НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №1

з дисципліни **«**Комп’ютерне моделювання**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІО-31

Долинний Олександр

Перевірив:

Марковський О.П.

Київ – 2015 р.

**Варіант завдання:**

Типи СМО: FIFO, RR

Г(2, 0.7)

K1=-5

K2=-1

K3=-1

T1=∞

T2=0

**Лістинг програми**

**package** km.lab1;  
  
  
**public class** Task {  
 **int taskNumber**;  
 **double entranceTime**;  
 **double leftTime**;  
  
 **public int** getTaskNumber() {  
 **return taskNumber**;  
 }  
  
 **public** Task(**int** taskNumber, **double** entranceTime, **double** leftTime) {  
 **this**.**taskNumber** = taskNumber;  
 **this**.**entranceTime** = entranceTime;  
 **this**.**leftTime** = leftTime;  
 }  
  
 **public void** setTaskNumber(**int** taskNumber) {  
 **this**.**taskNumber** = taskNumber;  
 }  
  
 **public double** getEntranceTime() {  
 **return entranceTime**;  
 }  
  
 **public void** setEntranceTime(**double** entranceTime) {  
 **this**.**entranceTime** = entranceTime;  
 }  
  
 **public double** getLeftTime() {  
 **return leftTime**;  
 }  
  
 **public void** setLeftTime(**double** leftTime) {  
 **this**.**leftTime** = leftTime;  
 }  
}

**package** km.lab1;  
  
**import** java.util.LinkedList;  
  
*/\*\*  
 \* Created by alex\_\_000 on 13.09.2015.  
 \*/***public class** Queue {  
 **double modelTime**;  
 **double loadTime**;  
 **int lastAddedTask**;  
 **int counter**;  
 LinkedList <Task> **tasks**;  
  
 Queue(){  
 **tasks** = **new** LinkedList <Task>();  
 **counter** = 0;  
 }  
 **public double** getModelTime() {  
 **return modelTime**;  
 }  
  
 **public int** size(){  
 **return tasks**.size();  
 }  
 **public void** setModelTime(**double** modelTime) {  
 **this**.**modelTime** = modelTime;  
 }  
  
 **public double** getLoadTime() {  
 **return loadTime**;  
 }  
  
 **public void** setLoadTime(**double** loadTime) {  
 **this**.**loadTime** = loadTime;  
 }  
  
 **public int** getLastAddedTask() {  
 **return lastAddedTask**;  
 }  
  
 **public void** incLastAddedTask() {  
 **lastAddedTask**++;  
 }  
  
  
 **public void** setLastAddedTask(**int** p) {  
 **lastAddedTask** = p;  
 }  
  
 **public int** getCounter() {  
 **return counter**;  
 }  
  
 **public void** incCounter() {  
 **counter**++;  
 }  
  
 **public void** addTask(Task task) {  
 **tasks**.add(task);  
 }  
  
 **public void** deleteFirstTask() {  
 **tasks**.remove(0);  
 }  
  
 **public** Task getFirstTask() {  
 **return tasks**.get(0);  
 }  
  
 **public** Task get(**int** i) {  
 **return tasks**.get(i);  
 }  
  
}

**package** km.lab1;  
  
**import** java.util.LinkedList;  
  
**public class** Handler {  
  
Generator **generator**;  
 **double averageTimeFIFO**;  
 **double dispersionFIFO**;  
 **double relaxationTimeFIFO**;  
 **double ratioFIFO**;  
 **double currencyFIFO**;  
 **double averageTimeRR**;  
 **double dispersionRR**;  
 **double relaxationTimeRR**;  
 **double ratioRR**;  
 **double currencyRR**;  
 **double functionFIFO**;  
 **double functionRR**;  
  
Handler(**double** a0, **double** a1, **double** a2, **double** a3, **double** a4){  
 **generator** = **new** Generator(a0, a1, a2, a3, a4);  
}  
 **void** generateTime(**double** genTime) {  
 **generator**.generateTime(genTime);  
  
 }  
  
 **void** generateSteps(**int** count) {  
 **generator**.generateSteps(count);  
  
 }  
  
  
 **void** hand(){  
 LinkedList<**double**[]> tasks = **generator**.getTasks();  
 **for**(**int** i=0;i<tasks.size();i++){  
 **averageTimeFIFO**+=(tasks.get(i)[4]-tasks.get(i)[0]);  
 **relaxationTimeFIFO**+=(tasks.get(i)[3]-tasks.get(i)[0]);  
 **ratioFIFO**+=tasks.get(i)[2];  
 **averageTimeRR**+=(tasks.get(i)[7]-tasks.get(i)[0]);  
 **relaxationTimeRR**+=(tasks.get(i)[6]-tasks.get(i)[0]);  
 **ratioRR**+=tasks.get(i)[5];  
 }  
  
 **averageTimeFIFO**=**averageTimeFIFO**/tasks.size();  
 **relaxationTimeFIFO**=**relaxationTimeFIFO**/tasks.size();  
 **ratioFIFO**=**ratioFIFO**/tasks.size();  
 **currencyFIFO**=1;  
 **averageTimeRR**=**averageTimeRR**/tasks.size();  
 **relaxationTimeRR**=**relaxationTimeRR**/tasks.size();  
 **ratioRR**=**ratioRR**/tasks.size();  
 **currencyRR**=1;  
 **for**(**int** i=0;i<tasks.size();i++){  
 **dispersionFIFO**+=(tasks.get(i)[4]-tasks.get(i)[0]-**averageTimeFIFO**)\*(tasks.get(i)[4]-tasks.get(i)[0]-**averageTimeFIFO**);  
 **dispersionRR**+=(tasks.get(i)[7]-tasks.get(i)[0]-**averageTimeRR**)\*(tasks.get(i)[7]-tasks.get(i)[0]-**averageTimeRR**);  
 }  
 **dispersionFIFO**=**dispersionFIFO**/(tasks.size()-1);  
 **dispersionRR**=**dispersionRR**/(tasks.size()-1);  
 **dispersionFIFO**-=**averageTimeFIFO**\***averageTimeFIFO**;  
 **dispersionRR**-=**averageTimeRR**\***averageTimeRR**;  
 **dispersionFIFO**=Math.*abs*(**dispersionFIFO**);  
 **dispersionRR**=Math.*abs*(**dispersionRR**);  
 **functionFIFO**=5\***averageTimeFIFO**+(-1)\***dispersionFIFO**+(-1)\***relaxationTimeFIFO**;  
 **functionRR**=5\***averageTimeRR**+(-1)\***dispersionRR**+(-1)\***relaxationTimeRR**;  
 System.***out***.println(**"FIFO:"**);  
 System.***out***.println(**"averageTimeFIFO: "** + **averageTimeFIFO**);  
 System.***out***.println(**"dispersionFIFO: "** + **dispersionFIFO**);  
 System.***out***.println(**"reactionTimeFIFO: "** + **relaxationTimeFIFO**);  
 System.***out***.println(**"ratioFIFO: "** + **ratioFIFO**);  
 System.***out***.println(**"currencyFIFO: "** + **currencyFIFO**);  
 System.***out***.println(**"functionFIFO: "** + **functionFIFO**);  
  
 System.***out***.println(**" "**);  
 System.***out***.println(**"RR:"**);  
 System.***out***.println(**"averageTimeRR: "** + **averageTimeRR**);  
 System.***out***.println(**"dispersionRR: "** + **dispersionRR**);  
 System.***out***.println(**"reactionTimeRR: "** + **relaxationTimeRR**);  
 System.***out***.println(**"ratioRR: "** + **ratioRR**);  
 System.***out***.println(**"currencyRR: "** + **currencyRR**);  
 System.***out***.println(**"functionRR: "** + **functionRR**);  
  
 }  
  
  
 **void** h() {  
 System.***out***.println(**"currencyRR: "**);  
 }  
}

**package** km.lab1;  
  
**import** java.util.LinkedList;  
**import** java.util.Random;  
  
**public class** Generator {  
 LinkedList<**double**[]> **tasks**;  
 Queue **RR**;  
 Queue **FIFO**;  
 **double**[] **constant**; *// lambda, mu1, mu2, alfa1,quantum* LinkedList<**double**[]> getTasks() {  
 **return tasks**;  
 }  
  
  
 **void** generateTime(**double** genTime) {  
  
 **double** time = 0;  
 **FIFO** = **new** Queue();  
 **RR** = **new** Queue();  
 **while** (time < genTime) {  
 **double**[] result = **new double**[8];  
 time += generateEvent(**constant**[0]);  
 result[0] = time;  
 Random rand = **new** Random();  
 **double** temp = rand.nextDouble();  
 **if** (temp < **constant**[3])  
 result[1] = generateEvent(**constant**[1]);  
 **else** result[1] = generateEvent(**constant**[2]);  
 **tasks**.add(result);  
 }  
  
  
 **while** (genTime > **FIFO**.getModelTime()) {  
 generateStepFIFO();  
 }  
*//  
// while (genTime > RR.getModelTime()) {  
// generateStepRR();  
// }* }  
  
  
 **void** generateSteps(**int** count) {  
 *//System.out.println("dd2");* **FIFO** = **new** Queue();  
 **RR** = **new** Queue();  
 **double** time = 0;  
 **for** (**int** i = 0; i < count; i++) {  
 **double**[] result = **new double**[8];  
 result[0] = time;  
 time += generateEvent(**constant**[0]);  
 Random rand = **new** Random();  
 **double** temp = rand.nextDouble();  
 **if** (temp < **constant**[3])  
 result[1] = generateEvent(**constant**[1]);  
 **else** result[1] = generateEvent(**constant**[2]);  
 *//System.out.println((tasks.size()) + " entryTime=" + result[0] + " solutionTime=" + result[1] /\*+ " time=" + time\*/);* **tasks**.add(result);  
  
 }  
  
  
 **for** (; **FIFO**.getCounter() < (count); ) {  
 generateStepFIFO();  
 }  
  
 **for** (; **RR**.getCounter() < (count); ) {  
 generateStepRR();  
 }  
  
  
 }  
  
  
 **void** generateStepFIFO() {  
 **if** (**FIFO**.getModelTime() != 0) {  
  
 **if** ((**FIFO**.size() == 0) && (**FIFO**.getLastAddedTask() != (**tasks**.size() - 1))) {  
 **int** indexTask = **FIFO**.getLastAddedTask() + 1;  
 **FIFO**.addTask(**new** Task(indexTask, **tasks**.get(indexTask)[0], **tasks**.get(indexTask)[1]));  
 **if** (**FIFO**.getModelTime() < **tasks**.get(indexTask)[0]) {  
 **FIFO**.setModelTime(**tasks**.get(indexTask)[0]);  
 }  
 *//System.out.println(" ");  
 //System.out.println("Add task: N=" + indexTask + "entranceTime=" + tasks.get(indexTask)[0] + "leftTime=" + tasks.get(indexTask)[1]);  
 //System.out.println("modelingTime=" + FIFO.getModelTime());* **FIFO**.incLastAddedTask();  
 } **else** {  
  
 **int** index = **FIFO**.getFirstTask().getTaskNumber();  
 **double**[] temp = **tasks**.get(index);  
 temp[2] = 1;  
 temp[3] = **FIFO**.getModelTime();  
 **FIFO**.setModelTime(**FIFO**.getModelTime() + **FIFO**.getFirstTask().getLeftTime());  
 temp[4] = **FIFO**.getModelTime();  
 *//System.out.println(" ");  
 //System.out.println("Delete task: N=" + index + " firstEntryTime=" + temp[3] + " solutionTime=" + temp[1] + " outTime=" + temp[4]);  
 //System.out.println("modelingTime=" + FIFO.getModelTime());* **tasks**.set(index, temp);  
 **FIFO**.deleteFirstTask();  
 **FIFO**.incCounter();  
  
 **while** ((**FIFO**.getLastAddedTask() != (**tasks**.size() - 1) && (**tasks**.get(**FIFO**.getLastAddedTask() + 1)[0] < **FIFO**.getModelTime()))) {  
 **int** indexTask = **FIFO**.getLastAddedTask() + 1;  
 **FIFO**.addTask(**new** Task(indexTask, **tasks**.get(indexTask)[0], **tasks**.get(indexTask)[1]));  
  
 **FIFO**.incLastAddedTask();  
 *//System.out.println("Add task: N=" + indexTask + "entranceTime=" + tasks.get(indexTask)[0] + "leftTime=" + tasks.get(indexTask)[1]);  
 //System.out.println("modelingTime=" + FIFO.getModelTime());* }  
  
 }  
  
  
 } **else** {  
 **FIFO**.addTask(**new** Task(0, **tasks**.get(0)[0], **tasks**.get(0)[1]));  
 **FIFO**.setModelTime(0.00000000001);  
 **FIFO**.setLastAddedTask(0);  
 *//System.out.println(" ");  
 // System.out.println("Add task: N=" + 0 + "entranceTime=" + tasks.get(0)[0] + "leftTime=" + tasks.get(0)[0]);  
 // System.out.println("modelingTime=" + FIFO.getModelTime());* }  
 }  
  
  
 **void** generateStepRR() {  
 **if** (**RR**.getModelTime() != 0) {  
  
 **if** ((**RR**.size() == 0) && (**RR**.getLastAddedTask() != (**tasks**.size() - 1))) {  
 **int** indexTask = **RR**.getLastAddedTask() + 1;  
 **RR**.addTask(**new** Task(indexTask, **tasks**.get(indexTask)[0], **tasks**.get(indexTask)[1]));  
 **if** (**RR**.getModelTime() < **tasks**.get(indexTask)[0]) {  
 **RR**.setModelTime(**tasks**.get(indexTask)[0]);  
 }  
 *//System.out.println(" ");  
 // System.out.println("Add task: N=" + indexTask + "entranceTime=" + tasks.get(indexTask)[0] + "leftTime=" + tasks.get(indexTask)[1]);  
 // System.out.println("modelingTime=" + RR.getModelTime());* **RR**.incLastAddedTask();  
 } **else** {  
  
 **if** (**RR**.size() != 0) {  
 **if** (**RR**.getFirstTask().getLeftTime() < **constant**[4]) {  
 **int** index = **RR**.getFirstTask().getTaskNumber();  
 **double**[] temp = **tasks**.get(index);  
 temp[5] = 1;  
 **if** (temp[6] == 0) {  
 temp[6] = **RR**.getModelTime();  
 }  
 **RR**.setModelTime(**RR**.getModelTime() + **RR**.getFirstTask().getLeftTime());  
 temp[7] = **RR**.getModelTime();  
 **tasks**.set(index, temp);  
 *//System.out.println(" ");  
 // System.out.println("Delete task: N=" + index + " firstEntryTime=" + temp[6] + " solutionTime=" + temp[1] + " outTime=" + temp[7]);  
 // System.out.println("modelingTime=" + RR.getModelTime());* **RR**.deleteFirstTask();  
 **RR**.incCounter();  
 } **else** {  
 **int** index = **RR**.getFirstTask().getTaskNumber();  
 **double**[] temp = **tasks**.get(index);  
 **if** (temp[6] == 0) {  
 temp[6] = **RR**.getModelTime();  
 **tasks**.set(index, temp);  
 }  
 **RR**.setModelTime(**RR**.getModelTime() + **constant**[4]);  
 **RR**.addTask(**new** Task(index, **RR**.getModelTime(), **RR**.getFirstTask().getLeftTime() - **constant**[4]));  
 System.***out***.println(**" "**);  
 *// System.out.println("Add task: N=" + index + "entryTime=" + RR.getModelTime() + "leftTime=" + (RR.getFirstTask().getLeftTime() - constant[4]));  
 // System.out.println("modelingTime=" + RR.getModelTime());* **RR**.deleteFirstTask();  
 }  
 }  
  
  
 **while** ((**RR**.getLastAddedTask() != (**tasks**.size() - 1) && (**tasks**.get(**RR**.getLastAddedTask() + 1)[0] < **RR**.getModelTime()))) {  
 **int** indexTask = **RR**.getLastAddedTask() + 1;  
 **RR**.addTask(**new** Task(indexTask, **tasks**.get(indexTask)[0], **tasks**.get(indexTask)[1]));  
 *//FIFO.setModelTime(tasks.get(indexTask)[1]);* **RR**.incLastAddedTask();  
 *// System.out.println("Add task: N=" + indexTask + "entranceTime=" + tasks.get(indexTask)[0] + "leftTime=" + tasks.get(indexTask)[1]);  
 // System.out.println("modelingTime=" + RR.getModelTime());* }  
  
 }  
  
 } **else** {  
 **RR**.addTask(**new** Task(0, **tasks**.get(0)[0], **tasks**.get(0)[1]));  
 **RR**.setModelTime(0.00000000001);  
 **RR**.setLastAddedTask(0);  
 System.***out***.println(**" "**);  
 *// System.out.println("Add task: N=" + 0 + "entranceTime=" + tasks.get(0)[0] + "leftTime=" + tasks.get(0)[0]);  
 //System.out.println("modelingTime=" + RR.getModelTime());* }  
  
 }  
  
 Generator(**double** a0, **double** a1, **double** a2, **double** a3, **double** a4) {  
 **tasks** = **new** LinkedList<**double**[]>();  
 **constant** = **new double**[5];  
 **constant**[0] = a0;  
 **constant**[1] = a1;  
 **constant**[2] = a2;  
 **constant**[3] = a3;  
 **constant**[4] = a4;  
 }  
  
 **double** generateEvent(**double** inp) {  
 Random rand = **new** Random();  
 **double** result = -1.0 / inp \* Math.*log*(rand.nextDouble());  
 **return** result;  
 }  
}