
- действия обозначаютя как «`!z6`».

- подциклы – циклы, находящиеся внутри данного, заключаются в скобки.

Например, запись

```
(<-A1(e1X))!z6<-e2(<-A1(e1X))->e21<-A1(e3)!z7
```

описывает следующий цикл:

- вызов автомата A1 с событием e1X произвольное количество раз;
- выполнение действия z6;
- вызов автомата A1 с событием e2;
- вызов автомата A1 с событием e1X произвольное количество раз;
- приход события e21;
- вызов автомата A1 с событием e3;
- выполнение действия z7.

4.5. Тестирование занятия – освобождения АТС и телефонов

4.5.1. Тестирование занятия – освобождения АТС

Получение названия, занятие и освобождение АТС производится действиями z6, z7, z8, z9 автомата A0. Действие z6 получает название АТС и занимает ее, действие z8 только занимает АТС, действие z7 освобождает АТС, и, наконец, действие z9 получает название АТС в случае, когда происходит вызов абонента, а название АТС было выбрано вызывающим абонентом.

Необходимо, чтобы последовательность действий с АТС имела вид:

- получение названия АТС;
- занятие АТС;
- освобождение АТС.

Для тестирования были построены всевозможные циклы на графе переходов автомата A0, проходящие через состояние «Ожидание действий». Отмечались только действия z6, z7, z8 и z9.

```
!z6!z7
!z9
!z9!z8!z7
```

Анализ полученных данных показывает, что автомат реализует требуемые последовательности действий с АТС и никакие другие.

4.5.2. Тестирование занятия – освобождения телефонов

Занимает и освобождает телефоны автомат A1. Производится это с помощью действий z2, z3, z4 и z5. В соответствии с описаниями действий, приведенных в пункте 3.3.2, действия должны выполнятся парно: z2-z4 и z3-z5. Пары означают занятие-освобождение вызывающего и вызываемого абонентов соответственно.

Для тестирования были построены всевозможные циклы на графе переходов автомата A1, проходящие через состояние «Ожидание действий». Отмечались только действия z2, z3, z4 и z5.

```
!z2!z4
!z2!z3!z4!z5
!z2!z3!z5!z4
```

Анализ полученных данных показывает, что автомат реализует требуемые последовательности действий и никакие другие последовательности невозможны.

4.6. Построение диаграмм последовательности

Взаимодействие автоматов A0 и A1 тестировалось следующим образом:
рассматривается граф переходов автомата A1;

- выбираются события, приходящие автомату A1 от автомата A0 и введенного дополнительно (разд. 3.3.2) автомата A2. Также выбираются все вызовы автоматов A0 и A2 автоматом A1;
- строятся все циклы, проходящие через состояние «Ожидание действий».

В результате таких действий получается следующий набор циклов:

```
->e2(->e1X)->e3
->e2(->e1X)<-A0(e22)->e3
->e2(->e1X)<-A0(e21)<-A2(e24)->e3<-A2(e22)
->e2(->e1X)<-A0(e21)<-A2(e24)->e22<-A0(e23)->e23<-A0(e22)->e3
->e2(->e1X)<-A0(e21)<-A2(e24)->e22<-A0(e23)->e3<-A2(e22)->e23
```

Эти циклы могут рассматриваться как всевозможные схемы взаимодействия автоматов – как всевозможные диаграммы последовательности. Действительно, в этих циклах описывается в какой последовательности происходит передача событий между автоматами. Например, рассмотрим второй цикл:

```
->e2(->e1X)<-A0(e22)->e3
```

В нем описывается следующая последовательность взаимодействия, сформированная с учетом схемы связей автомата A1 (из этой схемы, например, следует, что событие e2 поступает от автомата A0):

- приход события e2 от автомата A0;
- приход произвольного количества событий e1X от автомата A0;
- вызов автоматом A1 автомата A0 с событием e22;
- приход события e3 от автомата A0;

Анализ всех схем взаимодействия позволяет построить всевозможные диаграммы последовательности. Подробнее диаграммы последовательности описываются в следующем разделе.

4.6.1. Описание диаграммы последовательности

Для описания функций взаимодействия телефонов и АТС используется **диаграмма последовательности** [6]. Объекты на такой диаграмме изображаются прямоугольниками на вершине вертикальной пунктирной линии. Эта вертикальная линия называется **линией жизни** объекта. Каждое сообщение от одного объекта к другому изображается стрелкой

между линиями, а вызов собственного метода объекта – круглой стрелкой, начинающейся и заканчивающейся на одной вертикальной линии.

Как отмечалось выше, диаграммы последовательности могут быть построены из анализа полученных в разд. 4.6 данных. Рассмотрение графа переходов одного автомата гарантирует последовательное выполнение действий.

4.6.2. Диаграммы последовательности

Изобразим все построенные циклы в виде диаграмм последовательности. Тестирование заключается в проверке корректности каждой диаграммы и задании имени для нее.

Неполный набор номера

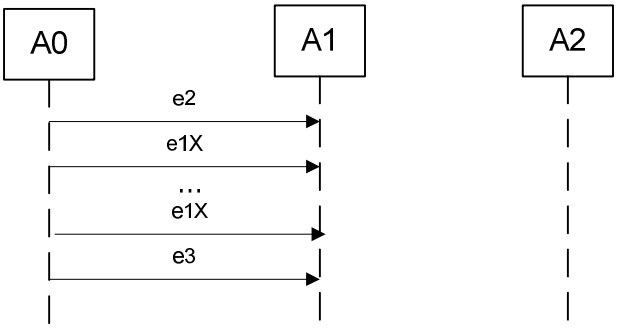

Рис. 8. Неполный набор номера

Абонент занят

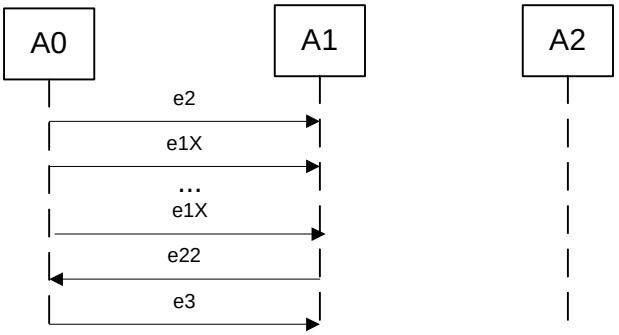

Рис. 9. Абонент занят

Абонент не отвечает

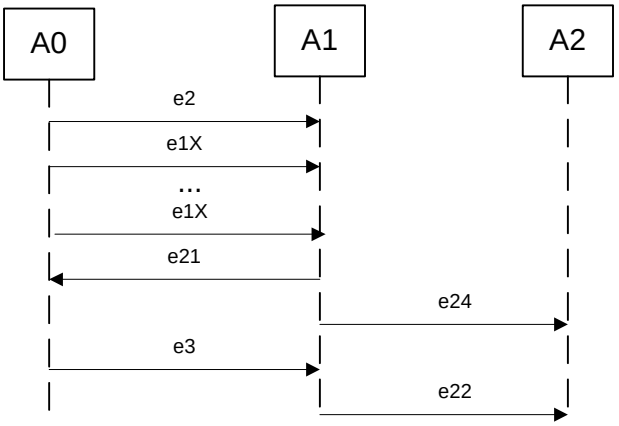

Рис. 10. Абонент не отвечает

Окончание разговора вызываемым абонентом

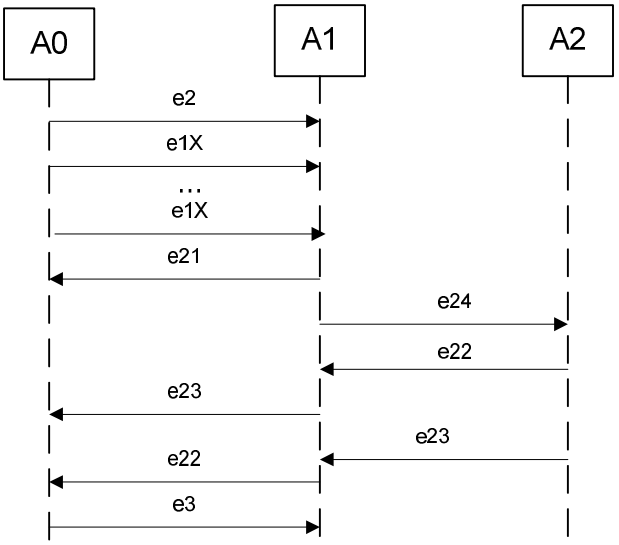

Рис. 11. Окончание разговора вызываемым абонентом

Абонент свободен

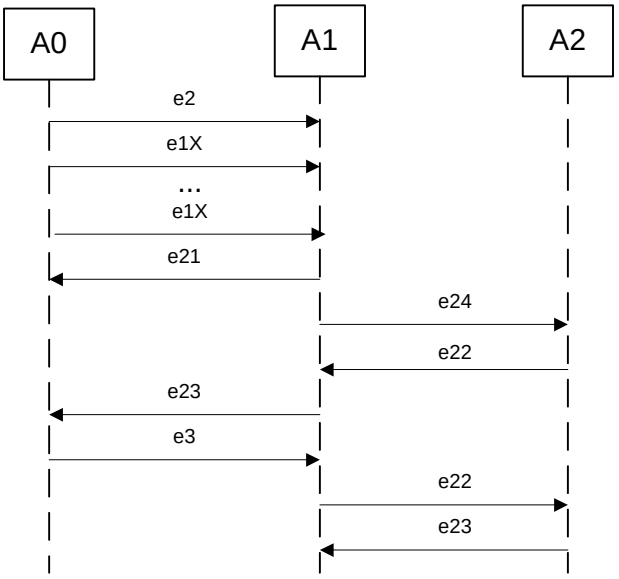

Рис. 12. Окончание разговора вызывающим абонентом

5. Заключение

В работе [7] рассмотрены методы совместного использования объектного и автоматного подходов. В настоящей работе представлен еще один метод взаимодействия автоматов в объектно-ориентированном программировании.

Достоинство предложенного метода состоит в устранении реентерабельности на уровне реализации автоматов с минимальными «потерями». Например, при устранении ее методом блокирования разделяемого ресурса (семафора), возникает опасность взаимной блокировки и полной остановки системы.

В предложенной реализации не происходит блокировки системы, и каждый автомат может рассматриваться как атомарная единица системы с независимыми от других автоматов входными и выходными воздействиями – автоматы связаны и как бы несвязаны одновременно.

Это обеспечивается за счет того, что одной из функций, реализуемых введенным базовым классом для реализации автоматов, является постановка приходящих автомату событий в очередь и последовательная их обработка в отдельном потоке, сформированном для этого автомата. Таким образом, при использовании данного метода применяется число потоков равное числу автоматов.

Недостаток метода состоит в невозможности утверждать, что вызванный автомат закончил обработку переданного события. Если в работе [2] использовался один поток и вызов автоматной функции останавливал выполнение действий до тех пор, пока событие не было обработано вызванным автоматом, то в данном случае вызов автомата не останавливает выполнение действий. При этом событие может быть обработано через неопределенное время, что определяется свойствами операционной системы в части организации переключений между потоками.

Также было произведено тестирование работы автоматов на основе анализа графа переходов этих автоматов. Тестирование на основе анализа графов переходов автоматов выполнялось, в том числе, и при помощи построенных по графам переходов диаграмм последовательности.

Отметим, что построение диаграмм последовательности по графам переходов отличает предлагаемый подход от классического [6], в котором эти два типа диаграмм обычно формально не связаны.

Литература

1. *Мясников А.И.* Моделирование кнопочного телефона с использованием SWITCH-технологии. СПбГИТМО (ТУ), 2003, http://is.ifmo.ru, раздел «Проекты».
2. *Шалыто А.А., Туккель Н.И.* Реализация автоматов при программировании событийных систем //Программист. 2002. №4. http://is.ifmo.ru, раздел «Статьи».
3. *Шалыто А.А., Туккель Н.И.* SWITCH-технология - автоматный подход к созданию программного обеспечения "реактивных" систем //Программирование. 2001. № 5. http://is.ifmo.ru, раздел «Статьи».
4. *Гуисов М.И., Кузнецов А.Б., Шалыто А.А.* Интеграция механизма обмена сообщениями в Switch-технологию. СПбГИТМО (ТУ), 2003, http://is.ifmo.ru, раздел «Проекты».
5. *Гуисов М.И., Кузнецов А.Б., Шалыто А.А.* Задача Д. Майхилла «Синхронизация цепи стрелков». Вариант 2. http://is.ifmo.ru, раздел «Проекты».
6. *Фаулер М., Скотт К.* UML. Основы. СПб.: Символ-Плюс, 2002.
7. *Шалыто А.А., Наумов Л.А.* Реализация автоматов в объектно-ориентированных программах //Мир ПК-Диск, 2004. №4. http://is.ifmo.ru, раздел «Статьи».

Приложение 1. Исходные коды программ

Класс BaseAutomata

```
import java.util.*;
import java.text.*;
/**
* Базовый автоматный класс. Реализует отложенный вызов автомата.<br><br>
*
* Потомки должны реализовать методы {@link #caseBlock(int p_Event)},
* {@link #stateChanged(int p_intOldState)}, Также потомки должны
* инициализировать {@link #m_strStateNames}, {@link #m_strEventNames},
* {@link #m_strActionNames}, {@link #m_strVarNames}.
*/
public abstract class BaseAutomata implements Runnable
{
  /**Контекст проекта.*/
  /**Контекст проекта.*/
  protected Context   ctx;

  /**Автомат, вызвавший текущий автомат*/
  private String m_strCaller = null;

  /**Происходит ли обработка события*/
  private boolean m_blnProcessingStarted = false;

  /**
   * Список состояний автомата.<br>
   * <b>Note:</b> Должен быть инициализирован в потомках.
   */
  protected String[]  m_strStateNames;

  /**
   * Список событий автомата.<br>
   * <b>Note:</b> Должен быть инициализирован в потомках.
   */
  protected String[]  m_strEventNames;

  /**
   * Список действий автомата.<br>
   * <b>Note:</b> Должен быть инициализирован в потомках.
   */
  protected String[]  m_strActionNames;

  /**
   * Список переменных автомата.<br>
   * <b>Note:</b> Должен быть инициализирован в потомках.
   */
  protected String[]  m_strVarNames;

  /**Уникальное в пределах контекста название автомата.*/
  protected String    m_strName;

  /**
   * Временной интервал между обработками событий.<br>
   * <b>Note:</b> Должен быть инициализирован в потомках.
   */
  protected int       m_intDelay;

  /**Текущее состояние автомата.*/
  protected int       y;
```

```
/**Очередь входных событий автомата.*/
private Vector      m_events;

/**Нить автомата. Служебная переменная для корректного запуска и остановки
            автомата.*/
private Thread      m_threadGo;

/**Строка "Пришло событие".*/
private String      m_strEventFired;

/**Строка "Начало обработки события".*/
private String      m_strStartProcess;

/**Строка "Начало конец события".*/
private String      m_strEndProcess;

/**Строка "Текущее состояние".*/
private String      m_strCurrState;

/**Строка "Новое состояние".*/
private String      m_strNewState;

/**Строка "Выполнить действие".*/
private String      m_strZ;

/**Строка "Проверить условие".*/
private String      m_strX;

/**Строка "Истина".*/
private String      m_strTrue;

/**Строка "Ложь".*/
private String      m_strFalse;

/**
 * Конструктор
 * @param p_strName  Имя автомата.
 * @param p_ctx      Контекст.
 */
public BaseAutomata(String p_strName, Context p_ctx)
{
  ctx = p_ctx;
  m_strName = p_strName;
  m_events = new Vector();


  m_strEventFired     = ctx.getParameter("EventFired_{0}.{1}");
  m_strStartProcess   = ctx.getParameter("StartProcess_{0}.{1}");
  m_strCurrState      = ctx.getParameter("CurrState_{0}.{1}");
  m_strZ              = ctx.getParameter("Z_{0}.{1}");
  m_strNewState       = ctx.getParameter("NewState_{0}.{1}");
  m_strEndProcess     = ctx.getParameter("EndProcess_{0}.{1}");
  m_strX              = ctx.getParameter("X_{0}.{1}.{2}");
  m_strTrue           = ctx.getParameter("TrueName");
  m_strFalse          = ctx.getParameter("FalseName");

  y = 0;
}
```

```
/**
 * Добавляет событие в очередь событий.
 * @param p_Event    Событие.
 */
public void A(BaseEvent p_Event)
{
  ctx.getLogger(getName()).log(MessageFormat.format(m_strEventFired, new
            Object[] {new Integer(p_Event.getEvent()),
            m_strEventNames[p_Event.getEvent()]}), getName(), 0);
  synchronized(m_events)
  {
    m_events.addElement(p_Event);
    if (m_threadGo == null) start();
  }
}

/**
 * Запускает автомат. После обработки всех событий из очереди событий,
    автомат
 * "засыпает" ({@link #stop()}). По приходу какого-либо события, автомат
    снова
 * начинает работу ({@link #start()}).
 */
private void start()
{
  m_threadGo = new Thread(this);
  m_threadGo.start();
}


/**
 * Останавливает работу автомата. После обработки всех событий из очереди
    событий,
 * автомат * "засыпает" ({@link #stop()}). По приходу какого-либо события,
    автомат
 * снова * начинает работу ({@link #start()}).
 */
private void stop()
{
  m_threadGo = null;
}

/**
 * Запускает {@link #caseBlock(int p_Event)} по-очереди для каждого события
    из
 * очереди событий. После обработки последнего - останавливается.
 */
public void run()
{
  Thread th = Thread.currentThread();
  boolean isEvent = false;
  BaseEvent event = null;

  while (th == m_threadGo)
  {
    isEvent = false;
    event = null;


    //синхронизируемся по объекту очереди
    synchronized(m_events)
    {
      if (!m_events.isEmpty())
      {
```

```
          event = (BaseEvent)m_events.firstElement();
          m_events.remove(0);
          isEvent = true;
        }
        else
        {
          // если очередь пуста - останавливаем поток, пока
          // очередной раз не вызовут метод A(int e)
          stop();
        }
      }
      //конец синхронизации


      if (isEvent)
      {
        int intOldState = y;
        ctx.getLogger(getName()).log(MessageFormat.format(m_strStartProcess,
               new Object[] {new Integer(event.getEvent()),
               m_strEventNames[event.getEvent()]}), getName(), 0);
        ctx.getLogger(getName()).log(MessageFormat.format(m_strCurrState, new
               Object[] {new Integer(y), m_strStateNames[y]}), getName(), 0);
        m_strCaller = event.getCaller();
        m_blnProcessingStarted = true;
        try
        {
          caseBlock(event.getEvent());
        }
        catch (Exception e)
        {
          e.printStackTrace();
          ctx.getLogger(getName()).log("Error occured" + e.getMessage());
        }
        m_blnProcessingStarted = false;
        ctx.getLogger(getName()).log(MessageFormat.format(m_strNewState, new
               Object[] {new Integer(y), m_strStateNames[y]}), getName(), 0);
        ctx.getLogger(getName()).log(MessageFormat.format(m_strEndProcess,
               new Object[] {new Integer(event.getEvent()),
               m_strEventNames[event.getEvent()]}), getName(), 0);

        if (intOldState != y) stateChanged(intOldState);
      }

      try
      {
        Thread.sleep(m_intDelay);
      }
      catch (InterruptedException e)
      {
        e.printStackTrace();
      }
    }
  }

  /**
   * Вызывает метод zXX данного класса. XX - номер, передаваемый в переменной
   * @param p_intNumber номер вызываемого метода
   */
  protected void z(int p_intNumber) throws Exception
  {
    ctx.getLogger(getName()).log(MessageFormat.format(m_strZ, new Object[]
               {new Integer(p_intNumber), m_strActionNames[p_intNumber]}),
               getName(), 1);
    getClass().getDeclaredMethod("z" + p_intNumber, null).invoke(this, null);
```

```
  }

  /**
   * Вычисляет значение условия xXX данного класса. XX - номер, передаваемый
                в переменной
   * @param p_intNumber номер вычисляемого значения
   */
  protected boolean x(int p_intNumber) throws Exception
  {
    boolean result = ((Boolean)getClass().getDeclaredMethod("x" +
              p_intNumber, null).invoke(this, null)).booleanValue();

    String strValueName;
    strValueName = result ? m_strTrue: m_strFalse;
    ctx.getLogger(getName()).log(MessageFormat.format(m_strX, new Object[]
              {new Integer(p_intNumber), m_strVarNames[p_intNumber],
              strValueName}), getName(), 1);
    return result;
  }

  /**
   * Основная логика автомата.
   * @param p_Event    Входное событие.
   */
  protected abstract void caseBlock(int p_Event) throws Exception;


  /**
   * Запускается каждый раз, когда автомат перешел в новое состояние.
   * @param p_intOldState  Старое состояние.
   */
  protected abstract void stateChanged(int p_intOldState);


  /**
   * Возвращает название вызвавшего автомата
   * @return название вызвавшего автомата
   */
  protected String getCaller() throws Exception
  {
    if (m_blnProcessingStarted == false)
    {
      throw new Exception("You can access this method only during event
              processing");
    }
    return m_strCaller;
  }

  /**
   * Возвращает имя автомата.
   * @return Имя автомата.
   */
  public String getName()
  {
    return m_strName;
  }
}
```

Класс BaseEvent

```
/**
Элементарное событие для автоматного взаимодействия
*/
public class BaseEvent
{
  /**
   * Номер события
   */
  private int m_event;

  /**
   * Название вызывающего автомата
   */
  private String m_strCaller;

  /**
   * Конструктор
   */
  public BaseEvent(int p_event, String p_strCaller)
  {
    m_event = p_event;
    m_strCaller = p_strCaller;
  }

  /**
   * Возвращает номер события
   */
  public int getEvent()
  {
    return m_event;
  }

  /**
   * Возвращает название вызывающего автомата
   */
  public String getCaller()
  {
    return m_strCaller;
  }

}
```

Класс A0

```
/**Автоматный класс, реализующий работу кнопочного телефона.*/
public class A0 extends BaseAutomata
{
  /**Название АТС, с которой соединен автомат. Не null только во время набора
              номера и звонка*/
  private String  m_strATSName;

  /**Набранный номер*/
  private int[]   m_digits;

  /**Количество набранных цифр номера*/
  private int     m_intLength;

  /**Интерфейс телефона*/
  private Phone   m_phone;
```

```
/**
 * Конструктор.
 * @param p_strName Название автомата (номер телефона).
 * @param p_phone   Интерфейс телефона.
 * @param p_ctx     Контекст проекта.
 */
public A0(String p_strName, Phone p_phone, Context p_ctx)
{
  super(p_strName, p_ctx);

  m_strStateNames = new String[10];
  m_strEventNames = new String[25];
  m_strVarNames = new String[20];
  m_strActionNames = new String[30];

  int i;
  for(i = 0; i < 10; i++) m_strStateNames[i] =  ctx.getParameter("A0.y" +
            i);
  for(i = 0; i < 25; i++) m_strEventNames[i] =  ctx.getParameter("A0.e" +
            i);
  for(i = 0; i < 20; i++) m_strVarNames[i] =    ctx.getParameter("A0.x" +
            i);
  for(i = 0; i < 30; i++) m_strActionNames[i] = ctx.getParameter("A0.z" +
            i);

  m_intDelay = Integer.parseInt(ctx.getParameter("Delay") != null ?
            ctx.getParameter("Delay"): "10");

  m_digits        = new int[10];
  m_intLength     = 0;
  m_phone         = p_phone;
  m_strATSName    = null;

  stateChanged(0);
}


protected void caseBlock(int p_Event) throws Exception
{
  int e = p_Event;
  int yOld = y;

  switch (y)
  {

    case 0:
      if         (e == 24)    {z(9); y=7;}

      else if    (e == 2)     {z(6);A1(2); y=3;}

      else if    (e == 10)    {z(10);z(20); y=1;}

      else if    (e == 11)    {z(11);z(21); y=1;}

      else if    (e == 12)    {z(12);z(22); y=1;}

      else if    (e == 13)    {z(13);z(23); y=1;}

      else if    (e == 14)    {z(14);z(24); y=1;}

      else if    (e == 15)    {z(15);z(25); y=1;}

      else if    (e == 16)    {z(16);z(26); y=1;}
```

```
  else if    (e == 17)    {z(17);z(27); y=1;}
  else if    (e == 18)    {z(18);z(28); y=1;}
  else if    (e == 19)    {z(19);z(29); y=1;}
  break;
case 1:
  if         (e == 2)     {z(6);A1(2);A0(0); y=2;}
  else if    (e == 24)    {z(4);z(5);z(9); y=7;}
  else if    ((e == 3)||(e == 4))    {z(4);z(5); y=0;}
  else if    (e == 10)    {z(10);z(20); y=1;}
  else if    (e == 11)    {z(11);z(21); y=1;}
  else if    (e == 12)    {z(12);z(22); y=1;}
  else if    (e == 13)    {z(13);z(23); y=1;}
  else if    (e == 14)    {z(14);z(24); y=1;}
  else if    (e == 15)    {z(15);z(25); y=1;}
  else if    (e == 16)    {z(16);z(26); y=1;}
  else if    (e == 17)    {z(17);z(27); y=1;}
  else if    (e == 18)    {z(18);z(28); y=1;}
  else if    (e == 19)    {z(19);z(29); y=1;}
  break;
case 2:
  if         (e == 4)     {A0(3);A0(2); y=3;}
  else if    (!(x(2)))    { y=3;}
  else if    (e == 3)     {z(4);z(5);A1(3);z(7); y=0;}
  else if    ((x(2))&&(x(10)))    {A1(10);z(3);A0(0); y=2;}
  else if    ((x(2))&&(x(11)))    {A1(11);z(3);A0(0); y=2;}
  else if    ((x(2))&&(x(12)))    {A1(12);z(3);A0(0); y=2;}
  else if    ((x(2))&&(x(13)))    {A1(13);z(3);A0(0); y=2;}
  else if    ((x(2))&&(x(14)))    {A1(14);z(3);A0(0); y=2;}
  else if    ((x(2))&&(x(15)))    {A1(15);z(3);A0(0); y=2;}
  else if    ((x(2))&&(x(16)))    {A1(16);z(3);A0(0); y=2;}
  else if    ((x(2))&&(x(17)))    {A1(17);z(3);A0(0); y=2;}
  else if    ((x(2))&&(x(18)))    {A1(18);z(3);A0(0); y=2;}
  else if    ((x(2))&&(x(19)))    {A1(19);z(3);A0(0); y=2;}
```

```
  break;

case 3:
  if         (e == 22)    { y=4;}

  else if    (e == 10)    {A1(10);z(20); y=3;}

  else if    (e == 11)    {A1(11);z(21); y=3;}

  else if    (e == 12)    {A1(12);z(22); y=3;}

  else if    (e == 13)    {A1(13);z(23); y=3;}

  else if    (e == 14)    {A1(14);z(24); y=3;}

  else if    (e == 15)    {A1(15);z(25); y=3;}

  else if    (e == 16)    {A1(16);z(26); y=3;}

  else if    (e == 17)    {A1(17);z(27); y=3;}

  else if    (e == 18)    {A1(18);z(28); y=3;}

  else if    (e == 19)    {A1(19);z(29); y=3;}

  else if    (e == 4)     {A0(3);A0(2); y=3;}

  else if    (e == 3)     {z(4);z(5);A1(3);z(7); y=0;}

  else if    (e == 21)    { y=5;}

  break;

case 4:
  if         (e == 4)    {A0(3);A0(2); y=3;}

  else if    (e == 3)    {z(4);z(5);A1(3);z(7); y=0;}

  break;

case 5:
  if         (e == 4)    {A0(3);A0(2); y=3;}

  else if    (e == 23)   { y=6;}

  else if    (e == 3)    {z(4);z(5);A1(3);z(7); y=0;}

  break;

case 6:
  if         (e == 22)   { y=4;}

  else if    (e == 4)    {A0(3);A0(2); y=3;}

  else if    (e == 3)    {z(4);z(5);A1(3);z(7); y=0;}

  break;

case 7:
  if         (e == 22)   { y=0;}

  else if    (e == 2)    {z(8);A1(22); y=8;}

  break;
```

```
    case 8:
      if         (e == 3)     {A1(23);z(7); y=0;}

      else if    (e == 22)    { y=9;}
      break;

    case 9:
      if         (e == 3)    {A1(23);z(7); y=0;}
      break;


  }

  if (y == yOld) return;
  switch (y)
  {

  }
}



protected void stateChanged(int p_intOldState)
{
  boolean b = true;
  if ((y == 0) || (y == 1) || (y == 7)) b = false;
  m_phone.setState(y, m_strStateNames[y], b);
}

/**
 * Добавить цифру к набранному к данному моменту номеру.
 * Можно набрать номер, а затем поднять трубку. В таком случае текущий
 * набранный номер будет посылаться АТС.
 * @param x Цифра.
 */
private void z1X(int x)
{
  if (m_intLength < 9)
  {
    m_digits[m_intLength] = x;
    m_intLength = m_intLength + 1;
  };
}

/**
 * Добавить цифрц на дисплей.
 * @param x Цифра.
 */
private void z2X(int x)
{
  m_phone.displayAddString("" + x);
}

/**Удалить первую цифру из набранного к данному моменту номера.*/
protected void z3()
{
  for(int i = 1; i < m_intLength; i++)
  {
    m_digits[i - 1] = m_digits[i];
  }
  m_intLength = m_intLength - 1;
}
```

```
/**Очистить набранный к данному моменту номер.*/
protected void z4()
{
  m_intLength = 0;
}
/**Очистить дисплей.*/
protected void z5()
{
  m_phone.displayReset();
}

/**Занять АТС.*/
protected void z6()
{
  m_strATSName = ctx.getFreeATS();
}

/**Освободить АТС.*/
protected void z7() throws Exception
{
  ctx.setATSFree(m_strATSName);
  m_strATSName = null;
}

/**Подтвердить соединение с АТС*/
protected void z8() throws Exception
{
  ctx.occupyATS(m_strATSName);
}

/**Получить название АТС*/
protected void z9() throws Exception
{
  m_strATSName = getCaller();
}

/**{@link #z1X(int x)}, x = 0*/
protected void z10() {z1X(0);}
/**{@link #z1X(int x)}, x = 1*/
protected void z11() {z1X(1);}
/**{@link #z1X(int x)}, x = 2*/
protected void z12() {z1X(2);}
/**{@link #z1X(int x)}, x = 3*/
protected void z13() {z1X(3);}
/**{@link #z1X(int x)}, x = 4*/
protected void z14() {z1X(4);}
/**{@link #z1X(int x)}, x = 5*/
protected void z15() {z1X(5);}
/**{@link #z1X(int x)}, x = 6*/
protected void z16() {z1X(6);}
/**{@link #z1X(int x)}, x = 7*/
protected void z17() {z1X(7);}
/**{@link #z1X(int x)}, x = 8*/
protected void z18() {z1X(8);}
/**{@link #z1X(int x)}, x = 9*/
protected void z19() {z1X(9);}

/**{@link #z2X(int x)}, x = 0*/
protected void z20() {z2X(0);}
/**{@link #z2X(int x)}, x = 1*/
protected void z21() {z2X(1);}
/**{@link #z2X(int x)}, x = 2*/
protected void z22() {z2X(2);}
/**{@link #z2X(int x)}, x = 3*/
```

```
protected void z23() {z2X(3);}
/**{@link #z2X(int x)}, x = 4*/
protected void z24() {z2X(4);}
/**{@link #z2X(int x)}, x = 5*/
protected void z25() {z2X(5);}
/**{@link #z2X(int x)}, x = 6*/
protected void z26() {z2X(6);}
/**{@link #z2X(int x)}, x = 7*/
protected void z27() {z2X(7);}
/**{@link #z2X(int x)}, x = 8*/
protected void z28() {z2X(8);}
/**{@link #z2X(int x)}, x = 9*/
protected void z29() {z2X(9);}


/**
 * Текущая цифра - цифра x
 * @param x - проверяемая цифра.
 * @return Текущая цифра из буфера совпадает с x.
 */
private boolean x1X(int x)
{
  return (m_intLength >= 1) && (m_digits[0] == x);
}

/**
 * Есть ли еще цифры.
 * @return Есть ли еще цифры в буфере телефона.
 */
protected boolean x2()
{
  return m_intLength >= 1;
}

/**{@link #x1X(int x)}, x = 0*/
protected boolean x10() {return x1X(0);}
/**{@link #x1X(int x)}, x = 1*/
protected boolean x11() {return x1X(1);}
/**{@link #x1X(int x)}, x = 2*/
protected boolean x12() {return x1X(2);}
/**{@link #x1X(int x)}, x = 3*/
protected boolean x13() {return x1X(3);}
/**{@link #x1X(int x)}, x = 4*/
protected boolean x14() {return x1X(4);}
/**{@link #x1X(int x)}, x = 5*/
protected boolean x15() {return x1X(5);}
/**{@link #x1X(int x)}, x = 6*/
protected boolean x16() {return x1X(6);}
/**{@link #x1X(int x)}, x = 7*/
protected boolean x17() {return x1X(7);}
/**{@link #x1X(int x)}, x = 8*/
protected boolean x18() {return x1X(8);}
/**{@link #x1X(int x)}, x = 9*/
protected boolean x19() {return x1X(9);}


/**
 * Запускает {@link A0} с событием e.
 * @param e Событие.
 */
private void A0(int e)
{
  this.A(new BaseEvent(e, getName()));
}
```

```
  /**
   * Запускает {@link A1} с событием e.
   * @param e Событие.
   */
  private void A1(int e)
  {
    ctx.getAutomata(m_strATSName).A(new BaseEvent(e, getName()));
  }
}
```

Класс A1

```
import java.util.*;
import java.util.*;
/**
Автоматный класс, реализующий работу АТС.
*/
public class A1 extends BaseAutomata
{
  /**Объект, который будет отображать АТС*/
  private ATS m_ats;

  /**Номер телефона вызывающего абонента.*/
  private String m_strFirstSubscriber;

  /**Номер телефона вызываемого абонента.*/
  private String m_strSecondSubscriber;

  /**Набранная к данному моменту часть номера.*/
  private String m_strNumber;

  /**
   * Конструктор.
   * @param p_strName  Название АТС.
   * @param p_ats      Интерфейс АТС.
   * @param p_ctx      Контекст проекта.
   */
  public A1(String p_strName, ATS p_ats, Context p_ctx)
  {
    super(p_strName, p_ctx);

    //gets the state names from applet parameters
    m_strStateNames = new String[6];
    m_strEventNames = new String[24];
    m_strVarNames = new String[3];
    m_strActionNames = new String[20];

    int i;
    for(i = 0; i < 6; i++) m_strStateNames[i] =  ctx.getParameter("A1.y"+ i);
    for(i = 0; i < 24; i++) m_strEventNames[i] = ctx.getParameter("A1.e"+ i);
    for(i = 0; i < 3; i++)  m_strVarNames[i] =  ctx.getParameter("A1.x" + i);
    for(i = 0; i < 20; i++) m_strActionNames[i]= ctx.getParameter("A1.z" +i);

    //initialize Delay Variable
    m_intDelay = Integer.parseInt(ctx.getParameter("Delay") != null ?
              ctx.getParameter("Delay"): "10");

    m_ats = p_ats;
    m_strNumber = "";
    stateChanged(0);
  }
```

```
protected void caseBlock(int p_Event) throws Exception
{
  int e = p_Event;
  int yOld = y;

  switch (y)
  {

    case 0:
      if    (e == 2)    {z(2); y=1;}

      break;

    case 1:
      if         ((x(1))&&(!(x(2))))       {A0(22); y=2;}

      else if    (e == 3)                  {z(4);z(6); y=0;}

      else if    ((e == 10)&&(!(x(1))))    {z(10);A1(0); y=1;}

      else if    ((e == 11)&&(!(x(1))))    {z(11);A1(0); y=1;}

      else if    ((e == 12)&&(!(x(1))))    {z(12);A1(0); y=1;}

      else if    ((e == 13)&&(!(x(1))))    {z(13);A1(0); y=1;}

      else if    ((e == 14)&&(!(x(1))))    {z(14);A1(0); y=1;}

      else if    ((e == 15)&&(!(x(1))))    {z(15);A1(0); y=1;}

      else if    ((e == 16)&&(!(x(1))))    {z(16);A1(0); y=1;}

      else if    ((e == 17)&&(!(x(1))))    {z(17);A1(0); y=1;}

      else if    ((e == 18)&&(!(x(1))))    {z(18);A1(0); y=1;}

      else if    ((e == 19)&&(!(x(1))))    {z(19);A1(0); y=1;}

      else if    ((x(1))&&(x(2)))          {z(3);A0(21);A2(24); y=4;}

      break;

    case 2:
      if        (e == 3)      {z(4);z(6); y=0;}

      break;

    case 3:
      if        (e == 23)     {z(5); y=0;}

      break;

    case 4:
      if         (e == 22)    {A0(23); y=5;}

      else if    (e == 3)     {z(4);z(5);z(6);A2(22); y=0;}

      break;

    case 5:
      if         (e == 23)    {z(5);A0(22); y=2;}

      else if    (e == 3)     {z(4);A2(22); y=3;}
```

```
      break;


  }

  if (y == yOld) return;
  switch (y)
  {

    case 0:
      z(6);
      break;

  }
}



protected void stateChanged(int p_OldState)
{
  m_ats.setState(m_strStateNames[y]);
}

/**
 * Добавить цифру к {@link #m_strNumber}.
 * @param x Цифра.
 */
private void z1X(int x)
{
  m_strNumber += x;
  m_ats.setNumber(m_strNumber);
}

/**{@link #z1X(int x)}, x = 0*/
protected void z10() {z1X(0);}
/**{@link #z1X(int x)}, x = 1*/
protected void z11() {z1X(1);}
/**{@link #z1X(int x)}, x = 2*/
protected void z12() {z1X(2);}
/**{@link #z1X(int x)}, x = 3*/
protected void z13() {z1X(3);}
/**{@link #z1X(int x)}, x = 4*/
protected void z14() {z1X(4);}
/**{@link #z1X(int x)}, x = 5*/
protected void z15() {z1X(5);}
/**{@link #z1X(int x)}, x = 6*/
protected void z16() {z1X(6);}
/**{@link #z1X(int x)}, x = 7*/
protected void z17() {z1X(7);}
/**{@link #z1X(int x)}, x = 8*/
protected void z18() {z1X(8);}
/**{@link #z1X(int x)}, x = 9*/
protected void z19() {z1X(9);}

/**Занять телефон вызывающего абонента.*/
protected void z2() throws Exception
{
  m_strFirstSubscriber = getCaller();
  ctx.m_busyAutomatas.add(m_strFirstSubscriber);
}
```

```
/**Занять телефон вызываемого абонента.*/
protected void z3()
{
  ctx.m_busyAutomatas.add(m_strSecondSubscriber);
}

/**Освободить телефон вызывающего абонента.*/
protected void z4()
{
  ctx.m_busyAutomatas.remove(m_strFirstSubscriber);
}

/**Освободить телефон вызываемого абонента.*/
protected void z5()
{
  ctx.m_busyAutomatas.remove(m_strSecondSubscriber);
}

/**Сбросить набранный к данному моменту номер.*/
protected void z6()
{
  m_strNumber = "";
  m_ats.setNumber(m_strNumber);
}

/**Валидный номер.*/
protected boolean x1()
{
  return m_strNumber.length() == 7;
}

/**Вызываемый абонент свободен.*/
protected boolean x2()
{
  //автомат в списке занятых абонентов
  if (ctx.m_busyAutomatas.contains(m_strNumber))
    return false;

  //такого телефона не существует
  if (ctx.getAutomata(m_strNumber) == null)
    return false;

  m_strSecondSubscriber = m_strNumber;
  return true;
}

/**
 * Послать событие p_intEvent автомату {@link A0}, реализующему телефон
    вызывающего абонента.
 * @param p_intEvent Событие.
 */
private void A0(int p_intEvent)
{
  ctx.getAutomata(m_strFirstSubscriber).A(new BaseEvent(p_intEvent,
            getName()));
}

/**
 * Послать событие p_intEvent автомату {@link A0}, реализующему телефон
    вызвывамого абонента.
 * @param p_intEvent Событие.
 */
private void A2(int p_intEvent)
{
```

```
    ctx.getAutomata(m_strSecondSubscriber).A(new BaseEvent(p_intEvent,
                getName()));
  }

  /**
   * Послать событие p_intEvent самому себе.
   * @param p_intEvent Событие.
   */
  private void A1(int p_intEvent)
  {
    this.A(new BaseEvent(p_intEvent, getName()));
  }
}
```

Класс Context

```
import java.util.*;
import java.applet.*;

/**
* Контекст проекта. Содержит названия всех автоматов.
*/
public class Context
{
  /**Логер для логирования ошибок.*/
  private Logger      m_logger = null;

  /**Список автоматов.*/
  private Hashtable   m_hashAutomatas;

  /**Запускающий апплет.*/
  private Applet  m_appMain;

  /**Список занятых автоматов.*/
  public Vector m_busyAutomatas = new Vector();

  /**Свободные АТС.*/
  public Hashtable m_hashFreeATS = new Hashtable();


  /**
   * Конструктор.
   * @param  p_app Главный апплет приложения.
   */
  public Context(Applet p_app)
  {
    m_hashAutomatas = new Hashtable();
    m_appMain       = p_app;
  }


  /**
   * Возвращает экземпляр класса, обеспечивающий логинирование.
   * @param  p_strLoggerName Главный апплет приложения. В данной реализации
      не функционален.
   * @return Экземпляр класса {@link Logger}.
   */
  public Logger getLogger(String p_strLoggerName)
  {
    if (m_logger == null)
    {
      m_logger = new Logger(p_strLoggerName);
```

```
  }
  return m_logger;
}

/**
 * Возвращает экземпляр класса, обеспечивающий логинирование.
 * @return Экземпляр класса {@link Logger}.
 */
public Logger getLogger() { return getLogger(""); }

/**
 * Добавляет автомат в контекст проекта.
 * @param p_automata Добавляемый автомат.
 */
public void addAutomata(BaseAutomata p_automata)
{
  if (m_hashAutomatas.containsKey(p_automata.getName()))
  {
    getLogger().log("Replace automata " + p_automata.getName(), "Context");
  }
  m_hashAutomatas.put(p_automata.getName(), p_automata);

  if (p_automata.getClass().getName() == "A1")
    m_hashFreeATS.put(p_automata.getName(), "free");

  getLogger().log("Add automata " + p_automata.getName(), "Context");
}

/**
 * Возвращает автомат с заданным именем.
 * @param Имя автомата.
 * @return Автомат с указанным именем.
 */
public BaseAutomata getAutomata(String p_strAutomataName)
{
  return (BaseAutomata) m_hashAutomatas.get(p_strAutomataName);
}


/**
 * Возвращвет параметр с заданным именем.
 * @param p_strParameterName Название параметра.
 * @return Параметр.
 */
public String getParameter(String p_strParameterName)
{
  String strTmp = m_appMain.getParameter(p_strParameterName);
  return (strTmp == null) ? "#" + p_strParameterName + "#": strTmp;
}

/**
 * Возвращает название свободной АТС. Может быть использовано в {@link
    Context#getAutomata(String)} в качестве параметра.
 * @return  Название свободной АТС.
 */
public String getFreeATS()
{
  for (Enumeration e = m_hashFreeATS.keys(); e.hasMoreElements(); )
  {
    String name = (String)e.nextElement();

    if (((String)m_hashFreeATS.get(name)).equals("free"))
    {
      m_hashFreeATS.put(name, "busy");
```

```
        return name;
      }
    }
    return null;
  }

  /**
   * Занять АТС
   * @param p_strATSName Название АТС
   */
  public void occupyATS(String p_strATSName) throws Exception
  {
    String atsState = (String)m_hashFreeATS.get(p_strATSName);

    if ( (atsState == null) || (atsState == "free") )
    {
      m_hashFreeATS.put(p_strATSName, "busy");
    }
    else if (atsState == "busy")
    {
      m_hashFreeATS.put(p_strATSName, "very busy");
    }
    else if (atsState == "very busy")
    {
      throw new Exception("ATS already busy");
    }
  }

  /**
   * Освобождает АТС.
   * @param p_strATSName Название АТС
   */
  public void setATSFree(String p_strATSName) throws Exception
  {
    String atsState = (String)m_hashFreeATS.get(p_strATSName);

    if ( (atsState == null) || (atsState == "free") )
    {
      throw new Exception("ATS already free");
    }
    else if (atsState == "busy")
    {
      m_hashFreeATS.put(p_strATSName, "free");
    }
    else if (atsState == "very busy")
    {
      m_hashFreeATS.put(p_strATSName, "busy");
    }
  }

}
```

Класс Logger

```
import java.text.*;
import java.util.*;

/**
* Класс обеспечивает логирование.
*/
public class Logger
{
```

```
  /**Имя логера. В данной реализации не функционально*/
  private String m_strName;

  /**
   * Конструктор.
   * @param p_strName Имя логгера. В данной реализации не функционально.
   */
  public Logger(String p_strName)
  {
    m_strName = p_strName;
  }

  /**
   * Логирует сообщение.
   * @param p_strMessage  Сообщение для логирования.
   * @param p_intLevel    Уровень вложенности сообщения.
   * @param p_strSource   Источник сообщения.
   */
  public void log(String p_strMessage, String p_strSource, int p_intLevel)
  {
    SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yy hh.mm.S");

    String strTmp = sdf.format(new Date()) + "\t" + p_strSource + "\t:\t";
    for(int i = 0; i < p_intLevel; i++) { strTmp = strTmp + "\t"; }

    System.out.print(strTmp + p_strMessage + "\n");
  }

  /**
   * Логирует сообщение. Уровень вложенности 0, Название источника сообщения
     - пустая строка.
   * @param p_strMessage  Сообщение для логирования.
   */
  public void log(String p_strMessage)
  {
    log(p_strMessage, "", 0);
  }

  /**
   * Логирует сообщение. Уровень вложенности сообщения 0.
   * @param p_strMessage  Сообщение для логирования.
   * @param p_strSource   Источник сообщения.
   */
  public void log(String p_strMessage, String p_strSource)
  {
    log(p_strMessage, p_strSource, 0);
  }
}
```

Класс Phone

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.net.*;
/**
* Интерфейс телефона. <br><br>
* Состоит из:
* <ul>
*   <li>слова "Телефон";</li>
*   <li>номера телефона;</li>
```

```
*   <li>дисплея;</li>
*   <li>стандартной кнопочной клавиатуры;</li>
*   <li>кнопки "сброс";</li>
*   <li>кнопки "поднять трубку";</li>
*   <li>строки состояния.</li>
* </ul>
*/
public class Phone extends Panel implements ActionListener, Runnable
{
  /**Контекст проекта.*/
  private Context  ctx;

  //interface controls
  /**Стандартная кнопочная клавиатура.*/
  private Button[] m_btnDigits;
  /**Кнопка "сброс".*/
  private Button   m_btnReset;
  /**Кнопка "поднять трубку."*/
  private Button   m_btnReciever;
  /**Дисплей.*/
  private Label    m_lblDisplay;
  /**Строка состояния.*/
  private Label    m_lblState;

  /**Номер телефона.*/
  private String   m_strAutomataName;
  /**Заголовок для кнопки "поднять трубку".*/
  private String   m_strRecieverUp;
  /**Заголовок для кнопки "опустить трубку".*/
  private String   m_strRecieverDown;

  /**Поток для визуализации звонка*/
  private Thread   m_threadGo;

  /**Аудио для проигрывания звонка*/
  private AudioClip m_audio;


  /**
   * Конструктор.
   * @param p_strAutomataName Номер телефона.
   * @param p_ctx Клнтекст проекта.
   */
  public Phone(String p_strAutomataName, Context p_ctx)
  {
    super(new BorderLayout(5, 5));

    ctx = p_ctx;
    m_strAutomataName = p_strAutomataName;

    m_strRecieverUp = ctx.getParameter("HangUp");
    m_strRecieverDown = ctx.getParameter("HangDown");

    initControls();
  }


  /**Инициализирует и размещает компоненты.*/
  private void initControls()
  {
    setBackground(new Color(0, 0, 128));

    Label lblTelephone = new Label(ctx.getParameter("Telephone"));
    lblTelephone.setFont(new Font("Arial", Font.BOLD, 20));
```

```
lblTelephone.setBackground(new Color(0, 0, 128));
lblTelephone.setForeground(Color.white);
lblTelephone.setAlignment(Label.CENTER);

Label lblNumber = new Label(m_strAutomataName);
lblNumber.setFont(new Font("Arial", Font.BOLD, 14));
lblNumber.setBackground(new Color(0, 0, 128));
lblNumber.setForeground(Color.white);
lblNumber.setAlignment(Label.CENTER);

m_lblDisplay = new Label("");
m_lblDisplay.setBackground(new Color(255, 128, 64));
m_lblDisplay.setForeground(Color.white);

m_btnDigits = new Button[10];
for(int i = 0; i < 10; i++)
{
  m_btnDigits[i] = new Button(i + "");
  m_btnDigits[i].setActionCommand("e1" + i);
  m_btnDigits[i].addActionListener(this);
}

m_btnReset = new Button("C");
m_btnReset.setActionCommand("e4");
m_btnReset.addActionListener(this);

m_btnReciever = new Button(m_strRecieverUp);
m_btnReciever.setActionCommand("e2");
m_btnReciever.addActionListener(this);

m_lblState = new Label("");
m_lblState.setBackground(Color.lightGray);


Panel pnlTitle    = new Panel(new BorderLayout(0, 0));
Panel pnlDisplay  = new Panel(new BorderLayout(10, 10));
Panel pnlButtons  = new Panel(new GridLayout(4, 3, 5, 5));
Panel pnlReciever = new Panel();
Panel pnlState    = new Panel(new GridLayout(1, 1));

pnlTitle.add(lblTelephone, BorderLayout.CENTER);
pnlTitle.add(lblNumber, BorderLayout.SOUTH);

pnlDisplay.add(m_lblDisplay, BorderLayout.CENTER);

for(int i = 1; i < 10; i++)
{
  pnlButtons.add(m_btnDigits[i]);
}

pnlButtons.add(new Label(""));
pnlButtons.add(m_btnDigits[0]);
pnlButtons.add(m_btnReset);

pnlReciever.add(m_btnReciever);

pnlState.add(m_lblState);

Panel tmp = new Panel(new BorderLayout(5, 5));
tmp.add(pnlTitle, BorderLayout.NORTH);
tmp.add(pnlDisplay, BorderLayout.SOUTH);
add(tmp, BorderLayout.NORTH);

add(pnlButtons, BorderLayout.CENTER);
```

```
  tmp = new Panel(new BorderLayout(5, 5));
  tmp.add(pnlReciever, BorderLayout.NORTH);
  tmp.add(pnlState, BorderLayout.CENTER);
  add(tmp, BorderLayout.SOUTH);

  add(new Label(" "), BorderLayout.EAST);
  add(new Label(" "), BorderLayout.WEST);
}


public void actionPerformed(ActionEvent e)
{
  String strTmp = e.getActionCommand();

  if (strTmp == "e2")
  {
    A(2);
    return;
  }

  if (strTmp == "e3")
  {
    A(3);
    return;
  }

  if (strTmp == "e4")
  {
    A(4);
    return;
  }

  for (int i = 10; i < 20; i++)
  {
    if (strTmp.equals("e" + i))
    {
      A(i);
      return;
    }
  }
}


/**
 * Добавляет цифры к дисплею.
 * @param p_s Цифры.
 */
public void displayAddString(String p_s)
{
  if (m_lblDisplay.getText().length() + p_s.length() < 11)
    m_lblDisplay.setText(m_lblDisplay.getText() + p_s);
}

/**Сбрасываетнабранный на дисплее номер.*/
public void displayReset()
{
  m_lblDisplay.setText("");
}

/**
 * Меняет строку состояния и надпись на кнопке "поднять трубку".
 * @param p_intState  Номер состояния
 * @param p_stateName Название состояния.
 * @param isRecieverUp Поднята ли трубка.
```

```
 */
public void setState(int p_intState, String p_stateName, boolean
             isRecieverUp)
{
  m_lblState.setText(p_stateName);
  if (isRecieverUp)
  {
    m_btnReciever.setLabel(m_strRecieverDown);
    m_btnReciever.setActionCommand("e3");
  }
  else
  {
    m_btnReciever.setLabel(m_strRecieverUp);
    m_btnReciever.setActionCommand("e2");
  }

  try
  {
    if (m_audio != null)
    {
      m_audio.stop();
      m_audio = null;
    }

    String strFileName = null;
    switch(p_intState)
    {
      case 7:
        strFileName = "ring.mid";
        break;
      case 5:
        strFileName = "long.mid";
        break;
      case 4:
      case 9:
        strFileName = "short.mid";
        break;
    }

    if (strFileName != null)
    {
      m_audio = Applet.newAudioClip(new URL("file:" + strFileName));
      m_audio.loop();
    }
  }
  catch(Exception e)
  {
    ctx.getLogger("").log("Mid files unavailable");
  }

  if (p_intState == 7)
  {
    m_threadGo = new Thread(this);
    m_threadGo.start();
  }
  else
  {
    m_threadGo = null;
  }
}

/**Поток визуализации звонка*/
public void run()
{
```

```
    Thread th = Thread.currentThread();

    m_lblDisplay.setBackground(new Color(0, 0, 0));
    m_lblDisplay.setText("            <<<>>>");

    boolean isNative = true;

    while (th == m_threadGo)
    {
      if (isNative)
      {
        m_lblDisplay.setBackground(new Color(255, 255, 255));
        isNative = false;
      }
      else
      {
        m_lblDisplay.setBackground(new Color(255, 128, 64));
        isNative = true;
      }

      try
      {
        Thread.sleep(200);
      }
      catch (InterruptedException e)
      {
        e.printStackTrace();
      }
    }

    if (m_lblDisplay.getText() == "            <<<>>>")
      m_lblDisplay.setText("");
    m_lblDisplay.setBackground(new Color(255, 128, 64));
  }


  /**
   * Запускает автомат {@link A0}.
   * @param p_event Событие.
   */
  private void A(int p_event)
  {
    ctx.getAutomata(m_strAutomataName).A(new BaseEvent(p_event, null));
  }
}
```

Класс ATS

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

/**
* Интерфейс АТС. Состоит из строки состояния и строки набранного к данному
      моменту номера.
*/
public class ATS extends Panel
{
  /**Строка состояния.*/
  private Label m_lblState;
  /**Набранный к данному моменту номер.*/
  private Label m_lblNumber;
```

```
/**Название АТС, которую визуализирует данный класс.*/
private String m_strAutomataName;


/**Контекст проекта.*/
private Context ctx;


/**
 * Конструктор.
 * @param p_strAutomataName Название АТС, которую визуализирует данный
    класс.
 * @param p_ctx             Контекст проекта.
 */
public ATS(String p_strAutomataName, Context p_ctx)
{
  super(new BorderLayout(5, 5));

  setBackground(Color.black);

  m_strAutomataName = p_strAutomataName;
  ctx = p_ctx;

  initControls();
}


/**
 * Инициализирует строки и расставляет их.
 */
private void initControls()
{
  m_lblState = new Label("");
  m_lblState.setBackground(Color.lightGray);
  m_lblState.setSize(200, 20);

  m_lblNumber = new Label("");
  m_lblNumber.setBackground(Color.lightGray);
  m_lblNumber.setSize(200, 20);

  add(m_lblState, BorderLayout.NORTH);
  add(m_lblNumber, BorderLayout.CENTER);
}


/**
 * Устанавливает новое значение строки состояния.
 * @param p_stateName Название нового состояния.
 */
public void setState(String p_stateName)
{
  m_lblState.setText(p_stateName);
}


/**
 * Устанавливает новое значение строки набранного к данному моменту номера.
 * @param p_strNumber Набранный к данному моменту номер.
 */
public void setNumber(String p_strNumber)
{
  m_lblNumber.setText(p_strNumber);
}
```

Класс PhoneNet

```
import java.awt.*;
import java.applet.*;

/**Главный апплет.*/
public class PhoneNet extends Applet
{
  /**Первый телефон*/
  Phone phone1;

  /**Второй телефон*/
  Phone phone2;

  /**Инициализация компонентов и их расположение на панели апплета*/
  public void init()
  {
    Context ctx;

    super.init();
    this.setLayout(new BorderLayout(10, 0));

    ctx = new Context(this);

    String phone1Name = ctx.getParameter("Phone1Name");
    String phone2Name = ctx.getParameter("Phone2Name");

    ATS ats1, ats2;
    String ats1Name = ctx.getParameter("ATS1Name");
    String ats2Name = ctx.getParameter("ATS2Name");

    phone1 = new Phone(phone1Name, ctx);
    phone2 = new Phone(phone2Name, ctx);
    ats1 = new ATS(ats1Name, ctx);
    ats2 = new ATS(ats2Name, ctx);

    ctx.addAutomata(new A0(phone1Name, phone1, ctx));
    ctx.addAutomata(new A0(phone2Name, phone2, ctx));
    ctx.addAutomata(new A1(ats1Name, ats1, ctx));
    ctx.addAutomata(new A1(ats2Name, ats2, ctx));


    add(phone1, BorderLayout.EAST);
    add(phone2, BorderLayout.WEST);
    Panel pnlATS = new Panel(new GridLayout(4, 1, 0, 10));
    pnlATS.add(ats2);
    pnlATS.add(ats1);

    add(pnlATS, BorderLayout.CENTER);
  }
}
```

Приложение 2. Документ, сформированый программой *JavaDoc*

Class BaseAutomata

`java.lang.Object`

`BaseAutomata`

All Implemented Interfaces:
java.lang.Runnable

Direct Known Subclasses:
A0, A1

---

public abstract class **BaseAutomata**
extends java.lang.Object
implements java.lang.Runnable
Базовый автоматный класс. Реализует отложенный вызов автомата.

Потомки должны реализовать методы `caseBlock(int p_Event)`, `stateChanged(int p_intOldState)`, Также потомки должны инициализировать `m_strStateNames`, `m_strEventNames`, `m_strActionNames`, `m_strVarNames`.

Field Summary	
protected Context	**ctx** Контекст проекта.
private boolean	**m_blnProcessingStarted** Происходит ли обработка события
private java.util.Vector	**m_events** Очередь входных событий автомата.
protected int	**m_intDelay** Временной интервал между обработками событий.
protected java.lang.String[]	**m_strActionNames** Список действий автомата.
private java.lang.String	**m_strCaller** Автомат, вызвавший текущий автомат
private java.lang.String	**m_strCurrState** Строка "Текущее состояние".
private java.lang.String	**m_strEndProcess** Строка "Начало конец события".
private java.lang.String	**m_strEventFired** Строка "Пришло событие".
protected java.lang.String[]	**m_strEventNames** Список событий автомата.
private	**m_strFalse**

java.lang.String	Строка "Ложь".
protected java.lang.String	**m_strName** Уникальное в пределах контекста название автомата.
private java.lang.String	**m_strNewState** Строка "Новое состояние".
private java.lang.String	**m_strStartProcess** Строка "Начало обработки события".
protected java.lang.String[]	**m_strStateNames** Список состояний автомата.
private java.lang.String	**m_strTrue** Строка "Истина".
protected java.lang.String[]	**m_strVarNames** Список переменных автомата.
private java.lang.String	**m_strX** Строка "Проверить условие".
private java.lang.String	**m_strZ** Строка "Выполнить действие".
private java.lang.Thread	**m_threadGo** Нить автомата.
protected int	**y** Текущее состояние автомата.

Constructor Summary

BaseAutomata(java.lang.String p_strName, Context p_ctx) Конструктор

Method Summary

void	**A**(BaseEvent p_Event) Добавляет событие в очередь событий.
protected abstract void	**caseBlock**(int p_Event) Основная логика автомата.
protected java.lang.String	**getCaller**() Возвращает название вызвавшего автомата
java.lang.String	**getName**() Возвращает имя автомата.
void	**run**() Запускает caseBlock(int p_Event) по-очереди для каждого события из очереди событий.
private void	**start**() Запускает автомат.
protected abstract void	**stateChanged**(int p_intOldState) Запускается каждый раз, когда автомат перешел в новое состояние.
private void	**stop**() Останавливает работу автомата.

protected boolean	x(int p_intNumber) Вычисляет значение условия xXX данного класса.
protected void	z(int p_intNumber) Вызывает метод zXX данного класса.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

ctx

protected Context **ctx**

Контекст проекта.

---

m_strCaller

private java.lang.String **m_strCaller**

Автомат, вызвавший текущий автомат

---

m_blnProcessingStarted

private boolean **m_blnProcessingStarted**

Происходит ли обработка события

---

m_strStateNames

protected java.lang.String[] **m_strStateNames**

Список состояний автомата.

Note: Должен быть инициализирован в потомках.

---

m_strEventNames

protected java.lang.String[] **m_strEventNames**

Список событий автомата.

Note: Должен быть инициализирован в потомках.

---

m_strActionNames

protected java.lang.String[] **m_strActionNames**

Список действий автомата.

Note: Должен быть инициализирован в потомках.

---

m_strVarNames

`protected java.lang.String[] m_strVarNames`

Список переменных автомата.
Note: Должен быть инициализирован в потомках.

---

m_strName

`protected java.lang.String m_strName`

Уникальное в пределах контекста название автомата.

---

m_intDelay

`protected int m_intDelay`

Временной интервал между обработками событий.
Note: Должен быть инициализирован в потомках.

---

y

`protected int y`

Текущее состояние автомата.

---

m_events

`private java.util.Vector m_events`

Очередь входных событий автомата.

---

m_threadGo

`private java.lang.Thread m_threadGo`

Нить автомата. Служебная переменная для корректного запуска и остановки автомата.

---

m_strEventFired

`private java.lang.String m_strEventFired`

Строка "Пришло событие".

---

m_strStartProcess

`private java.lang.String m_strStartProcess`

Строка "Начало обработки события".

---

m_strEndProcess

```
private java.lang.String m_strEndProcess
```
Строка "Начало конец события".

---

m_strCurrState

```
private java.lang.String m_strCurrState
```
Строка "Текущее состояние".

---

m_strNewState

```
private java.lang.String m_strNewState
```
Строка "Новое состояние".

---

m_strZ

```
private java.lang.String m_strZ
```
Строка "Выполнить действие".

---

m_strX

```
private java.lang.String m_strX
```
Строка "Проверить условие".

---

m_strTrue

```
private java.lang.String m_strTrue
```
Строка "Истина".

---

m_strFalse

```
private java.lang.String m_strFalse
```
Строка "Ложь".

Constructor Detail

BaseAutomata

```
public BaseAutomata(java.lang.String p_strName,
                    Context p_ctx)
```
Конструктор

Parameters:

`p_strName` - Имя автомата.

`p_ctx` - Контекст.

Method Detail

A

```
public void A(BaseEvent p_Event)
```

Добавляет событие в очередь событий.

Parameters:

`p_Event` - Событие.

---

start

```
private void start()
```

Запускает автомат. После обработки всех событий из очереди событий, автомат "засыпает" (`stop()`). По приходу какого-либо события, автомат снова начинает работу (`start()`).

---

stop

```
private void stop()
```

Останавливает работу автомата. После обработки всех событий из очереди событий, автомат * "засыпает" (`stop()`). По приходу какого-либо события, автомат снова * начинает работу (`start()`).

---

run

```
public void run()
```

Запускает `caseBlock(int p_Event)` по-очереди для каждого события из очереди событий. После обработки последнего - останавливается.

Specified by:

`run` in interface `java.lang.Runnable`

---

z

```
protected void z(int p_intNumber)
          throws java.lang.Exception
```

Вызывает метод zXX данного класса. XX - номер, передаваемый в переменной

Parameters:

`p_intNumber` - номер вызываемого метода

Throws:

`java.lang.Exception`

---

x

```
protected boolean x(int p_intNumber)
             throws java.lang.Exception
```

Вычисляет значение условия xXX данного класса. XX - номер, передаваемый в переменной

Parameters:

`p_intNumber` - номер вычисляемого значения

Throws:

`java.lang.Exception`

---

caseBlock

```
protected abstract void caseBlock(int p_Event)
                          throws java.lang.Exception
```

Основная логика автомата.

Parameters:

`p_Event` - Входное событие.

Throws:

`java.lang.Exception`

---

stateChanged

```
protected abstract void stateChanged(int p_intOldState)
```

Запускается каждый раз, когда автомат перешел в новое состояние.

Parameters:

`p_intOldState` - Старое состояние.

---

getCaller

```
protected java.lang.String getCaller()
                            throws java.lang.Exception
```

Возвращает название вызвавшего автомата

Returns:

название вызвавшего автомата

Throws:

`java.lang.Exception`

---

getName

```
public java.lang.String getName()
```

Возвращает имя автомата.

Returns:

Имя автомата.

Class BaseEvent

`java.lang.Object`

`BaseEvent`

---

public class **BaseEvent**
extends java.lang.Object
Елементарное событие для автоматного взаимодействия

Field Summary

`private int`	**`m_event`** Номер события

private java.lang.String	**m_strCaller** Название вызывающего автомата

Constructor Summary

BaseEvent(int p_event, java.lang.String p_strCaller) Конструктор

Method Summary

java.lang.String	**getCaller**() Возвращает название вызывающего автомата
int	**getEvent**() Возвращает номер события

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

m_event

```
private int m_event
```

Номер события

---

m_strCaller

```
private java.lang.String m_strCaller
```

Название вызывающего автомата

Constructor Detail

BaseEvent

```
public BaseEvent(int p_event,
                 java.lang.String p_strCaller)
```

Конструктор

Method Detail

getEvent

```
public int getEvent()
```

Возвращает номер события

---

getCaller

public java.lang.String **getCaller**()
Возвращает название вызывающего автомата

Class A0

java.lang.Object

BaseAutomata

A0

All Implemented Interfaces:
java.lang.Runnable

---

public class **A0**
extends BaseAutomata
Автоматный класс, реализующий работу кнопочного телефона.

Field Summary

private int[]	**m_digits** Набранный номер
private int	**m_intLength** Количество набранных цифр номера
private Phone	**m_phone** Интерфейс телефона
private java.lang.String	**m_strATSName** Название АТС, с которой соединен автомат.

Fields inherited from class BaseAutomata

ctx, m_intDelay, m_strActionNames, m_strEventNames, m_strName, m_strStateNames, m_strVarNames, y

Constructor Summary

A0(java.lang.String p_strName, Phone p_phone, Context p_ctx) Конструктор.

Method Summary

private void	**A0**(int e) Запускает A0 с событием e.
private void	**A1**(int e) Запускает A1 с событием e.
protected void	**caseBlock**(int p_Event) Основная логика автомата.
protected void	**stateChanged**(int p_intOldState) Запускается каждый раз, когда автомат перешел в новое состояние.
protected	**x10**()

boolean	x1X(int x), x = 0
protected boolean	**x11**() x1X(int x), x = 1
protected boolean	**x12**() x1X(int x), x = 2
protected boolean	**x13**() x1X(int x), x = 3
protected boolean	**x14**() x1X(int x), x = 4
protected boolean	**x15**() x1X(int x), x = 5
protected boolean	**x16**() x1X(int x), x = 6
protected boolean	**x17**() x1X(int x), x = 7
protected boolean	**x18**() x1X(int x), x = 8
protected boolean	**x19**() x1X(int x), x = 9
private boolean	**x1X**(int x) Текущая цифра - цифра x
protected boolean	**x2**() Есть ли еще цифры.
protected void	**z10**() z1X(int x), x = 0
protected void	**z11**() z1X(int x), x = 1
protected void	**z12**() z1X(int x), x = 2
protected void	**z13**() z1X(int x), x = 3
protected void	**z14**() z1X(int x), x = 4
protected void	**z15**() z1X(int x), x = 5
protected void	**z16**() z1X(int x), x = 6
protected void	**z17**() z1X(int x), x = 7
protected void	**z18**() z1X(int x), x = 8
protected void	**z19**() z1X(int x), x = 9
private void	**z1X**(int x) Добавить цифру к набранному к данному моменту номеру.

protected void	**z20**() z2X(int x), x = 0
protected void	**z21**() z2X(int x), x = 1
protected void	**z22**() z2X(int x), x = 2
protected void	**z23**() z2X(int x), x = 3
protected void	**z24**() z2X(int x), x = 4
protected void	**z25**() z2X(int x), x = 5
protected void	**z26**() z2X(int x), x = 6
protected void	**z27**() z2X(int x), x = 7
protected void	**z28**() z2X(int x), x = 8
protected void	**z29**() z2X(int x), x = 9
private void	**z2X**(int x) Добавить цифрц на дисплей.
protected void	**z3**() Удалить первую цифру из набранного к данному моменту номера.
protected void	**z4**() Очистить набранный к данному моменту номер.
protected void	**z5**() Очистить дисплей.
protected void	**z6**() Занять АТС.
protected void	**z7**() Освободить АТС.
protected void	**z8**() Подтвердить соединение с АТС
protected void	**z9**() Получить название АТС

Methods inherited from class BaseAutomata
A, getCaller, getName, run, x, z

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

m_strATSName

```
private java.lang.String m_strATSName
```

Название АТС, с которой соединен автомат. Не null только во время набора номера и звонка

---

m_digits

```
private int[] m_digits
```

Набранный номер

---

m_intLength

```
private int m_intLength
```

Количество набранных цифр номера

---

m_phone

```
private Phone m_phone
```

Интерфейс телефона

Constructor Detail

A0

```
public A0(java.lang.String p_strName,
          Phone p_phone,
          Context p_ctx)
```

Конструктор.

Parameters:

`p_strName` - Название автомата (номер телефона).

`p_phone` - Интерфейс телефона.

`p_ctx` - Контекст проекта.

Method Detail

caseBlock

```
protected void caseBlock(int p_Event)
                  throws java.lang.Exception
```

Description copied from class: BaseAutomata

Основная логика автомата.

Specified by:

caseBlock in class BaseAutomata

Parameters:

`p_Event` - Входное событие.

Throws:

`java.lang.Exception`

---

stateChanged

```
protected void stateChanged(int p_intOldState)
```

Description copied from class: BaseAutomata

Запускается каждый раз, когда автомат перешел в новое состояние.

Specified by:

stateChanged in class BaseAutomata

Parameters:

p_intOldState - Старое состояние.

---

z1X

```
private void z1X(int x)
```

Добавить цифру к набранному к данному моменту номеру. Можно набрать номер, а затем поднять трубку. В таком случае текущий набранный номер будет посылаться АТС.

Parameters:

x - Цифра.

---

z2X

```
private void z2X(int x)
```

Добавить цифрц на дисплей.

Parameters:

x - Цифра.

---

z3

```
protected void z3()
```

Удалить первую цифру из набранного к данному моменту номера.

---

z4

```
protected void z4()
```

Очистить набранный к данному моменту номер.

---

z5

```
protected void z5()
```

Очистить дисплей.

---

z6

```
protected void z6()
```

Занять АТС.

---

z7

```
protected void z7()
          throws java.lang.Exception
```

Освободить АТС.

Throws:

`java.lang.Exception`

---

z8

```
protected void z8()
          throws java.lang.Exception
```

Подтвердить соединение с АТС

Throws:

`java.lang.Exception`

---

z9

```
protected void z9()
          throws java.lang.Exception
```

Получить название АТС

Throws:

`java.lang.Exception`

---

z10

```
protected void z10()
```

`z1X(int x)`, x = 0

---

z11

```
protected void z11()
```

`z1X(int x)`, x = 1

---

z12

```
protected void z12()
```

`z1X(int x)`, x = 2

---

z13

```
protected void z13()
```

`z1X(int x)`, x = 3

---

z14

```
protected void z14()
```
z1X(int x), x = 4

---

z15

```
protected void z15()
```
z1X(int x), x = 5

---

z16

```
protected void z16()
```
z1X(int x), x = 6

---

z17

```
protected void z17()
```
z1X(int x), x = 7

---

z18

```
protected void z18()
```
z1X(int x), x = 8

---

z19

```
protected void z19()
```
z1X(int x), x = 9

---

z20

```
protected void z20()
```
z2X(int x), x = 0

---

z21

```
protected void z21()
```
z2X(int x), x = 1

---

z22

```
protected void z22()
```
z2X(int x), x = 2

---

z23

```
protected void z23()
```

z2X(int x), x = 3

---

z24

```
protected void z24()
```

z2X(int x), x = 4

---

z25

```
protected void z25()
```

z2X(int x), x = 5

---

z26

```
protected void z26()
```

z2X(int x), x = 6

---

z27

```
protected void z27()
```

z2X(int x), x = 7

---

z28

```
protected void z28()
```

z2X(int x), x = 8

---

z29

```
protected void z29()
```

z2X(int x), x = 9

---

x1X

```
private boolean x1X(int x)
```

Текущая цифра - цифра x

Parameters:

x - - проверяемая цифра.

Returns:

Текущая цифра из буфера совпадает с x.

---

x2

```
protected boolean x2()
```

Есть ли еще цифры.

Returns:

Есть ли еще цифры в буфере телефона.

---

x10

```
protected boolean x10()
```

x1X(int x), x = 0

---

x11

```
protected boolean x11()
```

x1X(int x), x = 1

---

x12

```
protected boolean x12()
```

x1X(int x), x = 2

---

x13

```
protected boolean x13()
```

x1X(int x), x = 3

---

x14

```
protected boolean x14()
```

x1X(int x), x = 4

---

x15

```
protected boolean x15()
```

x1X(int x), x = 5

---

x16

```
protected boolean x16()
```

x1X(int x), x = 6

---

x17

```
protected boolean x17()
```
x1X(int x), x = 7

---

x18

```
protected boolean x18()
```
x1X(int x), x = 8

---

x19

```
protected boolean x19()
```
x1X(int x), x = 9

A0

```
private void A0(int e)
```
Запускает A0 с событием e.

Parameters:

e - Событие.

A1

```
private void A1(int e)
```
Запускает A1 с событием e.

Parameters:

e - Событие.

Class A1

```
java.lang.Object
```

BaseAutomata

A1

All Implemented Interfaces:

java.lang.Runnable

---

public class **A1**
extends BaseAutomata
Автоматный класс, реализующий работу АТС.

Field Summary

private ATS	**m_ats** Объект, который будет отображать АТС
private java.lang.String	**m_strFirstSubscriber** Номер телефона вызывающего абонента.

private java.lang.String	**m_strNumber** Набранная к данному моменту часть номера.
private java.lang.String	**m_strSecondSubscriber** Номер телефона вызываемого абонента.

Fields inherited from class BaseAutomata
ctx, m_intDelay, m_strActionNames, m_strEventNames, m_strName, m_strStateNames, m_strVarNames, y

Constructor Summary

A1(java.lang.String p_strName, ATS p_ats, Context p_ctx) Конструктор.

Method Summary

private void	**A0**(int p_intEvent) Послать событие p_intEvent автомату A0, реализующему телефон вызывающего абонента.
private void	**A1**(int p_intEvent) Послать событие p_intEvent самому себе.
private void	**A2**(int p_intEvent) Послать событие p_intEvent автомату A0, реализующему телефон вызвывамого абонента.
protected void	**caseBlock**(int p_Event) Основная логика автомата.
protected void	**stateChanged**(int p_OldState) Запускается каждый раз, когда автомат перешел в новое состояние.
protected boolean	**x1**() Валидный номер.
protected boolean	**x2**() Вызываемый абонент свободен.
protected void	**z10**() z1X(int x), x = 0
protected void	**z11**() z1X(int x), x = 1
protected void	**z12**() z1X(int x), x = 2
protected void	**z13**() z1X(int x), x = 3
protected void	**z14**() z1X(int x), x = 4
protected void	**z15**() z1X(int x), x = 5
protected void	**z16**() z1X(int x), x = 6
protected	**z17**()

void	z1X(int x), x = 7
protected void	**z18**() z1X(int x), x = 8
protected void	**z19**() z1X(int x), x = 9
private void	**z1X**(int x) Добавить цифру к m_strNumber.
protected void	**z2**() Занять телефон вызывающего абонента.
protected void	**z3**() Занять телефон вызываемого абонента.
protected void	**z4**() Освободить телефон вызывающего абонента.
protected void	**z5**() Освободить телефон вызываемого абонента.
protected void	**z6**() Сбросить набранный к данному моменту номер.

Methods inherited from class BaseAutomata
A, getCaller, getName, run, x, z

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

m_ats

private ATS **m_ats**

Объект, который будет отображать АТС

---

m_strFirstSubscriber

private java.lang.String **m_strFirstSubscriber**

Номер телефона вызывающего абонента.

---

m_strSecondSubscriber

private java.lang.String **m_strSecondSubscriber**

Номер телефона вызываемого абонента.

---

m_strNumber

```
private java.lang.String m_strNumber
```
Набранная к данному моменту часть номера.

Constructor Detail

A1

```
public A1(java.lang.String p_strName,
          ATS p_ats,
          Context p_ctx)
```
Конструктор.

Parameters:

`p_strName` - Название АТС.

`p_ats` - Интерфейс АТС.

`p_ctx` - Контекст проекта.

Method Detail

caseBlock

```
protected void caseBlock(int p_Event)
                  throws java.lang.Exception
```
Description copied from class: `BaseAutomata`

Основная логика автомата.

Specified by:

`caseBlock` in class `BaseAutomata`

Parameters:

`p_Event` - Входное событие.

Throws:

`java.lang.Exception`

---

stateChanged

```
protected void stateChanged(int p_OldState)
```
Description copied from class: `BaseAutomata`

Запускается каждый раз, когда автомат перешел в новое состояние.

Specified by:

`stateChanged` in class `BaseAutomata`

Parameters:

`p_OldState` - Старое состояние.

---

z1X

```
private void z1X(int x)
```
Добавить цифру к `m_strNumber`.

Parameters:

`x` - Цифра.

---

z10

```
protected void z10()
```
z1X(int x), x = 0

---

z11

```
protected void z11()
```
z1X(int x), x = 1

---

z12

```
protected void z12()
```
z1X(int x), x = 2

---

z13

```
protected void z13()
```
z1X(int x), x = 3

---

z14

```
protected void z14()
```
z1X(int x), x = 4

---

z15

```
protected void z15()
```
z1X(int x), x = 5

---

z16

```
protected void z16()
```
z1X(int x), x = 6

---

z17

```
protected void z17()
```
z1X(int x), x = 7

---

z18

```
protected void z18()
```
z1X(int x), x = 8

---

z19

```
protected void z19()
```

z1X(int x), x = 9

---

z2

```
protected void z2()
          throws java.lang.Exception
```

Занять телефон вызывающего абонента.

Throws:

`java.lang.Exception`

---

z3

```
protected void z3()
```

Занять телефон вызываемого абонента.

---

z4

```
protected void z4()
```

Освободить телефон вызывающего абонента.

---

z5

```
protected void z5()
```

Освободить телефон вызываемого абонента.

---

z6

```
protected void z6()
```

Сбросить набранный к данному моменту номер.

---

x1

```
protected boolean x1()
```

Валидный номер.

---

x2

```
protected boolean x2()
```

Вызываемый абонент свободен.

A0

`private void A0(int p_intEvent)`

Послать событие p_intEvent автомату A0, реализующему телефон вызывающего абонента.

Parameters:

`p_intEvent` - Событие.

A2

`private void A2(int p_intEvent)`

Послать событие p_intEvent автомату A0, реализующему телефон вызвывамого абонента.

Parameters:

`p_intEvent` - Событие.

A1

`private void A1(int p_intEvent)`

Послать событие p_intEvent самому себе.

Parameters:

`p_intEvent` - Событие.

Class Context

`java.lang.Object`

`Context`

---

public class **Context**
extends java.lang.Object
Контекст проекта. Содержит названия всех автоматов,

Field Summary

private java.applet.Applet	**m_appMain** Запускающий апплет.
java.util.Vector	**m_busyAutomatas** Список занятых автоматов.
private java.util.Hashtable	**m_hashAutomatas** Список автоматов.
java.util.Hashtable	**m_hashFreeATS** Свободные АТС.
private Logger	**m_logger** Логер для логирования ошибок.

Constructor Summary

Context(java.applet.Applet p_app)

Конструктор.

Method Summary

void	**addAutomata**(BaseAutomata p_automata) Добавляет автомат в контекст проекта.
BaseAutomata	**getAutomata**(java.lang.String p_strAutomataName) Возвращает автомат с заданным именем.
java.lang.String	**getFreeATS**() Возвращает название свободной АТС.
Logger	**getLogger**() Возвращает экземпляр класса, обеспечивающий логинирование.
Logger	**getLogger**(java.lang.String p_strLoggerName) Возвращает экземпляр класса, обеспечивающий логинирование.
java.lang.String	**getParameter**(java.lang.String p_strParameterName) Возвращвет параметр с заданным именем.
void	**occupyATS**(java.lang.String p_strATSName) Занять АТС
void	**setATSFree**(java.lang.String p_strATSName) Освобождает АТС.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

m_logger

```
private Logger m_logger
```

Логер для логирования ошибок.

---

m_hashAutomatas

```
private java.util.Hashtable m_hashAutomatas
```

Список автоматов.

---

m_appMain

```
private java.applet.Applet m_appMain
```

Запускающий апплет.

---

m_busyAutomatas

```
public java.util.Vector m_busyAutomatas
```

Список занятых автоматов.

---

m_hashFreeATS

`public java.util.Hashtable` **`m_hashFreeATS`**

Свободные АТС.

Constructor Detail

Context

`public` **`Context`**`(java.applet.Applet p_app)`

Конструктор.

Parameters:

`p_app` - Главный апплет приложения.

Method Detail

getLogger

`public Logger` **`getLogger`**`(java.lang.String p_strLoggerName)`

Возвращает экземпляр класса, обеспечивающий логинирование.

Parameters:

`p_strLoggerName` - Главный апплет приложения. В данной реализации не функционален.

Returns:

Экземпляр класса `Logger`.

---

getLogger

`public Logger` **`getLogger`**`()`

Возвращает экземпляр класса, обеспечивающий логинирование.

Returns:

Экземпляр класса `Logger`.

---

addAutomata

`public void` **`addAutomata`**`(BaseAutomata p_automata)`

Добавляет автомат в контекст проекта.

Parameters:

`p_automata` - Добавляемый автомат.

---

getAutomata

`public BaseAutomata` **`getAutomata`**`(java.lang.String p_strAutomataName)`

Возвращает автомат с заданным именем.

Returns:

Автомат с указанным именем.

getParameter

```
public java.lang.String getParameter(java.lang.String p_strParameterName)
```

Возвращвет параметр с заданным именем.

Parameters:

`p_strParameterName` - Название параметра.

Returns:

Параметр.

---

getFreeATS

```
public java.lang.String getFreeATS()
```

Возвращает название свободной АТС. Может быть использовано в `getAutomata(String)` в качестве параметра.

Returns:

Название свободной АТС.

---

occupyATS

```
public void occupyATS(java.lang.String p_strATSName)
               throws java.lang.Exception
```

Занять АТС

Parameters:

`p_strATSName` - Название АТС

Throws:

`java.lang.Exception`

---

setATSFree

```
public void setATSFree(java.lang.String p_strATSName)
                throws java.lang.Exception
```

Освобождает АТС.

Parameters:

`p_strATSName` - Название АТС

Throws:

`java.lang.Exception`

Class Logger

```
java.lang.Object
```

`Logger`

---

public class **Logger**
extends java.lang.Object
Класс обеспечивает логирование.

Field Summary

private java.lang.String	**m_strName** Имя логера.

Constructor Summary

Logger(java.lang.String p_strName) Конструктор.

Method Summary

void	**log**(java.lang.String p_strMessage) Логирует сообщение.
void	**log**(java.lang.String p_strMessage, java.lang.String p_strSource) Логирует сообщение.
void	**log**(java.lang.String p_strMessage, java.lang.String p_strSource, int p_intLevel) Логирует сообщение.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

m_strName

```
private java.lang.String m_strName
```

Имя логера. В данной реализации не функционально

Constructor Detail

Logger

```
public Logger(java.lang.String p_strName)
```

Конструктор.

Parameters:

p_strName - Имя логгера. В данной реализации не функционально.

Method Detail

log

```
public void log(java.lang.String p_strMessage,
                java.lang.String p_strSource,
                int p_intLevel)
```

Логирует сообщение.

Parameters:

p_strMessage - Сообщение для логирования.

`p_intLevel` - Уровень вложенности сообщения.
`p_strSource` - Источник сообщения.

---

log

```
public void log(java.lang.String p_strMessage)
```

Логирует сообщение. Уровень вложенности 0, Название источника сообщения - пустая строка.
Parameters:
`p_strMessage` - Сообщение для логирования.

---

log

```
public void log(java.lang.String p_strMessage,
                java.lang.String p_strSource)
```

Логирует сообщение. Уровень вложенности сообщения 0.
Parameters:
`p_strMessage` - Сообщение для логирования.
`p_strSource` - Источник сообщения.

Class Phone

```
java.lang.Object

    java.awt.Component

        java.awt.Container

            java.awt.Panel

                Phone
```

All Implemented Interfaces:
javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.lang.Runnable, java.io.Serializable

---

public class **Phone**
extends java.awt.Panel
implements java.awt.event.ActionListener, java.lang.Runnable
Интерфейс телефона.

Состоит из:

- слова "Телефон";
- номера телефона;
- дисплея;
- стандартной кнопочной клавиатуры;
- кнопки "сброс";
- кнопки "поднять трубку";

- строки состояния.

See Also:
Serialized Form

Nested Class Summary

Nested classes inherited from class java.awt.Panel
java.awt.Panel.AccessibleAWTPanel

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

private Context	**ctx** Контекст проекта.
private java.applet.AudioClip	**m_audio** Аудио для проигрывания звонка
private java.awt.Button[]	**m_btnDigits** Стандартная кнопочная клавиатура.
private java.awt.Button	**m_btnReciever** Кнопка "поднять трубку."
private java.awt.Button	**m_btnReset** Кнопка "сброс".
private java.awt.Label	**m_lblDisplay** Дисплей.
private java.awt.Label	**m_lblState** Строка состояния.
private java.lang.String	**m_strAutomataName** Номер телефона.
private java.lang.String	**m_strRecieverDown** Заголовок для кнопки "опустить трубку".
private java.lang.String	**m_strRecieverUp** Заголовок для кнопки "поднять трубку".
private java.lang.Thread	**m_threadGo** Поток для визуализации звонка

Fields inherited from class java.awt.Panel

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

Phone(java.lang.String p_strAutomataName, Context p_ctx) Конструктор.	

Method Summary

private void	**A**(int p_event) Запускает автомат A0.
void	**actionPerformed**(java.awt.event.ActionEvent e)
void	**displayAddString**(java.lang.String p_s) Добавляет цифры к дисплею.
void	**displayReset**() Сбрасываетнабранный на дисплее номер.
private void	**initControls**() Инициализирует и размещает компоненты.
void	**run**() Поток визуализации звонка
void	**setState**(int p_intState, java.lang.String p_stateName, boolean isRecieverUp) Меняет строку состояния и надпись на кнопке "поднять трубку".

Methods inherited from class java.awt.Panel
addNotify, getAccessibleContext

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, addImpl, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, paramString, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove,

removeAll, removeContainerListener, removeNotify, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, setLayout, transferFocusBackward, transferFocusDownCycle, update, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

ctx

```
private Context ctx
```

Контекст проекта.

---

m_btnDigits

```
private java.awt.Button[] m_btnDigits
```
Стандартная кнопочная клавиатура.

---

m_btnReset

```
private java.awt.Button m_btnReset
```
Кнопка "сброс".

---

m_btnReciever

```
private java.awt.Button m_btnReciever
```
Кнопка "поднять трубку."

---

m_lblDisplay

```
private java.awt.Label m_lblDisplay
```
Дисплей.

---

m_lblState

```
private java.awt.Label m_lblState
```
Строка состояния.

---

m_strAutomataName

```
private java.lang.String m_strAutomataName
```
Номер телефона.

---

m_strRecieverUp

```
private java.lang.String m_strRecieverUp
```
Заголовок для кнопки "поднять трубку".

---

m_strRecieverDown

```
private java.lang.String m_strRecieverDown
```
Заголовок для кнопки "опустить трубку".

---

m_threadGo

```
private java.lang.Thread m_threadGo
```
Поток для визуализации звонка

---

m_audio

```
private java.applet.AudioClip m_audio
```
Аудио для проигрывания звонка

Constructor Detail

Phone

```
public Phone(java.lang.String p_strAutomataName,
             Context p_ctx)
```
Конструктор.

Parameters:

`p_strAutomataName` - Номер телефона.

`p_ctx` - Клнтекст проекта.

Method Detail

initControls

```
private void initControls()
```
Инициализирует и размещает компоненты.

---

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```
Specified by:

`actionPerformed` in interface `java.awt.event.ActionListener`

---

displayAddString

```
public void displayAddString(java.lang.String p_s)
```
Добавляет цифры к дисплею.

Parameters:

`p_s` - Цифры.

---

displayReset

```
public void displayReset()
```
Сбрасываетнабранный на дисплее номер.

---

setState

```
public void setState(int p_intState,
                     java.lang.String p_stateName,
                     boolean isRecieverUp)
```
Меняет строку состояния и надпись на кнопке "поднять трубку".

Parameters:

`p_intState` - Номер состояния

`p_stateName` - Название состояния.

`isRecieverUp` - Поднята ли трубка.

run

`public void run()`

Поток визуализации звонка

Specified by:

`run` in interface `java.lang.Runnable`

A

`private void A(int p_event)`

Запускает автомат `A0`.

Parameters:

`p_event` - Событие.

Class ATS

```
java.lang.Object

    java.awt.Component

        java.awt.Container

            java.awt.Panel

                ATS
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

---

public class **ATS**
extends java.awt.Panel

Интерфейс АТС. Состоит из строки состояния и строки набранного к данному моменту номера.

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class java.awt.Panel
`java.awt.Panel.AccessibleAWTPanel`

Nested classes inherited from class java.awt.Container
`java.awt.Container.AccessibleAWTContainer`

Nested classes inherited from class java.awt.Component

java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

private Context	**ctx** Контекст проекта.
private java.awt.Label	**m_lblNumber** Набранный к данному моменту номер.
private java.awt.Label	**m_lblState** Строка состояния.
private java.lang.String	**m_strAutomataName** Название АТС, которую визуализирует данный класс.

Fields inherited from class java.awt.Panel

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

ATS(java.lang.String p_strAutomataName, Context p_ctx) Конструктор.

Method Summary

private void	**initControls**() Инициализирует строки и расставляет их.
void	**setNumber**(java.lang.String p_strNumber) Устанавливает новое значение строки набранного к данному моменту номера.
void	**setState**(java.lang.String p_stateName) Устанавливает новое значение строки состояния.

Methods inherited from class java.awt.Panel

addNotify, getAccessibleContext

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addImpl, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, paramString, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, removeNotify, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, setLayout, transferFocusBackward, transferFocusDownCycle, update, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

m_lblState

```
private java.awt.Label m_lblState
```
Строка состояния.

---

m_lblNumber

```
private java.awt.Label m_lblNumber
```
Набранный к данному моменту номер.

---

m_strAutomataName

```
private java.lang.String m_strAutomataName
```
Название АТС, которую визуализирует данный класс.

---

ctx

```
private Context ctx
```
Контекст проекта.

Constructor Detail

ATS

```
public ATS(java.lang.String p_strAutomataName,
           Context p_ctx)
```
Конструктор.

Parameters:

`p_strAutomataName` - Название АТС, которую визуализирует данный класс.

`p_ctx` - Контекст проекта.

Method Detail

initControls

```
private void initControls()
```
Инициализирует строки и расставляет их.

---

setState

```
public void setState(java.lang.String p_stateName)
```
Устанавливает новое значение строки состояния.

Parameters:

`p_stateName` - Название нового состояния.

---

setNumber

`public void` **`setNumber`**`(java.lang.String p_strNumber)`

Устанавливает новое значение строки набранного к данному моменту номера.

Parameters:

`p_strNumber` - Набранный к данному моменту номер.

Class PhoneNet

```
java.lang.Object

      java.awt.Component

          java.awt.Container

              java.awt.Panel

                  java.applet.Applet

                      PhoneNet
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

---

public class **PhoneNet**
extends java.applet.Applet
Главный апплет.

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class java.applet.Applet
`java.applet.Applet.AccessibleApplet`

Nested classes inherited from class java.awt.Panel
`java.awt.Panel.AccessibleAWTPanel`

Nested classes inherited from class java.awt.Container
`java.awt.Container.AccessibleAWTContainer`

Nested classes inherited from class java.awt.Component
`java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy`

Field Summary

(package private) Phone	**phone1** Первый телефон
(package private) Phone	**phone2** Второй телефон

Fields inherited from class java.applet.Applet

Fields inherited from class java.awt.Panel

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

PhoneNet()

Method Summary

void	**init**() Инициализация компонентов и их расположение на панели аппдета

Methods inherited from class java.applet.Applet
destroy, getAccessibleContext, getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameter, getParameterInfo, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus, start, stop

Methods inherited from class java.awt.Panel
addNotify

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, addImpl, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents,

deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, paramString, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, removeNotify, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, setLayout, transferFocusBackward, transferFocusDownCycle, update, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

phone1

Phone **phone1**
Первый телефон

---

phone2

Phone **phone2**
Второй телефон

Constructor Detail

PhoneNet

```
public PhoneNet()
```

Method Detail

init

```
public void init()
```
Инициализация компонентов и их расположение на панели аппдета

Приложение 3. Пример log-файла

```
19-01-05 02.16.921  Context :   Add automata 1234567
10-12-04 02.16.937  Context :   Add automata 7654321
10-12-04 02.16.937  Context :   Add automata ATS1
10-12-04 02.16.937  Context :   Add automata ATS2
10-12-04 02.16.796  7654321 :   Произошло событие e11 (Нажать цифру 1).
10-12-04 02.16.812  7654321 :   Обработка события e11 (Нажать цифру 1).
10-12-04 02.16.812  7654321 :   Текущее состояние автомата 0 (Ожидание
                                 действий).
10-12-04 02.16.812  7654321 :       Выполнение действия z11 (Добавить 1).
10-12-04 02.16.812  7654321 :       Выполнение действия z21 (Добавить 1).
10-12-04 02.16.812  7654321 :   Автомат перешел в состояние 1 (Редактирование
                                 номера).
10-12-04 02.16.812  7654321 :   Конец обработки события e11 (Нажать цифру 1).
10-12-04 02.16.421  7654321 :   Произошло событие e12 (Нажать цифру 2).
10-12-04 02.16.421  7654321 :   Обработка события e12 (Нажать цифру 2).
10-12-04 02.16.421  7654321 :   Текущее состояние автомата 1 (Редактирование
                                 номера).
10-12-04 02.16.421  7654321 :       Выполнение действия z12 (Добавить 2).
10-12-04 02.16.421  7654321 :       Выполнение действия z22 (Добавить 2).
10-12-04 02.16.421  7654321 :   Автомат перешел в состояние 1 (Редактирование
                                 номера).
10-12-04 02.16.421  7654321 :   Конец обработки события e12 (Нажать цифру 2).
10-12-04 02.16.812  7654321 :   Произошло событие e13 (Нажать цифру 3).
10-12-04 02.16.812  7654321 :   Обработка события e13 (Нажать цифру 3).
10-12-04 02.16.812  7654321 :   Текущее состояние автомата 1 (Редактирование
                                 номера).
10-12-04 02.16.812  7654321 :       Выполнение действия z13 (Добавить 3).
10-12-04 02.16.812  7654321 :       Выполнение действия z23 (Добавить 3).
10-12-04 02.16.827  7654321 :   Автомат перешел в состояние 1 (Редактирование
                                 номера).
10-12-04 02.16.827  7654321 :   Конец обработки события e13 (Нажать цифру 3).
```

```
10-12-04 02.16.312  7654321 :  Произошло событие e14 (Нажать цифру 4).
10-12-04 02.16.312  7654321 :  Обработка события e14 (Нажать цифру 4).
10-12-04 02.16.312  7654321 :  Текущее состояние автомата 1 (Редактирование
                               номера).
10-12-04 02.16.312  7654321 :      Выполнение действия z14 (Добавить 4).
10-12-04 02.16.312  7654321 :      Выполнение действия z24 (Добавить 4).
10-12-04 02.16.312  7654321 :  Автомат перешел в состояние 1 (Редактирование
                               номера).
10-12-04 02.16.312  7654321 :  Конец обработки события e14 (Нажать цифру 4).
10-12-04 02.16.812  7654321 :  Произошло событие e2 (Поднять трубку).
10-12-04 02.16.812  7654321 :  Обработка события e2 (Поднять трубку).
10-12-04 02.16.812  7654321 :  Текущее состояние автомата 1 (Редактирование
                               номера).
10-12-04 02.16.812  7654321 :      Выполнение действия z6 (Занять АТС).
10-12-04 02.16.812  ATS2    :  Произошло событие e2 (Поднять трубку).
10-12-04 02.16.812  7654321 :  Произошло событие e0 (Проверка).
10-12-04 02.16.812  7654321 :  Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.812  7654321 :  Конец обработки события e2 (Поднять трубку).
10-12-04 02.16.812  ATS2    :  Обработка события e2 (Поднять трубку).
10-12-04 02.16.812  ATS2    :  Текущее состояние автомата 0 (Ожидание
                               действий).
10-12-04 02.16.812  ATS2    :      Выполнение действия z2 (Занять
                               вызывающего).
10-12-04 02.16.812  ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.812  ATS2    :  Конец обработки события e2 (Поднять трубку).
10-12-04 02.16.921  7654321 :  Обработка события e0 (Проверка).
10-12-04 02.16.921  7654321 :  Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.921  7654321 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.921  7654321 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.921  7654321 :      Проверика значения переменной x10 (Цифра
                               0). Значение: Ложь.
10-12-04 02.16.921  7654321 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.921  7654321 :      Проверика значения переменной x11 (Цифра
                               1). Значение: Истина.
10-12-04 02.16.921  ATS2    :  Произошло событие e11 (Набрать 1).
10-12-04 02.16.921  7654321 :      Выполнение действия z3 (Следующая цифра).
10-12-04 02.16.921  7654321 :  Произошло событие e0 (Проверка).
10-12-04 02.16.921  7654321 :  Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.921  7654321 :  Конец обработки события e0 (Проверка).
10-12-04 02.16.921  ATS2    :  Обработка события e11 (Набрать 1).
10-12-04 02.16.921  ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.921  ATS2    :      Проверика значения переменной x1
                               (Правильный номер). Значение: Ложь.
10-12-04 02.16.937  ATS2    :      Проверика значения переменной x1
                               (Правильный номер). Значение: Ложь.
10-12-04 02.16.937  ATS2    :      Выполнение действия z11 (Добавить 1).
10-12-04 02.16.937  ATS2    :  Произошло событие e0 (Проверка).
10-12-04 02.16.937  ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.937  ATS2    :  Конец обработки события e11 (Набрать 1).
10-12-04 02.16.30   7654321 :  Обработка события e0 (Проверка).
10-12-04 02.16.30   7654321 :  Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.30   7654321 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.30   7654321 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.30   7654321 :      Проверика значения переменной x10 (Цифра
                               0). Значение: Ложь.
10-12-04 02.16.30   7654321 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.30   7654321 :      Проверика значения переменной x11 (Цифра
                               1). Значение: Ложь.
```

```
10-12-04 02.16.30   7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.30   7654321 :      Проверика значения переменной x12 (Цифра
                                   2). Значение: Истина.
10-12-04 02.16.30   ATS2    :  Произошло событие e12 (Набрать 2).
10-12-04 02.16.30   7654321 :      Выполнение действия z3 (Следующая цифра).
10-12-04 02.16.30   7654321 :  Произошло событие e0 (Проверка).
10-12-04 02.16.30   7654321 :  Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.30   7654321 :  Конец обработки события e0 (Проверка).
10-12-04 02.16.46   ATS2    :  Обработка события e0 (Проверка).
10-12-04 02.16.46   ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.46   ATS2    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.46   ATS2    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.46   ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.46   ATS2    :  Конец обработки события e0 (Проверка).
10-12-04 02.16.140  7654321 :  Обработка события e0 (Проверка).
10-12-04 02.16.140  7654321 :  Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.140  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.140  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.140  7654321 :      Проверика значения переменной x10 (Цифра
                                   0). Значение: Ложь.
10-12-04 02.16.140  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.140  7654321 :      Проверика значения переменной x11 (Цифра
                                   1). Значение: Ложь.
10-12-04 02.16.140  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.140  7654321 :      Проверика значения переменной x12 (Цифра
                                   2). Значение: Ложь.
10-12-04 02.16.140  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.140  7654321 :      Проверика значения переменной x13 (Цифра
                                   3). Значение: Истина.
10-12-04 02.16.140  ATS2    :  Произошло событие e13 (Набрать 3).
10-12-04 02.16.140  7654321 :      Выполнение действия z3 (Следующая цифра).
10-12-04 02.16.140  7654321 :  Произошло событие e0 (Проверка).
10-12-04 02.16.140  7654321 :  Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.140  7654321 :  Конец обработки события e0 (Проверка).
10-12-04 02.16.140  ATS2    :  Обработка события e12 (Набрать 2).
10-12-04 02.16.140  ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.155  ATS2    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.155  ATS2    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.155  ATS2    :      Выполнение действия z12 (Добавить 2).
10-12-04 02.16.155  ATS2    :  Произошло событие e0 (Проверка).
10-12-04 02.16.155  ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.155  ATS2    :  Конец обработки события e12 (Набрать 2).
10-12-04 02.16.249  7654321 :  Обработка события e0 (Проверка).
10-12-04 02.16.249  7654321 :  Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.249  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.249  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.249  7654321 :      Проверика значения переменной x10 (Цифра
                                   0). Значение: Ложь.
10-12-04 02.16.249  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.249  7654321 :      Проверика значения переменной x11 (Цифра
                                   1). Значение: Ложь.
```

```
10-12-04 02.16.265  ATS2    :   Обработка события e13 (Набрать 3).
10-12-04 02.16.265  ATS2    :   Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.265  ATS2    :       Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.265  ATS2    :       Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.265  ATS2    :       Выполнение действия z13 (Добавить 3).
10-12-04 02.16.265  ATS2    :   Произошло событие e0 (Проверка).
10-12-04 02.16.265  ATS2    :   Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.265  ATS2    :   Конец обработки события e13 (Набрать 3).
10-12-04 02.16.249  7654321 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.265  7654321 :       Проверика значения переменной x12 (Цифра
                                2). Значение: Ложь.
10-12-04 02.16.265  7654321 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.265  7654321 :       Проверика значения переменной x13 (Цифра
                                3). Значение: Ложь.
10-12-04 02.16.265  7654321 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.265  7654321 :       Проверика значения переменной x14 (Цифра
                                4). Значение: Истина.
10-12-04 02.16.265  ATS2    :   Произошло событие e14 (Набрать 4).
10-12-04 02.16.265  7654321 :       Выполнение действия z3 (Следующая цифра).
10-12-04 02.16.265  7654321 :   Произошло событие e0 (Проверка).
10-12-04 02.16.265  7654321 :   Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.265  7654321 :   Конец обработки события e0 (Проверка).
10-12-04 02.16.358  ATS2    :   Обработка события e0 (Проверка).
10-12-04 02.16.358  ATS2    :   Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.358  ATS2    :       Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.358  ATS2    :       Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.358  ATS2    :   Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.358  ATS2    :   Конец обработки события e0 (Проверка).
10-12-04 02.16.358  7654321 :   Обработка события e0 (Проверка).
10-12-04 02.16.358  7654321 :   Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.358  7654321 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Ложь.
10-12-04 02.16.358  7654321 :   Автомат перешел в состояние 3 (Поднята трубка).
10-12-04 02.16.358  7654321 :   Конец обработки события e0 (Проверка).
10-12-04 02.16.468  ATS2    :   Обработка события e0 (Проверка).
10-12-04 02.16.468  ATS2    :   Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.468  ATS2    :       Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.468  ATS2    :       Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.468  ATS2    :   Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.468  ATS2    :   Конец обработки события e0 (Проверка).
10-12-04 02.16.546  7654321 :   Произошло событие e15 (Нажать цифру 5).
10-12-04 02.16.546  7654321 :   Обработка события e15 (Нажать цифру 5).
10-12-04 02.16.546  7654321 :   Текущее состояние автомата 3 (Поднята трубка).
10-12-04 02.16.546  ATS2    :   Произошло событие e15 (Набрать 5).
10-12-04 02.16.546  7654321 :       Выполнение действия z25 (Добавить 5).
10-12-04 02.16.546  7654321 :   Автомат перешел в состояние 3 (Поднята трубка).
10-12-04 02.16.546  7654321 :   Конец обработки события e15 (Нажать цифру 5).
10-12-04 02.16.562  ATS2    :   Обработка события e14 (Набрать 4).
10-12-04 02.16.562  ATS2    :   Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.562  ATS2    :       Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.562  ATS2    :       Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.562  ATS2    :       Выполнение действия z14 (Добавить 4).
10-12-04 02.16.577  ATS2    :   Произошло событие e0 (Проверка).
```

```
10-12-04 02.16.577  ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.577  ATS2    :  Конец обработки события e14 (Набрать 4).
10-12-04 02.16.671  ATS2    :  Обработка события e15 (Набрать 5).
10-12-04 02.16.671  ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.671  ATS2    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.671  ATS2    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.671  ATS2    :      Выполнение действия z15 (Добавить 5).
10-12-04 02.16.687  ATS2    :  Произошло событие e0 (Проверка).
10-12-04 02.16.687  ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.687  ATS2    :  Конец обработки события e15 (Набрать 5).
10-12-04 02.16.780  ATS2    :  Обработка события e0 (Проверка).
10-12-04 02.16.780  ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.780  ATS2    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.780  ATS2    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.780  ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.780  ATS2    :  Конец обработки события e0 (Проверка).
10-12-04 02.16.890  ATS2    :  Обработка события e0 (Проверка).
10-12-04 02.16.890  ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.890  ATS2    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.890  ATS2    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.890  ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.890  ATS2    :  Конец обработки события e0 (Проверка).
10-12-04 02.16.46   7654321 :  Произошло событие e16 (Нажать цифру 6).
10-12-04 02.16.46   7654321 :  Обработка события e16 (Нажать цифру 6).
10-12-04 02.16.46   7654321 :  Текущее состояние автомата 3 (Поднята трубка).
10-12-04 02.16.46   ATS2    :  Произошло событие e16 (Набрать 6).
10-12-04 02.16.46   7654321 :      Выполнение действия z26 (Добавить 6).
10-12-04 02.16.46   7654321 :  Автомат перешел в состояние 3 (Поднята трубка).
10-12-04 02.16.46   7654321 :  Конец обработки события e16 (Нажать цифру 6).
10-12-04 02.16.46   ATS2    :  Обработка события e16 (Набрать 6).
10-12-04 02.16.46   ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.46   ATS2    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.46   ATS2    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.46   ATS2    :      Выполнение действия z16 (Добавить 6).
10-12-04 02.16.46   ATS2    :  Произошло событие e0 (Проверка).
10-12-04 02.16.46   ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.46   ATS2    :  Конец обработки события e16 (Набрать 6).
10-12-04 02.16.140  ATS2    :  Обработка события e0 (Проверка).
10-12-04 02.16.140  ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.140  ATS2    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.140  ATS2    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.155  ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.155  ATS2    :  Конец обработки события e0 (Проверка).
10-12-04 02.16.546  7654321 :  Произошло событие e17 (Нажать цифру 7).
10-12-04 02.16.546  7654321 :  Обработка события e17 (Нажать цифру 7).
10-12-04 02.16.546  7654321 :  Текущее состояние автомата 3 (Поднята трубка).
10-12-04 02.16.546  ATS2    :  Произошло событие e17 (Набрать 7).
10-12-04 02.16.546  7654321 :      Выполнение действия z27 (Добавить 7).
10-12-04 02.16.546  7654321 :  Автомат перешел в состояние 3 (Поднята трубка).
10-12-04 02.16.546  7654321 :  Конец обработки события e17 (Нажать цифру 7).
10-12-04 02.16.546  ATS2    :  Обработка события e17 (Набрать 7).
10-12-04 02.16.546  ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.546  ATS2    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
```

```
10-12-04 02.16.546  ATS2    :      Проверика значения переменной x1
                                  (Правильный номер). Значение: Ложь.
10-12-04 02.16.546  ATS2    :      Выполнение действия z17 (Добавить 7).
10-12-04 02.16.546  ATS2    :  Произошло событие e0 (Проверка).
10-12-04 02.16.546  ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.546  ATS2    :  Конец обработки события e17 (Набрать 7).
10-12-04 02.16.655  ATS2    :  Обработка события e0 (Проверка).
10-12-04 02.16.655  ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.655  ATS2    :      Проверика значения переменной x1
                                  (Правильный номер). Значение: Истина.
10-12-04 02.16.655  ATS2    :      Проверика значения переменной x2 (Линия
                                  свободна). Значение: Истина.
10-12-04 02.16.655  ATS2    :      Проверика значения переменной x1
                                  (Правильный номер). Значение: Истина.
10-12-04 02.16.655  ATS2    :      Проверика значения переменной x2 (Линия
                                  свободна). Значение: Истина.
10-12-04 02.16.655  ATS2    :      Выполнение действия z3 (Занять
                                  вызываемого).
10-12-04 02.16.655  7654321 :  Произошло событие e21 (Длинные гудки).
10-12-04 02.16.655  1234567 :  Произошло событие e24 (Звонок).
10-12-04 02.16.655  7654321 :  Обработка события e21 (Длинные гудки).
10-12-04 02.16.655  7654321 :  Текущее состояние автомата 3 (Поднята трубка).
10-12-04 02.16.655  7654321 :  Автомат перешел в состояние 5 (Длинные гудки).
10-12-04 02.16.655  7654321 :  Конец обработки события e21 (Длинные гудки).
10-12-04 02.16.671  ATS2    :  Автомат перешел в состояние 4 (Длинные гудки).
10-12-04 02.16.671  ATS2    :  Конец обработки события e0 (Проверка).
10-12-04 02.16.671  1234567 :  Обработка события e24 (Звонок).
10-12-04 02.16.671  1234567 :  Текущее состояние автомата 0 (Ожидание
                                  действий).
10-12-04 02.16.671  1234567 :      Выполнение действия z9 (#A0.z9#).
10-12-04 02.16.671  1234567 :  Автомат перешел в состояние 7 (Звонок).
10-12-04 02.16.671  1234567 :  Конец обработки события e24 (Звонок).
10-12-04 02.16.249  1234567 :  Произошло событие e2 (Поднять трубку).
10-12-04 02.16.249  1234567 :  Обработка события e2 (Поднять трубку).
10-12-04 02.16.249  1234567 :  Текущее состояние автомата 7 (Звонок).
10-12-04 02.16.249  1234567 :      Выполнение действия z8 (Получить название
                                  АТС).
10-12-04 02.16.249  ATS2    :  Произошло событие e22 (Поднять трубку).
10-12-04 02.16.249  1234567 :  Автомат перешел в состояние 8 (Разговор).
10-12-04 02.16.249  1234567 :  Конец обработки события e2 (Поднять трубку).
10-12-04 02.16.265  ATS2    :  Обработка события e22 (Поднять трубку).
10-12-04 02.16.265  ATS2    :  Текущее состояние автомата 4 (Длинные гудки).
10-12-04 02.16.265  7654321 :  Произошло событие e23 (Разговор).
10-12-04 02.16.265  ATS2    :  Автомат перешел в состояние 5 (Разговор).
10-12-04 02.16.265  ATS2    :  Конец обработки события e22 (Поднять трубку).
10-12-04 02.16.265  7654321 :  Обработка события e23 (Разговор).
10-12-04 02.16.265  7654321 :  Текущее состояние автомата 5 (Длинные гудки).
10-12-04 02.16.265  7654321 :  Автомат перешел в состояние 6 (Разговор).
10-12-04 02.16.265  7654321 :  Конец обработки события e23 (Разговор).
10-12-04 02.16.827  7654321 :  Произошло событие e3 (Положить трубку).
10-12-04 02.16.827  7654321 :  Обработка события e3 (Положить трубку).
10-12-04 02.16.843  7654321 :  Текущее состояние автомата 6 (Разговор).
10-12-04 02.16.843  7654321 :      Выполнение действия z4 (Очистить).
10-12-04 02.16.843  7654321 :      Выполнение действия z5 (Очистить).
10-12-04 02.16.843  ATS2    :  Произошло событие e3 (Положить трубку).
10-12-04 02.16.843  7654321 :      Выполнение действия z7 (Освободить АТС).
10-12-04 02.16.843  7654321 :  Автомат перешел в состояние 0 (Ожидание
                                  действий).
10-12-04 02.16.843  7654321 :  Конец обработки события e3 (Положить трубку).
10-12-04 02.16.843  ATS2    :  Обработка события e3 (Положить трубку).
10-12-04 02.16.843  ATS2    :  Текущее состояние автомата 5 (Разговор).
10-12-04 02.16.843  ATS2    :      Выполнение действия z4 (Освободить
                                  вызывающего).
10-12-04 02.16.843  1234567 :  Произошло событие e22 (Короткие гудки).
```

```
10-12-04 02.16.858  ATS2    :  Автомат перешел в состояние 3 (Короткие гудки
                               вызываемого).
10-12-04 02.16.858  ATS2    :  Конец обработки события e3 (Положить трубку).
10-12-04 02.16.858  1234567 :  Обработка события e22 (Короткие гудки).
10-12-04 02.16.858  1234567 :  Текущее состояние автомата 8 (Разговор).
10-12-04 02.16.858  1234567 :  Автомат перешел в состояние 9 (Короткие гудки).
10-12-04 02.16.858  1234567 :  Конец обработки события e22 (Короткие гудки).
10-12-04 02.16.702  1234567 :  Произошло событие e3 (Положить трубку).
10-12-04 02.16.702  1234567 :  Обработка события e3 (Положить трубку).
10-12-04 02.16.702  1234567 :  Текущее состояние автомата 9 (Короткие гудки).
10-12-04 02.16.702  ATS2    :  Произошло событие e23 (Положить трубку).
10-12-04 02.16.702  1234567 :      Выполнение действия z7 (Освободить АТС).
10-12-04 02.16.702  1234567 :  Автомат перешел в состояние 0 (Ожидание
                               действий).
10-12-04 02.16.702  1234567 :  Конец обработки события e3 (Положить трубку).
10-12-04 02.16.702  ATS2    :  Обработка события e23 (Положить трубку).
10-12-04 02.16.702  ATS2    :  Текущее состояние автомата 3 (Короткие гудки
                               вызываемого).
10-12-04 02.16.702  ATS2    :      Выполнение действия z5 (Освободить
                               вызываемого).
10-12-04 02.16.702  ATS2    :      Выполнение действия z6 (Сброс цифр).
10-12-04 02.16.718  ATS2    :  Автомат перешел в состояние 0 (Ожидание
                               действий).
10-12-04 02.16.718  ATS2    :  Конец обработки события e23 (Положить трубку).
10-12-04 02.16.890  7654321 :  Произошло событие e11 (Нажать цифру 1).
10-12-04 02.16.890  7654321 :  Обработка события e11 (Нажать цифру 1).
10-12-04 02.16.890  7654321 :  Текущее состояние автомата 0 (Ожидание
                               действий).
10-12-04 02.16.890  7654321 :      Выполнение действия z11 (Добавить 1).
10-12-04 02.16.890  7654321 :      Выполнение действия z21 (Добавить 1).
10-12-04 02.16.890  7654321 :  Автомат перешел в состояние 1 (Редактирование
                               номера).
10-12-04 02.16.890  7654321 :  Конец обработки события e11 (Нажать цифру 1).
10-12-04 02.16.452  7654321 :  Произошло событие e12 (Нажать цифру 2).
10-12-04 02.16.452  7654321 :  Обработка события e12 (Нажать цифру 2).
10-12-04 02.16.452  7654321 :  Текущее состояние автомата 1 (Редактирование
                               номера).
10-12-04 02.16.452  7654321 :      Выполнение действия z12 (Добавить 2).
10-12-04 02.16.452  7654321 :      Выполнение действия z22 (Добавить 2).
10-12-04 02.16.452  7654321 :  Автомат перешел в состояние 1 (Редактирование
                               номера).
10-12-04 02.16.452  7654321 :  Конец обработки события e12 (Нажать цифру 2).
10-12-04 02.16.15   7654321 :  Произошло событие e2 (Поднять трубку).
10-12-04 02.16.15   7654321 :  Обработка события e2 (Поднять трубку).
10-12-04 02.16.15   7654321 :  Текущее состояние автомата 1 (Редактирование
                               номера).
10-12-04 02.16.15   7654321 :      Выполнение действия z6 (Занять АТС).
10-12-04 02.16.15   ATS2    :  Произошло событие e2 (Поднять трубку).
10-12-04 02.16.15   7654321 :  Произошло событие e0 (Проверка).
10-12-04 02.16.15   7654321 :  Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.15   7654321 :  Конец обработки события e2 (Поднять трубку).
10-12-04 02.16.15   ATS2    :  Обработка события e2 (Поднять трубку).
10-12-04 02.16.15   ATS2    :  Текущее состояние автомата 0 (Ожидание
                               действий).
10-12-04 02.16.15   ATS2    :      Выполнение действия z2 (Занять
                               вызывающего).
10-12-04 02.16.15   ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.15   ATS2    :  Конец обработки события e2 (Поднять трубку).
10-12-04 02.16.124  7654321 :  Обработка события e0 (Проверка).
10-12-04 02.16.124  7654321 :  Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.124  7654321 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.124  7654321 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
```

```
10-12-04 02.16.124  7654321 :      Проверика значения переменной x10 (Цифра
                                   0). Значение: Ложь.
10-12-04 02.16.124  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.124  7654321 :      Проверика значения переменной x11 (Цифра
                                   1). Значение: Истина.
10-12-04 02.16.124  ATS2    :  Произошло событие e11 (Набрать 1).
10-12-04 02.16.124  7654321 :      Выполнение действия z3 (Следующая цифра).
10-12-04 02.16.124  7654321 :  Произошло событие e0 (Проверка).
10-12-04 02.16.124  7654321 :  Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.124  7654321 :  Конец обработки события e0 (Проверка).
10-12-04 02.16.124  ATS2    :  Обработка события e11 (Набрать 1).
10-12-04 02.16.124  ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.124  ATS2    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.124  ATS2    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.124  ATS2    :      Выполнение действия z11 (Добавить 1).
10-12-04 02.16.124  ATS2    :  Произошло событие e0 (Проверка).
10-12-04 02.16.124  ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.124  ATS2    :  Конец обработки события e11 (Набрать 1).
10-12-04 02.16.218  7654321 :  Обработка события e0 (Проверка).
10-12-04 02.16.218  7654321 :  Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.218  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.218  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.218  7654321 :      Проверика значения переменной x10 (Цифра
                                   0). Значение: Ложь.
10-12-04 02.16.218  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.218  7654321 :      Проверика значения переменной x11 (Цифра
                                   1). Значение: Ложь.
10-12-04 02.16.218  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.218  7654321 :      Проверика значения переменной x12 (Цифра
                                   2). Значение: Истина.
10-12-04 02.16.218  ATS2    :  Произошло событие e12 (Набрать 2).
10-12-04 02.16.218  7654321 :      Выполнение действия z3 (Следующая цифра).
10-12-04 02.16.218  7654321 :  Произошло событие e0 (Проверка).
10-12-04 02.16.233  7654321 :  Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.233  7654321 :  Конец обработки события e0 (Проверка).
10-12-04 02.16.233  ATS2    :  Обработка события e0 (Проверка).
10-12-04 02.16.233  ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.233  ATS2    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.233  ATS2    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.233  ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.233  ATS2    :  Конец обработки события e0 (Проверка).
10-12-04 02.16.327  7654321 :  Обработка события e0 (Проверка).
10-12-04 02.16.327  7654321 :  Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.327  7654321 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Ложь.
10-12-04 02.16.327  7654321 :  Автомат перешел в состояние 3 (Поднята трубка).
10-12-04 02.16.327  7654321 :  Конец обработки события e0 (Проверка).
10-12-04 02.16.327  ATS2    :  Обработка события e12 (Набрать 2).
10-12-04 02.16.327  ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.327  ATS2    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.327  ATS2    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.327  ATS2    :      Выполнение действия z12 (Добавить 2).
10-12-04 02.16.327  ATS2    :  Произошло событие e0 (Проверка).
```

```
10-12-04 02.16.327  ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.327  ATS2    :  Конец обработки события e12 (Набрать 2).
10-12-04 02.16.437  ATS2    :  Обработка события e0 (Проверка).
10-12-04 02.16.437  ATS2    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.437  ATS2    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.437  ATS2    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.437  ATS2    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.437  ATS2    :  Конец обработки события e0 (Проверка).
10-12-04 02.16.46   1234567 :  Произошло событие e17 (Нажать цифру 7).
10-12-04 02.16.46   1234567 :  Обработка события e17 (Нажать цифру 7).
10-12-04 02.16.46   1234567 :  Текущее состояние автомата 0 (Ожидание
                                действий).
10-12-04 02.16.46   1234567 :      Выполнение действия z17 (Добавить 7).
10-12-04 02.16.62   1234567 :      Выполнение действия z27 (Добавить 7).
10-12-04 02.16.62   1234567 :  Автомат перешел в состояние 1 (Редактирование
                                номера).
10-12-04 02.16.62   1234567 :  Конец обработки события e17 (Нажать цифру 7).
10-12-04 02.16.608  1234567 :  Произошло событие e16 (Нажать цифру 6).
10-12-04 02.16.608  1234567 :  Обработка события e16 (Нажать цифру 6).
10-12-04 02.16.608  1234567 :  Текущее состояние автомата 1 (Редактирование
                                номера).
10-12-04 02.16.608  1234567 :      Выполнение действия z16 (Добавить 6).
10-12-04 02.16.608  1234567 :      Выполнение действия z26 (Добавить 6).
10-12-04 02.16.608  1234567 :  Автомат перешел в состояние 1 (Редактирование
                                номера).
10-12-04 02.16.608  1234567 :  Конец обработки события e16 (Нажать цифру 6).
10-12-04 02.16.999  1234567 :  Произошло событие e15 (Нажать цифру 5).
10-12-04 02.16.999  1234567 :  Обработка события e15 (Нажать цифру 5).
10-12-04 02.16.999  1234567 :  Текущее состояние автомата 1 (Редактирование
                                номера).
10-12-04 02.16.999  1234567 :      Выполнение действия z15 (Добавить 5).
10-12-04 02.16.999  1234567 :      Выполнение действия z25 (Добавить 5).
10-12-04 02.16.999  1234567 :  Автомат перешел в состояние 1 (Редактирование
                                номера).
10-12-04 02.16.999  1234567 :  Конец обработки события e15 (Нажать цифру 5).
10-12-04 02.16.437  1234567 :  Произошло событие e14 (Нажать цифру 4).
10-12-04 02.16.437  1234567 :  Обработка события e14 (Нажать цифру 4).
10-12-04 02.16.437  1234567 :  Текущее состояние автомата 1 (Редактирование
                                номера).
10-12-04 02.16.437  1234567 :      Выполнение действия z14 (Добавить 4).
10-12-04 02.16.437  1234567 :      Выполнение действия z24 (Добавить 4).
10-12-04 02.16.452  1234567 :  Автомат перешел в состояние 1 (Редактирование
                                номера).
10-12-04 02.16.452  1234567 :  Конец обработки события e14 (Нажать цифру 4).
10-12-04 02.16.937  1234567 :  Произошло событие e13 (Нажать цифру 3).
10-12-04 02.16.952  1234567 :  Обработка события e13 (Нажать цифру 3).
10-12-04 02.16.952  1234567 :  Текущее состояние автомата 1 (Редактирование
                                номера).
10-12-04 02.16.952  1234567 :      Выполнение действия z13 (Добавить 3).
10-12-04 02.16.952  1234567 :      Выполнение действия z23 (Добавить 3).
10-12-04 02.16.952  1234567 :  Автомат перешел в состояние 1 (Редактирование
                                номера).
10-12-04 02.16.952  1234567 :  Конец обработки события e13 (Нажать цифру 3).
10-12-04 02.16.312  1234567 :  Произошло событие e12 (Нажать цифру 2).
10-12-04 02.16.327  1234567 :  Обработка события e12 (Нажать цифру 2).
10-12-04 02.16.327  1234567 :  Текущее состояние автомата 1 (Редактирование
                                номера).
10-12-04 02.16.327  1234567 :      Выполнение действия z12 (Добавить 2).
10-12-04 02.16.327  1234567 :      Выполнение действия z22 (Добавить 2).
10-12-04 02.16.327  1234567 :  Автомат перешел в состояние 1 (Редактирование
                                номера).
10-12-04 02.16.327  1234567 :  Конец обработки события e12 (Нажать цифру 2).
```

```
10-12-04 02.16.952  1234567 :  Произошло событие e11 (Нажать цифру 1).
10-12-04 02.16.952  1234567 :  Обработка события e11 (Нажать цифру 1).
10-12-04 02.16.952  1234567 :  Текущее состояние автомата 1 (Редактирование
                               номера).
10-12-04 02.16.952  1234567 :      Выполнение действия z11 (Добавить 1).
10-12-04 02.16.952  1234567 :      Выполнение действия z21 (Добавить 1).
10-12-04 02.16.952  1234567 :  Автомат перешел в состояние 1 (Редактирование
                               номера).
10-12-04 02.16.952  1234567 :  Конец обработки события e11 (Нажать цифру 1).
10-12-04 02.16.671  1234567 :  Произошло событие e2 (Поднять трубку).
10-12-04 02.16.671  1234567 :  Обработка события e2 (Поднять трубку).
10-12-04 02.16.671  1234567 :  Текущее состояние автомата 1 (Редактирование
                               номера).
10-12-04 02.16.671  1234567 :      Выполнение действия z6 (Занять АТС).
10-12-04 02.16.671  ATS1    :  Произошло событие e2 (Поднять трубку).
10-12-04 02.16.671  1234567 :  Произошло событие e0 (Проверка).
10-12-04 02.16.671  1234567 :  Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.671  1234567 :  Конец обработки события e2 (Поднять трубку).
10-12-04 02.16.671  ATS1    :  Обработка события e2 (Поднять трубку).
10-12-04 02.16.671  ATS1    :  Текущее состояние автомата 0 (Ожидание
                               действий).
10-12-04 02.16.671  ATS1    :      Выполнение действия z2 (Занять
                               вызывающего).
10-12-04 02.16.687  ATS1    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.687  ATS1    :  Конец обработки события e2 (Поднять трубку).
10-12-04 02.16.780  1234567 :  Обработка события e0 (Проверка).
10-12-04 02.16.780  1234567 :  Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.780  1234567 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x10 (Цифра
                               0). Значение: Ложь.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x11 (Цифра
                               1). Значение: Ложь.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x12 (Цифра
                               2). Значение: Ложь.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x13 (Цифра
                               3). Значение: Ложь.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x14 (Цифра
                               4). Значение: Ложь.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x15 (Цифра
                               5). Значение: Ложь.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x16 (Цифра
                               6). Значение: Ложь.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x2 (Есть
                               цифры). Значение: Истина.
10-12-04 02.16.780  1234567 :      Проверика значения переменной x17 (Цифра
                               7). Значение: Истина.
10-12-04 02.16.780  ATS1    :  Произошло событие e17 (Набрать 7).
10-12-04 02.16.780  1234567 :      Выполнение действия z3 (Следующая цифра).
10-12-04 02.16.780  1234567 :  Произошло событие e0 (Проверка).
```

```
10-12-04 02.16.780  1234567 :   Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.780  1234567 :   Конец обработки события e0 (Проверка).
10-12-04 02.16.780  ATS1    :   Обработка события e17 (Набрать 7).
10-12-04 02.16.780  ATS1    :   Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.780  ATS1    :       Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.780  ATS1    :       Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.780  ATS1    :       Выполнение действия z17 (Добавить 7).
10-12-04 02.16.780  ATS1    :   Произошло событие e0 (Проверка).
10-12-04 02.16.796  ATS1    :   Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.796  ATS1    :   Конец обработки события e17 (Набрать 7).
10-12-04 02.16.890  1234567 :   Обработка события e0 (Проверка).
10-12-04 02.16.890  1234567 :   Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.890  1234567 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x10 (Цифра
                                0). Значение: Ложь.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x11 (Цифра
                                1). Значение: Ложь.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x12 (Цифра
                                2). Значение: Ложь.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x13 (Цифра
                                3). Значение: Ложь.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x14 (Цифра
                                4). Значение: Ложь.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x15 (Цифра
                                5). Значение: Ложь.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.890  1234567 :       Проверика значения переменной x16 (Цифра
                                6). Значение: Истина.
10-12-04 02.16.890  ATS1    :   Произошло событие e16 (Набрать 6).
10-12-04 02.16.890  1234567 :       Выполнение действия z3 (Следующая цифра).
10-12-04 02.16.890  1234567 :   Произошло событие e0 (Проверка).
10-12-04 02.16.890  1234567 :   Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.890  1234567 :   Конец обработки события e0 (Проверка).
10-12-04 02.16.890  ATS1    :   Обработка события e0 (Проверка).
10-12-04 02.16.890  ATS1    :   Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.890  ATS1    :       Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.890  ATS1    :       Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.890  ATS1    :   Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.890  ATS1    :   Конец обработки события e0 (Проверка).
10-12-04 02.16.983  1234567 :   Обработка события e0 (Проверка).
10-12-04 02.16.983  1234567 :   Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.983  1234567 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.983  1234567 :       Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
```

```
10-12-04 02.16.983  1234567 :      Проверика значения переменной x10 (Цифра
                                   0). Значение: Ложь.
10-12-04 02.16.999  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.999  1234567 :      Проверика значения переменной x11 (Цифра
                                   1). Значение: Ложь.
10-12-04 02.16.999  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.999  1234567 :      Проверика значения переменной x12 (Цифра
                                   2). Значение: Ложь.
10-12-04 02.16.999  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.999  1234567 :      Проверика значения переменной x13 (Цифра
                                   3). Значение: Ложь.
10-12-04 02.16.999  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.999  1234567 :      Проверика значения переменной x14 (Цифра
                                   4). Значение: Ложь.
10-12-04 02.16.999  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.999  1234567 :      Проверика значения переменной x15 (Цифра
                                   5). Значение: Истина.
10-12-04 02.16.999  ATS1    :  Произошло событие e15 (Набрать 5).
10-12-04 02.16.999  1234567 :      Выполнение действия z3 (Следующая цифра).
10-12-04 02.16.999  1234567 :  Произошло событие e0 (Проверка).
10-12-04 02.16.999  1234567 :  Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.999  1234567 :  Конец обработки события e0 (Проверка).
10-12-04 02.16.999  ATS1    :  Обработка события e16 (Набрать 6).
10-12-04 02.16.999  ATS1    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.999  ATS1    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.999  ATS1    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.999  ATS1    :      Выполнение действия z16 (Добавить 6).
10-12-04 02.16.999  ATS1    :  Произошло событие e0 (Проверка).
10-12-04 02.16.999  ATS1    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.999  ATS1    :  Конец обработки события e16 (Набрать 6).
10-12-04 02.16.93   1234567 :  Обработка события e0 (Проверка).
10-12-04 02.16.93   1234567 :  Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.93   1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.93   1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.93   1234567 :      Проверика значения переменной x10 (Цифра
                                   0). Значение: Ложь.
10-12-04 02.16.108  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.108  1234567 :      Проверика значения переменной x11 (Цифра
                                   1). Значение: Ложь.
10-12-04 02.16.108  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.108  1234567 :      Проверика значения переменной x12 (Цифра
                                   2). Значение: Ложь.
10-12-04 02.16.108  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.108  1234567 :      Проверика значения переменной x13 (Цифра
                                   3). Значение: Ложь.
10-12-04 02.16.108  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.108  1234567 :      Проверика значения переменной x14 (Цифра
                                   4). Значение: Истина.
10-12-04 02.16.108  ATS1    :  Произошло событие e14 (Набрать 4).
10-12-04 02.16.108  1234567 :      Выполнение действия z3 (Следующая цифра).
10-12-04 02.16.108  1234567 :  Произошло событие e0 (Проверка).
```

```
10-12-04 02.16.108  1234567 :  Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.108  1234567 :  Конец обработки события e0 (Проверка).
10-12-04 02.16.108  ATS1    :  Обработка события e15 (Набрать 5).
10-12-04 02.16.108  ATS1    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.108  ATS1    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.108  ATS1    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.108  ATS1    :      Выполнение действия z15 (Добавить 5).
10-12-04 02.16.108  ATS1    :  Произошло событие e0 (Проверка).
10-12-04 02.16.108  ATS1    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.108  ATS1    :  Конец обработки события e15 (Набрать 5).
10-12-04 02.16.202  1234567 :  Обработка события e0 (Проверка).
10-12-04 02.16.202  1234567 :  Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.202  1234567 :      Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.202  1234567 :      Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.202  1234567 :      Проверика значения переменной x10 (Цифра
                                0). Значение: Ложь.
10-12-04 02.16.202  1234567 :      Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.202  1234567 :      Проверика значения переменной x11 (Цифра
                                1). Значение: Ложь.
10-12-04 02.16.202  1234567 :      Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.202  1234567 :      Проверика значения переменной x12 (Цифра
                                2). Значение: Ложь.
10-12-04 02.16.202  1234567 :      Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.202  ATS1    :  Обработка события e0 (Проверка).
10-12-04 02.16.202  ATS1    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.202  ATS1    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.202  ATS1    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.202  ATS1    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.202  ATS1    :  Конец обработки события e0 (Проверка).
10-12-04 02.16.202  1234567 :      Проверика значения переменной x13 (Цифра
                                3). Значение: Истина.
10-12-04 02.16.202  ATS1    :  Произошло событие e13 (Набрать 3).
10-12-04 02.16.202  1234567 :      Выполнение действия z3 (Следующая цифра).
10-12-04 02.16.202  1234567 :  Произошло событие e0 (Проверка).
10-12-04 02.16.202  1234567 :  Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.202  1234567 :  Конец обработки события e0 (Проверка).
10-12-04 02.16.312  ATS1    :  Обработка события e14 (Набрать 4).
10-12-04 02.16.312  ATS1    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.312  ATS1    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.312  ATS1    :      Проверика значения переменной x1
                                (Правильный номер). Значение: Ложь.
10-12-04 02.16.312  ATS1    :      Выполнение действия z14 (Добавить 4).
10-12-04 02.16.312  ATS1    :  Произошло событие e0 (Проверка).
10-12-04 02.16.312  ATS1    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.312  ATS1    :  Конец обработки события e14 (Набрать 4).
10-12-04 02.16.312  1234567 :  Обработка события e0 (Проверка).
10-12-04 02.16.312  1234567 :  Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.312  1234567 :      Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.312  1234567 :      Проверика значения переменной x2 (Есть
                                цифры). Значение: Истина.
10-12-04 02.16.312  1234567 :      Проверика значения переменной x10 (Цифра
                                0). Значение: Ложь.
```

```
10-12-04 02.16.312  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.312  1234567 :      Проверика значения переменной x11 (Цифра
                                   1). Значение: Ложь.
10-12-04 02.16.312  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.312  1234567 :      Проверика значения переменной x12 (Цифра
                                   2). Значение: Истина.
10-12-04 02.16.312  ATS1    :  Произошло событие e12 (Набрать 2).
10-12-04 02.16.312  1234567 :      Выполнение действия z3 (Следующая цифра).
10-12-04 02.16.312  1234567 :  Произошло событие e0 (Проверка).
10-12-04 02.16.312  1234567 :  Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.312  1234567 :  Конец обработки события e0 (Проверка).
10-12-04 02.16.405  ATS1    :  Обработка события e0 (Проверка).
10-12-04 02.16.405  ATS1    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.405  ATS1    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.405  ATS1    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.405  ATS1    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.405  ATS1    :  Конец обработки события e0 (Проверка).
10-12-04 02.16.421  1234567 :  Обработка события e0 (Проверка).
10-12-04 02.16.421  1234567 :  Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.421  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.421  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.421  1234567 :      Проверика значения переменной x10 (Цифра
                                   0). Значение: Ложь.
10-12-04 02.16.421  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Истина.
10-12-04 02.16.421  1234567 :      Проверика значения переменной x11 (Цифра
                                   1). Значение: Истина.
10-12-04 02.16.421  ATS1    :  Произошло событие e11 (Набрать 1).
10-12-04 02.16.421  1234567 :      Выполнение действия z3 (Следующая цифра).
10-12-04 02.16.421  1234567 :  Произошло событие e0 (Проверка).
10-12-04 02.16.421  1234567 :  Автомат перешел в состояние 2 (Набор номера).
10-12-04 02.16.421  1234567 :  Конец обработки события e0 (Проверка).
10-12-04 02.16.515  ATS1    :  Обработка события e13 (Набрать 3).
10-12-04 02.16.515  ATS1    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.515  ATS1    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.515  ATS1    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.515  ATS1    :      Выполнение действия z13 (Добавить 3).
10-12-04 02.16.515  ATS1    :  Произошло событие e0 (Проверка).
10-12-04 02.16.515  ATS1    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.515  ATS1    :  Конец обработки события e13 (Набрать 3).
10-12-04 02.16.515  1234567 :  Обработка события e0 (Проверка).
10-12-04 02.16.515  1234567 :  Текущее состояние автомата 2 (Набор номера).
10-12-04 02.16.515  1234567 :      Проверика значения переменной x2 (Есть
                                   цифры). Значение: Ложь.
10-12-04 02.16.515  1234567 :  Автомат перешел в состояние 3 (Поднята трубка).
10-12-04 02.16.515  1234567 :  Конец обработки события e0 (Проверка).
10-12-04 02.16.608  ATS1    :  Обработка события e0 (Проверка).
10-12-04 02.16.608  ATS1    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.608  ATS1    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.608  ATS1    :      Проверика значения переменной x1
                                   (Правильный номер). Значение: Ложь.
10-12-04 02.16.608  ATS1    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.608  ATS1    :  Конец обработки события e0 (Проверка).
10-12-04 02.16.718  ATS1    :  Обработка события e12 (Набрать 2).
10-12-04 02.16.718  ATS1    :  Текущее состояние автомата 1 (Набор номера).
```

```
10-12-04 02.16.718  ATS1    :      Проверика значения переменной x1
                                  (Правильный номер). Значение: Ложь.
10-12-04 02.16.718  ATS1    :      Проверика значения переменной x1
                                  (Правильный номер). Значение: Ложь.
10-12-04 02.16.718  ATS1    :      Выполнение действия z12 (Добавить 2).
10-12-04 02.16.718  ATS1    :  Произошло событие e0 (Проверка).
10-12-04 02.16.718  ATS1    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.718  ATS1    :  Конец обработки события e12 (Набрать 2).
10-12-04 02.16.827  ATS1    :  Обработка события e11 (Набрать 1).
10-12-04 02.16.827  ATS1    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.827  ATS1    :      Проверика значения переменной x1
                                  (Правильный номер). Значение: Ложь.
10-12-04 02.16.827  ATS1    :      Проверика значения переменной x1
                                  (Правильный номер). Значение: Ложь.
10-12-04 02.16.827  ATS1    :      Выполнение действия z11 (Добавить 1).
10-12-04 02.16.827  ATS1    :  Произошло событие e0 (Проверка).
10-12-04 02.16.827  ATS1    :  Автомат перешел в состояние 1 (Набор номера).
10-12-04 02.16.827  ATS1    :  Конец обработки события e11 (Набрать 1).
10-12-04 02.16.921  ATS1    :  Обработка события e0 (Проверка).
10-12-04 02.16.921  ATS1    :  Текущее состояние автомата 1 (Набор номера).
10-12-04 02.16.921  ATS1    :      Проверика значения переменной x1
                                  (Правильный номер). Значение: Истина.
10-12-04 02.16.921  ATS1    :      Проверика значения переменной x2 (Линия
                                  свободна). Значение: Ложь.
10-12-04 02.16.921  1234567 :  Произошло событие e22 (Короткие гудки).
10-12-04 02.16.921  ATS1    :  Автомат перешел в состояние 2 (Короткие гудки
                                  вызывающего).
10-12-04 02.16.921  ATS1    :  Конец обработки события e0 (Проверка).
10-12-04 02.16.921  1234567 :  Обработка события e22 (Короткие гудки).
10-12-04 02.16.921  1234567 :  Текущее состояние автомата 3 (Поднята трубка).
10-12-04 02.16.921  1234567 :  Автомат перешел в состояние 4 (Короткие гудки).
10-12-04 02.16.937  1234567 :  Конец обработки события e22 (Короткие гудки).
10-12-04 02.16.30   ATS1    :  Обработка события e0 (Проверка).
10-12-04 02.16.30   ATS1    :  Текущее состояние автомата 2 (Короткие гудки
                                  вызывающего).
10-12-04 02.16.30   ATS1    :  Автомат перешел в состояние 2 (Короткие гудки
                                  вызывающего).
10-12-04 02.16.30   ATS1    :  Конец обработки события e0 (Проверка).
10-12-04 02.16.124  ATS1    :  Обработка события e0 (Проверка).
10-12-04 02.16.124  ATS1    :  Текущее состояние автомата 2 (Короткие гудки
                                  вызывающего).
10-12-04 02.16.124  ATS1    :  Автомат перешел в состояние 2 (Короткие гудки
                                  вызывающего).
10-12-04 02.16.124  ATS1    :  Конец обработки события e0 (Проверка).
10-12-04 02.16.77   1234567 :  Произошло событие e3 (Положить трубку).
10-12-04 02.16.77   1234567 :  Обработка события e3 (Положить трубку).
10-12-04 02.16.77   1234567 :  Текущее состояние автомата 4 (Короткие гудки).
10-12-04 02.16.77   1234567 :      Выполнение действия z4 (Очистить).
10-12-04 02.16.77   1234567 :      Выполнение действия z5 (Очистить).
10-12-04 02.16.77   ATS1    :  Произошло событие e3 (Положить трубку).
10-12-04 02.16.77   1234567 :      Выполнение действия z7 (Освободить АТС).
10-12-04 02.16.77   1234567 :  Автомат перешел в состояние 0 (Ожидание
                                  действий).
10-12-04 02.16.77   1234567 :  Конец обработки события e3 (Положить трубку).
10-12-04 02.16.77   ATS1    :  Обработка события e3 (Положить трубку).
10-12-04 02.16.77   ATS1    :  Текущее состояние автомата 2 (Короткие гудки
                                  вызывающего).
10-12-04 02.16.77   ATS1    :      Выполнение действия z4 (Освободить
                                  вызывающего).
10-12-04 02.16.77   ATS1    :      Выполнение действия z6 (Сброс цифр).
10-12-04 02.16.77   ATS1    :      Выполнение действия z6 (Сброс цифр).
10-12-04 02.16.77   ATS1    :  Автомат перешел в состояние 0 (Ожидание
                                  действий).
10-12-04 02.16.77   ATS1    :  Конец обработки события e3 (Положить трубку).
```