

Saint-Petersburg State University of Information Technologies, Mechanics and
Optics

Computer Technologies Department

V. R. Danilov, I. O. Varvaluk

Automated mobile phone payment system

State machine-oriented programming

Project Documentation

This project was created in terms of «Foundation for open project documentation».
<http://is.ifmo.ru/>

Saint-Petersburg
2006

Contents

Introduction	3
1. Problem statement.....	4
2. Automaton-based implementation	5
2.1 Event providers.....	5
2.2 Event providers.....	7
2.3 Controlled objects	8
3. Program implementation.....	9
3.1 Interpreter approach	9
3.2 Compilation approach	9
3.3 Using instructions.....	9
Conclusion	10
Appendix 1. Example of application log	11
Appendix 2. Generated XML-Definition.....	15
Appendix 3. Source code	24
Appendix 4. Generated <i>Java</i> source.....	40

Introduction

This work is an example of applying automata programming technique for design and development of mobile-phone payment terminal.

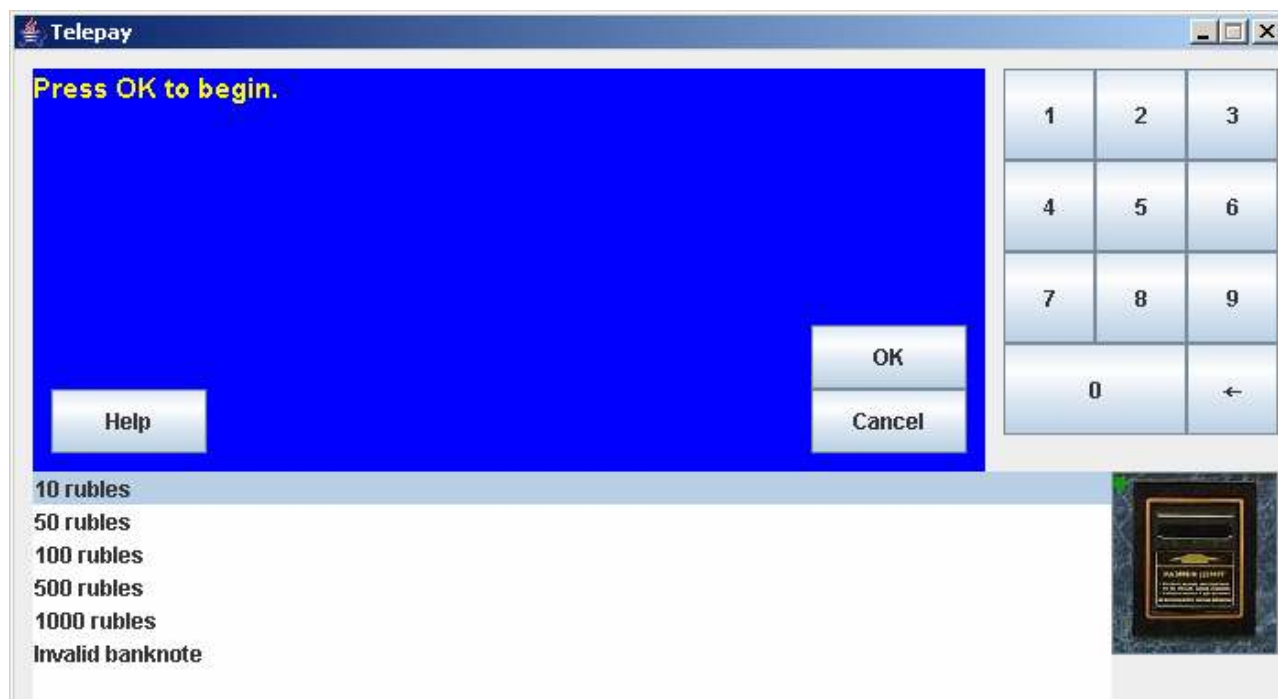
Automata approach is convenient for formalization and implementation of problems concerned with technical devices control. Graphs of automata transitions visually demonstrate the logic of processes. It makes it easy to see design errors such as unaccounted situations or absence of a transition.

Unimod development environment makes it possible to use automata approach from the start. While development with the tool one must design logic in terms of automata and then implement objects used by automata in *Java*. The program is partitioned into independent parts so error probability is decreased.

1. Problem statement

Task is to design and implement mobile phone payment terminal.

One can see control board of the device on picture 1.



Picture 1- Control board of the device

In the right side there is keyboard for phone number input.

To the bottom from the keyboard there are money receiver and light diodes reflecting the device's readiness for receiving money. Also there is a button to emulate work with the receiver.

- If the green light diode is shining then pressing button brings to entering banknote to client's account;
- If the red light diode is shining then pressing button brings to extracting "false banknote" from the receiver.

In the left side there is list of available banknotes.

In the left top corner there is a text screen showing instructions to the user in the current state.

There are two control touch keys on the screen: «OK» and «Cancel»:

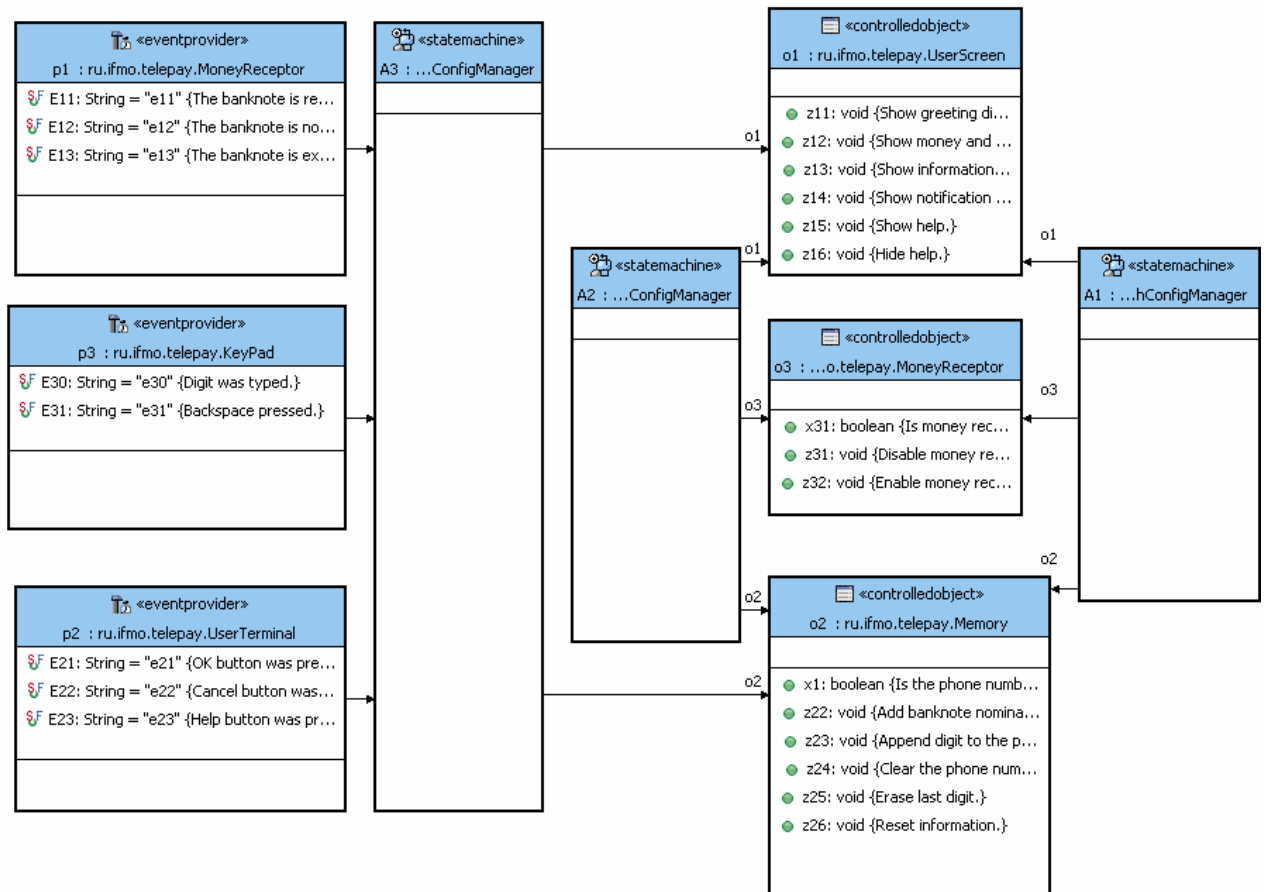
- «OK» — confirmation of current action and switching to next step.
- «Cancel» — cancel of current action and return to previous step.

«Help» touch key shows the help window and changes name to «Back». After pressing the «Back» touch key the device returns to work mode.

2. Automaton-based implementation

Described objects can be divided in the following way. The keyboard, control touch keys, money receiver and the «Help» touch key should be presented as event providers. The user screen, memory of the device and money receiver should be presented as controlled objects.

Scheme of the system is presented on the picture 2.



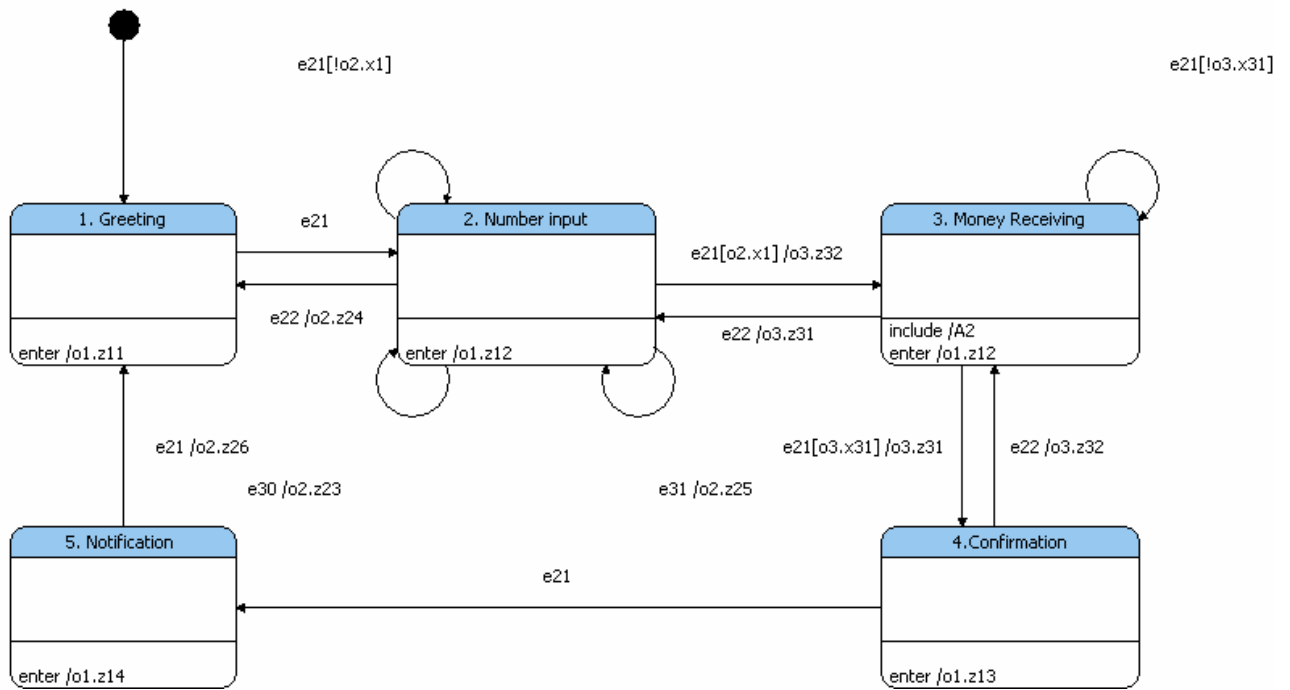
Picture 2- Scheme of the system

2.1 Event providers

As it is seen from the scheme of system there are three state machines: *A1*, *A2* and *A3*. The first state machine reflects the sequence of steps of payment. The second one controls functionality of money receiver. The third automaton controls switching between work and help modes.

Using inserted automatons simplifies the scheme of the system and decreases number of transitions.

One can see state machine *A1* on the picture 3.

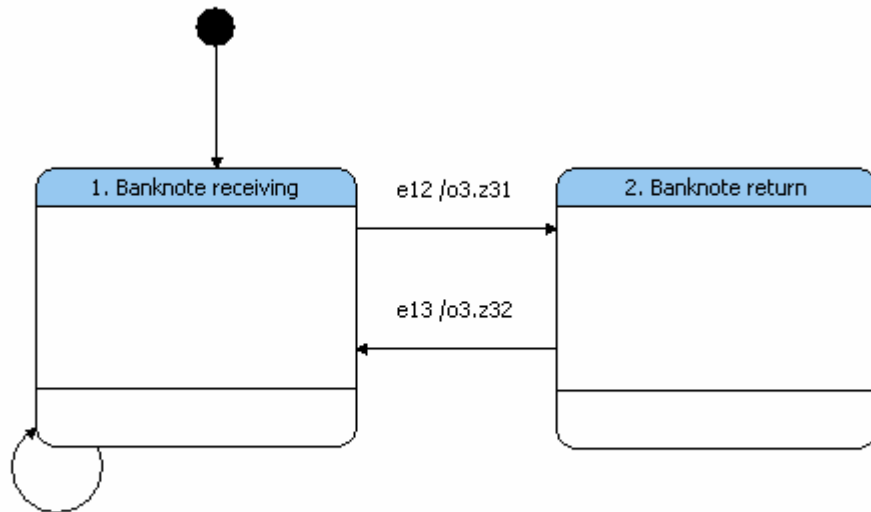


Picture 3- State machine *A1*

The state machine has five states:

- «*1. Greeting*» — this is the starting state signaling of the device’s readiness to work;
- «*2. Number Input*» — this is state of entering phone number to pay for;
- «*3. Money Input*» — this is state of receiving money from user. In order to simplify logic this state includes state machine *A2*;
- «*4. Confirmation*» — in this state user must confirm the correctness of provided information or return to previous steps to change it;
- «*5. Goodbye*» — this state signals that the payment was executed.

State machine *A2* controls the process of money receiving. Its scheme is shown on the picture 4.

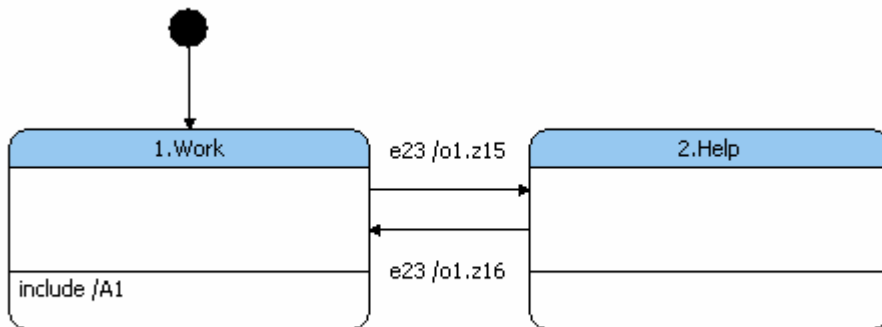


e11 /o2.z22, o1.z12

Picture 4- State machine A2

If in the state of «1. Banknote receiving» normal banknote is received then the device accepts it, moves to user account and continues receiving. If the banknote was invalid then the state machine transits to state «2. Banknote return» and returns only after the invalid banknote is extracted from receiver by user.

The state machine A3 controls switching between work and help modes of the device. It switches from «1.Work» to «2. Help» when the «Help» touch key is pressed and returns after the «Back» touch key is pressed. Its schema is shown on the picture 5 below.



Picture 5- State machine A3

2.2 Event providers

In this section event providers of the system are described.

Event provider p1

This object is concerned with money receiver events:

- **e11** — the received banknote was recognized;
- **e12** — the received banknote was invalid;
- **e13** — the user extracted invalid banknote.

Event provider *p2*

This object deals with controlling touch keys. Provided events are:

- **e21** — the «OK» button was pressed;
- **e22** — the «Cancel» button was pressed

Event provider *p3*

This object corresponds to keyboard for number input.

- **e31** — one of digit keys was pressed;
- **e32** — backspace was pressed.

2.3 Controlled objects

Controlled object *o1*

The object deals with user screen. It supports following methods:

- **z11** – shows the greeting dialog;
- **z12** – shows the number input dialog;
- **z13** – shows the confirmation dialog;
- **z14** – shows the dialog reflecting success of payment.

Controlled object *o2*

The object corresponds to the memory of device.

- **x1** – This method returns whether the provided number is valid;
- **z22** – This method adds the nominal of received banknote to user account;
- **z23** – This method appends digit to phone number;
- **z24** – This method clears the phone number;
- **z25** – This method removes last digit of phone number.

Controlled object *o3*

The object is concerned with light diodes and money receiver.

- **x31** – returns whether device is ready to receive banknote;
- **z31** – lights the red light diode and turns off the green. It signals to user that device can not accept banknote;
- **z32** – lights the green light diode and turns off the red. It signals to user that device is ready to receive money.

3. Program implementation

Program is implemented with using of *Unimod* plugin for «*Eclipse*» IDE. The plugin allows designing scheme of system and state machines. Then controlled objects, event providers and user interface are implemented in *Java*. The logic of application is programmed in state machines only.

The executable application can be created using two different approaches.

3.1 Interpreter approach

This approach needs using of *Unimod* libraries for execution.

The main state machine is saved as an *XML*-file and library containing compiled application classes is built. Now the application can be run using the built library, *Unimod* c libraries and the generated *XML*-file. *Unimod* will log events and transitions to console during application work.

The main drawback of the approach is usage of not necessary for application *Unimod* libraries.

3.2 Compilation approach

In this approach the source file is generated. The generated class explicitly creates all the application classes and implements logic defined in state machines. Some of *Unimod* libraries are still needed but less then in interpreter approach. The generated file is located in *default package* and is called `ModelEventProcessor.java`.

3.3 Using instructions

To start the application one must download archive with executable assembly, unpack it and start *run.bat*.

To open the project in *Unimod* one must select "*File/Import*" in system menu, then select "*Existing project in workspace*" in popup window. Then press "*Next >*" button. Directory containg the project files has to be selected.

Conclusion

Automaton-oriented approach ideally suits for design and implementation of technical systems. State machines' diagrams evidently present logic of process and simplify design errors search.

Unimod tool is very helpful in realizing the approach. It does not allow to generate programs from incorrect or incomplete specifications and shows the problems in transition graphs.

This device for payments was not designed for using in practice. The goal was to demonstrate convenience of using automata approach and *Unimod* tool.

Authors have opinion that designing programs using *Unimod* is much more correct then using traditional way because developer does not have to think about program structure in common but only about architecture. This fact forces to split program into common logic and architecture and implementation of independent modules as event providers and controlled objects. One more advantage is simplification of modifying program.

Appendix 1. Example of application log

One can see application log for paying 200 rubles for 88123883636 phone number below.

```
16:10:22,312 INFO [Run] Start event [e21] processing. In state [/A3:Top]
16:10:22,312 INFO [Run] Transition to go found [s1#1.Work##true]
16:10:22,312 INFO [Run] Start event [e21] processing. In state [/A3:1.Work/A1:Top]
16:10:22,312 INFO [Run] Transition to go found [s1#1.Greeting##true]
16:10:22,312 INFO [Run] Start on-enter action [o1.z11] execution
16:10:22,312 INFO [Run] Finish on-enter action [o1.z11] execution
16:10:22,312 DEBUG [Run] Try transition [1.Greeting#2.Number input#e21#true]
16:10:22,312 INFO [Run] Transition to go found [1.Greeting#2.Number input#e21#true]
16:10:22,375 INFO [Run] Start on-enter action [o1.z12] execution
16:10:22,375 INFO [Run] Finish on-enter action [o1.z12] execution
16:10:22,375 INFO [Run] Finish event [e21] processing. In state [/A3:1.Work/A1:2.Number
input]
16:10:22,375 INFO [Run] Finish event [e21] processing. In state [/A3:1.Work]
16:10:30,750 INFO [Run] Start event [e30] processing. In state [/A3:1.Work]
16:10:30,750 INFO [Run] Start event [e30] processing. In state [/A3:1.Work/A1:2.Number
input]
16:10:30,750 DEBUG [Run] Try transition [2.Number input#2.Number input#e30#true]
16:10:30,750 INFO [Run] Transition to go found [2.Number input#2.Number input#e30#true]
16:10:30,750 INFO [Run] Start output action [o2.z23] execution
16:10:30,750 INFO [Run] Finish output action [o2.z23] execution
16:10:30,750 INFO [Run] Start on-enter action [o1.z12] execution
16:10:30,750 INFO [Run] Finish on-enter action [o1.z12] execution
16:10:30,750 INFO [Run] Finish event [e30] processing. In state [/A3:1.Work/A1:2.Number
input]
16:10:30,750 INFO [Run] Finish event [e30] processing. In state [/A3:1.Work]
16:10:31,359 INFO [Run] Start event [e30] processing. In state [/A3:1.Work]
16:10:31,359 INFO [Run] Start event [e30] processing. In state [/A3:1.Work/A1:2.Number
input]
16:10:31,359 DEBUG [Run] Try transition [2.Number input#2.Number input#e30#true]
16:10:31,359 INFO [Run] Transition to go found [2.Number input#2.Number input#e30#true]
16:10:31,359 INFO [Run] Start output action [o2.z23] execution
16:10:31,359 INFO [Run] Finish output action [o2.z23] execution
16:10:31,359 INFO [Run] Start on-enter action [o1.z12] execution
16:10:31,359 INFO [Run] Finish on-enter action [o1.z12] execution
16:10:31,359 INFO [Run] Finish event [e30] processing. In state [/A3:1.Work/A1:2.Number
input]
16:10:31,359 INFO [Run] Finish event [e30] processing. In state [/A3:1.Work]
16:10:31,890 INFO [Run] Start event [e30] processing. In state [/A3:1.Work]
16:10:31,890 INFO [Run] Start event [e30] processing. In state [/A3:1.Work/A1:2.Number
input]
16:10:31,890 DEBUG [Run] Try transition [2.Number input#2.Number input#e30#true]
16:10:31,890 INFO [Run] Transition to go found [2.Number input#2.Number input#e30#true]
16:10:31,890 INFO [Run] Start output action [o2.z23] execution
16:10:31,890 INFO [Run] Finish output action [o2.z23] execution
16:10:31,890 INFO [Run] Start on-enter action [o1.z12] execution
16:10:31,890 INFO [Run] Finish on-enter action [o1.z12] execution
16:10:31,890 INFO [Run] Finish event [e30] processing. In state [/A3:1.Work/A1:2.Number
input]
16:10:31,890 INFO [Run] Finish event [e30] processing. In state [/A3:1.Work]
16:10:32,593 INFO [Run] Start event [e30] processing. In state [/A3:1.Work]
16:10:32,593 INFO [Run] Start event [e30] processing. In state [/A3:1.Work/A1:2.Number
input]
16:10:32,593 DEBUG [Run] Try transition [2.Number input#2.Number input#e30#true]
16:10:32,593 INFO [Run] Transition to go found [2.Number input#2.Number input#e30#true]
16:10:32,593 INFO [Run] Start output action [o2.z23] execution
16:10:32,593 INFO [Run] Finish output action [o2.z23] execution
16:10:32,593 INFO [Run] Start on-enter action [o1.z12] execution
16:10:32,593 INFO [Run] Finish on-enter action [o1.z12] execution
16:10:32,593 INFO [Run] Finish event [e30] processing. In state [/A3:1.Work/A1:2.Number
input]
16:10:32,609 INFO [Run] Finish event [e30] processing. In state [/A3:1.Work]
16:10:34,968 INFO [Run] Start event [e30] processing. In state [/A3:1.Work]
16:10:34,968 INFO [Run] Start event [e30] processing. In state [/A3:1.Work/A1:2.Number
input]
```



```

16:10:41,218 INFO [Run] Start event [e30] processing. In state [/A3:1.Work/A1:2.Number
input]
16:10:41,218 DEBUG [Run] Try transition [2.Number input#2.Number input#e30#true]
16:10:41,218 INFO [Run] Transition to go found [2.Number input#2.Number input#e30#true]
16:10:41,218 INFO [Run] Start output action [o2.z23] execution
16:10:41,218 INFO [Run] Finish output action [o2.z23] execution
16:10:41,218 INFO [Run] Start on-enter action [o1.z12] execution
16:10:41,218 INFO [Run] Finish on-enter action [o1.z12] execution
16:10:41,218 INFO [Run] Finish event [e30] processing. In state [/A3:1.Work/A1:2.Number
input]
16:10:41,218 INFO [Run] Finish event [e30] processing. In state [/A3:1.Work]
16:10:42,718 INFO [Run] Start event [e21] processing. In state [/A3:1.Work]
16:10:42,718 INFO [Run] Start event [e21] processing. In state [/A3:1.Work/A1:2.Number
input]
16:10:42,718 DEBUG [Run] Try transition [2.Number input#3.Money Input#e21#o2.x1]
16:10:42,718 INFO [Run] Start input action [o2.x1] calculation
16:10:42,718 INFO [Run] Finish input action [o2.x1] calculation. Its value is [true]
16:10:42,718 INFO [Run] Transition to go found [2.Number input#3.Money Input#e21#o2.x1]
16:10:42,734 INFO [Run] Start output action [o3.z32] execution
16:10:42,734 INFO [Run] Finish output action [o3.z32] execution
16:10:42,734 INFO [Run] Start on-enter action [o1.z12] execution
16:10:42,734 INFO [Run] Finish on-enter action [o1.z12] execution
16:10:42,734 INFO [Run] Start event [e21] processing. In state [/A3:1.Work/A1:3.Money
Input/A2:Top]
16:10:42,734 INFO [Run] Transition to go found [s4#1.Banknote receiving##true]
16:10:42,734 INFO [Run] Finish event [e21] processing. In state [/A3:1.Work/A1:3.Money
Input/A2:1.Banknote receiving]
16:10:42,734 INFO [Run] Finish event [e21] processing. In state [/A3:1.Work/A1:3.Money
Input]
16:10:42,734 INFO [Run] Finish event [e21] processing. In state [/A3:1.Work]
16:10:47,484 INFO [Run] Start event [e11] processing. In state [/A3:1.Work]
16:10:47,484 INFO [Run] Start event [e11] processing. In state [/A3:1.Work/A1:3.Money
Input]
16:10:47,484 INFO [Run] Start event [e11] processing. In state [/A3:1.Work/A1:3.Money
Input/A2:1.Banknote receiving]
16:10:47,484 DEBUG [Run] Try transition [1.Banknote receiving#1.Banknote
receiving#e11#true]
16:10:47,484 INFO [Run] Transition to go found [1.Banknote receiving#1.Banknote
receiving#e11#true]
16:10:47,484 INFO [Run] Start output action [o2.z22] execution
16:10:47,484 INFO [Run] Finish output action [o2.z22] execution
16:10:47,484 INFO [Run] Start output action [o1.z12] execution
16:10:47,484 INFO [Run] Finish output action [o1.z12] execution
16:10:47,484 INFO [Run] Finish event [e11] processing. In state [/A3:1.Work/A1:3.Money
Input/A2:1.Banknote receiving]
16:10:47,484 INFO [Run] Finish event [e11] processing. In state [/A3:1.Work/A1:3.Money
Input]
16:10:47,484 INFO [Run] Finish event [e11] processing. In state [/A3:1.Work]
16:10:49,375 INFO [Run] Start event [e11] processing. In state [/A3:1.Work]
16:10:49,375 INFO [Run] Start event [e11] processing. In state [/A3:1.Work/A1:3.Money
Input]
16:10:49,375 INFO [Run] Start event [e11] processing. In state [/A3:1.Work/A1:3.Money
Input/A2:1.Banknote receiving]
16:10:49,375 DEBUG [Run] Try transition [1.Banknote receiving#1.Banknote
receiving#e11#true]
16:10:49,375 INFO [Run] Transition to go found [1.Banknote receiving#1.Banknote
receiving#e11#true]
16:10:49,375 INFO [Run] Start output action [o2.z22] execution
16:10:49,375 INFO [Run] Finish output action [o2.z22] execution
16:10:49,375 INFO [Run] Start output action [o1.z12] execution
16:10:49,375 INFO [Run] Finish output action [o1.z12] execution
16:10:49,375 INFO [Run] Finish event [e11] processing. In state [/A3:1.Work/A1:3.Money
Input/A2:1.Banknote receiving]
16:10:49,375 INFO [Run] Finish event [e11] processing. In state [/A3:1.Work/A1:3.Money
Input]
16:10:49,375 INFO [Run] Finish event [e11] processing. In state [/A3:1.Work]
16:10:52,296 INFO [Run] Start event [e11] processing. In state [/A3:1.Work]
16:10:52,296 INFO [Run] Start event [e11] processing. In state [/A3:1.Work/A1:3.Money
Input]
16:10:52,296 INFO [Run] Start event [e11] processing. In state [/A3:1.Work/A1:3.Money
Input/A2:1.Banknote receiving]

```

```

16:10:52,296 DEBUG [Run] Try transition [1.Banknote receiving#1.Banknote
receiving#e11#true]
16:10:52,296 INFO [Run] Transition to go found [1.Banknote receiving#1.Banknote
receiving#e11#true]
16:10:52,296 INFO [Run] Start output action [o2.z22] execution
16:10:52,296 INFO [Run] Finish output action [o2.z22] execution
16:10:52,296 INFO [Run] Start output action [o1.z12] execution
16:10:52,296 INFO [Run] Finish output action [o1.z12] execution
16:10:52,296 INFO [Run] Finish event [e11] processing. In state [/A3:1.Work/A1:3.Money
Input/A2:1.Banknote receiving]
16:10:52,296 INFO [Run] Finish event [e11] processing. In state [/A3:1.Work/A1:3.Money
Input]
16:10:52,296 INFO [Run] Finish event [e11] processing. In state [/A3:1.Work]
16:10:54,843 INFO [Run] Start event [e21] processing. In state [/A3:1.Work]
16:10:54,843 INFO [Run] Start event [e21] processing. In state [/A3:1.Work/A1:3.Money
Input]
16:10:54,843 DEBUG [Run] Try transition [3.Money Input#4.Confirmation#e21#o3.x31]
16:10:54,843 INFO [Run] Start input action [o3.x31] calculation
16:10:54,843 INFO [Run] Finish input action [o3.x31] calculation. Its value is [true]
16:10:54,843 INFO [Run] Transition to go found [3.Money Input#4.Confirmation#e21#o3.x31]
16:10:54,843 INFO [Run] Start output action [o3.z31] execution
16:10:54,843 INFO [Run] Finish output action [o3.z31] execution
16:10:54,843 INFO [Run] Start on-enter action [o1.z13] execution
16:10:54,843 INFO [Run] Finish on-enter action [o1.z13] execution
16:10:54,843 INFO [Run] Finish event [e21] processing. In state
[/A3:1.Work/A1:4.Confirmation]
16:10:54,843 INFO [Run] Finish event [e21] processing. In state [/A3:1.Work]
16:10:56,093 INFO [Run] Start event [e23] processing. In state [/A3:1.Work]
16:10:56,093 DEBUG [Run] Try transition [1.Work#2.Help#e23#true]
16:10:56,093 INFO [Run] Transition to go found [1.Work#2.Help#e23#true]
16:10:56,093 INFO [Run] Start output action [o1.z15] execution
16:10:56,093 INFO [Run] Finish output action [o1.z15] execution
16:10:56,093 INFO [Run] Finish event [e23] processing. In state [/A3:2.Help]
16:10:56,734 INFO [Run] Start event [e23] processing. In state [/A3:2.Help]
16:10:56,734 DEBUG [Run] Try transition [2.Help#1.Work#e23#true]
16:10:56,734 INFO [Run] Transition to go found [2.Help#1.Work#e23#true]
16:10:56,734 INFO [Run] Start output action [o1.z16] execution
16:10:56,734 INFO [Run] Finish output action [o1.z16] execution
16:10:56,734 INFO [Run] Start event [e23] processing. In state
[/A3:1.Work/A1:4.Confirmation]
16:10:56,734 INFO [Run] Finish event [e23] processing. In state
[/A3:1.Work/A1:4.Confirmation]
16:10:56,734 INFO [Run] Finish event [e23] processing. In state [/A3:1.Work]
16:10:57,656 INFO [Run] Start event [e21] processing. In state [/A3:1.Work]
16:10:57,656 INFO [Run] Start event [e21] processing. In state
[/A3:1.Work/A1:4.Confirmation]
16:10:57,656 DEBUG [Run] Try transition [4.Confirmation#5.Goodbye#e21#true]
16:10:57,656 INFO [Run] Transition to go found [4.Confirmation#5.Goodbye#e21#true]
16:10:57,656 INFO [Run] Start output action [o1.z14] execution
16:10:57,656 INFO [Run] Finish output action [o1.z14] execution
16:10:57,656 INFO [Run] Finish event [e21] processing. In state
[/A3:1.Work/A1:5.Goodbye]
16:10:57,656 INFO [Run] Finish event [e21] processing. In state [/A3:1.Work]
16:10:58,718 INFO [Run] Start event [e21] processing. In state [/A3:1.Work]
16:10:58,718 INFO [Run] Start event [e21] processing. In state [/A3:1.Work/A1:5.Goodbye]
16:10:58,718 DEBUG [Run] Try transition [5.Goodbye#1.Greeting#e21#true]
16:10:58,718 INFO [Run] Transition to go found [5.Goodbye#1.Greeting#e21#true]
16:10:58,718 INFO [Run] Start output action [o2.z26] execution
16:10:58,718 INFO [Run] Finish output action [o2.z26] execution
16:10:58,718 INFO [Run] Start on-enter action [o1.z11] execution
16:10:58,718 INFO [Run] Finish on-enter action [o1.z11] execution
16:10:58,718 INFO [Run] Finish event [e21] processing. In state
[/A3:1.Work/A1:1.Greeting]
16:10:58,718 INFO [Run] Finish event [e21] processing. In state [/A3:1.Work]

```

Appendix 2. Generated XML-Definition

Telepay.unimod

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

```

```

</composited>
</node>
<node id="id14" type="com.evelopers.unimod.plugin.eclipse.model.GStateMachine">
  <property name="name" value="A2"/>
  <property name="configManagerClassName"
value="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
  <composited role="top">
    <node id="id15" type="com.evelopers.unimod.plugin.eclipse.model.GTopState">
      <property name="name" value="Top"/>
      <composited role="substates[0]">
        <node id="id17" type="com.evelopers.unimod.plugin.eclipse.model.GInitialState">
          <property name="name" value="s4"/>
        </node>
      </composited>
      <composited role="substates[1]">
        <node id="id18" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
          <property name="name" value="1. Banknote receiving"/>
          <property name="actions" value=""/>
          <property name="subMachineHandles" value=""/>
        </node>
      </composited>
      <composited role="substates[2]">
        <node id="id16" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
          <property name="name" value="2. Banknote return"/>
          <property name="actions" value=""/>
          <property name="subMachineHandles" value=""/>
        </node>
      </composited>
    </node>
  </composited>
</node>
<node id="id6" type="com.evelopers.unimod.plugin.eclipse.model.GStateMachine">
  <property name="name" value="A1"/>
  <property name="configManagerClassName"
value="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
  <composited role="top">
    <node id="id7" type="com.evelopers.unimod.plugin.eclipse.model.GTopState">
      <property name="name" value="Top"/>
      <composited role="substates[0]">
        <node id="id10" type="com.evelopers.unimod.plugin.eclipse.model.GInitialState">
          <property name="name" value="s1"/>
        </node>
      </composited>
      <composited role="substates[1]">
        <node id="id9" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
          <property name="name" value="1. Greeting"/>
          <property name="actions" value="o1.z11"/>
          <property name="subMachineHandles" value=""/>
        </node>
      </composited>
      <composited role="substates[2]">
        <node id="id11" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
          <property name="name" value="2. Number input"/>
          <property name="actions" value="o1.z12"/>
          <property name="subMachineHandles" value=""/>
        </node>
      </composited>
      <composited role="substates[3]">
        <node id="id8" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
          <property name="name" value="3. Money Receiving"/>
          <property name="actions" value="o1.z12"/>
          <property name="subMachineHandles" value="A2"/>
        </node>
      </composited>
    </node>
  </composited>
</node>

```



```

</composited>
<composited role="substates[4]">
  <node id="id12" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
    <property name="name" value="5. Notification" />
    <property name="actions" value="o1.z14" />
    <property name="subMachineHandles" value="" />
  </node>
</composited>
<composited role="substates[5]">
  <node id="id13" type="com.evelopers.unimod.plugin.eclipse.model.GNormalState">
    <property name="name" value="4. Confirmation" />
    <property name="actions" value="o1.z13" />
    <property name="subMachineHandles" value="" />
  </node>
</composited>
</node>
</composited>
<edge client="id3" id="eid1160567883343-1819818" supplier="id19"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
  <property name="name" value="" />
  <property name="supplierRole" value="" />
</edge>
<edge client="id5" id="eid1160567883343-20479628" supplier="id19"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
  <property name="name" value="" />
  <property name="supplierRole" value="" />
</edge>
<edge client="id4" id="eid1160567883343-4576850" supplier="id19"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
  <property name="name" value="" />
  <property name="supplierRole" value="" />
</edge>
<edge client="id19" id="id29" supplier="id0"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
  <property name="name" value="" />
  <property name="supplierRole" value="o1" />
</edge>
<edge client="id19" id="id34" supplier="id1"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
  <property name="name" value="" />
  <property name="supplierRole" value="o2" />
</edge>
<edge client="id22" id="id35" supplier="id23"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
  <property name="labelText" value="" />
</edge>
<edge client="id23" id="id49" supplier="id21"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
  <property name="labelText" value="e23 /o1.z15" />
</edge>
<edge client="id21" id="id31" supplier="id23"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
  <property name="labelText" value="e23 /o1.z16" />
</edge>
<edge client="id14" id="id47" supplier="id0"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
  <property name="name" value="" />
  <property name="supplierRole" value="o1" />
</edge>
<edge client="id14" id="id30" supplier="id2"
type="com.evelopers.unimod.plugin.eclipse.model.GAssociation">
  <property name="name" value="" />

```

```

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

```

```

    <edge client="id11" id="id33" supplier="id8"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e21[o2.x1] /o3.z32"/>
    </edge>
    <edge client="id8" id="id44" supplier="id11"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e22 /o3.z31"/>
    </edge>
    <edge client="id8" id="id39" supplier="id8"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e21[!o3.x31]"/>
    </edge>
    <edge client="id8" id="id43" supplier="id13"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e21[o3.x31] /o3.z31"/>
    </edge>
    <edge client="id12" id="id48" supplier="id9"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e21 /o2.z26"/>
    </edge>
    <edge client="id13" id="id37" supplier="id8"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e22 /o3.z32"/>
    </edge>
    <edge client="id13" id="id27" supplier="id12"
type="com.evelopers.unimod.plugin.eclipse.model.GTransition">
    <property name="labelText" value="e21"/>
    </edge>
</elements>
<diagrams>
<diagram>
    <node-ref node="id0">
    <property name="location" value="(560, 10)"/>
    <property name="size" value="(181, 191)"/>
    </node-ref>
    <node-ref node="id2">
    <property name="location" value="(560, 231)"/>
    <property name="size" value="(181, 140)"/>
    </node-ref>
    <node-ref node="id1">
    <property name="location" value="(560, 413)"/>
    <property name="size" value="(201, 228)"/>
    </node-ref>
    <node-ref node="id3">
    <property name="location" value="(25, 20)"/>
    <property name="size" value="(236, 181)"/>
    </node-ref>
    <node-ref node="id5">
    <property name="location" value="(20, 221)"/>
    <property name="size" value="(241, 160)"/>
    </node-ref>
    <node-ref node="id4">
    <property name="location" value="(25, 420)"/>
    <property name="size" value="(236, 161)"/>
    </node-ref>
    <node-ref node="id19">
    <property name="location" value="(280, 20)"/>
    <property name="size" value="(121, 561)"/>
    </node-ref>
    <node-ref node="id14">
    <property name="location" value="(420, 180)"/>
    <property name="size" value="(121, 281)"/>
    </node-ref>

```

```

<node-ref node="id6">
  <property name="location" value="(780, 180)" />
  <property name="size" value="(130, 252)" />
</node-ref>
<edge-ref edge="eid1160567883343-1819818">
  <property name="labelPosition" value="(100%) -&gt; (-10, -10)" />
</edge-ref>
<edge-ref edge="eid1160567883343-20479628">
  <property name="labelPosition" value="(100%) -&gt; (-10, -10)" />
</edge-ref>
<edge-ref edge="eid1160567883343-4576850">
  <property name="labelPosition" value="(100%) -&gt; (-10, -10)" />
</edge-ref>
<edge-ref edge="id29">
  <property name="labelPosition" value="(100%) -&gt; (-10, -10)" />
</edge-ref>
<edge-ref edge="id34">
  <property name="labelPosition" value="(100%) -&gt; (-10, -10)" />
</edge-ref>
<edge-ref edge="id47">
  <property name="labelPosition" value="(100%) -&gt; (-10, -10)" />
</edge-ref>
<edge-ref edge="id30">
  <property name="labelPosition" value="(100%) -&gt; (-10, -10)" />
</edge-ref>
<edge-ref edge="id24">
  <property name="labelPosition" value="(100%) -&gt; (-10, -10)" />
</edge-ref>
<edge-ref edge="id46">
  <property name="labelPosition" value="(100%) -&gt; (19, -20)" />
</edge-ref>
<edge-ref edge="id36">
  <property name="labelPosition" value="(100%) -&gt; (19, -21)" />
</edge-ref>
<edge-ref edge="id32">
  <property name="labelPosition" value="(100%) -&gt; (-1, -22)" />
</edge-ref>
</diagram>
<diagram>
  <node-ref node="id20" />
  <node-ref node="id22">
    <property name="location" value="(280, 140)" />
  </node-ref>
  <node-ref node="id23">
    <property name="location" value="(200, 260)" />
    <property name="size" value="(181, 101)" />
  </node-ref>
  <node-ref node="id21">
    <property name="location" value="(460, 260)" />
    <property name="size" value="(181, 101)" />
  </node-ref>
  <edge-ref edge="id35">
    <property name="sourceAnchor" value="" />
    <property name="targetAnchor" value="NORTH:5" />
    <property name="bendpoints" value="" />
    <property name="labelPosition" value="(50%) -&gt; (20, -10)" />
  </edge-ref>
  <edge-ref edge="id49">
    <property name="sourceAnchor" value="EAST:2" />
    <property name="targetAnchor" value="WEST:2" />
    <property name="bendpoints" value="" />
    <property name="labelPosition" value="(50%) -&gt; (0, -20)" />
  </edge-ref>

```

```

<edge-ref edge="id31">
  <property name="sourceAnchor" value="WEST:4"/>
  <property name="targetAnchor" value="EAST:4"/>
  <property name="bendpoints" value=""/>
  <property name="labelPosition" value="(50%) -&gt; (0, 20)"/>
</edge-ref>
</diagram>
<diagram>
  <node-ref node="id15"/>
  <node-ref node="id17">
    <property name="location" value="(280, 160)"/>
  </node-ref>
  <node-ref node="id18">
    <property name="location" value="(200, 240)"/>
    <property name="size" value="(161, 140)"/>
  </node-ref>
  <node-ref node="id16">
    <property name="location" value="(460, 240)"/>
    <property name="size" value="(161, 141)"/>
  </node-ref>
  <edge-ref edge="id28">
    <property name="sourceAnchor" value=""/>
    <property name="targetAnchor" value="NORTH:5"/>
    <property name="bendpoints" value=""/>
    <property name="labelPosition" value="(50%) -&gt; (0, -20)"/>
  </edge-ref>
  <edge-ref edge="id40">
    <property name="sourceAnchor" value="SOUTH:2"/>
    <property name="targetAnchor" value="WEST:7"/>
    <property name="bendpoints" value=""/>
    <property name="labelPosition" value="(50%) -&gt; (-35, 45)"/>
  </edge-ref>
  <edge-ref edge="id26">
    <property name="sourceAnchor" value="EAST:2"/>
    <property name="targetAnchor" value="WEST:2"/>
    <property name="bendpoints" value=""/>
    <property name="labelPosition" value="(50%) -&gt; (0, -20)"/>
  </edge-ref>
  <edge-ref edge="id38">
    <property name="sourceAnchor" value="WEST:5"/>
    <property name="targetAnchor" value="EAST:5"/>
    <property name="bendpoints" value=""/>
    <property name="labelPosition" value="(50%) -&gt; (0, -20)"/>
  </edge-ref>
</diagram>
<diagram>
  <node-ref node="id7"/>
  <node-ref node="id10">
    <property name="location" value="(101, 60)"/>
  </node-ref>
  <node-ref node="id9">
    <property name="location" value="(40, 180)"/>
    <property name="size" value="(141, 101)"/>
  </node-ref>
  <node-ref node="id11">
    <property name="location" value="(280, 180)"/>
    <property name="size" value="(161, 101)"/>
  </node-ref>
  <node-ref node="id8">
    <property name="location" value="(580, 180)"/>
    <property name="size" value="(161, 101)"/>
  </node-ref>
  <node-ref node="id12">

```

```

    <property name="location" value="(40, 380)" />
    <property name="size" value="(141, 101)" />
  </node-ref>
  <node-ref node="id13">
    <property name="location" value="(580, 380)" />
    <property name="size" value="(161, 101)" />
  </node-ref>
  <edge-ref edge="id25">
    <property name="sourceAnchor" value="" />
    <property name="targetAnchor" value="NORTH:4" />
    <property name="bendpoints" value="" />
    <property name="labelPosition" value="(50%) -&gt; (26, -20)" />
  </edge-ref>
  <edge-ref edge="id45">
    <property name="sourceAnchor" value="EAST:2" />
    <property name="targetAnchor" value="WEST:2" />
    <property name="bendpoints" value="" />
    <property name="labelPosition" value="(50%) -&gt; (0, -20)" />
  </edge-ref>
  <edge-ref edge="id50">
    <property name="sourceAnchor" value="WEST:3" />
    <property name="targetAnchor" value="EAST:3" />
    <property name="bendpoints" value="" />
    <property name="labelPosition" value="(50%) -&gt; (1, 20)" />
  </edge-ref>
  <edge-ref edge="id42">
    <property name="sourceAnchor" value="EAST:5" />
    <property name="targetAnchor" value="SOUTH:7" />
    <property name="bendpoints" value="" />
    <property name="labelPosition" value="(50%) -&gt; (35, 45)" />
  </edge-ref>
  <edge-ref edge="id51">
    <property name="sourceAnchor" value="WEST:1" />
    <property name="targetAnchor" value="NORTH:2" />
    <property name="bendpoints" value="" />
    <property name="labelPosition" value="(50%) -&gt; (-34, -55)" />
  </edge-ref>
  <edge-ref edge="id41">
    <property name="sourceAnchor" value="SOUTH:2" />
    <property name="targetAnchor" value="WEST:5" />
    <property name="bendpoints" value="" />
    <property name="labelPosition" value="(50%) -&gt; (-65, 45)" />
  </edge-ref>
  <edge-ref edge="id33">
    <property name="sourceAnchor" value="EAST:3" />
    <property name="targetAnchor" value="WEST:3" />
    <property name="bendpoints" value="" />
    <property name="labelPosition" value="(50%) -&gt; (0, -20)" />
  </edge-ref>
  <edge-ref edge="id44">
    <property name="sourceAnchor" value="WEST:4" />
    <property name="targetAnchor" value="EAST:4" />
    <property name="bendpoints" value="" />
    <property name="labelPosition" value="(50%) -&gt; (1, 10)" />
  </edge-ref>
  <edge-ref edge="id39">
    <property name="sourceAnchor" value="NORTH:7" />
    <property name="targetAnchor" value="EAST:1" />
    <property name="bendpoints" value="" />
    <property name="labelPosition" value="(50%) -&gt; (55, -55)" />
  </edge-ref>
  <edge-ref edge="id43">
    <property name="sourceAnchor" value="SOUTH:3" />

```

```

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

```

Appendix 3. Source code

Keypad.java

```
package ru.ifmo.telepay;

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

import java.awt.event.*;

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

    /**
     * @unimod.event.descr Digit was typed.
     */
    public static final String E30 = "e30";

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

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

        KeypadPanel keypadPanel = TelepayScreen.getInstance().getKeyPadPanel();

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

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

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



```

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

```

KeyPadPanel.java

```

package ru.ifmo.telepay;

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

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

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

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

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

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

        buttons = new JButton[10];

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

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

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

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

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

```

```

        this.getBounds().y + 3 * btnHeight,
        btnWidth, btnHeight));
    }

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

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

```

Memory.java

```

package ru.ifmo.telepay;

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

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

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

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

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

    /**
     * Returns if the number entered is valid.
     * In this model number is considered valid if consists of 11 digits.
     */
}

```

```

    * @return Returns true if provided number is valid.
    */
    public static boolean isNumberValid() {
        return (number.length() == 11);
    }

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

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

    /**
     * @unimod.action.descr Add banknote nominal to user's account.
     */
    public void z22(StateMachineContext context) {
        /*TODO: automatically generated by UniMod method*/
        money += ((Integer)context.getEventContext().getParameter(Memory.MONEY)).intValue();
    }

    /**
     * @unimod.action.descr Append digit to the phone number.
     */
    public void z23(StateMachineContext context) {
        number += context.getEventContext().getParameter(Memory.DIGIT).toString();
    }

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

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

    /**
     * @unimod.action.descr Is the phone number valid.
     */
    public boolean x1(StateMachineContext context) {
        /*TODO: automatically generated by UniMod method*/
        return (isNumberValid());
    }

```

```

/**
 * @animod.action.descr Reset information.
 */
public void z26(StateMachineContext context) {
    /*TODO: automatically generated by UniMod method*/
    number = "";
    money = 0;
}
}

```

MoneyReceiverPanel.java

```

package ru.ifmo.telepay;

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

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

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

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

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

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

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

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

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

    /**
     * Image of receiver turned off.
     */

```

```

private ImageIcon iconOff;

/**
 * Image of receiver turned on.
 */
private ImageIcon iconOn;

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

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

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

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

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

    setReceiverValid(false);
}

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

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

/**

```

```

    * Sets receiver to valid or invalid state.
    *
    * @param valid If valid is true then receiver is set to valid state.
    */
    public void setReceiverValid(boolean valid) {
        receiverButton.setIcon(valid ? iconOn : iconOff);
    }

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

```

MoneyReceptor.java

```

package ru.ifmo.telepay;

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

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

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

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

    /**
     * @unimod.event.descr The banknote is recognized.
     */
    public static final String E11 = "e11";

    /**
     * @unimod.event.descr The banknote is not recognized.
     */
    public static final String E12 = "e12";

    /**
     * @unimod.event.descr The banknote is extracted.
     */
    public static final String E13 = "e13";
}

```

```

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

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

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

}

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

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

/**
 * @unimod.action.descr Is money receiving enabled.
 */
public boolean x31(StateMachineContext context) {
    /*TODO: automatically generated by UniMod method*/
    MoneyReceiverPanel panel = TelepayScreen.getInstance().getMoneyReceiverPanel();
    return panel.isReceiverValid();
}
}

```

TelepayScreen.java

```
package ru.ifmo.telepay;

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

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

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

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

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

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

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

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

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

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



```

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

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

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

this.getContentPane().add(content);

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

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

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

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

```

UserScreen.java

```

package ru.ifmo.telepay;

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

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

```

```

/**
 * Text to be shown as greeting.
 */
public static final String GREETING = "Press OK to start.";

/**
 * Text to be shown at text numbering.
 */
public static final String ENTER_NUMBER = "Please enter your number: %s \n";

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

/**
 * Text to be shown at confirmation.
 */
public static final String CONFIRM_INFORMATION = "Please confirm provided information: \n"
    + "%d rubles will be paid for number %s. \n";

/**
 * Text to be shown at the end of work.
 */
public static final String GOODBYE = "Your payment was successfully delivered. \n" +
    "Thank you for using Telepay.";

/**
 * Text to be shown as help.
 */
public static final String HELP = "1) Enter the number you want to pay for.\n" +
    "2) Enter money through the money receptor.\n" +
    "3) Confirm provided information.\n" +
    "4) Wait for payment confirmation on the screen.\n" +
    "\n" +
    "You can go to next step by pressing \"OK\" button or return to previous\n" +
    "step to modify provided information by pressing \"Cancel\"";

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

/**
 * @unimod.action.descr Show money and number input dialog.
 */
public void z12(StateMachineContext context) {
    /*TODO: automatically generated by UniMod method*/
    UserScreenPanel screen = TelepayScreen.getInstance().getUserScreenPanel();
    screen.setTerminalText(ENTER_MONEY, new Object[] {Memory.getNumber(), new
Integer(Memory.getMoney())});
    screen.enableOK(Memory.isNumberValid());
}

/**
 * @unimod.action.descr Show information confirmation dialog.

```

```

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

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

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

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

```

UserScreenPanel.java

```

package ru.ifmo.telepay;

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

import java.io.*;

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

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

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

    /**

```

```

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

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

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

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

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

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

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

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

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

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

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

    help = new JButton(HELP_TEXT);
    help.setVisible(true);
    this.add(help);
    help.setBounds(this.getBounds().x + space,
        this.getBounds().y + this.getBounds().height - btnHeight,

```

```

        btnWidth, btnHeight);

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

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

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

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

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

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

/**
 * Sets text to terminal.

```

```

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

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

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

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

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

```

UserTerminal.java

```

package ru.ifmo.telepay;

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

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

```

```

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

    /**
     * @unimod.event.descr OK button was pressed.
     */
    public static final String E21 = "e21";
    /**
     * @unimod.event.descr Cancel button was pressed.
     */
    public static final String E22 = "e22";

    /**
     * @unimod.event.descr Help button was pressed.
     */
    public static final String E23 = "e23";

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

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

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

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

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

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

```

Appendix 4. Generated Java source.

ModelEventProcessor.java

```
/**
 * This file was generated from model [Model1] on [Sat Oct 14 04:07:16 MSD 2006].
 * Do not change content of this file.
 */

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

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

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

public class ModelEventProcessor extends AbstractEventProcessor {

    private ModelStructure modelStructure;

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

    private int decodeStateMachine(String sm) {

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

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

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

        return -1;
    }

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

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

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



```

    _A1 = new A1EventProcessor();
}

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

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

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

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

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

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

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

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

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

    public ControlledObjectsManager getControlledObjectsManager() {
        return controlledObjectsManager;
    }

    public EventProvidersManager getEventProvidersManager() {
        return eventProvidersManager;
    }
}

```

```

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

    public void dispose() {
    }

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

public ModelStructure getModelStructure() {
    return modelStructure;
}

```

```

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

```

```

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

```

```

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

```

// get state machine from path

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

```
try {
```

```
    switch (sm) {
```

```
        case A3:
```

```
            return _A3.process(event, context, path, config);
```

```
        case A2:
```

```
            return _A2.process(event, context, path, config);
```

```
        case A1:
```

```
            return _A1.process(event, context, path, config);
```

```
        default:
```

```
            throw new EventProcessorException("Unknown state machine [" + path.getStateMachine() + "]);
```

```
    }
```

```
} catch (Exception e) {
```

```
    if (e instanceof SystemException) {
```

```
        throw (SystemException) e;
```

```
    } else {
```

```
        throw new SystemException(e);
```

```
    }
```

```
}
```

```

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

```

// get state machine from path

```

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

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

```

```

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

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

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

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

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

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

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

```

```

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

```

```

private class A3EventProcessor {

  // states
  private static final int Top = 1;
  private static final int s1 = 2;
  private static final int _1_Work = 3;
  private static final int _2_Help = 4;

  private int decodeState(String state) {

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

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

    if ("1.Work".equals(state)) {
      return _1_Work;
    } else

    if ("2.Help".equals(state)) {
      return _2_Help;
    }

    return -1;
  }

  // events
  private static final int e23 = 1;

  private int decodeEvent(String event) {

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

```

```

    return -1;
}

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

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

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

    config = transiteToStableState(context, path, config);

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

    return config;
}

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

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

                return;
            case _1_Work:
                // 1.Work includes A1

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

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

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

                return;
            case _2_Help:

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

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

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

    switch (s) {
        case Top:

```

```

fireComeToState(context, path, "s1");

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

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

// 1.Work []

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

return config;
}

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

```

```

    BitSet calculatedInputActions = new BitSet(0);

```

```

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

```

```

    int e = decodeEvent(event.getName());

```

```

    while (true) {
        switch (s) {
            case _1_Work:

```

```

                switch (e) {
                    case e23:

```

```

                        // 1.Work->2.Help e23[true]/o1.z15

```

```

                        fireTransitionCandidate(context, path, "1.Work", event, "1.Work#2.Help#e23#true");

```

```

                        fireTransitionFound(context, path, "1.Work", event, "1.Work#2.Help#e23#true");

```

```

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

```

```

                        o1.z15(context);

```

```

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

```

```

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

```

```

                        // 2.Help []

```

```

                        return new StateMachineConfig("2.Help");

```

```

                    default:

```

```

                        // transition not found

```

```

                        return config;

```

```

                    }

```

```

            case _2_Help:

```

```

switch (e) {
    case e23:

        // 2.Help->1.Work e23[true]/o1.z16

        fireTransitionCandidate(context, path, "2.Help", event, "2.Help#1.Work#e23#true");

        fireTransitionFound(context, path, "2.Help", event, "2.Help#1.Work#e23#true");

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

        o1.z16(context);

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

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

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

    default:

        // transition not found
        return config;
}

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

```

```

private class A2EventProcessor {

    // states
    private static final int Top = 1;
    private static final int s4 = 2;
    private static final int _1__Banknote_receiving = 3;
    private static final int _2__Banknote_return = 4;

    private int decodeState(String state) {

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

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

        if ("1. Banknote receiving".equals(state)) {
            return _1__Banknote_receiving;
        } else

        if ("2. Banknote return".equals(state)) {
            return _2__Banknote_return;
        }
    }
}

```



```

    }

    return -1;
}

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

private int decodeEvent(String event) {

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

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

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

    return -1;
}

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

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

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

    config = transiteToStableState(context, path, config);

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

    return config;
}

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

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

                return;
            case _1__Banknote_receiving:

                return;
            case _2__Banknote_return:

```

```

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

```

```

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

```

```

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

```

```

    switch (s) {
        case Top:

```

```

            fireComeToState(context, path, "s4");

```

```

            // s4->1. Banknote receiving [true]/

```

```

            event = Event.NO_EVENT;

```

```

            fireTransitionFound(context, path, "s4", event, "s4#1. Banknote receiving##true");

```

```

            fireComeToState(context, path, "1. Banknote receiving");

```

```

            // 1. Banknote receiving []

```

```

            return new StateMachineConfig("1. Banknote receiving");

```

```

        }

```

```

    return config;
}

```

```

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

```

```

    BitSet calculatedInputActions = new BitSet(0);

```

```

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

```

```

    int e = decodeEvent(event.getName());

```

```

    while (true) {

```

```

        switch (s) {

```

```

            case _1__Banknote_receiving:

```

```

                switch (e) {

```

```

                    case e11:

```

```

                        // 1. Banknote receiving->1. Banknote receiving e11[true]/o2.z22,o1.z12

```

```

                        fireTransitionCandidate(context, path, "1. Banknote receiving", event, "1. Banknote
receiving#1. Banknote receiving#e11#true");

```

```

                        fireTransitionFound(context, path, "1. Banknote receiving", event, "1. Banknote receiving#1.
Banknote receiving#e11#true");

```

```

                        fireBeforeOutputActionExecution(context, path, "1. Banknote receiving#1. Banknote
receiving#e11#true", "o2.z22");

```

```

        o2.z22(context);

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

        o1.z12(context);

        fireAfterOutputActionExecution(context, path, "1. Banknote receiving#1. Banknote
receiving#e11#true", "o1.z12");

        fireComeToState(context, path, "1. Banknote receiving");

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

    case e12:

        // 1. Banknote receiving->2. Banknote return e12[true]/o3.z31

        fireTransitionCandidate(context, path, "1. Banknote receiving", event, "1. Banknote
receiving#2. Banknote return#e12#true");

        fireTransitionFound(context, path, "1. Banknote receiving", event, "1. Banknote receiving#2.
Banknote return#e12#true");

        fireBeforeOutputActionExecution(context, path, "1. Banknote receiving#2. Banknote
return#e12#true", "o3.z31");

        o3.z31(context);

        fireAfterOutputActionExecution(context, path, "1. Banknote receiving#2. Banknote
return#e12#true", "o3.z31");

        fireComeToState(context, path, "2. Banknote return");

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

    default:

        // transition not found
        return config;
    }

    case _2__Banknote_return:

        switch (e) {
            case e13:

                // 2. Banknote return->1. Banknote receiving e13[true]/o3.z32

                fireTransitionCandidate(context, path, "2. Banknote return", event, "2. Banknote return#1.
Banknote receiving#e13#true");

```

```

        fireTransitionFound(context, path, "2. Banknote return", event, "2. Banknote return#1.
Banknote receiving#e13#true");

        fireBeforeOutputActionExecution(context, path, "2. Banknote return#1. Banknote
receiving#e13#true", "o3.z32");

        o3.z32(context);

        fireAfterOutputActionExecution(context, path, "2. Banknote return#1. Banknote
receiving#e13#true", "o3.z32");

        fireComeToState(context, path, "1. Banknote receiving");

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

    default:

        // transition not found
        return config;
    }

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

}

}

}

private class A1EventProcessor {

    // states
    private static final int Top = 1;
    private static final int s1 = 2;
    private static final int _1__Greeting = 3;
    private static final int _2__Number_input = 4;
    private static final int _3__Money_Receiving = 5;
    private static final int _5__Notification_ = 6;
    private static final int _4__Confirmation = 7;

    private int decodeState(String state) {

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

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

        if ("1. Greeting".equals(state)) {
            return _1__Greeting;
        } else

        if ("2. Number input".equals(state)) {
            return _2__Number_input;
        } else

```

```

    if ("3. Money Receiving".equals(state)) {
        return _3__Money_Receiving;
    } else

    if ("5. Notification ".equals(state)) {
        return _5__Notification_;
    } else

    if ("4.Confirmation".equals(state)) {
        return _4__Confirmation;
    }

    return -1;
}

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

private int decodeEvent(String event) {

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

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

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

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

    return -1;
}

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

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

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

    config = transiteToStableState(context, path, config);

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

    return config;
}

```

```

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

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

                return;
            case _1__Greeting:

                return;
            case _2__Number_input:

                return;
            case _3__Money_Receiving:
                // 3. Money Receiving includes A2

                fireBeforeSubmachineExecution(context, event, path, "3. Money Receiving", "A2");

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

                fireAfterSubmachineExecution(context, event, path, "3. Money Receiving", "A2");

                return;
            case _5__Notification_:

                return;
            case _4__Confirmation:

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

```

```

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

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

    switch (s) {
        case Top:

            fireComeToState(context, path, "s1");

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

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

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

            o1.z11(context);

```

```

        fireAfterOutputActionExecution(context, path, "s1#1. Greeting##true", "o1.z11");
        return new StateMachineConfig("1. Greeting");
    }
    return config;
}

```

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

boolean

```

    o2_x1 = false;
    o3_x31 = false;

```

BitSet calculatedInputActions = **new** BitSet(2);

```

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

```

```

while (true) {
    switch (s) {
        case _1__ Greeting:

```

```

            switch (e) {
                case e21:

```

// 1. Greeting->2. Number input e21[true]/

```

                fireTransitionCandidate(context, path, "1. Greeting", event, "1. Greeting#2. Number
input#e21#true");

```

```

                fireTransitionFound(context, path, "1. Greeting", event, "1. Greeting#2. Number
input#e21#true");

```

```

                fireComeToState(context, path, "2. Number input");

```

// 2. Number input [o1.z12]

```

                fireBeforeOutputActionExecution(context, path, "1. Greeting#2. Number input#e21#true",
"o1.z12");

```

```

                o1.z12(context);

```

```

                fireAfterOutputActionExecution(context, path, "1. Greeting#2. Number input#e21#true",
"o1.z12");
                return new StateMachineConfig("2. Number input");

```

default:

```

        // transition not found
        return config;
    }

```

```

    case _2__ Number_input:

```

```

switch (e) {
  case e31:

    // 2. Number input->2. Number input e31[true]/o2.z25

    fireTransitionCandidate(context, path, "2. Number input", event, "2. Number input#2. Number
input#e31#true");

    fireTransitionFound(context, path, "2. Number input", event, "2. Number input#2. Number
input#e31#true");

    fireBeforeOutputActionExecution(context, path, "2. Number input#2. Number input#e31#true",
"o2.z25");

    o2.z25(context);

    fireAfterOutputActionExecution(context, path, "2. Number input#2. Number input#e31#true",
"o2.z25");

    fireComeToState(context, path, "2. Number input");

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

    o1.z12(context);

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

  case e22:

    // 2. Number input->1. Greeting e22[true]/o2.z24

    fireTransitionCandidate(context, path, "2. Number input", event, "2. Number input#1.
Greeting#e22#true");

    fireTransitionFound(context, path, "2. Number input", event, "2. Number input#1.
Greeting#e22#true");

    fireBeforeOutputActionExecution(context, path, "2. Number input#1. Greeting#e22#true",
"o2.z24");

    o2.z24(context);

    fireAfterOutputActionExecution(context, path, "2. Number input#1. Greeting#e22#true",
"o2.z24");

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

    // 1. Greeting [o1.z11]
    fireBeforeOutputActionExecution(context, path, "2. Number input#1. Greeting#e22#true",
"o1.z11");

    o1.z11(context);

```



```

"o1.z11");
    fireAfterOutputActionExecution(context, path, "2. Number input#1. Greeting#e22#true",
    return new StateMachineConfig("1. Greeting");

    case e30:
        // 2. Number input->2. Number input e30[true]/o2.z23
        fireTransitionCandidate(context, path, "2. Number input", event, "2. Number input#2. Number
input#e30#true");

        fireTransitionFound(context, path, "2. Number input", event, "2. Number input#2. Number
input#e30#true");

        fireBeforeOutputActionExecution(context, path, "2. Number input#2. Number input#e30#true",
"o2.z23");

        o2.z23(context);

        fireAfterOutputActionExecution(context, path, "2. Number input#2. Number input#e30#true",
"o2.z23");

        fireComeToState(context, path, "2. Number input");

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

        o1.z12(context);

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

    case e21:
        // 2. Number input->2. Number input e21[!o2.x1]/
        fireTransitionCandidate(context, path, "2. Number input", event, "2. Number input#2. Number
input#e21#!o2.x1");

        if (!isInputActionCalculated(calculatedInputActions, _o2_x1)) {
            fireBeforeInputActionExecution(context, path, "2. Number input#2. Number
input#e21#!o2.x1", "o2.x1");

            o2_x1 = o2.x1(context);

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

        if (!o2_x1) {
            fireTransitionFound(context, path, "2. Number input", event, "2. Number input#2. Number
input#e21#!o2.x1");

            fireComeToState(context, path, "2. Number input");

```

```

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

        o1.z12(context);

        fireAfterOutputActionExecution(context, path, "2. Number input#2. Number
input#e21#!o2.x1", "o1.z12");
        return new StateMachineConfig("2. Number input");
    }
    // 2. Number input->3. Money Receiving e21[o2.x1]/o3.z32

    fireTransitionCandidate(context, path, "2. Number input", event, "2. Number input#3. Money
Receiving#e21#o2.x1");

    if (o2_x1) {

        fireTransitionFound(context, path, "2. Number input", event, "2. Number input#3. Money
Receiving#e21#o2.x1");

        fireBeforeOutputActionExecution(context, path, "2. Number input#3. Money
Receiving#e21#o2.x1", "o3.z32");

        o3.z32(context);

        fireAfterOutputActionExecution(context, path, "2. Number input#3. Money
Receiving#e21#o2.x1", "o3.z32");

        fireComeToState(context, path, "3. Money Receiving");

        // 3. Money Receiving [o1.z12]
        fireBeforeOutputActionExecution(context, path, "2. Number input#3. Money
Receiving#e21#o2.x1", "o1.z12");

        o1.z12(context);

        fireAfterOutputActionExecution(context, path, "2. Number input#3. Money
Receiving#e21#o2.x1", "o1.z12");
        return new StateMachineConfig("3. Money Receiving");
    }

    // transition not found
    return config;
default:

    // transition not found
    return config;
}

case _3__Money_Receiving:

    switch (e) {
        case e22:

            // 3. Money Receiving->2. Number input e22[true]/o3.z31

            fireTransitionCandidate(context, path, "3. Money Receiving", event, "3. Money Receiving#2.
Number input#e22#true");

```

```

        fireTransitionFound(context, path, "3. Money Receiving", event, "3. Money Receiving#2.
Number input#e22#true");

        fireBeforeOutputActionExecution(context, path, "3. Money Receiving#2. Number
input#e22#true", "o3.z31");

        o3.z31(context);

        fireAfterOutputActionExecution(context, path, "3. Money Receiving#2. Number
input#e22#true", "o3.z31");

        fireComeToState(context, path, "2. Number input");

        // 2. Number input [o1.z12]
        fireBeforeOutputActionExecution(context, path, "3. Money Receiving#2. Number
input#e22#true", "o1.z12");

        o1.z12(context);

        fireAfterOutputActionExecution(context, path, "3. Money Receiving#2. Number
input#e22#true", "o1.z12");
        return new StateMachineConfig("2. Number input");

    case e21:

        // 3. Money Receiving->3. Money Receiving e21[!o3.x31]/

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

        if (!isInputActionCalculated(calculatedInputActions, _o3_x31)) {

            fireBeforeInputActionExecution(context, path, "3. Money Receiving#3. Money
Receiving#e21#!o3.x31", "o3.x31");

            o3_x31 = o3.x31(context);

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

        if (!o3_x31) {

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

            fireComeToState(context, path, "3. Money Receiving");

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

            o1.z12(context);

            fireAfterOutputActionExecution(context, path, "3. Money Receiving#3. Money
Receiving#e21#!o3.x31", "o1.z12");
            return new StateMachineConfig("3. Money Receiving");
        }

```

```

    }
    // 3. Money Receiving->4. Confirmation e21[o3.x31]/o3.z31

    fireTransitionCandidate(context, path, "3. Money Receiving", event, "3. Money
Receiving#4. Confirmation#e21#o3.x31");

    if (o3_x31) {

        fireTransitionFound(context, path, "3. Money Receiving", event, "3. Money
Receiving#4. Confirmation#e21#o3.x31");

        fireBeforeOutputActionExecution(context, path, "3. Money
Receiving#4. Confirmation#e21#o3.x31", "o3.z31");

        o3.z31(context);

        fireAfterOutputActionExecution(context, path, "3. Money
Receiving#4. Confirmation#e21#o3.x31", "o3.z31");

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

        // 4. Confirmation [o1.z13]
        fireBeforeOutputActionExecution(context, path, "3. Money
Receiving#4. Confirmation#e21#o3.x31", "o1.z13");

        o1.z13(context);

        fireAfterOutputActionExecution(context, path, "3. Money
Receiving#4. Confirmation#e21#o3.x31", "o1.z13");
        return new StateMachineConfig("4. Confirmation");

    }

    // transition not found
    return config;
default:

    // transition not found
    return config;
}

case _5__Notification_:

    switch (e) {
        case e21:

            // 5. Notification ->1. Greeting e21[true]/o2.z26

            fireTransitionCandidate(context, path, "5. Notification ", event, "5. Notification #1.
Greeting#e21#true");

            fireTransitionFound(context, path, "5. Notification ", event, "5. Notification #1.
Greeting#e21#true");

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

            o2.z26(context);

```

```

"o2.z26");
    fireAfterOutputActionExecution(context, path, "5. Notification #1. Greeting#e21#true",

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

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

    o1.z11(context);

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

    default:

        // transition not found
        return config;
    }

    case _4_ Confirmation:

        switch (e) {
            case e22:

                // 4. Confirmation->3. Money Receiving e22[true]/o3.z32

                fireTransitionCandidate(context, path, "4. Confirmation", event, "4. Confirmation#3. Money
                Receiving#e22#true");

                fireTransitionFound(context, path, "4. Confirmation", event, "4. Confirmation#3. Money
                Receiving#e22#true");

                fireBeforeOutputActionExecution(context, path, "4. Confirmation#3. Money
                Receiving#e22#true", "o3.z32");

                o3.z32(context);

                fireAfterOutputActionExecution(context, path, "4. Confirmation#3. Money Receiving#e22#true",
                "o3.z32");

                fireComeToState(context, path, "3. Money Receiving");

                // 3. Money Receiving [o1.z12]
                fireBeforeOutputActionExecution(context, path, "4. Confirmation#3. Money
                Receiving#e22#true", "o1.z12");

                o1.z12(context);

                fireAfterOutputActionExecution(context, path, "4. Confirmation#3. Money Receiving#e22#true",
                "o1.z12");
                return new StateMachineConfig("3. Money Receiving");

            case e21:

                // 4. Confirmation->5. Notification e21[true]/

```

```

        fireTransitionCandidate(context, path, "4.Confirmation", event, "4.Confirmation#5.
Notification #e21#true");

        fireTransitionFound(context, path, "4.Confirmation", event, "4.Confirmation#5. Notification
#e21#true");

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

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

        o1.z14(context);

        fireAfterOutputActionExecution(context, path, "4.Confirmation#5. Notification #e21#true",
"o1.z14");

        return new StateMachineConfig("5. Notification ");

    default:

        // transition not found
        return config;
    }

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

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

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

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

    return b;
}
}
}

```