Fuzzy Laboratory 2

Components with (FLETPN) Models

1. Laboratory Objectives

- Acquiring the concepts
  - implementing the comparator in Java using the All_Petri_Nets_Framework,
  - components with FLETPN models,
  - Object Enhanced Time Petri Nets Capsules (OETPNc)
  - the controller model of a first-order system with PI and PID implemented with FLETPN.
  - The plant model with FLETPN
- Developments and tests

2. The comparator

Considering the mathematical model for the comparator:

$X_5 = 1$ and $x_6 = -1 \leftrightarrow (x_0 \pm \delta) > x_1$,

The corresponding tokens are: $x_5 = [0,0,0,0,1]$ and $x_6 = [1,0,0,0,0]$

$X_5 = -1$ and $x_6 = 1 \leftrightarrow (x_0 \pm \delta) < x_1$;

$X_5 = 1$ and $x_6 = 1 \leftrightarrow |x_0 \pm \delta| \leq x_1$;

with $\delta \in [-0.1, 0.1]$.

It is required to build a program that accomplishes:

$(x_0 - x_1) > \varepsilon \Rightarrow x_5 = 1, x_6 = \Phi$

$(x_1 - x_0) > \varepsilon \Rightarrow x_6 = -1, x_5 = \Phi$

$| x_0 - x_1| < \varepsilon \Rightarrow x_5 \neq 1 \ \ x_6 \neq -1$

Figure 2.1 shows the comparator Petri Net where x_0, x_1, x_5, and x_6 are the tokens in the places P0, P1, P5, and P6 respectively. The comparator can be integrated into a component, where the places P0 and P1 are input ports. The comparator implements the logic: If P0> P1 the output transition TOut is executed with the token <0,0,0,0, 1> at P5, and if P1> P0 executes TOut with the token <1,0,0,0,0> at P6.

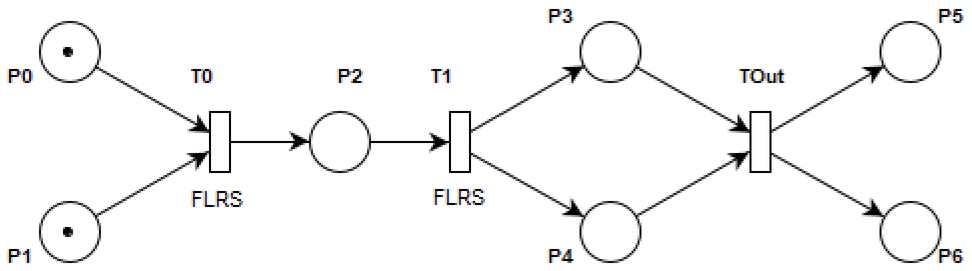

Figure 2.1 The comparator Petri Net

For implementation, the All_Petri_Nets_Framework is also used. The code is implemented with eclipse and can be found here: https://github.com/dahliajanabi/All_Petri_Nets_Framework in the *DCS_FuzzyLab.Comparator*, the *Comparator* class and the *ComparatorInput* class.

Next, use the fuzzy set:

FS={NL, NM, ZR, PM, PL}, with extension *EFS = FS ∪Φ*

Where Φ signifies the non-existence of a token, or equivalent "there is no information about that variable at this time", and in the implementation is coded with FF.

Code Explanation:

In code sequence 1, the *ComparatorInput* Class showed in code sequence 1, a text file is created with the protocol presented as a string: (`"P0:"+f1+"F"+","+"P1:"+f2+"F\n");` This means that the 1st input place is P0 with the float value f1 followed by the letter F, then separated by a comma with no spaces, then the 2nd place value which is P1 also followed by the float value f2 followed by F, then a feed line to mark that this line is for the first input; as each line of the created file will be sent to the input places when they will be null after that they are consumed by transition t1.

These kind of input files can be created manually as well but the protocol for who the input places is written should be respected.

In the *Comparator* class showed in code sequence 2, the input text file is added as follows:

```
pn.SetInputFile( "D:\\PetriInputData\\comparator.txt" );
```

Two constant values are needed for Tout:

```
DataFuzzy c1 = new DataFuzzy();
c1.SetName("NLToken");
c1.SetValue(new Fuzzy(-1.0F));
pn.ConstantPlaceList.add(c1);

DataFuzzy c2 = new DataFuzzy();
c2.SetName("PLToken");
c2.SetValue(new Fuzzy(1.0F));
pn.ConstantPlaceList.add(c2);
```

Those kind of places are added to the ConstantPlaceList so they are not a part of the Petri net structure, and they will never be null after a transition is executed that has them as input places.

TOut will use the *Move* operation to move the token from the constant place to a specific one, and *MakeNull* is used to make either P5 or P6 null so the token from the previous tic would be consumed, this is because those places are not inputs for another transition so the tokens remain there, and they can only be placed with new tokens.

Also the delay for the PN should be changed to 0 so that it can execute faster:

```
pn.Delay = 0;
```

This means that the PN will get its input from the text file and no need to input the values using the GUI/ InputFuzzy. The FLETPN thread will be terminated when the final line of the input text file is reached and a graph will be displayed showing all the places values in all the tics as shown in figure 2.2.

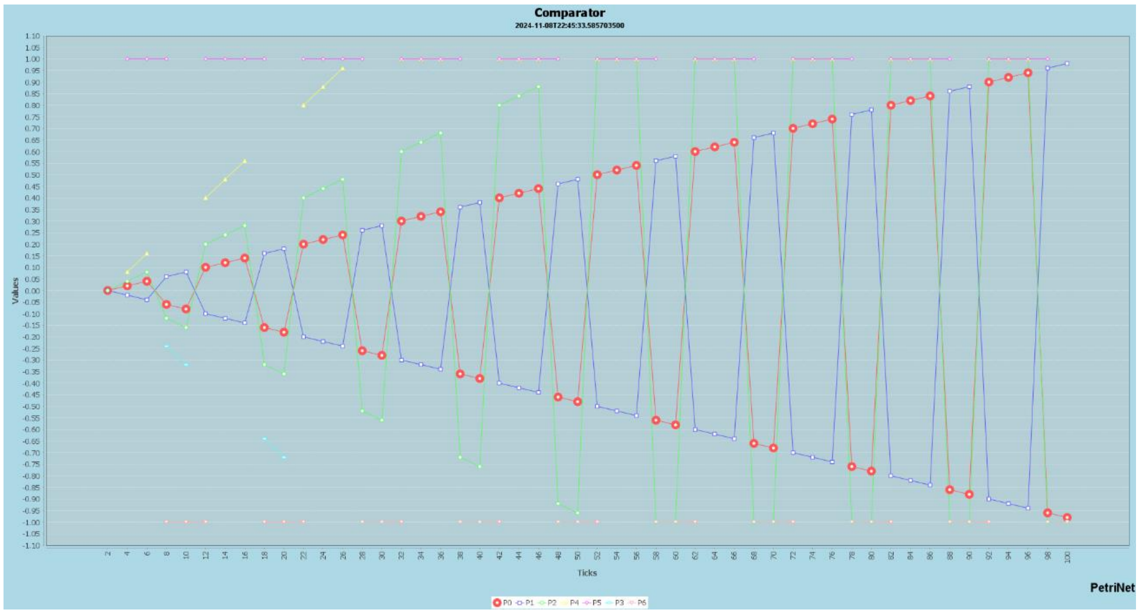

Figure 2.2 Comparator class output graph

Code sequence 1: ComparatorInput class

```
package DCS_FuzzyLab.Comparator;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.Files;

public class ComparatorInput {
	public static void main(String[] args) throws InterruptedException,
IOException {
```

```
            File file = new File("D:\\PetriInputData\\comparator.txt");
            Files.deleteIfExists(file.toPath());
            FileWriter fw = new FileWriter(file.getPath());
            Float f1, f2;
            for (float i = 0; i < 100; i++) {
                  if (i % 10 < 5) {
                        f1 = i/100;
                        f2 = i/-100;
                  } else {
                        f1 = i/-100;
                        f2 = i/100;
                  }
                  fw.write("P0:"+f1+"F"+","+"P1:"+f2+"F\n");
            }
            fw.close();
            System.out.println("Done!");
      }
}
```

Code sequence 2: Comparator class

```
package DCS_FuzzyLab.Comparator;

import java.io.FileNotFoundException;
import java.util.ArrayList;
import Components.Activation;
import Components.Condition;
import Components.GuardMapping;
import Components.PetriNet;
import Components.PetriNetWindow;
import Components.PetriTransition;
import DataObjects.DataFuzzy;
import DataOnly.FLRS;
import DataOnly.FV;
import DataOnly.Fuzzy;
import DataOnly.FuzzyVector;
import DataOnly.PlaceNameWithWeight;
import Enumerations.FZ;
import Enumerations.LogicConnector;
import Enumerations.TransitionCondition;
import Enumerations.TransitionOperation;

public class Comparator {
       public static void main (String[]args) throws FileNotFoundException {
       PetriNet pn = new PetriNet();
       pn.PetriNetName = "Comparator";
       pn.NetworkPort = 1081;

       FLRS differentiator = new FLRS(new FV(FZ.ZR), new FV(FZ.NM), new FV(FZ.NL), new
FV(FZ.NL),new FV(FZ.NL),
                                                             new FV(FZ.PM), new FV(FZ.ZR),
new FV(FZ.NM), new FV(FZ.NL), new FV(FZ.NL),
                                                             new FV(FZ.PL), new FV(FZ.PM),
new FV(FZ.ZR), new FV(FZ.NM), new FV(FZ.NL),
                                                                  new FV(FZ.PL), new FV(FZ.PL),
new FV(FZ.PM), new FV(FZ.ZR), new FV(FZ.NM),
                                                                  new FV(FZ.PL), new FV(FZ.PL),
new FV(FZ.PL), new FV(FZ.PM),new FV(FZ.ZR));

       FLRS separator = new FLRS(new FV(FZ.NL,FZ.FF), new FV(FZ.NL,FZ.FF), new
FV(FZ.FF,FZ.FF), new FV(FZ.FF,FZ.PL),new FV(FZ.FF,FZ.PL));

       differentiator.Print();
       separator.Print();

       pn.SetInputFile("D:\\PetriInputData\\comparator.txt");

       DataFuzzy p0 = new DataFuzzy();
       p0.SetName("P0");
//     p0.SetValue(new Fuzzy(0.0F));
       pn.PlaceList.add(p0);

       DataFuzzy p1 = new DataFuzzy();
       p1.SetName("P1");
//     p1.SetValue(new Fuzzy(0.0F));
       pn.PlaceList.add(p1);

       DataFuzzy p2 = new DataFuzzy();
       p2.SetName("P2");
       pn.PlaceList.add(p2);

       DataFuzzy p3 = new DataFuzzy();
       p3.SetName("P3");
       pn.PlaceList.add(p3);

       DataFuzzy p4 = new DataFuzzy();
       p4.SetName("P4");
       pn.PlaceList.add(p4);

       DataFuzzy c1 = new DataFuzzy();
       c1.SetName("NLToken");
       c1.SetValue(new Fuzzy(-1.0F));
       pn.ConstantPlaceList.add(c1);

       DataFuzzy c2 = new DataFuzzy();
       c2.SetName("PLToken");
       c2.SetValue(new Fuzzy(1.0F));
       pn.ConstantPlaceList.add(c2);

       DataFuzzy p5 = new DataFuzzy();
       p5.SetName("P5");
       pn.PlaceList.add(p5);

       DataFuzzy p6 = new DataFuzzy();
       p6.SetName("P6");
       pn.PlaceList.add(p6);

       // T0 ------------------------------------------------
                      PetriTransition t0 = new PetriTransition(pn);
                      t0.TransitionName = "T0";
                      t0.InputPlaceName.add("P0");
                      t0.InputPlaceName.add("P1");

                      Condition T0Ct1 = new Condition(t0, "P0",
TransitionCondition.NotNull);
```

```
                        Condition T0Ct2 = new Condition(t0, "P1",
TransitionCondition.NotNull);
                        T0Ct1.SetNextCondition(LogicConnector.AND, T0Ct2);

                        GuardMapping grdT0 = new GuardMapping();
                        grdT0.condition = T0Ct1;

                        ArrayList<PlaceNameWithWeight> input0 = new ArrayList<>();
                        input0.add(new PlaceNameWithWeight("P0", 1F));
                        input0.add(new PlaceNameWithWeight("P1", 1F));

                        ArrayList<String> Output0 = new ArrayList<>();
                        Output0.add("P2");


                        grdT0.Activations.add(new Activation(t0, differentiator, input0,
TransitionOperation.FLRS, Output0));

                        t0.GuardMappingList.add(grdT0);

                        t0.Delay = 0;
                        pn.Transitions.add(t0);


                        // T1 ------------------------------------------------
                        PetriTransition t1 = new PetriTransition(pn);
                        t1.TransitionName = "T1";
                        t1.InputPlaceName.add("P2");

                        Condition T1Ct1 = new Condition(t1, "P2",
TransitionCondition.NotNull);

                        GuardMapping grdT1 = new GuardMapping();
                        grdT1.condition = T1Ct1;

                        ArrayList<PlaceNameWithWeight> input1 = new ArrayList<>();
                        input1.add(new PlaceNameWithWeight("P2", 1F));


                        ArrayList<String> Output1 = new ArrayList<>();
                        Output1.add("P3");
                        Output1.add("P4");


                        grdT1.Activations.add(new Activation(t1, separator, input1,
TransitionOperation.FLRS, Output1));

                        t1.GuardMappingList.add(grdT1);

                        t1.Delay = 0;
                        pn.Transitions.add(t1);

                        // tOut ------------------------------------------------
                        PetriTransition tOut = new PetriTransition(pn);
                        tOut.TransitionName = "TOut";
                        tOut.InputPlaceName.add("P3");
                        tOut.InputPlaceName.add("P4");

                        Condition TOutCt1 = new Condition(tOut, "P3",
TransitionCondition.NotNull);
```

```
                        GuardMapping grdtOut1 = new GuardMapping();
                        grdtOut1.condition = TOutCt1;

                        grdtOut1.Activations.add(new Activation(tOut, "NLToken",
TransitionOperation.Move, "P6"));
                        grdtOut1.Activations.add(new Activation(tOut, "",
TransitionOperation.MakeNull, "P5")); //to consume the token from the previous tic
                        tOut.GuardMappingList.add(grdtOut1);


                        Condition TOutCt2 = new Condition(tOut, "P4",
TransitionCondition.NotNull);

                        GuardMapping grdtOut2 = new GuardMapping();
                        grdtOut2.condition = TOutCt2;

                        grdtOut2.Activations.add(new Activation(tOut, "PLToken",
TransitionOperation.Move, "P5"));
                        grdtOut2.Activations.add(new Activation(tOut, "",
TransitionOperation.MakeNull, "P6")); //to consume the token from the previous tic
                        tOut.GuardMappingList.add(grdtOut2);


                        tOut.Delay = 0;
                        pn.Transitions.add(tOut);

                        // -------------------------------------------

                        // PetriTransition t3 = new PetriTransition(pn);
                        // pn.Transitions.add(t3);

                        System.out.println("Comparator started \n ----------------------------
--");

                        pn.Delay = 10;
                        pn.PrintingSpeed=10;

                        pn.ShowLogInWindow=true;
                        // pn.Start();

                        PetriNetWindow frame = new PetriNetWindow(false);
                        frame.petriNet = pn;
                        frame.setVisible(true);
        }
}
```

Exercises:

1. Enter a sine signal at P0 and at P1 a cosine signal.
2. Change the code so that P5 has a token if P0<P1 and P6 has a token when P1>P0.

3. Object Enhanced Time Petri Net Capsules (OETPN-C)

The OETPN-c are components used to integrate OETPN models. in this lab those models have different types: FLETPN (float in the domain [-1,1]), input, and output channels. The input channels can be both synchronous and asynchronous. The integrated OETPN models can have two kinds of transitions: asynchronous (T_a) and synchronous (T_s). $T = T_a \cup T_s$. T_a can be executed between ticks at any moment of time, unlike the T_s which can only be executed at ticks. Figure 3.1 shows the OETPN-c.

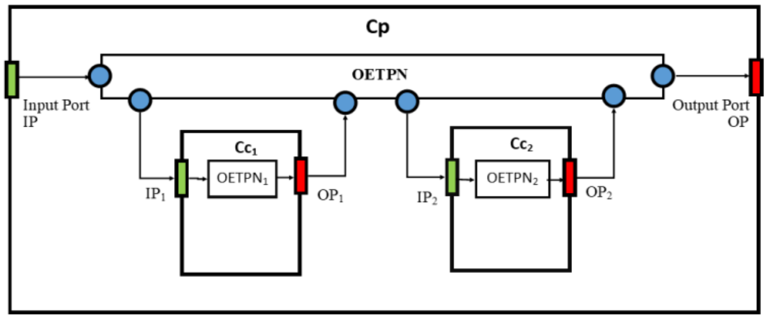

Figure 3.1 OETPN-c

Each OETPN-C is executed by a separate thread. Cc1 (the controller) and Cc2 (the plant) are created by the OETPN (the parent).

3.1 The OETPN model

This model is linked to Input and output ports and another OETPN-c. Figure 3.2 shows the OETPN model. The operator injects the reference value in the r input channel, the Cc_2_y receives the current state y from the plant, Cc_1_r and Cc_1_y are output channels that are connected to the controller to send it r and y values respectively. Cp_o_y can be an output channel if the OETPN-c is connected to another capsule. Cc_1_c receives the command from the controller and Cc_2_u sends the command to the plant.

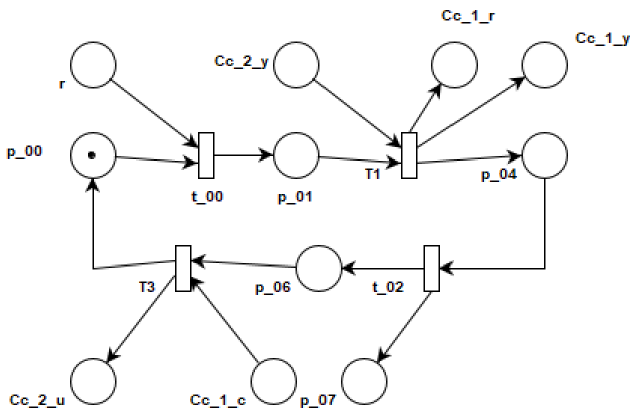

Figure 3.2 The OETPN model

3.3 The Plant model

Consider the mathematical model for the plant:

x(k+1) = a * x(k) + b * u(k)

y(k) = x(k+1)

where:

a= 0.5, b= 0.7

Figure 3.5 shows the FLETPN model of the plant.

The input channel u receives the controller's command from the OETPN model. Transition t_21 computes the current status of the system y(k) according to the above formula and place the result in y and x, transition t_22 sends y through the output channel Cc_2_y to the OETPN model.

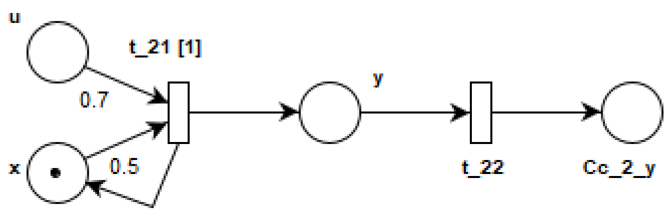

Figure 3.5 The FLETPN model of the Plant

3.2 The Controller with a Proportional–Integral (PI) Action

The controller of the type PI has the mathematical model:

u(k) = u(k-1) + Δu(k)

Δu(k) = K· e(k) + K_I·e(k-1)

e(k) = r(k) – y(k),

where *r* is a reference and *e* is an error, e(k-1) is equal with e(k) after a time unit.

The diagram of components that simulate the control application is shown in Figure 3.4.

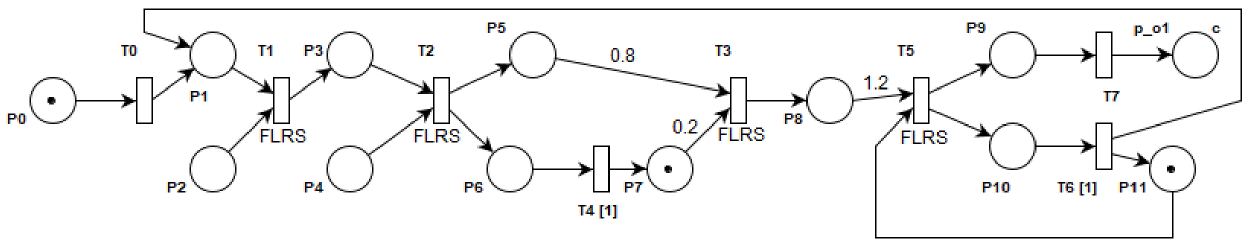

Figure 3.4 FLETPN model for the PI Controller

The Interpretation of the Petri Net is as follows:

P2: receives the system status y(k) from the OETPN model;

T1: Takes the system status from the P2 input port and stores it in the place P3;

P4: receives reference value r(k) from the OETPN model;

T2: Calculates the error e(k) = r(k) - y(k) and stores it in P5 and P6;

T4: Stores the previous error e_{ant}(k)= e(k-1) after a delay of 1 t.u. (time unit) at place P7;

T3: calculate the relation w_{7_3}*e(k-1) + w_{5_3}·e(k) where w_{7_3} = 0.20 and w_{5_3} = 0.80. Deviation of the command Δu(k) is stored in place P8;

P11: contains the previous value of the command u(k-1), initially set to zero;

T5: Calculates the current value of the command u(k) = u(k-1) + Δu(k) by summing the two values stored in P9 and P10;

T6: Updates with delay of 1 t.u. the command value u(k) = u(k-1) and signals the restart of a new calculation of the order;

T7: Sends the token of P9 through the output channel p_o1 c to the OETPN model.

The coefficients associated with the corresponding arcs from P5 to T3, from P7 to T3 and from P8 to T5, specify the constants of the PI controller. Changing their value may lead to better performance.

3.4 The Implementation

The following application implements the controller of a first order system. The input data for r(k) are create in the OETPNInput class and is added to the OETPN class that is listed in code sequence 3. The implementation of the OETPN is listed in code sequence 4, the PIController in sequence 5, and the plant is in sequence 6. For testing, run the OETPNInput class, don’t forget to create the PetriInputData folder on the D drive. After that, run the PI and the Plant models first, then the OETPN, you can click on show graph on the OETPN before starting it to see the graph in real-time. After the execution ends, click on show graph button on the OETPN to see filtered output by selecting which places to show on the graph. Figure3.4 shows the current status of the plant y(k) and the reference value r(k).

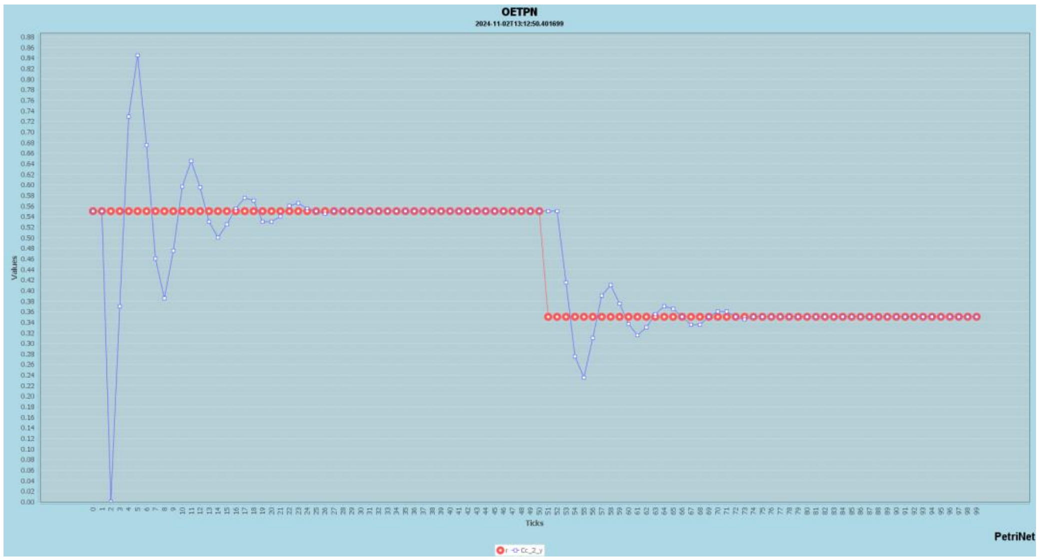

Figure 3.5 The output of the OETPN model.

Code sequence 3: OETPNInput class

```
package DCS_FuzzyLab.OETPN_C;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.Files;

//For OETPN-c
public class OETPNInput {
	public static void main(String[] args) throws InterruptedException, IOException {
		File file = new File("D:\\PetriInputData\\OETPNInput.txt");
		Files.deleteIfExists(file.toPath());
		FileWriter fw = new FileWriter(file.getPath());
		Float command = 0.55F;

		for (float i = 0; i < 100; i++) {
			if (i > 50)
				command = 0.35f;

			fw.write("r:" + command + "F\n");
		}
		fw.close();
		System.out.println("Done!");
	}
}
```

Code sequence 4: OETPN class

```
package DCS_FuzzyLab.OETPN_C;

import java.io.FileNotFoundException;
import Components.Activation;
import Components.Condition;
import Components.GuardMapping;
import Components.PetriNet;
import Components.PetriNetWindow;
import Components.PetriTransition;
import DataObjects.DataFuzzy;
import DataObjects.DataTransfer;
import DataOnly.Fuzzy;
import DataOnly.TransferOperation;
import Enumerations.LogicConnector;
import Enumerations.TransitionCondition;
import Enumerations.TransitionOperation;

public class OETPN {
	public static void main (String[]args) throws FileNotFoundException {
	PetriNet pn = new PetriNet();
	pn.PetriNetName = "OETPN";
	pn.NetworkPort = 1080;

	pn.SetInputFile("D:\\PetriInputData\\OETPNInput.txt");

	DataFuzzy p_00 = new DataFuzzy();
	p_00.SetName("p_00");
	p_00.SetValue(new Fuzzy(0.0F));
	pn.PlaceList.add(p_00);

	DataFuzzy p_02 = new DataFuzzy(); //from operator
	p_02.SetName("r");
	pn.PlaceList.add(p_02);

	DataFuzzy p_01 = new DataFuzzy();
	p_01.SetName("p_01");
	pn.PlaceList.add(p_01);

	DataFuzzy p_03 = new DataFuzzy(); //from plant
	p_03.SetName("Cc_2_y");
	p_03.SetValue(new Fuzzy(0.55F));
	pn.PlaceList.add(p_03);

	DataFuzzy p_04 = new DataFuzzy();
	p_04.SetName("p_04");
	pn.PlaceList.add(p_04);

	DataTransfer p_05_r = new DataTransfer();
	p_05_r.SetName("Cc_1_r");
	p_05_r.Value = new TransferOperation("localhost", "1081", "P4"); //to controller
	pn.PlaceList.add(p_05_r);

	DataTransfer p_05_y = new DataTransfer();
	p_05_y.SetName("Cc_1_y");
	p_05_y.Value = new TransferOperation("localhost", "1081", "P2");  //to controller
	pn.PlaceList.add(p_05_y);

	DataFuzzy p_06 = new DataFuzzy();
	p_06.SetName("p_06");
	pn.PlaceList.add(p_06);

	DataFuzzy p_07 = new DataFuzzy();
	p_07.SetName("Cp_o_y");
	pn.PlaceList.add(p_07);

	DataFuzzy p_08 = new DataFuzzy(); //from controller
```

```
        p_08.SetName("Cc_1_c");
        pn.PlaceList.add(p_08);

        DataTransfer p_09 = new DataTransfer();
        p_09.SetName("Cc_2_u");
        p_09.Value = new TransferOperation("localhost", "1082", "u"); //to plant
        pn.PlaceList.add(p_09);


        // T_00 ------------------------------------------------
                              PetriTransition t_00 = new PetriTransition(pn);
                              t_00.TransitionName = "t_00";
                              t_00.InputPlaceName.add("r");
                              t_00.InputPlaceName.add("p_00");


                              Condition T_10Ct1 = new Condition(t_00, "r",
TransitionCondition.NotNull);
                              Condition T_10Ct2 = new Condition(t_00, "p_00",
TransitionCondition.NotNull);
                              T_10Ct1.SetNextCondition(LogicConnector.AND, T_10Ct2);

                              GuardMapping grdt_00 = new GuardMapping();
                              grdt_00.condition = T_10Ct1;

                              grdt_00.Activations.add(new Activation(t_00, "r",
TransitionOperation.Move, "p_01"));

                              t_00.GuardMappingList.add(grdt_00);

                              t_00.Delay = 1;
                              pn.Transitions.add(t_00);



        // T_01 ------------------------------------------------
                       PetriTransition t_01 = new PetriTransition(pn);
                       t_01.TransitionName = "t_01";
                       t_01.InputPlaceName.add("Cc_2_y");
                       t_01.InputPlaceName.add("p_01");
                       //t_01.IsAsync= true;

                       Condition T_01Ct1 = new Condition(t_01, "Cc_2_y",
TransitionCondition.NotNull);
                       Condition T_01Ct2 = new Condition(t_01, "p_01",
TransitionCondition.NotNull);
                       T_01Ct1.SetNextCondition(LogicConnector.AND, T_01Ct2);

                       GuardMapping grdt_01 = new GuardMapping();
                       grdt_01.condition = T_01Ct1;

                       grdt_01.Activations.add(new Activation(t_01, "Cc_2_y",
TransitionOperation.SendOverNetwork, "Cc_1_y"));
                       grdt_01.Activations.add(new Activation(t_01, "p_01",
TransitionOperation.SendOverNetwork, "Cc_1_r"));
                       grdt_01.Activations.add(new Activation(t_01, "Cc_2_y",
TransitionOperation.Move, "p_04"));

                       t_01.GuardMappingList.add(grdt_01);

                       t_01.Delay = 0;
```

```
                      pn.Transitions.add(t_01);


                      // T_02 -----------------------------------------------
                      PetriTransition t_02 = new PetriTransition(pn);
                      t_02.TransitionName = "t_02";
                      t_02.InputPlaceName.add("p_04");

                      Condition T_02Ct1 = new Condition(t_02, "p_04",
TransitionCondition.NotNull);


                      GuardMapping grdt_02 = new GuardMapping();
                      grdt_02.condition = T_02Ct1;

                      grdt_02.Activations.add(new Activation(t_02, "p_04",
TransitionOperation.Move, "p_05"));
                      grdt_02.Activations.add(new Activation(t_02, "p_04",
TransitionOperation.Move, "p_06"));

                      t_02.GuardMappingList.add(grdt_02);

                      t_02.Delay = 0;
                      pn.Transitions.add(t_02);



                      // T_03 -----------------------------------------------
                      PetriTransition t_03 = new PetriTransition(pn);
                      t_03.TransitionName = "t_03";
                      t_03.InputPlaceName.add("p_06");
                      t_03.InputPlaceName.add("Cc_1_c");
                      //t_03.IsAsync= true;

                      Condition T_03Ct1 = new Condition(t_03, "p_06",
TransitionCondition.NotNull);
                      Condition T_03Ct2 = new Condition(t_03, "Cc_1_c",
TransitionCondition.NotNull);
                      T_03Ct1.SetNextCondition(LogicConnector.AND, T_03Ct2);

                      GuardMapping grdt_03 = new GuardMapping();
                      grdt_03.condition = T_03Ct1;

                      grdt_03.Activations.add(new Activation(t_03,"Cc_1_c",
TransitionOperation.SendOverNetwork, "Cc_2_u"));
                      grdt_03.Activations.add(new Activation(t_03,"p_06",
TransitionOperation.Move, "p_00"));

                      t_03.GuardMappingList.add(grdt_03);

                      t_03.Delay = 0;
                      pn.Transitions.add(t_03);


                      // -------------------------------------------

                      System.out.println("OETPN started \n ------------------------------");
                      pn.Delay = 10;
                      pn.PrintingSpeed=50;
                      pn.ShowLogInWindow=false;
                      // pn.Start();
```

```
				PetriNetWindow frame = new PetriNetWindow(false);
				frame.petriNet = pn;
				frame.setVisible(true);
	}
}
```

Code sequence 5: PIController class

```
package DCS_FuzzyLab.OETPN_C;

import java.io.FileNotFoundException;
import java.util.ArrayList;
import Components.Activation;
import Components.Condition;
import Components.GuardMapping;
import Components.PetriNet;
import Components.PetriNetWindow;
import Components.PetriTransition;
import DataObjects.DataFuzzy;
import DataObjects.DataTransfer;
import DataOnly.FLRS;
import DataOnly.FV;
import DataOnly.Fuzzy;
import DataOnly.FuzzyVector;
import DataOnly.PlaceNameWithWeight;
import DataOnly.TransferOperation;
import Enumerations.FZ;
import Enumerations.LogicConnector;
import Enumerations.TransitionCondition;
import Enumerations.TransitionOperation;
// Cc1 The controller:
public class PIController {
	public static void main (String[]args) throws FileNotFoundException {
	PetriNet pn = new PetriNet();
	pn.PetriNetName = "PI Controller";
	pn.NetworkPort = 1081;

//	pn.SetInputFile("D:\\PetriInputData\\PIController.txt"); //for testing PI
controller alone, put initial marking in p2 and p4, T7 must all be commented

	FLRS reader = new FLRS(new FV(FZ.NL), new FV(FZ.NM), new FV(FZ.ZR), new
FV(FZ.PM), new FV(FZ.PL),
							new FV(FZ.NL), new FV(FZ.NM), new
FV(FZ.ZR), new FV(FZ.PM), new FV(FZ.PL),
							new FV(FZ.NL), new FV(FZ.NM), new
FV(FZ.ZR), new FV(FZ.PM), new FV(FZ.PL),
							new FV(FZ.NL), new FV(FZ.NM), new
FV(FZ.ZR), new FV(FZ.PM), new FV(FZ.PL),
							new FV(FZ.NL), new FV(FZ.NM), new
FV(FZ.ZR), new FV(FZ.PM), new FV(FZ.PL));


	FLRS doubleChannelAdder = new FLRS(new FV(FZ.NL, FZ.NL), new FV(FZ.NL,
FZ.NL), new FV(FZ.NL, FZ.NL), new FV(FZ.NM, FZ.NM),new FV(FZ.ZR, FZ.ZR),
                                                            new FV(FZ.NL, FZ.NL),
new FV(FZ.NL, FZ.NL), new FV(FZ.NM, FZ.NM), new FV(FZ.ZR, FZ.ZR),new FV(FZ.PM,
FZ.PM),
                                                            new FV(FZ.NL, FZ.NL),
new FV(FZ.NM, FZ.NM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM, FZ.PM),new FV(FZ.PL,
FZ.PL),
                                                            new FV(FZ.NM, FZ.NM),
new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM, FZ.PM), new FV(FZ.PL, FZ.PL),new FV(FZ.PL,
FZ.PL),
                                                            new FV(FZ.ZR, FZ.ZR),
new FV(FZ.PM, FZ.PM), new FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL),new FV(FZ.PL,
FZ.PL));

      FLRS doubleChannelDifferentiator = new FLRS(new FV(FZ.ZR, FZ.ZR), new
FV(FZ.PM, FZ.PM), new FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL),new FV(FZ.PL, FZ.PL),
                                                                            new
FV(FZ.NM, FZ.NM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM, FZ.PM), new FV(FZ.PL,
FZ.PL),new FV(FZ.PL, FZ.PL),
                                                                            new
FV(FZ.NL, FZ.NL), new FV(FZ.NM, FZ.NM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM,
FZ.PM),new FV(FZ.PL, FZ.PL),
                                                                            new
FV(FZ.NL, FZ.NL), new FV(FZ.NL, FZ.NL), new FV(FZ.NM, FZ.NM), new FV(FZ.ZR,
FZ.ZR),new FV(FZ.PM, FZ.PM),
                                                                            new
FV(FZ.NL, FZ.NL), new FV(FZ.NL, FZ.NL), new FV(FZ.NL, FZ.NL), new FV(FZ.NM,
FZ.NM),new FV(FZ.ZR, FZ.ZR));

      FLRS doubleChannelDifferentiator2 = new FLRS(new FV(FZ.ZR, FZ.ZR), new
FV(FZ.NM, FZ.NM), new FV(FZ.NL, FZ.NL), new FV(FZ.NL, FZ.NL),new FV(FZ.NL, FZ.NL),
                                                                             new
FV(FZ.PM, FZ.PM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.NM, FZ.NM), new FV(FZ.NL,
FZ.NL),new FV(FZ.NL, FZ.NL),
                                                                             new
FV(FZ.PL, FZ.PL), new FV(FZ.PM, FZ.PM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.NM,
FZ.NM),new FV(FZ.NL, FZ.NL),
                                                                             new
FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL), new FV(FZ.PM, FZ.PM), new FV(FZ.ZR,
FZ.ZR),new FV(FZ.NM, FZ.NM),
                                                                             new
FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL), new FV(FZ.PL, FZ.PL), new FV(FZ.PM,
FZ.PM),new FV(FZ.ZR, FZ.ZR));


      FLRS adder = new FLRS(new FV(FZ.NL), new FV(FZ.NL), new FV(FZ.NL), new
FV(FZ.NM), new FV(FZ.ZR),
                                        new FV(FZ.NL), new FV(FZ.NL), new
FV(FZ.NM), new FV(FZ.ZR), new FV(FZ.PM),
                                        new FV(FZ.NL), new FV(FZ.NM), new
FV(FZ.ZR), new FV(FZ.PM), new FV(FZ.PL),
                                        new FV(FZ.NM), new FV(FZ.ZR), new
FV(FZ.PM), new FV(FZ.PL), new FV(FZ.PL),
                                        new FV(FZ.ZR), new FV(FZ.PM), new
FV(FZ.PL), new FV(FZ.PL), new FV(FZ.PL));
```

```
	FLRS OneXOneDefaultTable = new FLRS(new FV(FZ.NL), new FV(FZ.NM), new
FV(FZ.ZR), new FV(FZ.PM),new FV(FZ.PL));

	FLRS OneXTwoDefaultTable = new FLRS(new FV(FZ.NL,FZ.NL), new
FV(FZ.NM,FZ.NM), new FV(FZ.ZR,FZ.ZR), new FV(FZ.PM,FZ.PM),new FV(FZ.PL,FZ.PL));

	reader.Print();
	doubleChannelAdder.Print();
	doubleChannelDifferentiator.Print();
	doubleChannelDifferentiator2.Print();
	adder.Print();
	OneXOneDefaultTable.Print();
	OneXTwoDefaultTable.Print();


	DataFuzzy p0 = new DataFuzzy();
	p0.SetName("P0");
	p0.SetValue(new Fuzzy(0.0F));
	pn.PlaceList.add(p0);

	DataFuzzy p1 = new DataFuzzy();
	p1.SetName("P1");
	pn.PlaceList.add(p1);

	DataFuzzy p2 = new DataFuzzy(); //from OETPN
	p2.SetName("P2");
	pn.PlaceList.add(p2);

	DataFuzzy p3 = new DataFuzzy();
	p3.SetName("P3");
	pn.PlaceList.add(p3);

	DataFuzzy p4 = new DataFuzzy();  //from OETPN
	p4.SetName("P4");
	pn.PlaceList.add(p4);

	DataFuzzy p5 = new DataFuzzy();
	p5.SetName("P5");
	pn.PlaceList.add(p5);

	DataFuzzy p6 = new DataFuzzy();
	p6.SetName("P6");
	pn.PlaceList.add(p6);

	DataFuzzy p7 = new DataFuzzy();
	p7.SetName("P7");
	p7.SetValue(new Fuzzy(0.0F));
	pn.PlaceList.add(p7);

	DataFuzzy p8 = new DataFuzzy();
	p8.SetName("P8");
	pn.PlaceList.add(p8);

	DataFuzzy p9 = new DataFuzzy();
	p9.SetName("P9");
```

```
      pn.PlaceList.add(p9);

      DataFuzzy p10 = new DataFuzzy();
      p10.SetName("P10");
      pn.PlaceList.add(p10);

      DataFuzzy p11 = new DataFuzzy();
      p11.SetName("P11");
      p11.SetValue(new Fuzzy(0.0F));
      pn.PlaceList.add(p11);

      DataTransfer p_o_1 = new DataTransfer();
      p_o_1.SetName("c");
      p_o_1.Value = new TransferOperation("localhost", "1080", "Cc_1_c"); //to
OETPN
      pn.PlaceList.add(p_o_1);


      // T0 ------------------------------------------------
                          PetriTransition t0 = new PetriTransition(pn);
                          t0.TransitionName = "T0";
                          t0.InputPlaceName.add("P0");


                          Condition T0Ct1 = new Condition(t0, "P0",
TransitionCondition.NotNull);

                          GuardMapping grdT0 = new GuardMapping();
                          grdT0.condition = T0Ct1;

                          ArrayList<PlaceNameWithWeight> input0 = new
ArrayList<>();
                          input0.add(new PlaceNameWithWeight("P0", 1F));

                          ArrayList<String> Output0 = new ArrayList<>();
                          Output0.add("P1");


                          grdT0.Activations.add(new Activation(t0,
OneXOneDefaultTable, input0, TransitionOperation.FLRS, Output0));

                          t0.GuardMappingList.add(grdT0);

                          t0.Delay = 0;
                          pn.Transitions.add(t0);



      // T1 ------------------------------------------------
                   PetriTransition t1 = new PetriTransition(pn);
                   t1.TransitionName = "T1";
                   t1.InputPlaceName.add("P1");
                   t1.InputPlaceName.add("P2");
```

```
                    Condition T1Ct1 = new Condition(t1, "P1",
TransitionCondition.NotNull);
                    Condition T1Ct2 = new Condition(t1, "P2",
TransitionCondition.NotNull);
                    T1Ct1.SetNextCondition(LogicConnector.AND, T1Ct2);

                    GuardMapping grdT1 = new GuardMapping();
                    grdT1.condition = T1Ct1;

                    ArrayList<PlaceNameWithWeight> input1 = new ArrayList<>();
                    input1.add(new PlaceNameWithWeight("P1", 1F));
                    input1.add(new PlaceNameWithWeight("P2", 1F));

                    ArrayList<String> Output1 = new ArrayList<>();
                    Output1.add("P3");


                    grdT1.Activations.add(new Activation(t1, reader, input1,
TransitionOperation.FLRS, Output1));

                    t1.GuardMappingList.add(grdT1);

                    t1.Delay = 0;
                    pn.Transitions.add(t1);


                    // T2 ------------------------------------------------
                    PetriTransition t2 = new PetriTransition(pn);
                    t2.TransitionName = "T2";
                    t2.InputPlaceName.add("P3");
                    t2.InputPlaceName.add("P4");

                    Condition T2Ct1 = new Condition(t2, "P3",
TransitionCondition.NotNull);
                    Condition T2Ct2 = new Condition(t2, "P4",
TransitionCondition.NotNull);
                    T2Ct1.SetNextCondition(LogicConnector.AND, T2Ct2);

                    GuardMapping grdT2 = new GuardMapping();
                    grdT2.condition = T2Ct1;

                    ArrayList<PlaceNameWithWeight> input2 = new ArrayList<>();
                    input2.add(new PlaceNameWithWeight("P3", 1F));
                    input2.add(new PlaceNameWithWeight("P4", 1F));


                    ArrayList<String> Output2 = new ArrayList<>();
                    Output2.add("P5");
                    Output2.add("P6");


                    grdT2.Activations.add(new Activation(t2,
doubleChannelDifferentiator, input2, TransitionOperation.FLRS, Output2));

                    t2.GuardMappingList.add(grdT2);
                    t2.Delay = 0;
                    pn.Transitions.add(t2);



                    // T3 ------------------------------------------------
                    PetriTransition t3 = new PetriTransition(pn);
                    t3.TransitionName = "T3";
                    t3.InputPlaceName.add("P5");
                    t3.InputPlaceName.add("P7");

                    Condition T3Ct1 = new Condition(t3, "P5",
TransitionCondition.NotNull);
                    Condition T3Ct2 = new Condition(t3, "P7",
TransitionCondition.NotNull);
                    T3Ct1.SetNextCondition(LogicConnector.AND, T3Ct2);

                    GuardMapping grdT3 = new GuardMapping();
                    grdT3.condition = T3Ct1;

                    ArrayList<PlaceNameWithWeight> input3 = new ArrayList<>();
                    input3.add(new PlaceNameWithWeight("P5", 0.8F));
                    input3.add(new PlaceNameWithWeight("P7", 0.2F));


                    ArrayList<String> Output3 = new ArrayList<>();
                    Output3.add("P8");


                    grdT3.Activations.add(new Activation(t3, adder, input3,
TransitionOperation.FLRS, Output3));

                    t3.GuardMappingList.add(grdT3);

                    t3.Delay = 0;
                    pn.Transitions.add(t3);


                    // T4 ------------------------------------------------
                    PetriTransition t4 = new PetriTransition(pn);
                    t4.TransitionName = "T4";
                    t4.InputPlaceName.add("P6");

                    Condition T4Ct1 = new Condition(t4, "P6",
TransitionCondition.NotNull);

                    GuardMapping grdT4 = new GuardMapping();
                    grdT4.condition = T4Ct1;

                    ArrayList<PlaceNameWithWeight> input4 = new ArrayList<>();
                    input4.add(new PlaceNameWithWeight("P6", 1F));

                    ArrayList<String> Output4 = new ArrayList<>();
                    Output4.add("P7");
                    grdT4.Activations.add(new Activation(t4, OneXOneDefaultTable, 
input4, TransitionOperation.FLRS, Output4));

                    t4.GuardMappingList.add(grdT4);

                    t4.Delay = 1;
                    pn.Transitions.add(t4);



                    // T5 ------------------------------------------------
                    PetriTransition t5 = new PetriTransition(pn);
                    t5.TransitionName = "T5";
                    t5.InputPlaceName.add("P8");
                    t5.InputPlaceName.add("P11");

                    Condition T5Ct1 = new Condition(t5, "P8", 
TransitionCondition.NotNull);
                    Condition T5Ct2 = new Condition(t5, "P11", 
TransitionCondition.NotNull);
                    T5Ct1.SetNextCondition(LogicConnector.AND, T5Ct2);

                    GuardMapping grdT5 = new GuardMapping();
                    grdT5.condition = T5Ct1;

                    ArrayList<PlaceNameWithWeight> input5 = new ArrayList<>();
                    input5.add(new PlaceNameWithWeight("P8", 1.2F));
                    input5.add(new PlaceNameWithWeight("P11", 1F));


                    ArrayList<String> Output5 = new ArrayList<>();
                    Output5.add("P9");
                    Output5.add("P10");


                    grdT5.Activations.add(new Activation(t5, doubleChannelAdder, 
input5, TransitionOperation.FLRS, Output5));

                    t5.GuardMappingList.add(grdT5);

                    t5.Delay = 0;
                    pn.Transitions.add(t5);


                    // T6 ------------------------------------------------
                    PetriTransition t6 = new PetriTransition(pn);
                    t6.TransitionName = "T6";
                    t6.InputPlaceName.add("P10");

                    Condition T6Ct1 = new Condition(t6, "P10", 
TransitionCondition.NotNull);

                    GuardMapping grdT6 = new GuardMapping();
                    grdT6.condition = T6Ct1;

                    ArrayList<PlaceNameWithWeight> input6 = new ArrayList<>();
                    input6.add(new PlaceNameWithWeight("P10", 1F));

                    ArrayList<String> Output6 = new ArrayList<>();
                    Output6.add("P1");
                    Output6.add("P11");


                    grdT6.Activations.add(new Activation(t6, OneXTwoDefaultTable,
input6, TransitionOperation.FLRS, Output6));

                    t6.GuardMappingList.add(grdT6);

                    t6.Delay = 1;
                    pn.Transitions.add(t6);

// T7 ------------------------------------------------
                    PetriTransition t7 = new PetriTransition(pn);
                    t7.TransitionName = "T7";
                    t7.InputPlaceName.add("P9");

                    Condition T7Ct1 = new Condition(t6, "P9",
TransitionCondition.NotNull);

                    GuardMapping grdT7 = new GuardMapping();
                    grdT7.condition = T7Ct1;

                    grdT7.Activations.add(new Activation(t7, "P9",
TransitionOperation.SendOverNetwork, "c"));

                    t7.GuardMappingList.add(grdT7);

                    t7.Delay = 0;
                    pn.Transitions.add(t7);

                    // -------------------------------------------

                    System.out.println("PIController started \n -------------------
-----------");
                    pn.Delay = 10;
                    pn.PrintingSpeed=50;
                    pn.ShowLogInWindow=false;
                    // pn.Start();

                    PetriNetWindow frame = new PetriNetWindow(false);
                    frame.petriNet = pn;
                    frame.setVisible(true);
       }
}
```

Code sequence 6: Plant class

```
package DCS_FuzzyLab.OETPN_C;
```

```java
import java.io.FileNotFoundException;
import java.util.ArrayList;
import Components.Activation;
import Components.Condition;
import Components.GuardMapping;
import Components.PetriNet;
import Components.PetriNetWindow;
import Components.PetriTransition;
import DataObjects.DataFuzzy;
import DataObjects.DataTransfer;
import DataOnly.FLRS;
import DataOnly.FV;
import DataOnly.Fuzzy;
import DataOnly.FuzzyVector;
import DataOnly.PlaceNameWithWeight;
import DataOnly.TransferOperation;
import Enumerations.FZ;
import Enumerations.LogicConnector;
import Enumerations.TransitionCondition;
import Enumerations.TransitionOperation;

public class Plant {
	public static void main (String[]args) throws FileNotFoundException {
	PetriNet pn = new PetriNet();
	pn.PetriNetName = "Plant";
	pn.NetworkPort = 1082;

	FLRS doubleChannelAdder = new FLRS(new FV(FZ.NL, FZ.NL), new FV(FZ.NL, FZ.NL), new
FV(FZ.NL, FZ.NL), new FV(FZ.NM, FZ.NM),new FV(FZ.ZR, FZ.ZR),
				new FV(FZ.NL, FZ.NL), new FV(FZ.NL, FZ.NL), new FV(FZ.NM, FZ.NM),
new FV(FZ.ZR, FZ.ZR),new FV(FZ.PM, FZ.PM),
				new FV(FZ.NL, FZ.NL), new FV(FZ.NM, FZ.NM), new FV(FZ.ZR, FZ.ZR),
new FV(FZ.PM, FZ.PM),new FV(FZ.PL, FZ.PL),
				new FV(FZ.NM, FZ.NM), new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM, FZ.PM),
new FV(FZ.PL, FZ.PL),new FV(FZ.PL, FZ.PL),
				new FV(FZ.ZR, FZ.ZR), new FV(FZ.PM, FZ.PM), new FV(FZ.PL, FZ.PL),
new FV(FZ.PL, FZ.PL),new FV(FZ.PL, FZ.PL));

	doubleChannelAdder.Print();

	DataFuzzy p_i2 = new DataFuzzy(); //from OETPN
	p_i2.SetName("u");
	pn.PlaceList.add(p_i2);

	DataFuzzy p_20 = new DataFuzzy(); //from operator
	p_20.SetName("x");
	p_20.SetValue(new Fuzzy(0.0F));
	pn.PlaceList.add(p_20);

	DataFuzzy p_21 = new DataFuzzy();
	p_21.SetName("y");
	pn.PlaceList.add(p_21);

	DataTransfer p_o2 = new DataTransfer();
	p_o2.SetName("Cc_2_y");
	p_o2.Value = new TransferOperation("localhost", "1080", "Cc_2_y"); //to OETPN
	pn.PlaceList.add(p_o2);



	// T_21 ------------------------------------------------
```

```
                              PetriTransition t_21 = new PetriTransition(pn);
                              t_21.TransitionName = "t_21";
                              t_21.InputPlaceName.add("u");
                              t_21.InputPlaceName.add("x");


                              Condition T_21Ct1 = new Condition(t_21, "u",
TransitionCondition.NotNull);
                              Condition T_21Ct2 = new Condition(t_21, "x",
TransitionCondition.NotNull);
                              T_21Ct1.SetNextCondition(LogicConnector.AND, T_21Ct2);

                              GuardMapping grdt_21 = new GuardMapping();
                              grdt_21.condition = T_21Ct1;

                              ArrayList<PlaceNameWithWeight> input1 = new ArrayList<>();
                              input1.add(new PlaceNameWithWeight("x", 0.5F)); //a*x
                              input1.add(new PlaceNameWithWeight("u", 0.7F)); //b*u

                              ArrayList<String> output1 = new ArrayList<>();
                              output1.add("x");
                              output1.add("y");


                              grdt_21.Activations.add(new Activation(t_21,
doubleChannelAdder, input1, TransitionOperation.FLRS, output1));



                              t_21.GuardMappingList.add(grdt_21);

                              t_21.Delay = 1;
                              pn.Transitions.add(t_21);


                              // T22 ------------------------------------------------
                              PetriTransition t_22 = new PetriTransition(pn);
                              t_22.TransitionName = "t_22";
                              t_22.InputPlaceName.add("y");

                              Condition T_22Ct1 = new Condition(t_22, "y",
TransitionCondition.NotNull);

                              GuardMapping grdt_22 = new GuardMapping();
                              grdt_22.condition = T_22Ct1;

                              grdt_22.Activations.add(new Activation(t_22, "y",
TransitionOperation.SendOverNetwork, "Cc_2_y"));

                              t_22.GuardMappingList.add(grdt_22);

                              t_22.Delay = 0;
                              pn.Transitions.add(t_22);

       // -------------------------------------------

                      System.out.println("Plant started \n ------------------------------");
                      pn.Delay = 10;
                      pn.PrintingSpeed=50;
                      pn.ShowLogInWindow=false;
                      // pn.Start();
```

```
                        PetriNetWindow frame = new PetriNetWindow(false);
                        frame.petriNet = pn;
                        frame.setVisible(true);
        }
}
```

4. Exercises:

1. Change the weight of the arcs of the PI controller so that the performance of the system would be as good as possible.

2. Modify the first order system constants of the plant. Find the values of the driver and the coefficient values so the system has the best performance behavior.

3. Starting from the previous application, develop an application to implement the proportional integral derivative (PID) controller. The Petri Net will be in Figure 4.1.

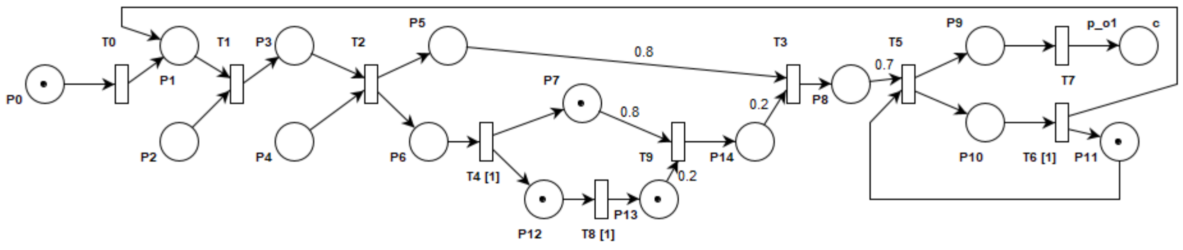

Figure 4.1 FLETPN for PID.

5. Verification of knowledge

1. How to build a Petri Net using the All_Petri_Framework?
2. What is the OETPN-c and what it is used for?
3. Using the input file, when is the executior launched and how much the PN delay should be and why?
4. How the performance of the system is influenced by the weight of the arcs?
5. How to attach FLRS tables to transitions using the All_Petri_Framework?