НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО”

КАФЕДРА АВТОМАТИЗОВАНИХ СИСТЕМ ОБРОБКИ ІНФОРМАЦІЇ І УПРАВЛІННЯ

Комп’ютерний практикум № 7

з дисципліни

“Моделювання систем”

Виконав:

студент групи ІТ-04

Чапча Святослав

Перевірив:

Дифучин А.Ю.

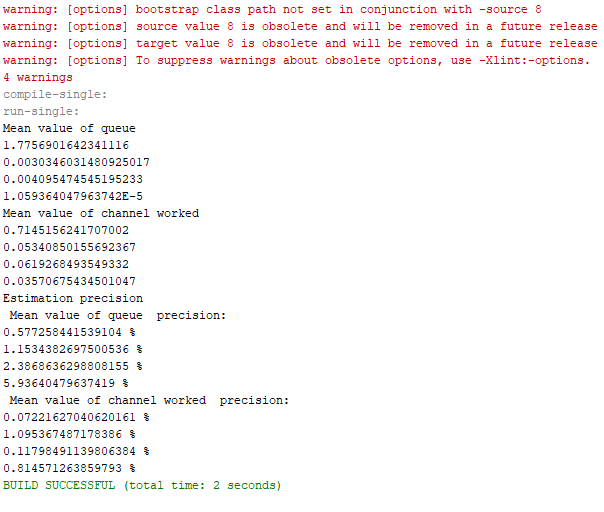
Київ-2023

**Тема:** «Формалізація дискретно-подійних систем стохастичною мережею Петрі»

**Завдання 1 до практичної роботи :**

Розглянути алгоритм Петрі-об’єткного моделювання, реалізований в бібліотеці PetriObjModelPaint (див. github StetsenkoInna). Виконати тестування запропонованого алгоритму на моделі мережі маcового обслуговування. 15 балів

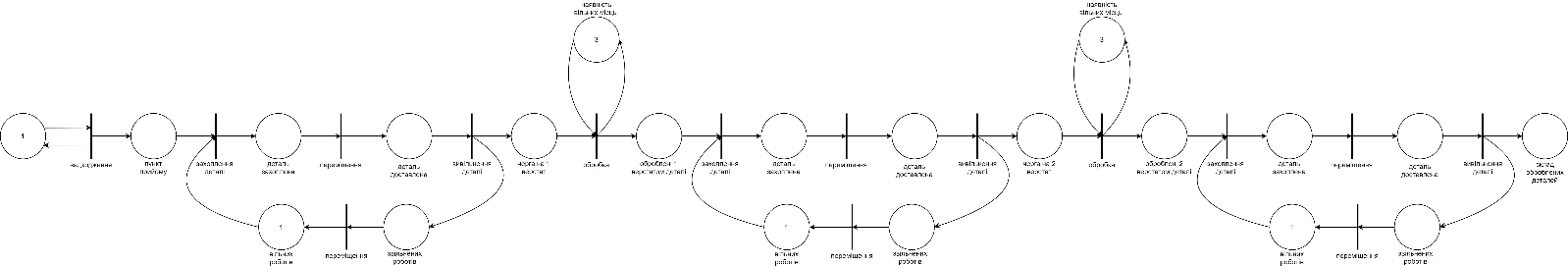
**Результат виконання:**

****

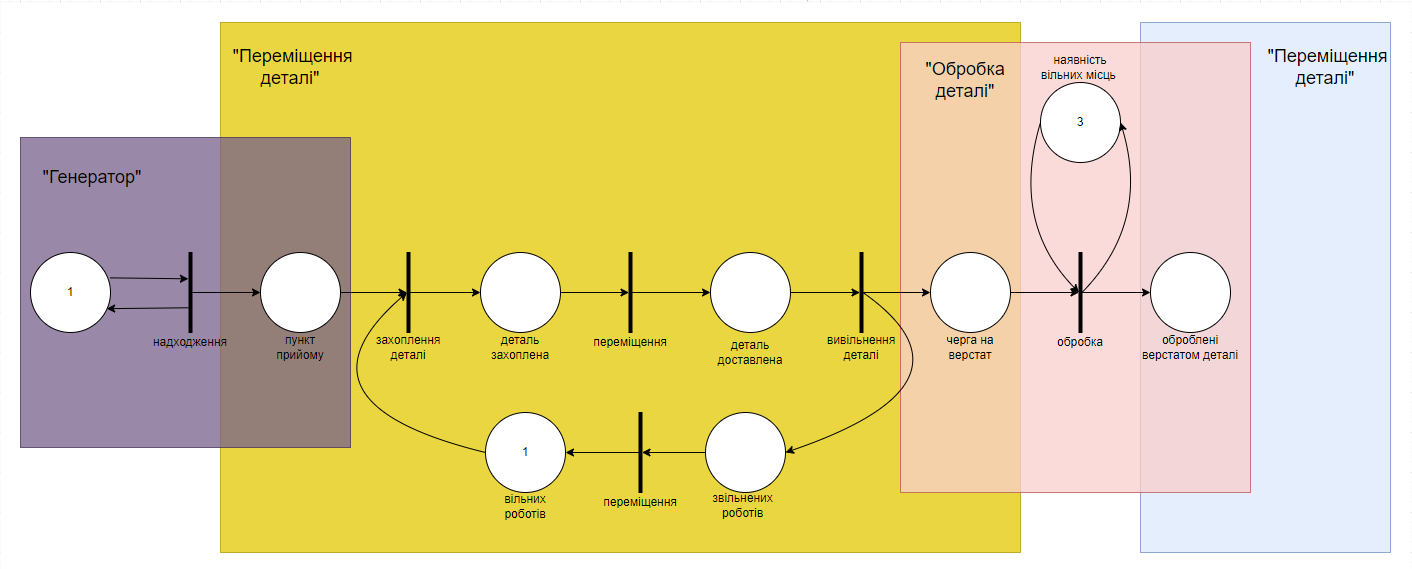
**Завдання 2 до практичної роботи :**

За текстом завдання 2 практикуму 5 розробити відповідні Петрі-об’єктри та побудувати Петрі-об’єктну модель системи. Отримати результати імітаційного моделювання. Зробити висновки про функціонування моделі. 30 балів.

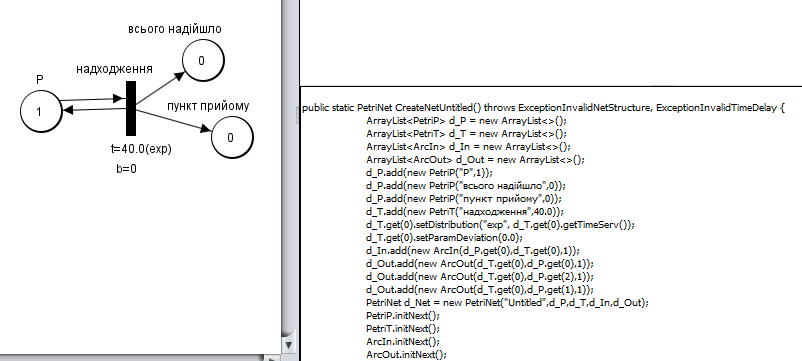
**Початкова схема:**

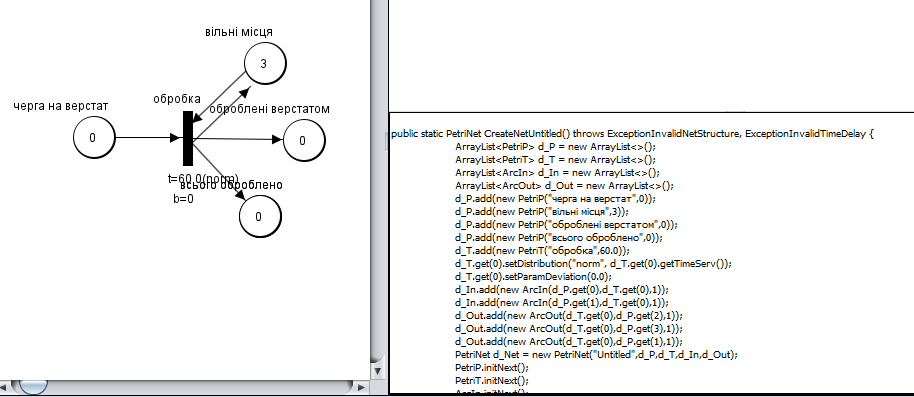


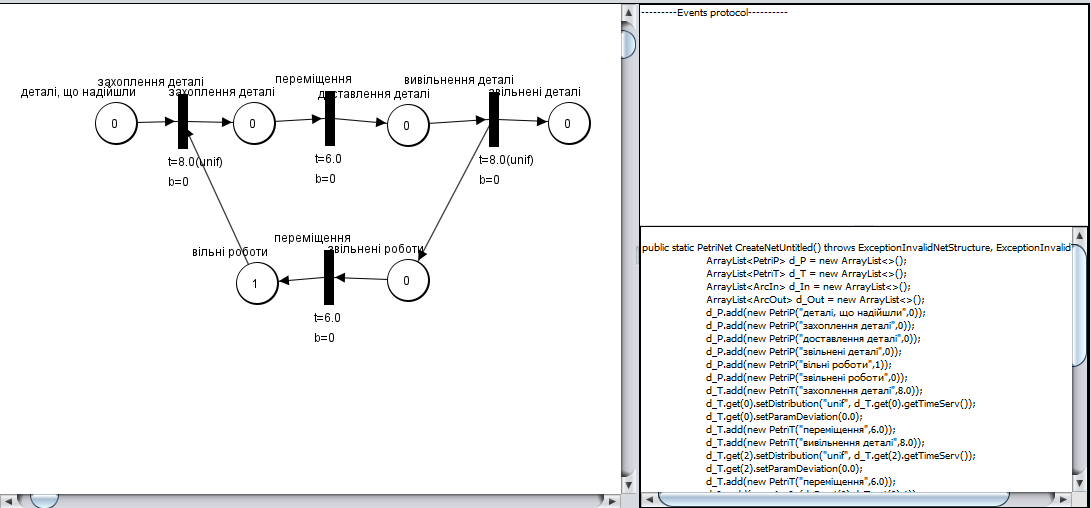
**Петрі-об’єктна схема:**

****

**Згенеровані методи для Петрі-об’єктів:**

****

****

****

**Лістинг коду:**

import PetriObj.PetriObjModel;

import PetriObj.PetriSim;

import PetriObj.ArcIn;

import PetriObj.ArcOut;

import PetriObj.ExceptionInvalidNetStructure;

import PetriObj.ExceptionInvalidTimeDelay;

import PetriObj.PetriNet;

import PetriObj.PetriP;

import PetriObj.PetriT;

import java.util.ArrayList;

public class robots {

public static void main(String[] args) throws ExceptionInvalidTimeDelay, ExceptionInvalidNetStructure {

firstMain();

}

public static void firstMain() throws ExceptionInvalidTimeDelay, ExceptionInvalidNetStructure {

PetriObjModel model = getModel();

model.setIsProtokol(false);

double timeModeling = 1000000;

model.go(timeModeling);

System.out.println("Total Arrived");

System.out.println(model.getListObj().get(0).getNet().getListP()[2].getMark());

System.out.println("Processed By Machine 1");

System.out.println(model.getListObj().get(4).getNet().getListP()[3].getMark());

System.out.println("Processed By Machine 2");

System.out.println(model.getListObj().get(5).getNet().getListP()[3].getMark());

System.out.println("Arrived to Storage");

System.out.println(model.getListObj().get(3).getNet().getListP()[3].getMark());

System.out.println("Percentage of Fully Processed");

System.out.println((double) model.getListObj().get(3).getNet().getListP()[3].getMark()

/ model.getListObj().get(0).getNet().getListP()[2].getMark() \* 100);

}

public static PetriObjModel getModel() throws ExceptionInvalidTimeDelay, ExceptionInvalidNetStructure {

ArrayList<PetriSim> list = new ArrayList<>();

list.add(new PetriSim(CreateGenerator()));

list.add(new PetriSim(Create1Transportation(6, "Transport start-1")));

list.add(new PetriSim(Create1Transportation(7, "Transport 1-2")));

list.add(new PetriSim(Create1Transportation(5, "Transport 2-finish")));

list.add(new PetriSim(CreateProcess("norm", 60, 10, "process 1")));

list.add(new PetriSim(CreateProcess("exp", 100, "process 2")));

list.get(0).getNet().getListP()[1] = list.get(1).getNet().getListP()[0];

list.get(1).getNet().getListP()[3] = list.get(4).getNet().getListP()[0];

list.get(4).getNet().getListP()[1] = list.get(2).getNet().getListP()[0];

list.get(2).getNet().getListP()[3] = list.get(5).getNet().getListP()[0];

list.get(5).getNet().getListP()[1] = list.get(3).getNet().getListP()[0];

PetriObjModel model = new PetriObjModel(list);

return model;

}

public static PetriNet CreateGenerator() throws ExceptionInvalidNetStructure, ExceptionInvalidTimeDelay {

ArrayList<PetriP> d\_P = new ArrayList<>();

ArrayList<PetriT> d\_T = new ArrayList<>();

ArrayList<ArcIn> d\_In = new ArrayList<>();

ArrayList<ArcOut> d\_Out = new ArrayList<>();

d\_P.add(new PetriP("P", 1));

d\_P.add(new PetriP("пункт прийому", 0));

d\_P.add(new PetriP("всього надійшло", 0));

d\_T.add(new PetriT("надходження", 40.0));

d\_T.get(0).setDistribution("exp", d\_T.get(0).getTimeServ());

d\_T.get(0).setParamDeviation(0.0);

d\_In.add(new ArcIn(d\_P.get(0), d\_T.get(0), 1));

d\_Out.add(new ArcOut(d\_T.get(0), d\_P.get(0), 1));

d\_Out.add(new ArcOut(d\_T.get(0), d\_P.get(1), 1));

d\_Out.add(new ArcOut(d\_T.get(0), d\_P.get(2), 1));

PetriNet d\_Net = new PetriNet("Generator", d\_P, d\_T, d\_In, d\_Out);

PetriP.initNext();

PetriT.initNext();

ArcIn.initNext();

ArcOut.initNext();

return d\_Net;

}

public static PetriNet Create1Transportation(double time, String name) throws ExceptionInvalidNetStructure, ExceptionInvalidTimeDelay {

ArrayList<PetriP> d\_P = new ArrayList<>();

ArrayList<PetriT> d\_T = new ArrayList<>();

ArrayList<ArcIn> d\_In = new ArrayList<>();

ArrayList<ArcOut> d\_Out = new ArrayList<>();

d\_P.add(new PetriP("деталі, що надійшли", 0));

d\_P.add(new PetriP("захоплені деталі", 0));

d\_P.add(new PetriP("доставлені деталі", 0));

d\_P.add(new PetriP("звільнені деталі", 0));

d\_P.add(new PetriP("вільні роботи", 1));

d\_P.add(new PetriP("звільнені роботи", 0));

d\_T.add(new PetriT("захоплення деталі", 8.0));

d\_T.get(0).setDistribution("unif", d\_T.get(0).getTimeServ());

d\_T.get(0).setParamDeviation(1.0);

d\_T.add(new PetriT("переміщення", time));

d\_T.add(new PetriT("вивільнення деталі", 8.0));

d\_T.get(2).setDistribution("unif", d\_T.get(2).getTimeServ());

d\_T.get(2).setParamDeviation(1.0);

d\_T.add(new PetriT("переміщення", time));

d\_In.add(new ArcIn(d\_P.get(0), d\_T.get(0), 1));

d\_In.add(new ArcIn(d\_P.get(1), d\_T.get(1), 1));

d\_In.add(new ArcIn(d\_P.get(2), d\_T.get(2), 1));

d\_In.add(new ArcIn(d\_P.get(5), d\_T.get(3), 1));

d\_In.add(new ArcIn(d\_P.get(4), d\_T.get(0), 1));

d\_Out.add(new ArcOut(d\_T.get(0), d\_P.get(1), 1));

d\_Out.add(new ArcOut(d\_T.get(1), d\_P.get(2), 1));

d\_Out.add(new ArcOut(d\_T.get(2), d\_P.get(3), 1));

d\_Out.add(new ArcOut(d\_T.get(2), d\_P.get(5), 1));

d\_Out.add(new ArcOut(d\_T.get(3), d\_P.get(4), 1));

PetriNet d\_Net = new PetriNet(name, d\_P, d\_T, d\_In, d\_Out);

PetriP.initNext();

PetriT.initNext();

ArcIn.initNext();

ArcOut.initNext();

return d\_Net;

}

public static PetriNet CreateProcess(String dist, double mean, String name) throws ExceptionInvalidNetStructure, ExceptionInvalidTimeDelay {

ArrayList<PetriP> d\_P = new ArrayList<>();

ArrayList<PetriT> d\_T = new ArrayList<>();

ArrayList<ArcIn> d\_In = new ArrayList<>();

ArrayList<ArcOut> d\_Out = new ArrayList<>();

d\_P.add(new PetriP("черга на верстат", 0));

d\_P.add(new PetriP("оброблені верстатом", 0));

d\_P.add(new PetriP("вільні місця", 3));

d\_P.add(new PetriP("всього оброблено", 0));

d\_T.add(new PetriT("обробка", mean));

d\_T.get(0).setDistribution(dist, d\_T.get(0).getTimeServ());

d\_In.add(new ArcIn(d\_P.get(0), d\_T.get(0), 1));

d\_In.add(new ArcIn(d\_P.get(2), d\_T.get(0), 1));

d\_Out.add(new ArcOut(d\_T.get(0), d\_P.get(1), 1));

d\_Out.add(new ArcOut(d\_T.get(0), d\_P.get(2), 1));

d\_Out.add(new ArcOut(d\_T.get(0), d\_P.get(3), 1));

PetriNet d\_Net = new PetriNet(name, d\_P, d\_T, d\_In, d\_Out);

PetriP.initNext();

PetriT.initNext();

ArcIn.initNext();

ArcOut.initNext();

return d\_Net;

}

public static PetriNet CreateProcess(String dist, double mean, double deviation, String name) throws ExceptionInvalidNetStructure, ExceptionInvalidTimeDelay {

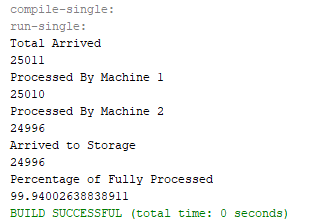
PetriNet d\_Net = CreateProcess(dist, mean, name);

d\_Net.getListT()[0].setParamDeviation(deviation);

return d\_Net;

}

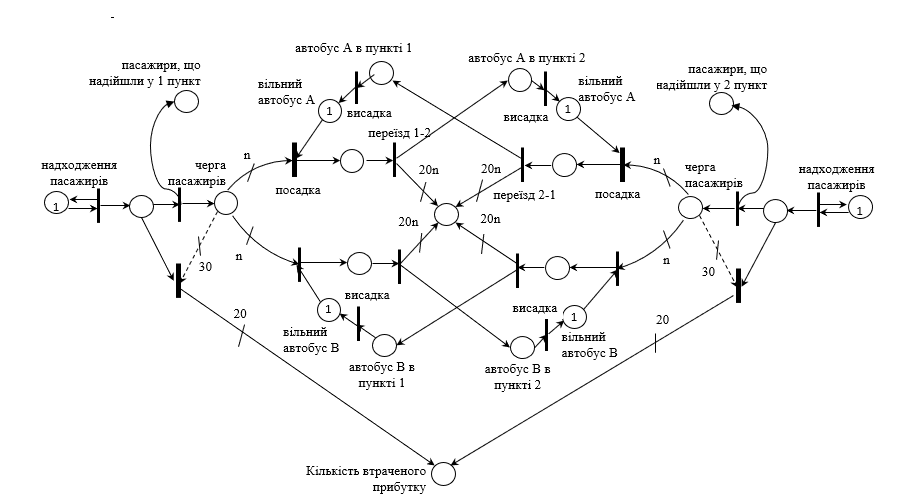
**Результат виконання моделювання схеми:**



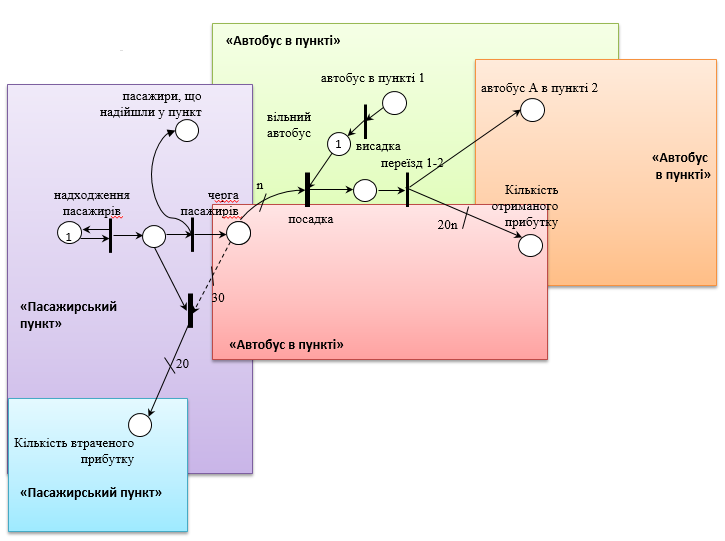
**Завдання 3 до практичної роботи :**

За текстом завдання 3 практикуму 5 розробити відповідні Петрі-об’єктри та побудувати Петрі-об’єктну модель системи. Отримати результати імітаційного моделювання. Зробити висновки про функціонування моделі. 30 балів.

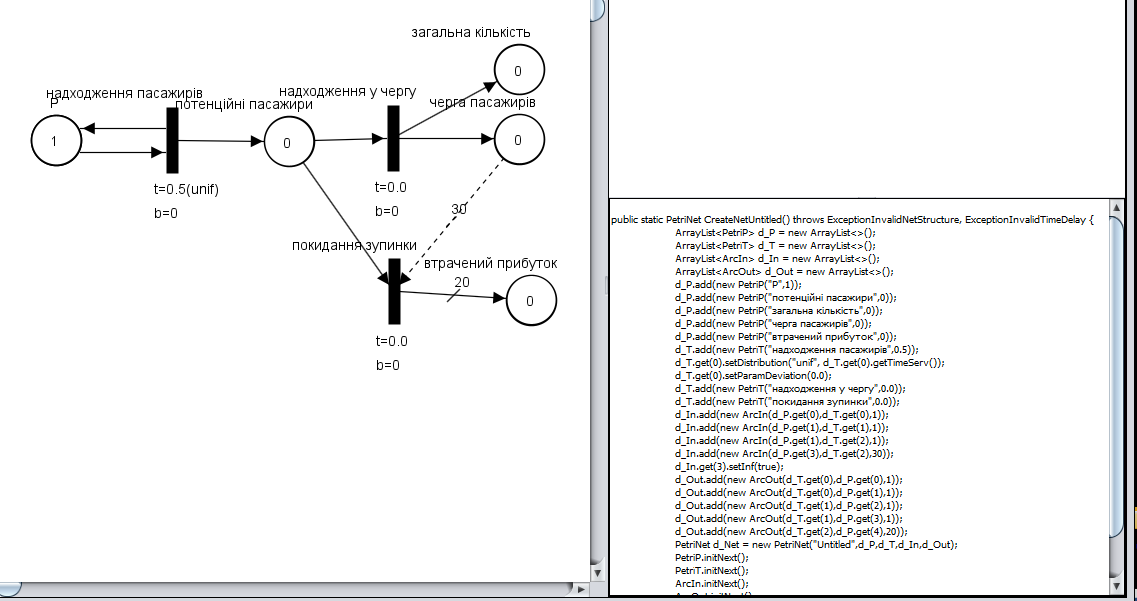
**Початкова схема:**

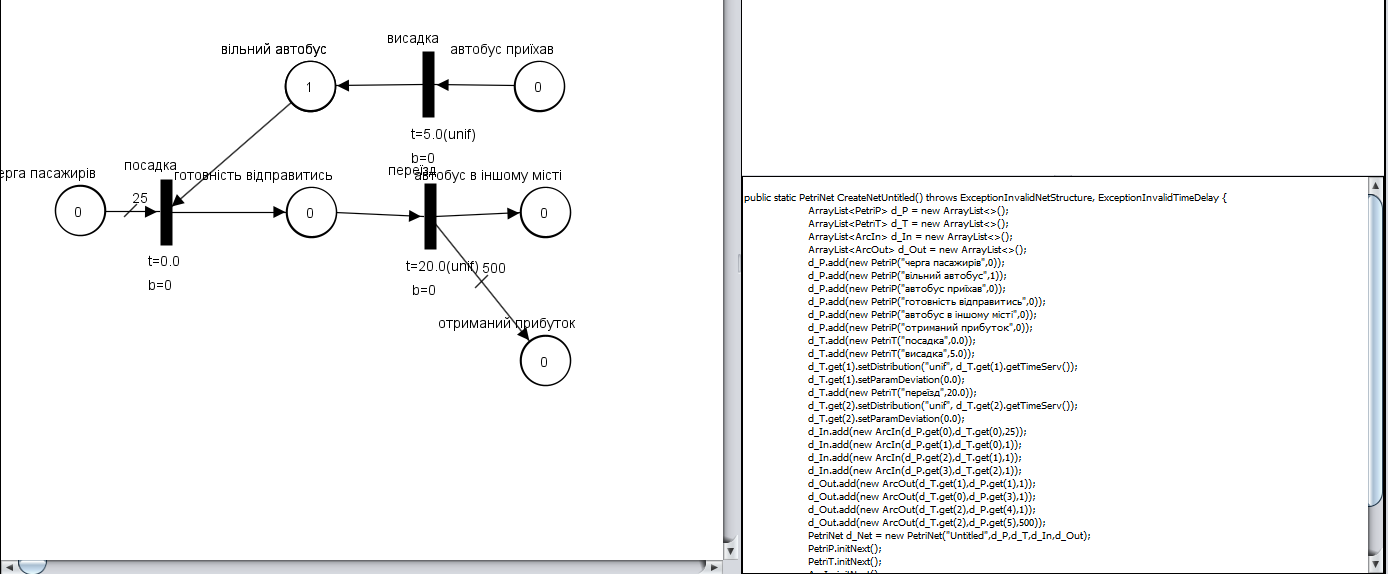


**Петрі-об’єктна схема:**

****

**Згенеровані методи для Петрі-об’єктів:**

****

****

**Лістинг коду:**

import LibNet.NetLibrary;

import PetriObj.PetriObjModel;

import PetriObj.PetriSim;

import PetriObj.ArcIn;

import PetriObj.ArcOut;

import PetriObj.ExceptionInvalidNetStructure;

import PetriObj.ExceptionInvalidTimeDelay;

import PetriObj.PetriNet;

import PetriObj.PetriP;

import PetriObj.PetriT;

import java.util.ArrayList;

public class buses {

public static void main(String[] args) throws ExceptionInvalidTimeDelay, ExceptionInvalidNetStructure {

PetriObjModel model = null;

int people = 10;

model = getModel(people);

model.setIsProtokol(false);

double timeModeling = 600;

model.go(timeModeling);

System.out.println("Transfered to Kyiv");

System.out.println(model.getListObj().get(0).getNet().getListP()[2].getMark());

System.out.println("Transfered to Lviv");

System.out.println(model.getListObj().get(1).getNet().getListP()[2].getMark());

System.out.println("Lost Money");

System.out.println(model.getListObj().get(0).getNet().getListP()[4].getMark());

System.out.println("Gain Money");

System.out.println(model.getListObj().get(2).getNet().getListP()[5].getMark());

System.out.println("Average Queue in Kyiv");

System.out.println(model.getListObj().get(0).getNet().getListP()[3].getMean());

System.out.println("Average Queue in Lviv");

System.out.println(model.getListObj().get(1).getNet().getListP()[3].getMean());

System.out.println("Waiting Time in Kyiv");

System.out.println(model.getListObj().get(0).getNet().getListP()[3].getMean() \* timeModeling

/ model.getListObj().get(0).getNet().getListP()[2].getMark());

System.out.println("Waiting Time in Lviv");

System.out.println(model.getListObj().get(1).getNet().getListP()[3].getMean() \* timeModeling

/ model.getListObj().get(1).getNet().getListP()[2].getMark());

System.out.println("Num Of Seats, When Waiting Time Will Be Min");

System.out.println(getMinTime(choise));

}

public static int getMinTime(int choise) throws ExceptionInvalidTimeDelay, ExceptionInvalidNetStructure {

int peopleMin = Integer.MAX\_VALUE;

double timeMin = Double.MAX\_VALUE;

PetriObjModel model = null;

for(int people = 1; people <= 25; people++){

model = getModel(people);

model.setIsProtokol(false);

double timeModeling = 600;

model.go(timeModeling);

double time = model.getListObj().get(0).getNet().getListP()[3].getMean() \* timeModeling

/ model.getListObj().get(0).getNet().getListP()[2].getMark()

+ model.getListObj().get(1).getNet().getListP()[3].getMean() \* timeModeling

/ model.getListObj().get(1).getNet().getListP()[2].getMark();

if(time < timeMin){

timeMin = time;

peopleMin = people;

}

}

return peopleMin;

}

public static PetriObjModel getModel(int people) throws ExceptionInvalidTimeDelay, ExceptionInvalidNetStructure {

ArrayList<PetriSim> list = new ArrayList<PetriSim>();

list.add(new PetriSim(CreateBusStop("Kyiv", 0.5, 0.2)));

list.add(new PetriSim(CreateBusStop("Lviv", 0.5, 0.2)));

list.add(new PetriSim(CreateTravel("bus A in Kyiv", 1, people)));

list.add(new PetriSim(CreateTravel("bus B in Kyiv", 1, people)));

list.add(new PetriSim(CreateTravel("bus A in Lviv", 1, people)));

list.add(new PetriSim(CreateTravel("bus B in Lviv", 1, people)));

list.get(2).getNet().getListT()[0].setPriority(1);

list.get(4).getNet().getListT()[0].setPriority(1);

list.get(0).getNet().getListP()[3] = list.get(2).getNet().getListP()[0];

list.get(0).getNet().getListP()[3] = list.get(3).getNet().getListP()[0];

list.get(0).getNet().getListP()[4] = list.get(1).getNet().getListP()[4];

list.get(1).getNet().getListP()[3] = list.get(4).getNet().getListP()[0];

list.get(1).getNet().getListP()[3] = list.get(5).getNet().getListP()[0];

list.get(2).getNet().getListP()[4] = list.get(4).getNet().getListP()[3];

list.get(2).getNet().getListP()[3] = list.get(4).getNet().getListP()[4];

list.get(3).getNet().getListP()[4] = list.get(5).getNet().getListP()[3];

list.get(3).getNet().getListP()[3] = list.get(5).getNet().getListP()[4];

list.get(3).getNet().getListP()[5] = list.get(2).getNet().getListP()[5];

list.get(4).getNet().getListP()[5] = list.get(2).getNet().getListP()[5];

list.get(5).getNet().getListP()[5] = list.get(2).getNet().getListP()[5];

PetriObjModel model = new PetriObjModel(list);

return model;

}

public static PetriNet CreateBusStop(String name, double mean, double deviation) throws ExceptionInvalidNetStructure, ExceptionInvalidTimeDelay {

ArrayList<PetriP> d\_P = new ArrayList<>();

ArrayList<PetriT> d\_T = new ArrayList<>();

ArrayList<ArcIn> d\_In = new ArrayList<>();

ArrayList<ArcOut> d\_Out = new ArrayList<>();

d\_P.add(new PetriP("P", 1));

d\_P.add(new PetriP("потенційні пасажири", 0));

d\_P.add(new PetriP("загальна кількість", 0));

d\_P.add(new PetriP("черга пасажирів", 0));

d\_P.add(new PetriP("втрачений прибуток", 0));

d\_T.add(new PetriT("надходження пасажирів", mean));

d\_T.get(0).setDistribution("unif", d\_T.get(0).getTimeServ());

d\_T.get(0).setParamDeviation(deviation);

d\_T.add(new PetriT("надходження у чергу", 0.0));

d\_T.add(new PetriT("покидання зупинки", 0.0));

d\_T.get(2).setPriority(1);

d\_In.add(new ArcIn(d\_P.get(0), d\_T.get(0), 1));

d\_In.add(new ArcIn(d\_P.get(1), d\_T.get(1), 1));

d\_In.add(new ArcIn(d\_P.get(1), d\_T.get(2), 1));

d\_In.add(new ArcIn(d\_P.get(3), d\_T.get(2), 30));

d\_In.get(3).setInf(true);

d\_Out.add(new ArcOut(d\_T.get(0), d\_P.get(0), 1));

d\_Out.add(new ArcOut(d\_T.get(0), d\_P.get(1), 1));

d\_Out.add(new ArcOut(d\_T.get(1), d\_P.get(3), 1));

d\_Out.add(new ArcOut(d\_T.get(2), d\_P.get(4), 20));

d\_Out.add(new ArcOut(d\_T.get(1), d\_P.get(2), 1));

PetriNet d\_Net = new PetriNet(name, d\_P, d\_T, d\_In, d\_Out);

PetriP.initNext();

PetriT.initNext();

ArcIn.initNext();

ArcOut.initNext();

return d\_Net;

}

public static PetriNet CreateTravel(String name, int buses, int people) throws ExceptionInvalidNetStructure, ExceptionInvalidTimeDelay {

ArrayList<PetriP> d\_P = new ArrayList<>();

ArrayList<PetriT> d\_T = new ArrayList<>();

ArrayList<ArcIn> d\_In = new ArrayList<>();

ArrayList<ArcOut> d\_Out = new ArrayList<>();

d\_P.add(new PetriP("черга пасажирів", 0));

d\_P.add(new PetriP("готовність відправитись", 0));

d\_P.add(new PetriP("вільний автобус", buses));

d\_P.add(new PetriP("автобус приїхав", 0));

d\_P.add(new PetriP("автобус в іншому місті", 0));

d\_P.add(new PetriP("отриманий прибуток", 0));

d\_T.add(new PetriT("посадка", 0.0));

d\_T.add(new PetriT("пареїзд", 20.0));

d\_T.get(1).setDistribution("unif", d\_T.get(1).getTimeServ());

d\_T.get(1).setParamDeviation(5.0);

d\_T.add(new PetriT("висадка", 5.0));

d\_T.get(2).setDistribution("unif", d\_T.get(2).getTimeServ());

d\_T.get(2).setParamDeviation(1.0);

d\_In.add(new ArcIn(d\_P.get(0), d\_T.get(0), people));

d\_In.add(new ArcIn(d\_P.get(1), d\_T.get(1), 1));

d\_In.add(new ArcIn(d\_P.get(2), d\_T.get(0), 1));

d\_In.add(new ArcIn(d\_P.get(3), d\_T.get(2), 1));

d\_Out.add(new ArcOut(d\_T.get(0), d\_P.get(1), 1));

d\_Out.add(new ArcOut(d\_T.get(1), d\_P.get(4), 1));

d\_Out.add(new ArcOut(d\_T.get(1), d\_P.get(5), 500));

d\_Out.add(new ArcOut(d\_T.get(2), d\_P.get(2), 1));

PetriNet d\_Net = new PetriNet(name, d\_P, d\_T, d\_In, d\_Out);

PetriP.initNext();

PetriT.initNext();

ArcIn.initNext();

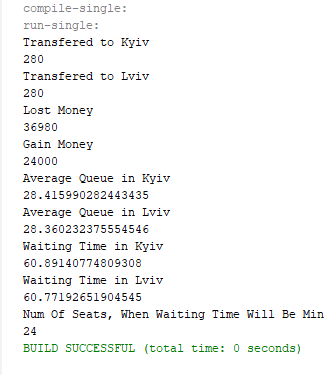
ArcOut.initNext();

return d\_Net;

}

}

**Результат виконання моделювання схеми:**

****

**Завдання 4 до практичної роботи :**

Побудувати математичні рівняння, що описують побудовану за текстом завдання 1 практикуму 5 Петрі-об’єктну модель.20 балів.

Вхідні дані:

𝐓 = {(T0), (T1) , (T2) , (T3) , (T4) , (T5) , (T6) , (T7) , (T8) , (T9) , (T10)}

**P** = {(P0, P1, P2), (P2, P3, P4), (P2, P5), (P5, P6, P7), (P5, P8), (P8, P9, P10), (P8, P11), (P11, P12, P13), (P13, P14), (P14, P15, P16), (P14, P2)}

**A** = {(P0, T0), (T0, P0), (T0, P1), (T0, P2), (P2, T1), (P2, T2), (T1, P3), (P3, T1), (T1, P4), (T2, P5), (P5, T3), (P5, T4), (T3, P6), (P6, T3), (T3, P7), (T4, P8), (P8, T5), (P8, T6), (T5, P9), (P9, T5), (T5, P10), (T6, P11), (P11, T7), (P11, T8), (T7, P12), (P12, T7), (T7, P13), (T8, P14), (P14, T9), (P14, T10), (T9, P15), (P15, T9), (T9, P16), (T10, P2)}

**K** = {(0, 1), (1, 1), (0, 1), (1, 1), (0, 1), (1, 1), (0, 1), (1, 1), (0, 1), (1, 1), (0, 1)}

**R** = {1, -1lnξ, 1, -1lnξ, 1, -1lnξ, 1, -1lnξ, 1, -1lnξ, 5}

**I** = ∅

**W** =

Початковий стан:

**Ітерація №1:**

|  |  |  |
| --- | --- | --- |
| MP0(0.0)≥4 | ⟹ | Z(T0, 0.0)=1 |
| MP2(0.0)<1, MP3(0.0)≥1 | ⟹ | Z(T1, 0.0)=0 |
| MP2(0.0)<1 | ⟹ | Z(T2, 0.0)=0 |
| MP5(0.0)<1, MP6(0.0)≥1 | ⟹ | Z(T3, 0.0)=0 |
| MP5(0.0)<1 | ⟹ | Z(T4, 0.0)=0 |
| MP8(0.0)<1, MP9(0.0)≥1 | ⟹ | Z(T5, 0.0)=0 |
| MP8(0.0)<1 | ⟹ | Z(T6, 0.0)=0 |
| MP11(0.0)<1, MP12(0.0)≥1 | ⟹ | Z(T7, 0.0)=0 |
| MP11(0.0)<1 | ⟹ | Z(T8, 0.0)=0 |
| MP14(0.0)<1, MP15(0.0)≥1 | ⟹ | Z(T9, 0.0)=0 |
| MP14(0.0)<1 | ⟹ | Z(T10, 0.0)=0 |

Ψ={T0} ⟹ X(T0) = 1

|  |  |  |
| --- | --- | --- |
|  |  |  |
| D-: |  |  |
| MP0(0.0) = 0 |  | ET0(0.0) = {0.0 + 1} |
|  | | |
|  |  |  |

t1 = min{1, ∞, ∞, ∞, ∞, ∞, ∞, ∞, ∞, ∞, ∞} = 1

|  |  |  |
| --- | --- | --- |
| Y(T0, 1) | = | 1 |
| D+:  MP0(1) = 4 |  | ET0(1) = {∞} |
| MP1(1) = 4 |  |  |
| MP2(1) = 4 |  |  |

**Ітерація №2:**

|  |  |  |
| --- | --- | --- |
| MP0(1)≥4 | ⟹ | Z(T0, 1)=1 |
| MP2(1)≥1, MP3(1)≥1 | ⟹ | Z(T1, 1)=1 |
| MP2(1)≥1 | ⟹ | Z(T2, 1)=1 |
| MP5(1)<1, MP6(1)≥1 | ⟹ | Z(T3, 1)=0 |
| MP5(1)<1 | ⟹ | Z(T4, 1)=0 |
| MP8(1)<1, MP9(1)≥1 | ⟹ | Z(T5, 1)=0 |
| MP8(1)<1 | ⟹ | Z(T6, 1)=0 |
| MP11(1)<1, MP12(1)≥1 | ⟹ | Z(T7, 1)=0 |
| MP11(1)<1 | ⟹ | Z(T8, 1)=0 |
| MP14(1)<1, MP15(1)≥1 | ⟹ | Z(T9, 1)=0 |
| MP14(1)<1 | ⟹ | Z(T10, 1)=0 |
|  |  |  |

Ψ = {T0, T1, T2}. P2 - конфліктна позиція, можливі варіанти T1 або T2. Згідно з приорітетами вибір пав на T1.

Ψ’ = {T0, T1} ⟹ X(T0) = 1, X(T1, 1) = 1

|  |  |  |
| --- | --- | --- |
| D-: |  |  |
| MP0(1) = 4 – (4\*1) = 0 |  | ET0(1) = {1 + 1} |
|  | | |
| MP2(1) = 4 – (1\*1 +1\*0) = 3 |  | ET1(1) = {1 + 1.2} |
| MP3(1) = 1 – 1\*1 = 0 |  |  |

t2 = min{2, 2.2, ∞, ∞, ∞, ∞, ∞, ∞, ∞, ∞, ∞} = 2

D+:

|  |  |  |
| --- | --- | --- |
| Y(T0, 2)=1 |  |  |
| MP0(2) = 0 + 4\*1 = 4 |  | ET0(1) = {∞} |
| MP1(0.5) = 4 + 4\*1 = 8 |  |  |
| MP2(0.5) = 3 + 4\*1 = 7 |  |  |

**Завдання 5 до практичної роботи:**

Переваги:

1. Гнучкість та розширюваність: Петрі-об'єктне моделювання дозволяє гнучко визначати об'єкти та їх взаємодію в системі, що дозволяє моделювати широкий спектр процесів. Легко розширюється для включення нових об'єктів або змін у вже існуючих.

2. Візуалізація та зрозумілість: Графічне відображення процесів у вигляді діаграм полегшує розуміння та взаємодію з учасниками проекту. Забезпечує чітке відображення порядку виконання подій, що полегшує аналіз та виявлення проблем.

3. Паралельність та конкурентність: Здатність виражати паралельність дій і конкурентність подій дозволяє ефективно моделювати складні системи з багатьма взаємодіючими елементами.

Недоліки:

1. Складність моделювання: Створення складних Петрі-об'єктних моделей може виявитися трудомістким завданням, особливо для великих і складних систем.

2. Неоднозначність: Деякі аспекти моделювання можуть бути неоднозначними, що може призвести до різних інтерпретацій та ускладнювати аналіз.

3. Велика кількість змінних: У складних системах може виникнути велика кількість об'єктів та зв'язків, що ускладнює аналіз та відслідковування всіх можливих станів системи.

4. Обмеженість для деяких задач: Деякі типи завдань можуть виявитися неефективними для моделювання за допомогою Петрі-об'єктного підходу, що може вимагати використання інших методів.

**ВИСНОВОК**

Петрі-об’єктні моделі є зручними в тому плані, що зі складною системою значно легше працювати, якщо розбити її на підсистеми, та до того ж, не треба дублювати код і легше тиражувати окремі підсистеми. Також виграє візуальне представлення – схема представляється компактніше та зрозуміліше. Але недоліком є те, що схема може не давати повного уявлення про систему і частину умов треба зберігати окремо.