

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

Computer Technologies Department

A. V. Vokin, I. A. Pimenov, A. A. Shalyto

Modeling of the Car's Auto Gear

Programming with switch-technology and UniMod developer environment

Project documentation

Project was created under
«Foundation for open project documentation»

<http://is.ifmo.ru>

Saint - Petersburg

2006

Contents

Introduction.....	3
1. Target setting.....	4
2. Performance script	5
2.1 Informal script text.....	5
2.2. Formal script	5
3. Class diagram.....	5
4. Classes descriptions	7
4.1. Interface HydroVisualizer	7
4.1.1. Methods descriptions	7
4.2. Class HydroVisualizerForm	7
4.3 Class JFrame.....	7
4.4. Class CustomEventProvider	7
4.5. Class EventProvider.....	7
4.6. Class Hydro	7
4.7. Interface HydroHandler.....	8
4.8. Class ControlledObject.....	8
5. Automata Automata.....	9
5.1. Description.....	9
5.2. Connections diagram.....	9
5.3. Transitions graph.....	10
6. Implementation	10
6.1 Interpretation.....	10
6.2 Compilation.....	10
Resume.....	10
Literature	11
Application1. XML automata description	11
Application2. Source codes for events providers and controlled objects	12
CustomEventProvider.java	12
Hydro.java.....	13
Application 3. Source code of <i>Java</i> -class representing the automata.....	16
Application 4. Source code of the application interfaces.....	29
HydroVisualizerForm.java	29
TachometrPanel.java	32

Introduction

As shown in this work, *SWITCH*-technology, suggested in [1, 2], is, as we think, the most natural decision for a wide classes of problems to control events-based systems. That's why its application is expedient for solving problems of developing simulators of such systems.

The goal of this work – modeling car auto gear basing on *SWITCH*-technology and *UniMod development environment*, that is used for automata programming.

You can find more on this technology at <http://is.ifmo.ru>, and more about *UniMod development environment* – at <http://unimod.sourceforge.net>.

Application is created using IDE *Eclipse 3.1*. At the same time *UniMod* is a plug-in to mentioned above IDE. Release 1.2.15 was used, that supports *JDK 1.5*.

1. Target setting

Goal of this project – is to build model to imitate auto car gear. It should follows the down below requirements.

1. Management of car auto gear is accomplished with three actions:
 - a) switching on the ignition;
 - b) switching direction;
 - c) holding on accelerator pedal.
2. Management system, that is made basing on finite automata, should provide control over number of rotations and in-time gear switching.
3. It is thought that engine is unable to work with more then 3000 rotations a minute.
4. Engine is turned on by pressing button *ON*.
5. User chooses direction using panel *DIRECTION*:
 - a) Forward direction – *FORWARD*;
 - b) Backward direction – *BACKWARD*;
 - c) Neutral – *NEUTRAL*. In this case engine is turned on, but is not moving.
6. While keeping *GAS* button pressed on, engine is increasing rotations number. At the moment number of 2000 rotations is reached, switching on a higher gear is happened;
7. User may stop engine, only in case being on *NEUTRAL* gear, by pressing the button *OFF*.

Application screenshot is shown on Fig. 1

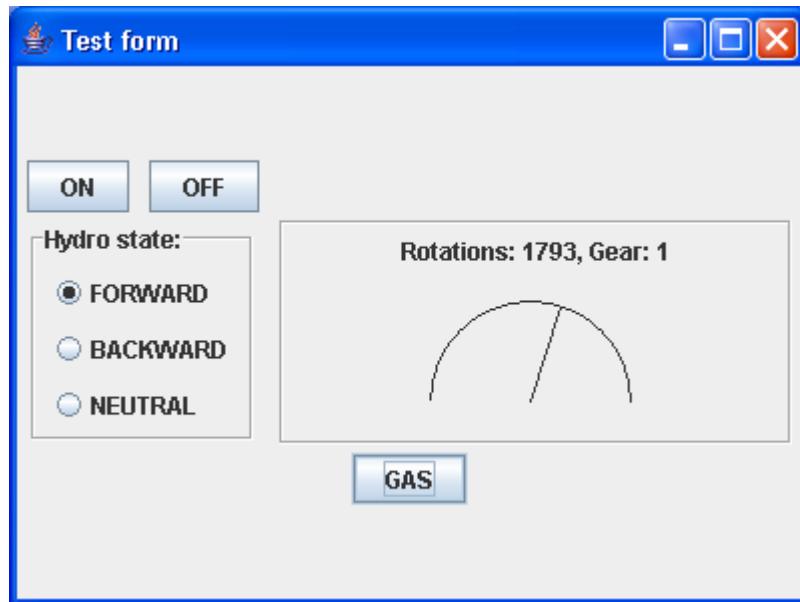


Fig. 1. Example of running application

2. Performance script

2.1 Informal script text

Car's engine is turned off, that's why all user interactions, except turning the engine on, doesn't change vehicle state.

User turns engine on. Tachometer's arrow gets up to 3000 transactions in minute. Now vehicle gives response on pushing down gas – it increases its transactions (tachometers arrow moves right). At this moment, even if number of transactions will exceed 3000 transactions in a minute. Turning gear to another state would not happen, because we don't know the direction where we are moving (auto gear's state in neutral).

After user has chosen moving direction, indicator appears on the form, ,that shows us number of engines transactions and state of auto gear. Now, if we chose forward direction, then after exceeding 3000 transactions in a minute switching to next gear state happens. Now turning the engine off or changing direction can be done only through neutral gear state.

And if we are exceeding 3000 transactions a minute, but we are at 5 gear state, than there no switching will happen.

2.2. Formal script

After running application, to turn on engine user have to press ON button. To turn engine off and exit program, user should press OFF button. Then we should chose directions using buttons FORWARD or BACKWARD. To increase number of transactions user should press GAS button. When exceeding 2000 transactions, switching to higher gear state happens (but only if it was not five gear state). While GAS pedal is not pressed engine decreases transactions and when they are just about 2000 switching to a lower gear state happens (but only if it was not first gear state).

3. Class diagram

On fig. 2 you can see class diagram for model of car auto gear that is made with *SWITCH*-technology.

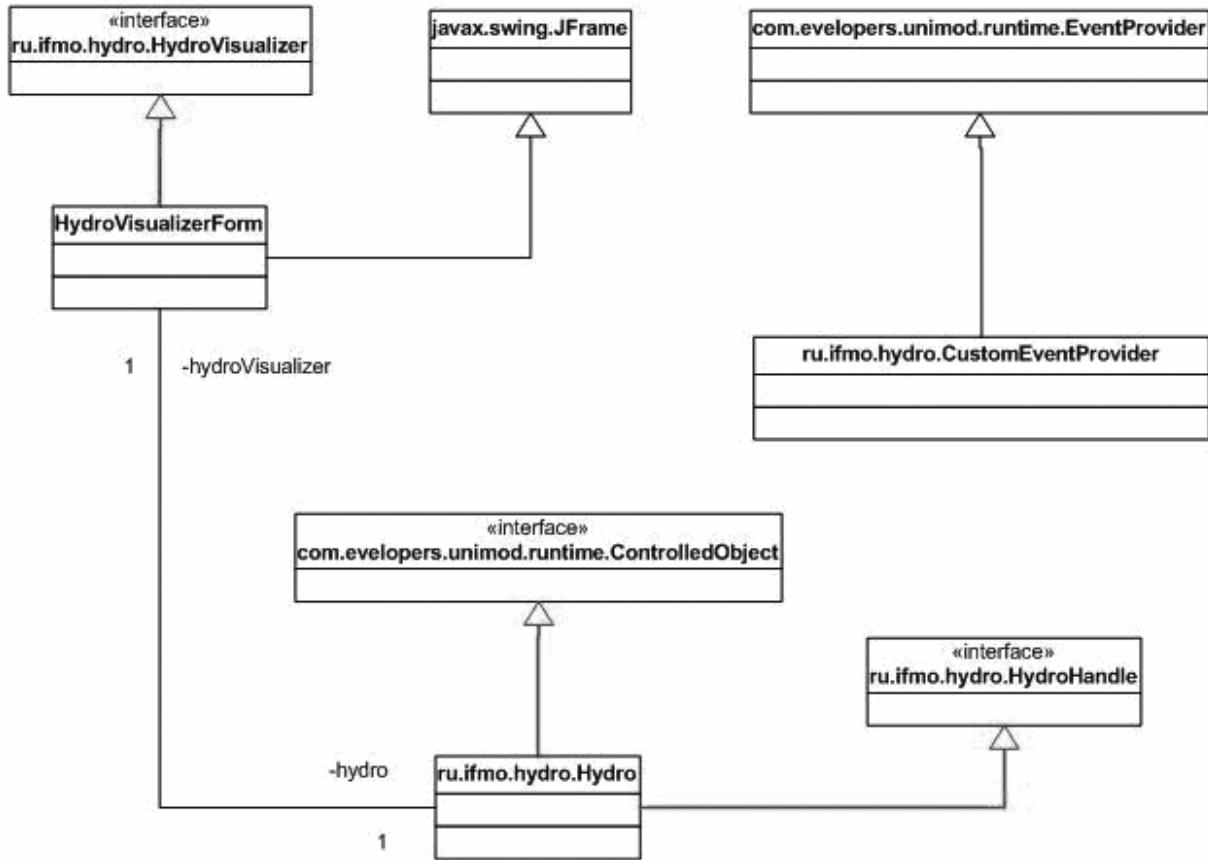


Fig.2. Class diagram

Let's shortly describe how classes on fig. 2 are connected:

- object of class **Hydro** – control object;
- interface **HydroHandler** – consists of methods, that affect user interactions to automata;
- interface **ControlledObject** – Interface of package *UniMod*. Tells us, that implemented class from it is an controlled object;
- object **CutromEventProvider** – event provider;
- interface **EventProvider** – interface of package *UniMod*. Tells us that class that implements it is event provider;
- object of class **HydroVisualizerForm** - realizes user interactions with application;
- interface **HydroVisualizer** – consists of methods, within GUI lets user control automata;
- class **JFrame** – class of *Swing* package that implements window.

Controlled object in this project is object of class **Hydro**. As it is thought in *UniMod* package **Hydro** implements interface **ControlledObject**. Also this class implements interface **HydroHandle**, that has methods to control class **Hydro** (user actions). For visualizing state of controlled object class **HydroVisualizerForm** is used. **HydroVisualizerForm** inherits window class **JFrame** from package

Swing and implements interface `HydroVisualizer`, that has visualization methods.

All events are set up in class `CustomEventProvider`, that, as it is thought in *UniMod* implements interface `EventProvider`. Class `CustomEventProvider` has all actions that happen in this project.

4. Classes descriptions

4.1. Interface `HydroVisualizer`

It consists of methods for visualizing states of finite automata, that simulates auto gear behavior and user interactions.

4.1.1. Methods descriptions

1. Method `public void setRotations(int rotations)` – sets engine rotations for drawing tachometer on the user form.
2. Method `public void setGear(int gear)` – sets number of current gear on visualizer.
3. Method `public void setStatus(int status)` – turns engine on or off.

4.2. Class `HydroVisualizerForm`

Implements `HydroVisualizer` interface and extends `JFrame` class from package *Swing*. Consists of elements of automata manipulation, that simulates car auto gear.

4.3 Class `JFrame`

Standard class from package *SWING* for drawing graphics window.

4.4. Class `CustomEventProvider`

Controlled object – auto gear. Implements interface `EventProvider` from package *UniMod*. Consists of variables, that describes auto gear state at this time:

- engine rotations;
- gear.

4.5. Class `EventProvider`

Interface from package *UniMod*, that implements all event providers.

4.6. Class `Hydro`

Implements interface `HydroHandler` and interface `ControlledObject` from package *UniMod*. Shows automata states. There the following attributes of this class:

- engine condition (turned on/off);
- movement direction (forward/backward);
- current gear;
- engine rotations.

4.7. Interface HydroHandler

Consists of methods for transmitting messages to automata, that are invoked by user interactions.
Lets list used methods:

- `public void stop()` – user tried to turn engine off.
- `public void start()` – user turned engine on.
- `public void gasPedalPressed()` – user pressed accelerate pedal.
- `public void gasPedalReleased()` – user released accelerate pedal.
- `public void switchToForward()` – user selected *FORWARD* direction.
- `public void switchToBackward()` – user selected *BACKWARD* direction.

4.8. Class ControlledObject

Interface of package *UniMod*, that implements all controlled objects.

5. Automata Automata

5.1. Description

The automata emulates work of auto car gear.

5.2. Connections diagram

On the fig.3 you can see connections diagram for event provider, automata and controlled object.

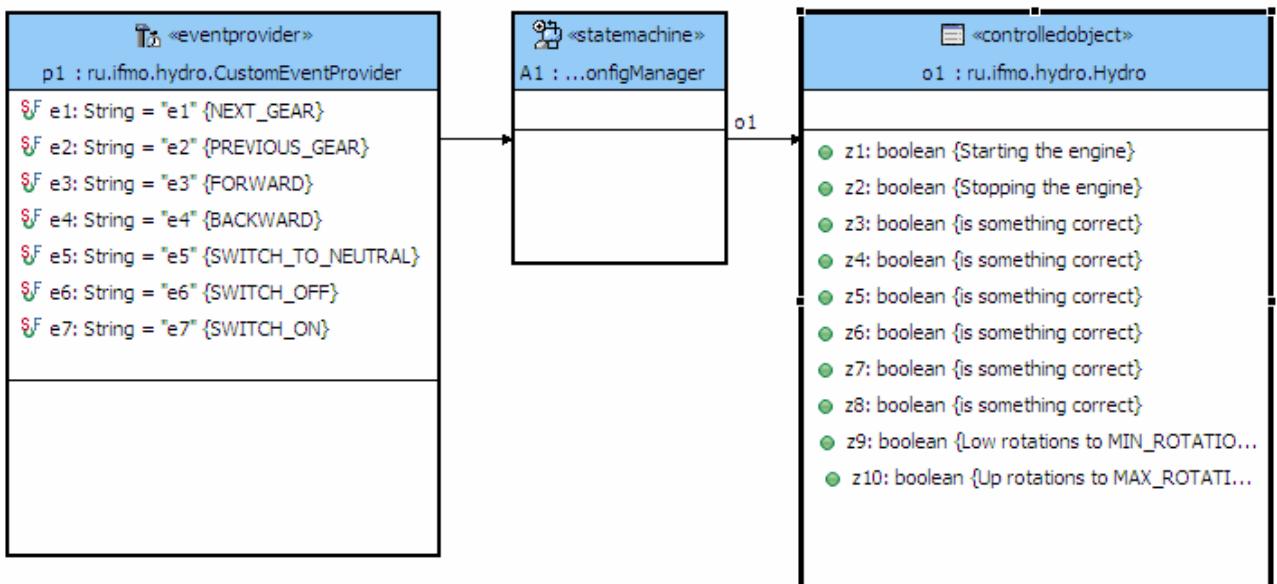


Fig. 3 Automata connections diagram

5.3. Transitions graph

On fig. 4 you can see transitions graph for automata

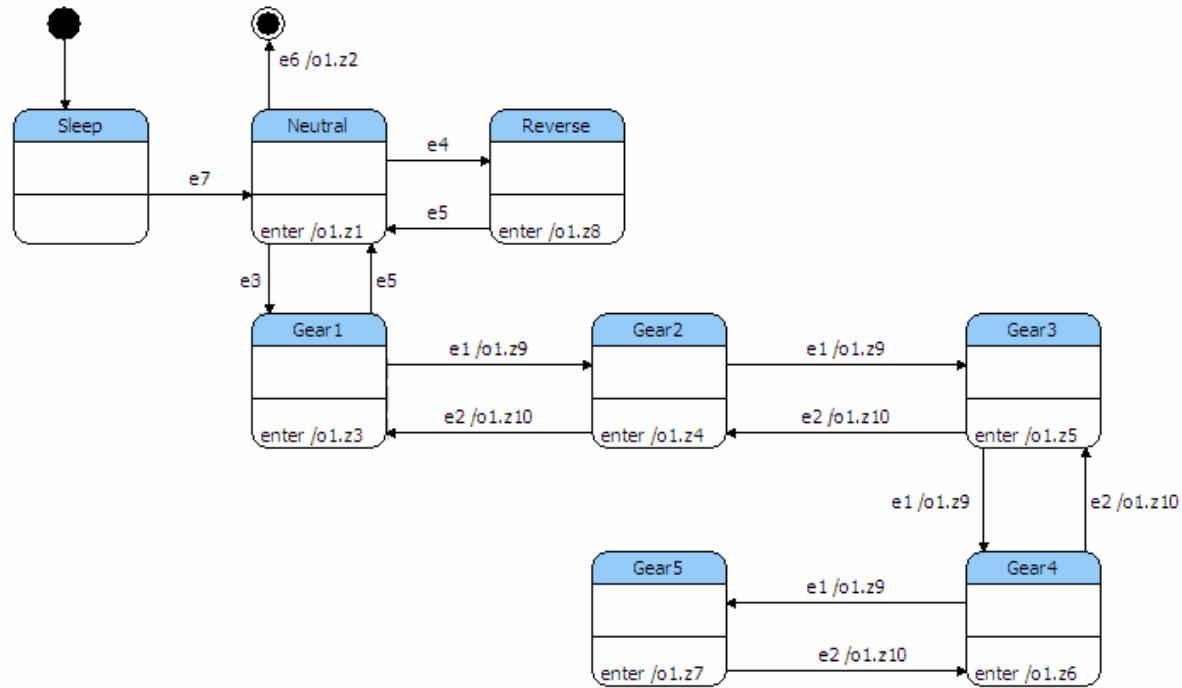


Fig. 4 Transition graph

6. Implementation

6.1 Interpretation

Interpretation way is thought as using xml-description of automata at every step of execution. This way has *UniMod* package in requirements.

In Application 1 you can see xml automata description that was created with the help of the *UniMod* package.

In Application 2 you can see source codes of events providers and controlled objects.

6.2 Compilation

Compilation allows creating Java class from xml description that represents automata. Following that we can avoid using *UniMod* libraries for launching application.

In Application 3 you can see source codes that are generated by *UniMod* package on transitions graph.

In Application 4 you can see sources of application interfaces.

Resume

SWITCH-technology can be used more than effective with the help of *UniMod* package, while developing applications.

At the same moment diagrams are used not only on designing application step, but also when debugging and working on feedback.

Package *UniMod* gives you ability to check for formal properties of transmission graph, the last one allows you to (despite of other products of the same type) get right working applications at first step.

Literature

1. *Shalito A. A.* SWITCH-technology. Algorithmic and developing logic control problems . Saint-Petersburg.: Nauka, 1998. <http://is.ifmo.ru/books/switch/1>
2. *Shalito A. A., Tuckel N.I* SWITCH-technology – automata approach in creation software for "reactive" systems //Programming. 2001. № 5. <http://is.ifmo.ru/works/switch/1/>

Application1. XML automata description

```
01 <?xml version="1.0" encoding="UTF-8"?><!DOCTYPE model PUBLIC "-//eDevelopers Corp.//DTD
State machine model V1.0//EN" "http://www.evelopers.com/dtd/unimod/statemachine.dtd">
02 <model name="Model1">
03   <controlledObject class="ru.ifmo.hydro.Hydro" name="o1"/>
04   <eventProvider class="ru.ifmo.hydro.CustomEventProvider" name="p1">
05     <association clientRole="p1" targetRef="Automata"/>
06   </eventProvider>
07   <rootStateMachine>
08     <stateMachineRef name="Automata"/>
09   </rootStateMachine>
10   <stateMachine name="Automata">
11     <configStore
12       class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
13     <association clientRole="Automata" supplierRole="o1" targetRef="o1"/>
14     <state name="Top" type="NORMAL">
15       <state name="Gear3" type="NORMAL">
16         <outputAction ident="o1.z5"/>
17       </state>
18       <state name="Neutral" type="NORMAL">
19         <outputAction ident="o1.z1"/>
20       </state>
21       <state name="Gear1" type="NORMAL">
22         <outputAction ident="o1.z3"/>
23       </state>
24       <state name="Sleep" type="NORMAL"/>
25       <state name="Reverce" type="NORMAL">
26         <outputAction ident="o1.z8"/>
27       </state>
28       <state name="Gear2" type="NORMAL">
29         <outputAction ident="o1.z4"/>
30       </state>
31       <state name="Gear5" type="NORMAL">
32         <outputAction ident="o1.z7"/>
33       </state>
34       <state name="s1" type="INITIAL"/>
35       <state name="Gear4" type="NORMAL">
36         <outputAction ident="o1.z6"/>
37       </state>
38       <state name="s2" type="FINAL"/>
39     </state>
40     <transition event="e2" sourceRef="Gear3" targetRef="Gear2">
41       <outputAction ident="o1.z10"/>
42     </transition>
43     <transition event="e1" sourceRef="Gear3" targetRef="Gear4">
44       <outputAction ident="o1.z9"/>
45     </transition>
46     <transition event="e3" sourceRef="Neutral" targetRef="Gear1"/>
47     <transition event="e4" sourceRef="Neutral" targetRef="Reverce"/>
48     <transition event="e6" sourceRef="Neutral" targetRef="s2">
```

```

48     <outputAction ident="o1.z2"/>
49 </transition>
50 <transition event="e5" sourceRef="Gear1" targetRef="Neutral"/>
51 <transition event="e1" sourceRef="Gear1" targetRef="Gear2">
52     <outputAction ident="o1.z9"/>
53 </transition>
54 <transition event="e7" sourceRef="Sleep" targetRef="Neutral"/>
55 <transition event="e5" sourceRef="Reverce" targetRef="Neutral"/>
56 <transition event="e2" sourceRef="Gear2" targetRef="Gear1">
57     <outputAction ident="o1.z10"/>
58 </transition>
59 <transition event="e1" sourceRef="Gear2" targetRef="Gear3">
60     <outputAction ident="o1.z9"/>
61 </transition>
62 <transition event="e2" sourceRef="Gear5" targetRef="Gear4">
63     <outputAction ident="o1.z10"/>
64 </transition>
65 <transition sourceRef="s1" targetRef="Sleep"/>
66 <transition event="e1" sourceRef="Gear4" targetRef="Gear5">
67     <outputAction ident="o1.z9"/>
68 </transition>
69 <transition event="e2" sourceRef="Gear4" targetRef="Gear3">
70     <outputAction ident="o1.z10"/>
71 </transition>
72 </stateMachine>
73 </model>
```

Application2. Source codes for events providers and controlled objects

CustomEventProvider.java

Events provider. That are generated with the use if user.

```

01 package ru.ifmo.hydro;
02
03 import com.evelopers.common.exception.CommonException;
04 import com.evelopers.unimod.runtime.EventProvider;
05 import com.evelopers.unimod.runtime.ModelEngine;
06
07 public class CustomEventProvider implements EventProvider {
08
09     /**
10      * @unimod.event.descr NEXT_GEAR
11      */
12     public static final String e1 = "e1";
13
14     /**
15      * @unimod.event.descr PREVIOUS_GEAR
16      */
17     public static final String e2 = "e2";
18
19     /**
20      * @unimod.event.descr FORWARD
21      */
22     public static final String e3 = "e3";
23
24     /**
25      * @unimod.event.descr BACKWARD
26      */
27     public static final String e4 = "e4";
28
29     /**
30      * @unimod.event.descr SWITCH_TO_NEUTRAL
31      */
32     public static final String e5 = "e5";
33
34     /**
35      * @unimod.event.descr SWITCH_OFF
36      */
37     public static final String e6 = "e6";
38
39     /**
40      * @unimod.event.descr SWITCH_ON
41      */
```

```

35     */
36     public static final String e7 = "e7";
37
38     public void init(ModelEngine engine) throws CommonException {
39         Hydro.init(engine);
40     }
41
42     public void dispose() {
43         // TODO Auto-generated method stub
44     }
45 }
46
47 }

```

Hydro.java

Control object. Represents car auto gear by itself.

```

001 package ru.ifmo.hydro;
002
003 import com.evelopers.unimod.core.stateworks.Event;
004 import com.evelopers.unimod.runtime.ControlledObject;
005 import com.evelopers.unimod.runtime.ModelEngine;
006 import com.evelopers.unimod.runtime.context.StateMachineContext;
007 import com.evelopers.unimod.runtime.context.StateMachineContextImpl;
008
009
010
011 public class Hydro implements ControlledObject, HydroHandle {
012
013     public static final int MAX_ROTATIONS = 2000;
014     public static final int MIN_ROTATIONS = 3000;
015
016     private ModelEngine engine;
017
018     private static Hydro o1;
019
020     private HydroVisualizer hydroVisualizer;
021
022     private volatile int rotations = 0;
023     public volatile int gear = 0;
024     private volatile boolean gasPedalPressed;
025     private volatile boolean switchedOn;
026
027     private CustomThread customThread;
028
029     public Hydro() {
030         System.out.println("Default constructor");
031     }
032
033     private Hydro(ModelEngine engine) {
034         System.out.println("Constructor");
035         this.engine = engine;
036         hydroVisualizer = new HydroVisualizerForm("Эмуляция работы АКПП", this);
037         hydroVisualizer.setStatus(0);
038         customThread = new CustomThread();
039     }
040
041     public static void init(ModelEngine engine) {
042         o1 = new Hydro(engine);
043     }
044
045     private void lowGear() {
046         notify(CustomEventProvider.e2);
047     }
048
049     private void upGear() {
050         notify(CustomEventProvider.e1);
051     }

```

```

052
053     private void checkRotations() {
054         if (o1.rotations >= MAX_ROTATIONS) {
055             if (o1.gear > 0 && o1.gear < 5) {
056                 o1.upGear();
057             }
058         } else if (o1.rotations <= MIN_ROTATIONS) {
059             if (o1.gear > 1) {
060                 o1.lowGear();
061             }
062         }
063         if (o1.rotations < MIN_ROTATIONS) {
064             o1.rotations = MIN_ROTATIONS;
065         }
066         if (o1.rotations > MAX_ROTATIONS) {
067             o1.rotations = MAX_ROTATIONS;
068         }
069     }
070
071     public void notify(String event) {
072         o1.engine.getEventManager().handle(new Event(event),
073         StateMachineContextImpl.create());
074     }
075
076     public void start() {
077         o1.notify(CustomEventProvider.e7);
078     }
079
080     public void stop() {
081         o1.notify(CustomEventProvider.e6);
082     }
083
084     public void gasPedalPressed() {
085         o1.gasPedalPressed = true;
086     }
087
088     public void gasPedalReleased() {
089         o1.gasPedalPressed = false;
090     }
091
092     public void switchToForward() {
093         notify(CustomEventProvider.e3);
094     }
095
096     public void switchToBackward() {
097         notify(CustomEventProvider.e4);
098     }
099
100    public void switchToNeutral() {
101        notify(CustomEventProvider.e5);
102    }
103
104    /**
105     * @unimod.action.descr is something correct
106     */
107    public boolean z3(StateMachineContext context) {
108        o1.gear = 1;
109        o1.hydroVisualizer.setGear(o1.gear);
110        return true;
111    }
112
113    /**
114     * @unimod.action.descr is something correct
115     */
116    public boolean z4(StateMachineContext context) {
117        o1.gear = 2;
118        o1.hydroVisualizer.setGear(o1.gear);
119        return true;
120    }
121    /**

```

```

122     * @unimod.action.descr is something correct
123     */
124     public boolean z5(StateMachineContext context) {
125         o1.gear = 3;
126         o1.hydroVisualizer.setGear(o1.gear);
127         return true;
128     }
129
130     /**
131     * @unimod.action.descr is something correct
132     */
133     public boolean z6(StateMachineContext context) {
134         o1.gear = 4;
135         o1.hydroVisualizer.setGear(o1.gear);
136         return true;
137     }
138
139     /**
140     * @unimod.action.descr is something correct
141     */
142     public boolean z7(StateMachineContext context) {
143         o1.gear = 5;
144         o1.hydroVisualizer.setGear(o1.gear);
145         return true;
146     }
147
148     /**
149     * @unimod.action.descr is something correct
150     */
151     public boolean z8(StateMachineContext context) {
152         o1.gear = -1;
153         o1.hydroVisualizer.setGear(o1.gear);
154         return true;
155     }
156
157     /**
158     * @unimod.action.descr Starting the engine
159     */
160     public boolean z1(StateMachineContext context) {
161         o1.switchedOn = true;
162         o1.rotations = MIN_ROTATIONS;
163         o1.gear = 0;
164         o1.hydroVisualizer.setGear(o1.gear);
165         o1.hydroVisualizer.setStatus(1);
166
167         if (!o1.customThread.isAlive()) {
168             o1.customThread.start();
169         }
170         return false;
171     }
172
173     /**
174     * @unimod.action.descr Stopping the engine
175     */
176     public boolean z2(StateMachineContext context) {
177         o1.switchedOn = false;
178         o1.rotations = 0;
179         o1.hydroVisualizer.setRotations(o1.rotations);
180         o1.hydroVisualizer.setStatus(0);
181         System.exit(0);
182         return false;
183     }
184
185     public class CustomThread extends Thread {
186         public void run() {
187             System.out.println("In the customThread.run");
188             while (o1.switchedOn) {
189                 if (o1.gasPedalPressed) {
190                     o1.rotations++;
191                 } else {
192                     o1.rotations--;

```

```

193         }
194         if (ol.rotations < MIN_ROTATIONS) {
195             ol.rotations = MIN_ROTATIONS;
196         }
197         if (ol.gear != 0) {
198             checkRotations();
199         }
200         ol.hydroVisualizer.setRotations(ol.rotations);
201         try {
202             Thread.sleep(10);
203         } catch (InterruptedException e) {
204             e.printStackTrace();
205         }
206     }
207 }
210
211 /**
212 * @unimod.action.descr Low rotations to MIN_ROTATIONS value
213 */
214 public boolean z9(StateMachineContext context) {
215     ol.rotations = MIN_ROTATIONS;
216     return true;
217 }
218
219 /**
220 * @unimod.action.descr Up rotations to MAX_ROTATIONS value
221 */
222 public boolean z10(StateMachineContext context) {
223     ol.rotations = MAX_ROTATIONS;
224     return true;
225 }
226 }
```

Application 3. Source code of Java-class representing the automata

```

001 package ru.ifmo.hydro;
002
003 /**
004 * This file was generated from model [Model1] on [Tue Mar 07 14:21:15 MSK 2006].
005 * Do not change content of this file.
006 */
007
008 //
009
010 import java.util.*;
011
012 import com.evelopers.common.exception.*;
013 import com.evelopers.unimod.core.stateworks.*;
014 import com.evelopers.unimod.runtime.*;
015 import com.evelopers.unimod.runtime.context.*;
016
017
018 public class Model1EventProcessor extends AbstractEventProcessor {
019
020     private ModelStructure modelStructure;
021
022     private static final int A1 = 1;
023
024     private int decodeStateMachine(String sm) {
025
026         if ("A1".equals(sm)) {
027             return A1;
028         }
029     }
```

```

030         return -1;
031     }
032
033     private A1EventProcessor _A1;
034
035     public Model1EventProcessor() {
036         modelStructure = new Model1ModelStructure();
037
038         _A1 = new A1EventProcessor();
039     }
040
041     public ModelStructure getModelStructure() {
042         return modelStructure;
043     }
044
045     public void setControlledObjectsMap(ControlledObjectsMap controlledObjectsMap) {
046         super.setControlledObjectsMap(controlledObjectsMap);
047
048         _A1.init(controlledObjectsMap);
049     }
050
051     protected StateMachineConfig process(
052         Event event, StateMachineContext context,
053         StateMachinePath path, StateMachineConfig config) throws SystemException
054     {
055
056         // get state machine from path
057         int sm = decodeStateMachine(path.getStateMachine());
058
059         try {
060             switch (sm) {
061                 case A1:
062                     return _A1.process(event, context, path, config);
063                 default:
064                     throw new EventProcessorException("Unknown state machine [" +
path.getStateMachine() + "]");
065             }
066         } catch (Exception e) {
067             if (e instanceof SystemException) {
068                 throw (SystemException)e;
069             } else {
070                 throw new SystemException(e);
071             }
072         }
073     }
074
075     protected StateMachineConfig transiteToStableState(
076         StateMachineContext context,
077         StateMachinePath path, StateMachineConfig config) throws SystemException
078     {
079
080         // get state machine from path
081         int sm = decodeStateMachine(path.getStateMachine());
082
083         try {
084             switch (sm) {
085                 case A1:
086                     return _A1.transiteToStableState(context, path, config);
087                 default:
088                     throw new EventProcessorException("Unknown state machine [" +
path.getStateMachine() + "]");
089             }
090         } catch (Exception e) {
091             if (e instanceof SystemException) {
092                 throw (SystemException)e;
093             } else {
094                 throw new SystemException(e);
095             }
096         }

```

```

097     private class Model1ModelStructure implements ModelStructure {
098
099     private Map configManagers = new HashMap();
100
101     private Model1ModelStructure() {
102         configManagers.put("A1", new
103             com.evelopers.unimod.runtime.config.DistinguishConfigManager());
104     }
105
106     public StateMachinePath getRootPath() throws EventProcessorException {
107         return new StateMachinePath("A1");
108     }
109
110     public StateMachineConfigManager getConfigManager(String stateMachine) throws
111         EventProcessorException {
112         return (StateMachineConfigManager) configManagers.get(stateMachine);
113     }
114
115     public StateMachineConfig getTopConfig(String stateMachine) throws
116         EventProcessorException {
117         int sm = decodeStateMachine(stateMachine);
118
119         switch (sm) {
120             case A1:
121                 return new StateMachineConfig("Top");
122             default:
123                 throw new EventProcessorException("Unknown state machine [" +
stateMachine + "]");
124         }
125     }
126
127     public boolean isFinal(String stateMachine, StateMachineConfig config) throws
128         EventProcessorException {
129         // get state machine from path
130         int sm = decodeStateMachine(stateMachine);
131         int state;
132
133         switch (sm) {
134             case A1:
135                 state = _A1.decodeState(config.getActiveState());
136                 switch (state) {
137                     case A1EventProcessor.s2:
138                         return true;
139                     default:
140                         return false;
141                 }
142             default:
143                 throw new EventProcessorException("Unknown state machine [" +
stateMachine + "]");
144         }
145     }
146
147
148
149
150
151     private class A1EventProcessor {
152
153         // states
154         private static final int Top = 1;
155         private static final int s5 = 2;
156         private static final int Sleep = 3;
157         private static final int s1 = 4;
158         private static final int s9 = 5;
159         private static final int s7 = 6;
160         private static final int Gear1 = 7;
161         private static final int s4 = 8;

```

```

162     private static final int s2 = 9;
163     private static final int s10 = 10;
164     private static final int s8 = 11;
165
166     private int decodeState(String state) {
167
168         if ("Top".equals(state)) {
169             return Top;
170         } else
171
172             if ("s5".equals(state)) {
173                 return s5;
174             } else
175
176                 if ("Sleep".equals(state)) {
177                     return Sleep;
178                 } else
179
180                     if ("s1".equals(state)) {
181                         return s1;
182                     } else
183
184                         if ("s9".equals(state)) {
185                             return s9;
186                         } else
187
188                         if ("s7".equals(state)) {
189                             return s7;
190                         } else
191
192                         if ("Gear1".equals(state)) {
193                             return Gear1;
194                         } else
195
196                         if ("s4".equals(state)) {
197                             return s4;
198                         } else
199
200                             if ("s2".equals(state)) {
201                                 return s2;
202                             } else
203
204                             if ("s10".equals(state)) {
205                                 return s10;
206                             } else
207
208                             if ("s8".equals(state)) {
209                                 return s8;
210                             }
211
212             return -1;
213     }
214
215     // events
216     private static final int e4 = 1;
217     private static final int e2 = 2;
218     private static final int e7 = 3;
219     private static final int e5 = 4;
220     private static final int e1 = 5;
221     private static final int e3 = 6;
222     private static final int e6 = 7;
223
224     private int decodeEvent(String event) {
225
226         if ("e4".equals(event)) {
227             return e4;
228         } else
229
230             if ("e2".equals(event)) {
231                 return e2;
232             } else

```

```

233
234     if ("e7".equals(event)) {
235         return e7;
236     } else
237
238     if ("e5".equals(event)) {
239         return e5;
240     } else
241
242     if ("e1".equals(event)) {
243         return e1;
244     } else
245
246     if ("e3".equals(event)) {
247         return e3;
248     } else
249
250     if ("e6".equals(event)) {
251         return e6;
252     }
253
254     return -1;
255 }
256
257 private ru.ifmo.hydro.Hydro ol;
258
259 private void init(ControlledObjectsMap controlledObjectsMap) {
260     ol = (ru.ifmo.hydro.Hydro)controlledObjectsMap.getControlledObject("ol");
261 }
262
263 private StateMachineConfig process(Event event, StateMachineContext context,
264 StateMachinePath path, StateMachineConfig config) throws Exception {
265     config = lookForTransition(event, context, path, config);
266
267     config = transiteToStableState(context, path, config);
268
269     // execute included state machines
270     executeSubmachines(event, context, path, config);
271
272     return config;
273 }
274
275 private void executeSubmachines(Event event, StateMachineContext context,
276 StateMachinePath path, StateMachineConfig config) throws Exception {
277     int state = decodeState(config.getActiveState());
278
279     while (true) {
280         switch (state) {
281             case s5:
282
283                 return;
284             case Sleep:
285
286                 return;
287             case s1:
288
289                 return;
290             case s9:
291
292                 return;
293             case s7:
294
295                 return;
296             case Gear1:
297
298                 return;
299             case s4:
300
301                 return;
302             case s2:

```

```

302             return;
303         case s10:
304
305             return;
306         case s8:
307
308             return;
309         default:
310             throw new EventProcessorException("State with name [" +
config.getActiveState() + "] is unknown for state machine [A1]");
311         }
312     }
313 }
314
315     private StateMachineConfig transiteToStableState(StateMachineContext context,
StateMachinePath path, StateMachineConfig config) throws Exception {
316
317         int s = decodeState(config.getActiveState());
318         Event event;
319
320         switch (s) {
321             case Top:
322
323                 fireComeToState(context, path, "s1");
324
325                 // s1->Sleep [true]/
326                 event = Event.NO_EVENT;
327                 fireTransitionFound(context, path, "s1", event,
328 "s1#Sleep##true");
329
330
331                 fireComeToState(context, path, "Sleep");
332
333                 // Sleep []
334
335                 return new StateMachineConfig("Sleep");
336         }
337
338         return config;
339     }
340
341     private StateMachineConfig lookForTransition(Event event, StateMachineContext
context, StateMachinePath path, StateMachineConfig config) throws Exception {
342
343
344
345
346
347         BitSet calculatedInputActions = new BitSet(0);
348
349         int s = decodeState(config.getActiveState());
350         int e = decodeEvent(event.getName());
351
352         while (true) {
353             switch (s) {
354                 case s5:
355
356
357                     switch (e) {
358                         case e5:
359
360                             // s5->s4 e5[true]/
361
362                             fireTransitionCandidate(context, path, "s5", event,
363 "s5#s4#e5#true");
364
365
366
367

```

```

368                     fireTransitionFound(context, path, "s5", event,
369 "s5#s4#e5#true");
370
371                     fireComeToState(context, path, "s4");
372
373                     // s4 [o1.z1]
374                     fireBeforeOutputActionExecution(context, path,
375 "s5#s4#e5#true", "o1.z1");
376
377                     o1.z1(context);
378
379                     fireAfterOutputActionExecution(context, path,
380 "s5#s4#e5#true", "o1.z1");
381
382                     return new StateMachineConfig("s4");
383
384
385                     // transition not found
386                     return config;
387                 }
388
389             case Sleep:
390
391
392             switch (e) {
393                 case e7:
394
395                     // Sleep->s4 e7[true]/
396
397                     fireTransitionCandidate(context, path, "Sleep",
398 event, "Sleep#s4#e7#true");
399
400
401
402                     fireTransitionFound(context, path, "Sleep", event,
403 "Sleep#s4#e7#true");
404
405
406                     fireComeToState(context, path, "s4");
407
408                     // s4 [o1.z1]
409                     fireBeforeOutputActionExecution(context, path,
410 "Sleep#s4#e7#true", "o1.z1");
411
412                     o1.z1(context);
413
414                     fireAfterOutputActionExecution(context, path,
415 "Sleep#s4#e7#true", "o1.z1");
416
417                     return new StateMachineConfig("s4");
418
419
420                     // transition not found
421                     return config;
422                 }
423
424             case s9:
425
426             switch (e) {
427                 case e2:
428
429                     // s9->s10 e2[true]/o1.z10
430
431

```

```

432                               fireTransitionCandidate(context, path, "s9", event,
433 "s9#s10#e2#true");
434
435
436
437                               fireTransitionFound(context, path, "s9", event,
438 "s9#s10#e2#true");
439
440                               fireBeforeOutputActionExecution(context, path,
441 "s9#s10#e2#true", "o1.z10");
442
443                               o1.z10(context);
444
445                               fireAfterOutputActionExecution(context, path,
446 "s9#s10#e2#true", "o1.z10");
447
448                               fireComeToState(context, path, "s10");
449
450                               // s10 [o1.z6]
451                               fireBeforeOutputActionExecution(context, path,
452 "s9#s10#e2#true", "o1.z6");
453
454                               o1.z6(context);
455
456
457                               fireAfterOutputActionExecution(context, path,
458 "s9#s10#e2#true", "o1.z6");
459
460                               return new StateMachineConfig("s10");
461
462
463
464                               default:
465
466
467                               // transition not found
468                               return config;
469
470
471
472                               case s7:
473
474
475
476
477
478                               switch (e) {
479                               case e2:
480
481                               // s7->Gear1 e2[true]/o1.z10
482
483                               fireTransitionCandidate(context, path, "s7", event,
484 "s7#Gear1#e2#true");
485
486
487                               fireTransitionFound(context, path, "s7", event,
488 "s7#Gear1#e2#true");
489
490                               fireBeforeOutputActionExecution(context, path,
491 "s7#Gear1#e2#true", "o1.z10");
492
493
494                               o1.z10(context);
495
496                               fireAfterOutputActionExecution(context, path,
497 "s7#Gear1#e2#true", "o1.z10");
498
499
500                               fireComeToState(context, path, "Gear1");
501
502                               // Gear1 [o1.z3]
503                               fireBeforeOutputActionExecution(context, path,
504 "s7#Gear1#e2#true", "o1.z3");
505
506                               o1.z3(context);

```



```

"Gear1#s4#e5#true", "o1.z1");
554
555                               o1.z1(context);
556
557                               fireAfterOutputActionExecution(context, path,
558                               return new StateMachineConfig("s4");
559
560
561             case e1:
562
563                 // Gear1->s7 e1[true]/o1.z9
564
565                 fireTransitionCandidate(context, path, "Gear1",
566                 event, "Gear1#s7#e1#true");
567
568
569
570                 fireTransitionFound(context, path, "Gear1", event,
571                 "Gear1#s7#e1#true");
572
573                 fireBeforeOutputActionExecution(context, path,
574
575                 "Gear1#s7#e1#true", "o1.z9");
576
577
578                 fireAfterOutputActionExecution(context, path,
579
580                 fireComeToState(context, path, "s7");
581
582                 // s7 [o1.z4]
583                 fireBeforeOutputActionExecution(context, path,
584
585                 "Gear1#s7#e1#true", "o1.z4");
586
587                 o1.z4(context);
588
589                 fireAfterOutputActionExecution(context, path,
590
591                 "Gear1#s7#e1#true", "o1.z4");
592
593
594                 return new StateMachineConfig("s7");
595             }
596
597             case s4:
598
599                 switch (e) {
600                     case e4:
601
602                         // s4->s5 e4[true]/
603
604                         fireTransitionCandidate(context, path, "s4", event,
605                         "s4#s5#e4#true");
606
607
608
609
610                         fireTransitionFound(context, path, "s4", event,
611                         "s4#s5#e4#true");
612
613                         fireComeToState(context, path, "s5");
614

```

```

615
616 // s5 [o1.z8]
617 fireBeforeOutputActionExecution(context, path,
618 o1.z8(context);
619
620 fireAfterOutputActionExecution(context, path,
621
622 return new StateMachineConfig("s5");
623
624
625 case e3:
626
627 // s4->Gear1 e3[true]/
628
629 fireTransitionCandidate(context, path, "s4", event,
630 "s4#Gear1#e3#true");
631
632
633
634
635 fireTransitionFound(context, path, "s4", event,
636 "s4#Gear1#e3#true");
637
638
639
640
641 fireComeToState(context, path, "Gear1");
642
643 // Gear1 [o1.z3]
644 fireBeforeOutputActionExecution(context, path,
645 "s4#Gear1#e3#true", "o1.z3");
646
647
648
649 case e6:
650
651 // s4->s2 e6[true]/o1.z2
652
653 fireTransitionCandidate(context, path, "s4", event,
654 "s4#s2#e6#true");
655
656
657
658
659 fireTransitionFound(context, path, "s4", event,
660 "s4#s2#e6#true");
661
662 fireBeforeOutputActionExecution(context, path,
663 "s4#s2#e6#true", "o1.z2");
664
665
666
667
668
669
670
671
672
673
674
675

```

```

676                                     // transition not found
677                                     return config;
678                                 }
679
680             case s10:
681
682                 switch (e) {
683                     case e2:
684
685                         // s10->s8 e2[true]/o1.z10
686
687                         fireTransitionCandidate(context, path, "s10", event,
688
689                         "s10#s8#e2#true");
690
691
692
693
694
695                         fireTransitionFound(context, path, "s10", event,
696
697                         "s10#s8#e2#true");
698
699
700
701                         fireBeforeOutputActionExecution(context, path,
702
703
704                         o1.z10(context);
705
706                         fireAfterOutputActionExecution(context, path,
707
708
709                         fireComeToState(context, path, "s8");
710
711                         // s8 [o1.z5]
712                         fireBeforeOutputActionExecution(context, path,
713
714
715                         o1.z5(context);
716
717                         fireAfterOutputActionExecution(context, path,
718
719
720
721
722
723
724                         fireTransitionCandidate(context, path, "s10", event,
725
726
727                         "s10#s9#e1#true");
728
729
730                         fireTransitionFound(context, path, "s10", event,
731
732
733
734
735                         fireBeforeOutputActionExecution(context, path,
736
737
738                         o1.z9(context);
739
740                         fireAfterOutputActionExecution(context, path,
741
742
743                         fireComeToState(context, path, "s9");
744
745                         // s9 [o1.z7]
746                         fireBeforeOutputActionExecution(context, path,
747
748                         "s10#s9#e1#true", "o1.z7");
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2489
2490
2491
2492
2493
2494
2495
2496
2497
2497
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2698
2699
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2789
2790
2791
2792
2793
2794
2795
2796
2797
2797
2798
2799
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2898
2899
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3
```



```

797                     fireAfterOutputActionExecution(context, path,
798
799                     "s8#s10#e1#true", "o1.z9");
800
801                     fireComeToState(context, path, "s10");
802
803                     // s10 [o1.z6]
804                     fireBeforeOutputActionExecution(context, path,
805
806                     "s8#s10#e1#true", "o1.z6");
807
808                     o1.z6(context);
809
810                     fireAfterOutputActionExecution(context, path,
811
812                     "s8#s10#e1#true", "o1.z6");
813
814                     return new StateMachineConfig("s10");
815
816
817                     default:
818
819                     // transition not found
820                     return config;
821
822                 }
823
824
825             }
826
827             private static boolean isInputActionCalculated(BitSet calculatedInputActions, int
k) {
828
829                 boolean b = calculatedInputActions.get(k);
830
831                 if (!b) {
832                     calculatedInputActions.set(k);
833
834                 return b;
835             }
836
837         }

```

Application 4. Source code of the application interfaces

HydroVisualizerForm.java

```

001 package ru.ifmo.hydro;
002
003 import javax.swing.*;
004 import javax.swing.border.Border;
005 import javax.swing.border.TitledBorder;
006 import java.awt.*;
007 import java.awt.event.ActionEvent;
008 import java.awt.event.ActionListener;
009 import java.awt.eventMouseListener;
010 import java.awt.event.MouseEvent;
011
012 public class HydroVisualizerForm extends JFrame implements HydroVisualizer,
ActionListener, MouseListener {
013     public static final long serialVersionUID = 1234567890521;
014     public static final String OFF = "ВЫКЛ";
015     public static final String ON = "БКЛ";

```

```
016     public static final String FORWARD = "ВПЕРЕД";
017     public static final String BACKWARD = "НАЗАД";
018     public static final String NEUTRAL = "НЕЙТРАЛЬ";
019     public static final String GAS = "ГАЗ";
020     public static final String HYDRO_STATE = "НАПРАВЛЕНИЕ";
021
022     private JButton btnRotations;
023     private JButton btnOn;
024     private JButton btnOff;
025     private JRadioButton rbnForward;
026     private JRadioButton rbnBackward;
027     private JRadioButton rbnNeutral;
028     private TachometerPanel tachometerPanel;
029
030     private Hydro hydro;
031
032     public HydroVisualizerForm(String title, Hydro hydro) {
033         super(title);
034
035         this.hydro = hydro;
036
037         setSize(400, 300);
038
039         getContentPane().setLayout(new GridBagLayout());
040         GridBagConstraints c = new GridBagConstraints();
041         c.insets = new Insets(2, 5, 2, 5);
042
043         c.gridx = 0;
044         c.gridy = 0;
045         btnOn = new JButton(ON);
046         btnOn.addActionListener(this);
047         getContentPane().add(btnOn, c);
048
049         c.gridx = 1;
050         btnOff = new JButton(OFF);
051         btnOff.addActionListener(this);
052         getContentPane().add(btnOff, c);
053
054         JPanel switchPanel = new JPanel(new GridBagLayout());
055         Border etched = BorderFactory.createEtchedBorder();
056         TitledBorder titledBorder = BorderFactory.createTitledBorder(etched,
057             HYDRO_STATE + ":" );
058         switchPanel.setBorder(titledBorder);
059
060         c.gridx = 0;
061         c.gridy = 1;
062         c.gridwidth = 2;
063         getContentPane().add(switchPanel, c);
064
065         c.gridx = 2;
066         c.weightx = 1.0;
067         tachometerPanel = new TachometerPanel();
068         tachometerPanel.setBorder(BorderFactory.createEtchedBorder());
069         getContentPane().add(tachometerPanel, c);
070
071
072         rbnNeutral = new JRadioButton(NEUTRAL);
073         rbnNeutral.addActionListener(this);
074         rbnForward = new JRadioButton(FORWARD);
075         rbnForward.addActionListener(this);
076         rbnBackward = new JRadioButton(BACKWARD);
077         rbnBackward.addActionListener(this);
078         ButtonGroup buttonGroup = new ButtonGroup();
079         buttonGroup.add(rbnNeutral);
080         buttonGroup.add(rbnForward);
081         buttonGroup.add(rbnBackward);
082
083         c.gridx = 0;
084         c.weightx = 0.0;
085         c.gridy = 0;
```

```

086     c.anchor = GridBagConstraints.WEST;
087     switchPanel.add(rbnForward, c);
088
089     c.gridx = 1;
090     switchPanel.add(rbnBackward, c);
091
092     c.gridx = 2;
093     switchPanel.add(rbnNeutral, c);
094
095     c.gridx = 2;
096     c.gridwidth = 4;
097     c.anchor = GridBagConstraints.CENTER;
098     c.fill = GridBagConstraints.NONE;
099     c.weightx = 1.0;
100    btnRotations = new JButton(GAS);
101    btnRotations.addMouseListener(this);
102    getContentPane().add(btnRotations, c);
103
104    setVisible(true);
105
106    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
107 }
108
109 public void setRotations(int rotations) {
110     tachometerPanel.setRotations(rotations);
111     repaint();
112 }
113
114 public void setGear(int gear) {
115     tachometerPanel.setGear(gear);
116 }
117
118 public void setStatus(int status) {
119 }
120
121 public void actionPerformed(ActionEvent ae) {
122     if (ae.getSource() == btnOn) {
123         hydro.start();
124     }
125     if (ae.getSource() == btnOff) {
126         hydro.stop();
127     }
128
129     if (ae.getSource() == rbnForward) {
130         hydro.switchToForward();
131     }
132     if (ae.getSource() == rbnBackward) {
133         hydro.switchToBackward();
134     }
135     if (ae.getSource() == rbnNeutral) {
136         hydro.switchToNeutral();
137     }
138 }
139
140 public void mouseClicked(MouseEvent e) {
141     //To change body of implemented methods use File | Settings | File Templates.
142 }
143
144 public void mousePressed(MouseEvent e) {
145     hydro.gasPedalPressed();
146 }
147
148 public void mouseReleased(MouseEvent e) {
149     hydro.gasPedalReleased();
150 }
151
152 public void mouseEntered(MouseEvent e) {
153     //To change body of implemented methods use File | Settings | File Templates.
154 }
155
156 public void mouseExited(MouseEvent e) {

```

```
157         //To change body of implemented methods use File | Settings | File Templates.
158     }
159 }
```

TachometrPanel.java

```
01 package ru.ifmo.hydro;
02
03 import javax.swing.*;
04 import java.awt.*;
05
06 /**
07 * Created by Andrey Vokin.
08 * User: admin
09 * Date: 28.12.2005
10 * Time: 1:00:40
11 */
12 public class TachometerPanel extends JPanel {
13
14     public static final long serialVersionUID = 1234567890561;
15
16     public static final int RADIUS_X = 50;
17     public static final int RADIUS_Y = 50;
18     public static final int CENTER_X = 125;
19     public static final int CENTER_Y = 90;
20
21     public static final String ROTATIONS = "Обороты";
22     public static final String GEAR = "передача";
23
24     /**
25      * Maximal value of the engine rotations.
26      */
27     public static final int MAX_ROTATIONS = 3000;
28
29     private JLabel lblRotations;
30
31     private int rotations;
32
33     private int gear;
34
35     public TachometerPanel() {
36         super();
37
38         lblRotations = new JLabel();
39         add(lblRotations);
40     }
41
42     public void drawLine(Graphics g, double angle) {
43         int y = CENTER_Y - (int) (Math.sin(Math.PI * angle / 180) * RADIUS_Y);
44         int x = CENTER_X - (int) (Math.cos(Math.PI * angle / 180) * RADIUS_X);
45         g.drawLine(CENTER_X, CENTER_Y, x, y);
46     }
47
48     public void paint(Graphics g) {
49         super.paint(g);
50         double angle = 1.0 * rotations / MAX_ROTATIONS * 180;
51         g.drawArc(CENTER_X - RADIUS_X, CENTER_Y - RADIUS_Y, 2 * RADIUS_X, 2 *
RADIUS_Y, 0, 180);
52         drawLine(g, angle);
53         g.drawOval(200, 150, 2, 2);
54     }
55
56     private void write() {
57         lblRotations.setText(ROTATIONS + ": " + rotations + ", " + GEAR + "; " +
gear);
58     }
59
60     public void setRotations(int rotations) {
61         this.rotations = rotations;
62         write();
```

```
63     }
64
65     public void setGear(int gear) {
66         this.gear = gear;
67         write();
68     }
69 }
```