НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО”

КАФЕДРА АВТОМАТИЗОВАНИХ СИСТЕМ ОБРОБКИ ІНФОРМАЦІЇ І УПРАВЛІННЯ

Комп’ютерний практикум № 2

з дисципліни

“Моделювання систем”

Виконала:

студентка групи ІС-71

Алпаєва Ю.С

Перевірила:

старший викладач

Новікова П.А.

Комп’ютерний практикум №2

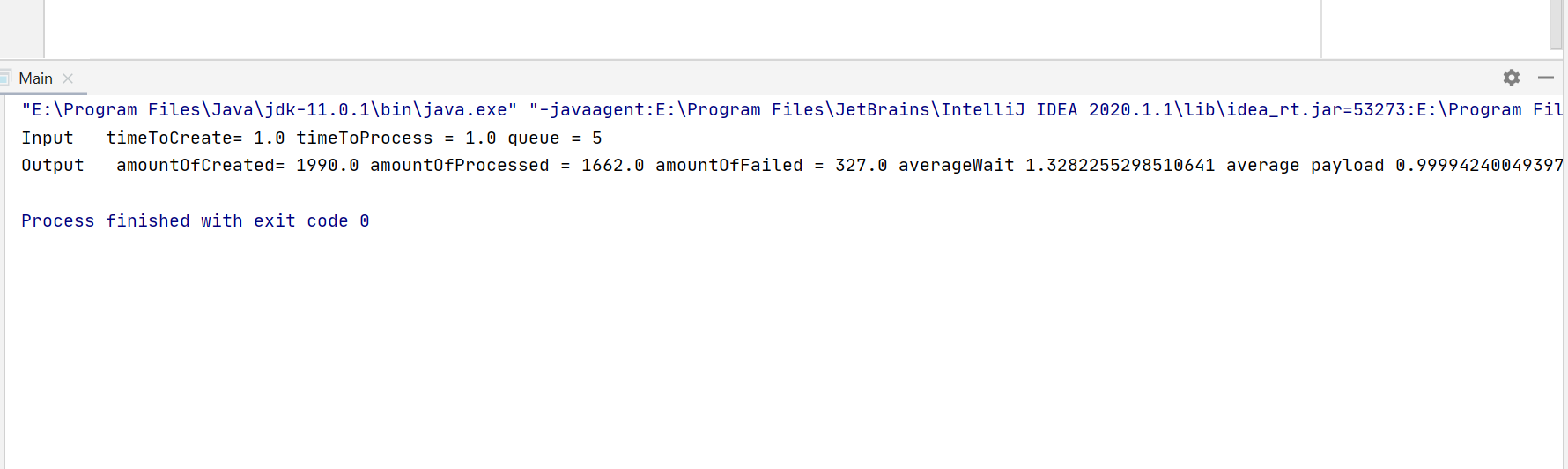
1. Реалізувати алгоритм імітації простої моделі обслуговування одним пристроєм (рис 2.1). **10 балів.**

Початкові параметри системи час створення = 1

Час обробки = 1

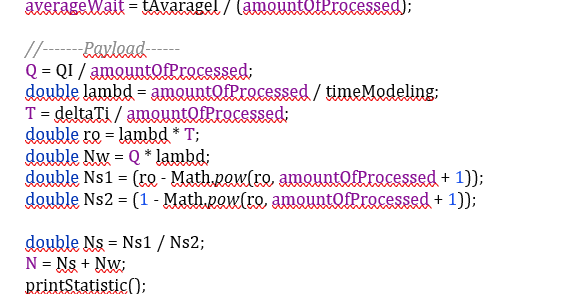
Максимальна черга =5

**Вивід:**

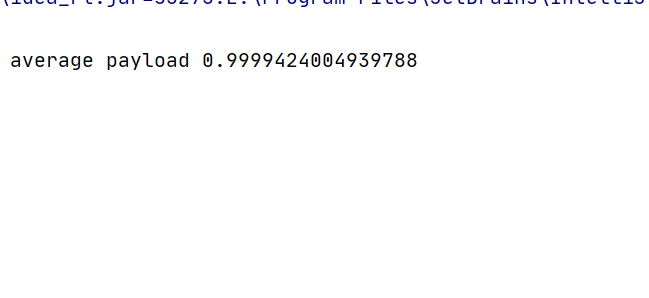


1. Модифікувати алгоритм, додавши обчислення середнього завантаження пристрою. **30 балів.**

**Алгоритм пошуку**



**Вивід**



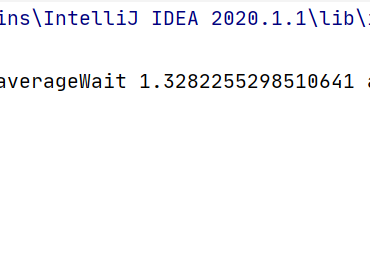
1. Модифікувати алгоритм, додавши обчислення середнього очікування в черзі. **30 балів.**

Алгоритм пошуку:

tAvarageI = tAvarageI + (queue \* (current - timeCurrentPref));

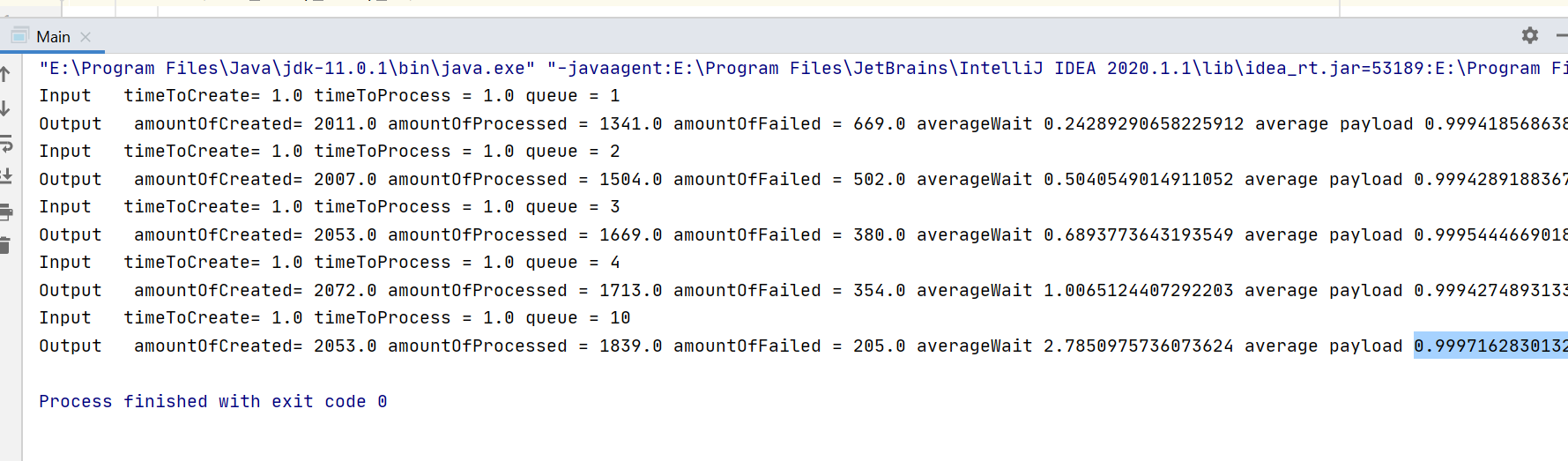
averageWait = tAvarageI / (amountOfProcessed);

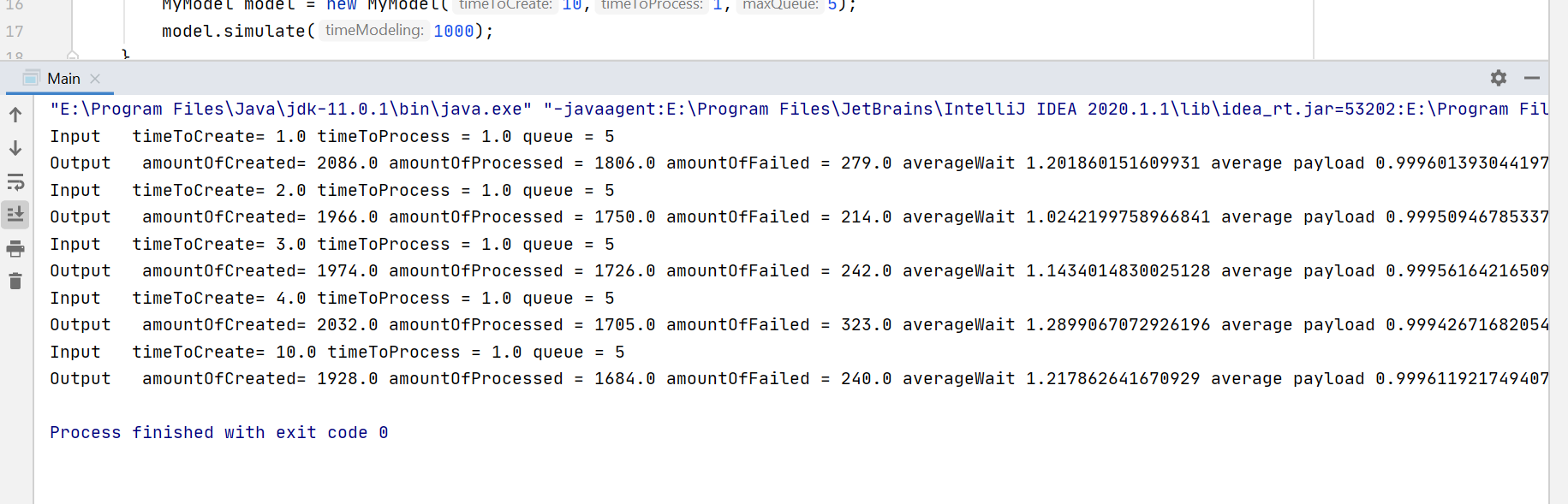
**Вивід**

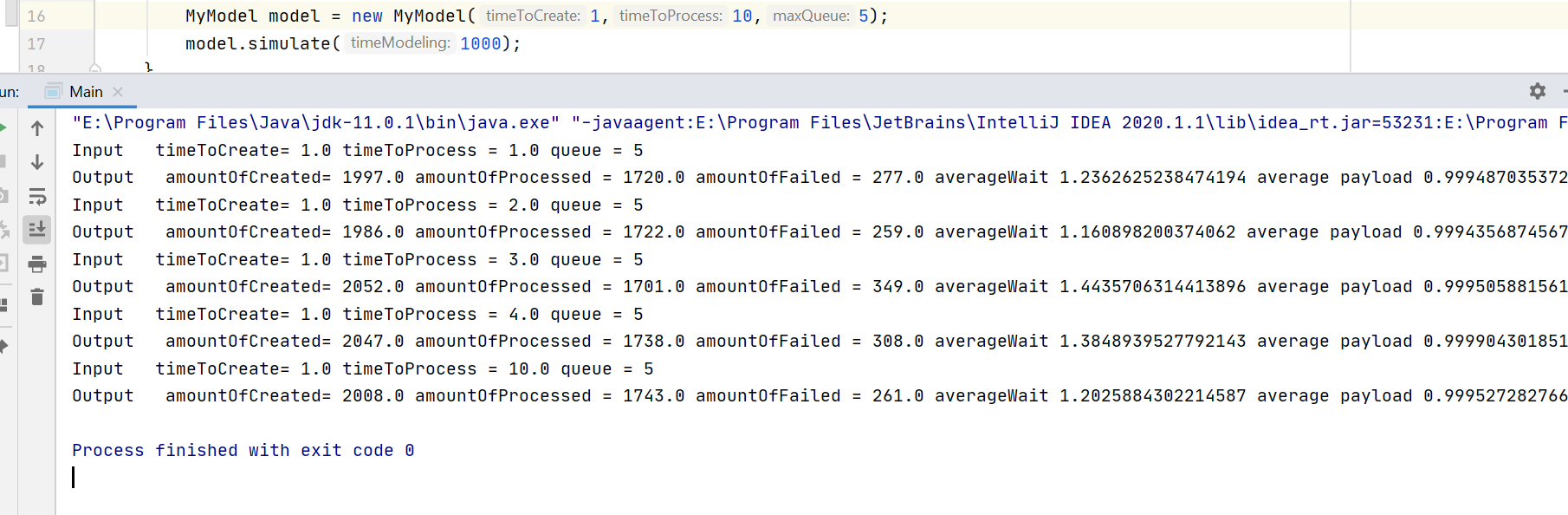


1. Виконати верифікацію моделі, змінюючи значення вхідних змінних та параметрів моделі. Навести результати верифікації у таблиці. **30 балів.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| iteration | timeToCreate | timeToProcess | maxQueue | amountOfCreated | amountOfProcessed | amountOfFailed | averageWait | payload |
| 1 | 1 | 1 | 1 | 2011 | 1341 | 669 | 0.24289 | 0.999418568638726 |
| 2 | 1 | 1 | 2 | 2007 | 1504 | 502 | 0.504 | 0.9994289188367934 |
| 3 | 1 | 1 | 3 | 2053 | 1669 | 380 | 0.689 | 0.9995444669018146 |
| 4 | 1 | 1 | 4 | 2072 | 1713 | 354 | 1.006 | 0.999427489313345 |
| 6 | 1 | 1 | 10 | 2053 | 1839 | 205 | 2.785 | 0.9997162830132811 |
| 7 | 2 | 1 | 5 | 1966 | 1750 | 214 | 1.024 | 0.9995094678533758 |
| 8 | 3 | 1 | 5 | 1974 | 1726 | 242 | 1.14 | 0.9995616421650941 |
| 9 | 4 | 1 | 5 | 2032 | 1705 | 323. | 1.28 | 0.9994267168205407 |
| 11 | 10 | 1 | 5 | 1928 | 1684 | 240 | 1.21 | 0.9996119217494079 |
| 12 | 1 | 2 | 5 | 1986 | 1722 | 259 | 1.160 | 0.9994356874567393 |
| 13 | 1 | 3 | 5 | 2052 | 1701 | 349 | 1.44 | 0.9995058815617088 |
| 14 | 1 | 4 | 5 | 2047 | 1738 | 308 | 1.38 | 0.9999043018514089 |
| 16 | 1 | 10 | 5 | 2008 | 1743 | 261 | 1.202 | 0.9995272827668971 |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |







**Висновок щодо верифікаціі**

Верифікація проводилась таким способом: по черзі запускалась програма з зміною одного параметра, на один, після встановлення параметру величиною 4, а потім збільшення проводилось до 10.

Висновки після проведення Верифікації системи

|  |  |  |
| --- | --- | --- |
| Параметр,що був змінений | Поведінка системи | Висновок |
| Max queue | При збільшенні параметра поступово зменшувалась кількість відмов, збільшилось навантаження. | Поведінка системи є очікуваною  Чим більше значення максимальної черги тим менша кількість відмов |
| Time To Create | При збільшенні параметра зростає середній час очікування, зменшується кількість оброблених заяв | Поведінка системи  є очікуваною  Збільшення пояснюється тим що час для створення стає більшим отже ,заявка створюється більше часу – зростає час очікування.  Зменшення кількості оброблених заяв зумовлено тим що час затримки стає більшим. |
| Time To Process | Зі збільшенням параметру зростає час очікування | Поведінка системи  є очікуваною  Збільшення часу очікування пояснюється тим що на обробку однієї заявки витрачається більше часу, отже затримка стає більшою, збільшується час очікування. |

**Висновок**

В ході данної лабораторної роботи було розроблено алгоритм симуляції та зроблениа верифікація моделі.

При зміні параметрів в усіх випадках поведінка системи є очікованою, отже робота алгоритму є коректною .

**Лістинг**

package com.company;  
  
public class MyModel {  
 private double next;  
 private double current;  
 private double t0;  
 private double t1;  
 private double timeToCreate;  
 private double timeToProcess;  
 private double amountOfCreated;  
 private double amountOfProcessed;  
 private double amountOfFailed;  
  
 private int stateNumber;  
 private int maxQueue;  
 private int queue;  
  
 private int nextEvent;  
 private double averageWait;  
 private double averageProcess;  
 private double tGenerateCur;  
 public double T;  
 public double N;  
 public double Q;  
  
 public MyModel(double timeToCreate, double timeToProcess, int maxQueue) {  
 this.timeToCreate = timeToCreate;  
 this.timeToProcess = timeToProcess;  
 this.maxQueue = maxQueue;  
 }  
  
 public void simulate(double timeModeling) {  
 double tAvarageI = 0;  
 double deltaTi = 0;  
 double QI = 0;  
 double timeCurrentPref = current;  
 while (current < timeModeling) {  
 timeCurrentPref = current;  
 next = t0;  
 nextEvent = 0;  
  
 if (t1 < next) {  
 next = t1;  
 nextEvent = 1;  
 }  
 current = next;  
 if(nextEvent == 1){  
 doEvent1();  
  
 }  
 if(nextEvent == 0){  
 doEvent0();  
  
 }  
 tAvarageI = tAvarageI + (queue \* (current - timeCurrentPref));  
 deltaTi = deltaTi + (current - timeCurrentPref);  
 QI = QI + (current - timeCurrentPref) \* queue;  
 ;  
  
 }  
 *//-------Time------* averageWait = tAvarageI / (amountOfProcessed);  
  
 *//-------Payload------* Q = QI / amountOfProcessed;  
 double lambd = amountOfProcessed / timeModeling;  
 T = deltaTi / amountOfProcessed;  
 double ro = lambd \* T;  
 double Nw = Q \* lambd;  
 double Ns1 = (ro - Math.*pow*(ro, amountOfProcessed + 1));  
 double Ns2 = (1 - Math.*pow*(ro, amountOfProcessed + 1));  
  
 double Ns = Ns1 / Ns2;  
 N = Ns + Nw;  
 printStatistic();  
 }  
  
 private void doEvent0() {  
 *//delay* t0 = current + TimeGenerator.*generate*();  
 amountOfCreated++;  
 if (stateNumber == 0) {  
 stateNumber = 1;  
 *//process* t1 = current + TimeGenerator.*generate*();  
 } else {  
 if (queue < maxQueue)  
 queue++;  
 else  
 amountOfFailed++;  
 }  
  
  
 }  
  
 private void doEvent1() {  
 t1 = Double.*MAX\_VALUE*;  
 stateNumber = 0;  
 if (queue > 0) {  
 queue--;  
 stateNumber = 1;  
 t1 = current + TimeGenerator.*generate*();  
 }  
 amountOfProcessed++;  
  
 }  
  
public void printStatistic() {  
 System.*out*.println("Input "+" timeToCreate= " + timeToCreate + " timeToProcess = " + timeToProcess + " max queue = "  
 + maxQueue);  
 System.*out*.println("Output "+" amountOfCreated= " + amountOfCreated + " amountOfProcessed = " + amountOfProcessed + " amountOfFailed = "  
 + amountOfFailed + " averageWait " + averageWait + " average payload " + N);  
}

public void printInfo() {  
 System.*out*.println(" current = " + current + " stateNumber = " + stateNumber + " queue = " + queue);  
 }  
  
}

package com.company;  
public class TimeGenerator {  
 public static double generate() {  
 double n = 10000.0;  
 double mPrf = 0.4987551736746871;  
 double a = Math.*random*();  
 return *formula*(a, 1 / mPrf);  
 }  
  
 private static double formula(double a, double lambda) {  
 return ((-1.0 / lambda) \* Math.*log*(a));  
 }  
}

package com.company;  
  
import java.util.Random;  
  
package com.company;  
  
import java.util.Random;  
  
public class Main {  
  
 public static void main(String[] args) {  
MyModel model = new MyModel(1, 1, 5);  
 model.simulate(1000);  
 }  
}

}

}