НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №2

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

Виконав:

студент 3 курсу

ФІОТ гр. ІО-31

Долинний Олександр

Перевірив:

Марковський О.П.

Київ – 2015 р.

**Лістинг програми**

**Model.java**

**package** km.lab2;  
  
*/\*  
Computer modelling.  
Lab 2  
Dolinniy Alexandr  
IO-31  
01.11.2015  
\*/*

**import** java.util.LinkedList;  
**import** java.util.Random;  
  
**public class** Model {  
 **private** LinkedList<Task> **tasks**;  
 **private** LinkedList<Block> **blocks**;  
 **private double**[][] **matrixCrossing** = {{0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},*// ЦП* {0.3, 0.3, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},*// Кеш* {0.0, 0.1, 0.1, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},*// ПнМ* {0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},*// ОП* {0.0, 0.0, 0.3, 0.3, 0.3, 0.4, 0.5, 0.65, 0.8, 1.0, 1.0, 1.0},*// ПдМ* {1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},*// АУ* {0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},*// КДП* {0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},*// СА* {0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},*// КОП* {0.0, 0.0, 0.0, 0.0, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 1.0},*// PSI* {1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},*// LPT* {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0},*// USB  
 // ЦП, Кеш, ПнМ, ОП, ПдМ, АУ, КДП, СА, КОП, PSI, LPT, USB* };  
  
 **private** String[] **str** = {**"ЦП"**, **"Кеш"**, **"ПнМіст"**, **"ОП"**, **"ПдМіст"**, **"АУ"**, **"КДП"**, **"СА"**, **"КОП"**, **"PSI"**, **"LPT"**, **"USB"**};  
**private double**[] **arrayTime** = {0.01, 10, 1, 0.1, 0.1, 0.00316, 0.0316, 0.1, 0.1, 0.0316, 0.001, 0.01};  
**private int**[] **startState** =  
 {0, 0, 0, 0, 0, 0};  
 **private int**[] **arrayRestriction** =  
 {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1};  
  
 **private double modelTime**;  
 **private int stepNumber** = 0;  
 **private double**[] **stat**;  
 LinkedList[] **timeDevice**;  
  
 Model() {  
 **tasks** = **new** LinkedList<Task>();  
 **blocks** = **new** LinkedList<Block>();  
 **stat** = **new double**[**matrixCrossing**.**length**];  
 **timeDevice** = **new** LinkedList[**matrixCrossing**.**length**];  
  
 **for** (**int** i = 0; i < **matrixCrossing**.**length**; i++)  
 **timeDevice**[i] = **new** LinkedList<Block>();  
**for** (**int** i = 0; i < **startState**.**length**; i++)  
 **tasks**.add(**new** Task(i, **startState**[i], nextDevice(**startState**[i]), 0, genTimeSol(**startState**[i])));sortTasks();}  
  
 **void** sortTasks() {  
 **if** (**tasks**.size() > 1) {  
 **for** (**int** i = 0; i < (**tasks**.size() - 1); i++) {  
 **for** (**int** j = i + 1; j < (**tasks**.size()); j++) {  
 **if** (**tasks**.get(i).getTimeEntrance() > **tasks**.get(j).getTimeEntrance()) {  
 Task buf = **tasks**.get(i);  
 **tasks**.set(i, **tasks**.get(j));  
 **tasks**.set(j, buf);  
 }  
 }  
 }  
 }  
 }  
  
 **void** tasksString() {  
 **for** (**int** i = 0; i < **tasks**.size(); i++)  
 System.***out***.println(**tasks**.get(i).toString());  
 }  
  
 **int** nextDevice(**int** num) {  
 **int** result = 0;  
 Random rand = **new** Random();  
 **double** temp = rand.nextDouble();  
 **for** (result = 0; (result < **matrixCrossing**.**length**) && (temp > **matrixCrossing**[num][result]); result++) {  
 }  
 **return** result;  
 }  
  
 **double** genTimeSol(**int** num) {  
 **double** inp = **arrayTime**[num];  
 Random rand = **new** Random();  
 **double** result = -1.0 / inp \* Math.*log*(rand.nextDouble());  
 **return** result;  
 }  
  
  
 **void** checkingBlock() {  
 **for** (**int** i = **blocks**.size() - 1; i >= 0; i--) {  
 **if** (**tasks**.get(0).getTimeEntrance() > **blocks**.get(i).getTimeOut())  
 **blocks**.remove(i);  
 }  
 }  
  
 **void** sortBlocks(**int** n) {  
 **if** (**timeDevice**[n].size() > 1) {  
 **for** (**int** i = 0; i < (**timeDevice**[n].size() - 1); i++) {  
 **for** (**int** j = i + 1; j < (**timeDevice**[n].size()); j++) {  
 **if** (((Block) **timeDevice**[n].get(i)).getTimeIn() > ((Block) **timeDevice**[n].get(j)).getTimeIn()) {  
 Block buf = (Block) **timeDevice**[n].get(i);  
 **timeDevice**[n].set(i, **timeDevice**[n].get(j));  
 **timeDevice**[n].set(j, buf);  
 }  
 }  
 }  
 }  
  
 **for** (**int** j = **timeDevice**[n].size() - 1; j > 0; j--) {  
 **if** (((Block) **timeDevice**[n].get(j)).getTimeIn() < ((Block) **timeDevice**[n].get(j - 1)).getTimeOut()) {  
 **timeDevice**[n].set(j - 1, **new** Block(((Block) **timeDevice**[n].get(j - 1)).getTimeIn(), ((Block) **timeDevice**[n].get(j)).getTimeOut(), n));  
 **timeDevice**[n].remove(j);  
 }  
 }  
  
 **for** (**int** j = **timeDevice**[n].size() - 1; j > 0; j--) {  
 **if** (((Block) **timeDevice**[n].get(j)).getTimeOut() < **modelTime**) {  
 **stat**[n] += (((Block) **timeDevice**[n].get(j)).getTimeOut() - ((Block) **timeDevice**[n].get(j)).getTimeIn());  
 **timeDevice**[n].remove(j);  
 }  
 }  
 }  
  
 **void** generateStep() {  
 checkingBlock();  
 **int** restriction = **arrayRestriction**[**tasks**.get(0).getNumberDevice()];  
 **int** restrCounter = 0;  
 **double** timeEntrance = **tasks**.get(0).getTimeEntrance();  
 **for** (**int** i = 0; i < **blocks**.size(); i++) {  
 **if** ((**tasks**.get(0).getNumberDevice() == **blocks**.get(i).getNumberDevice()) &&  
 (timeEntrance <= **blocks**.get(i).getTimeOut()) &&  
 (timeEntrance >= **blocks**.get(i).getTimeIn()))  
 restrCounter++;  
 }  
 **if** (restriction > restrCounter) {  
 **if** (**tasks**.get(0).getTimeEntrance() < **modelTime**) {  
 **blocks**.add(**new** Block(**tasks**.get(0).getTimeEntrance(),  
 **tasks**.get(0).getTimeEntrance() + **tasks**.get(0).getTimeSolution(),  
 **tasks**.get(0).getNumberDevice()));  
  
 **timeDevice**[**tasks**.get(0).getNumberDevice()].add(**new** Block(**tasks**.get(0).getTimeEntrance(),  
 **tasks**.get(0).getTimeEntrance() + **tasks**.get(0).getTimeSolution(),  
 **tasks**.get(0).getNumberDevice()));  
 sortBlocks(**tasks**.get(0).getNumberDevice());  
Task buf = **tasks**.get(0);  
 buf.setNumberDevice(buf.getNumberNextDevice());  
 buf.setTimeEntrance(**tasks**.get(0).getTimeEntrance() + **tasks**.get(0).getTimeSolution() + Math.*pow*(10, -12));  
 buf.setNumberNextDevice(nextDevice(buf.getNumberDevice()));  
 buf.setTimeSolution(genTimeSol(buf.getNumberDevice()));  
 **tasks**.add(buf);  
 **tasks**.remove(0);  
 sortTasks();  
 } **else** {  
 **blocks**.add(**new** Block(**tasks**.get(0).getTimeEntrance(),  
 **tasks**.get(0).getTimeEntrance() + **tasks**.get(0).getTimeSolution(),  
 **tasks**.get(0).getNumberDevice()));  
 **timeDevice**[**tasks**.get(0).getNumberDevice()].add(**new** Block(**tasks**.get(0).getTimeEntrance(),  
 **tasks**.get(0).getTimeEntrance() + **tasks**.get(0).getTimeSolution(),  
 **tasks**.get(0).getNumberDevice()));**modelTime** = **tasks**.get(0).getTimeEntrance() + **tasks**.get(0).getTimeSolution();  
  
 Task buf = **tasks**.get(0);  
 buf.setNumberDevice(buf.getNumberNextDevice());  
 buf.setTimeEntrance(**modelTime** + Math.*pow*(10, -12));  
 buf.setNumberNextDevice(nextDevice(buf.getNumberDevice()));  
 buf.setTimeSolution(genTimeSol(buf.getNumberDevice()));  
 **tasks**.add(buf);  
 **tasks**.remove(0);  
 sortTasks();  
 }  
 } **else** {  
  
 LinkedList<Block> tempBlocks = **new** LinkedList<Block>(); **for** (**int** i = 0; i < **blocks**.size(); i++) {  
 **if** ((**tasks**.get(0).getNumberDevice() == **blocks**.get(i).getNumberDevice()) &&  
 (**tasks**.get(0).getTimeEntrance() <= **blocks**.get(i).getTimeOut()) &&  
 (**tasks**.get(0).getTimeEntrance() >= **blocks**.get(i).getTimeIn())) {  
 tempBlocks.add(**new** Block(**blocks**.get(i).getTimeIn(), **blocks**.get(i).getTimeOut(), **blocks**.get(i).getNumberDevice()));  
 }  
 }  
  
 **for** (**int** i = 0; i < (tempBlocks.size() - 1); i++) {  
 **for** (**int** j = i + 1; j < (tempBlocks.size()); j++) {  
 **if** (tempBlocks.get(i).getTimeIn() > tempBlocks.get(j).getTimeIn()) {  
 Block temp = tempBlocks.get(i);  
 tempBlocks.set(i, tempBlocks.get(j));  
 tempBlocks.set(j, temp);  
 }  
 }  
 }  
  
 **double** tempTime = tempBlocks.get(0).getTimeOut();  
 **for** (**int** i = 1; i < restriction; i++) {  
 **if** (tempTime > tempBlocks.get(i).getTimeOut())  
 tempTime = tempBlocks.get(i).getTimeOut();  
 }  
  
 Task buf = **tasks**.get(0);  
 buf.setTimeEntrance(tempTime + Math.*pow*(10, -12));  
 **tasks**.add(buf);**tasks**.remove(0);  
 sortTasks();  
 }}  
  
 **void** generateTime(**double** inp) {  
 **while** (inp > **tasks**.get(0).getTimeEntrance())  
 generateStep();  
 **for** (**int** i = 0; i < **matrixCrossing**.**length**; i++) {  
  
 **for** (**int** j = **timeDevice**[i].size() - 1; j > 0; j--) {  
 **if** (((Block) **timeDevice**[i].get(j)).getTimeOut() > inp) {  
 **timeDevice**[i].set(j, **new** Block(((Block) **timeDevice**[i].get(j - 1)).getTimeIn(), inp, i));  
 }  
 **if** (((Block) **timeDevice**[i].get(j)).getTimeIn() < ((Block) **timeDevice**[i].get(j - 1)).getTimeOut()) {  
 **timeDevice**[i].set(j - 1, **new** Block(((Block) **timeDevice**[i].get(j - 1)).getTimeIn(), ((Block) **timeDevice**[i].get(j)).getTimeOut(), i));  
 **timeDevice**[i].remove(j);  
 }  
 }  
 **for** (**int** j = **timeDevice**[i].size() - 1; j > 0; j--) {  
 **stat**[i] += (((Block) **timeDevice**[i].get(j)).getTimeOut() - ((Block) **timeDevice**[i].get(j)).getTimeIn());  
 }  
 **stat**[i] /= inp;  
 }  
 }  
  
 **void** getStatistic() {  
 **double** sumStat = 0;  
 **for** (**int** i = 0; i < **matrixCrossing**.**length**; i++) {  
 sumStat += **stat**[i];  
 System.***out***.println(**str**[i] + **": rate = "** + String.*format*(**" %.10f"**, **stat**[i]));  
 }  
 System.***out***.println(**"SUM = "** + String.*format*(**" %.10f"**, sumStat));  
  
 }  
  
 **public double**[] getStat() {  
 **return stat**;  
 }  
  
 **void** bigStat(**double** time, **int** iteration) {  
 **double**[] bigStat = **new double**[**matrixCrossing**.**length**];  
 **for** (**int** i = 0; i < iteration; i++) {  
 Model model = **new** Model();  
 model.generateTime(time);  
 **for** (**int** j = 0; j < **matrixCrossing**.**length**; j++) {  
 bigStat[j] += model.getStat()[j];  
 }  
 }  
  
 **for** (**int** i = 0; i < **matrixCrossing**.**length**; i++) {  
 bigStat[i] /= iteration;  
 System.***out***.println(**str**[i] + **": rate = "** + String.*format*(**" %.10f"**, bigStat[i]));  
 }  
 }  
}

**Main.java**

**package** km.lab2;  
*/\*  
Computer modelling.  
Lab 2  
Dolinniy Alexandr  
IO-31  
01.11.2015  
\*/*  
**public class** Main {  
 **public static void** main(String[] args) {  
 Model model = **new** Model();model.bigStat(10000, 10000);  
 }  
}

**Task.java**

**package** km.lab2;  
*/\*  
Computer modelling.  
Lab 2  
Dolinniy Alexandr  
IO-31  
01.11.2015  
\*/*  
**public class** Task {  
 **private int numberTask**;  
 **private int numberDevice**;  
 **private int numberNextDevice**;  
 **private double timeEntrance**;  
 **private double timeSolution**;  
  
 **public** Task(**int** numberTask, **int** numberDevice, **int** numberNextDevice, **double** timeEntrance, **double** timeSolution) {  
 **this**.**numberTask** = numberTask;  
 **this**.**numberDevice** = numberDevice;  
 **this**.**numberNextDevice** = numberNextDevice;  
 **this**.**timeEntrance** = timeEntrance;  
 **this**.**timeSolution** = timeSolution;  
 }  
  
 **public int** getNumberTask() {  
 **return numberTask**;  
 }  
  
 **public void** setNumberTask(**int** numberTask) {  
 **this**.**numberTask** = numberTask;  
 }  
  
 **public int** getNumberDevice() {  
 **return numberDevice**;  
 }  
  
 **public void** setNumberDevice(**int** numberDevice) {  
 **this**.**numberDevice** = numberDevice;  
 }  
  
 **public int** getNumberNextDevice() {  
 **return numberNextDevice**;  
 }  
  
 **public void** setNumberNextDevice(**int** numberNextDevice) {  
 **this**.**numberNextDevice** = numberNextDevice;  
 }  
  
 **public double** getTimeEntrance() {  
 **return timeEntrance**;  
 }  
  
 **public void** setTimeEntrance(**double** timeEntrance) {  
 **this**.**timeEntrance** = timeEntrance;  
 }  
  
 **public double** getTimeSolution() {  
 **return timeSolution**;  
 }  
  
 **public void** setTimeSolution(**double** timeSolution) {  
 **this**.**timeSolution** = timeSolution;  
 }  
  
 @Override  
 **public** String toString() {  
 String result = **"numberTask:"** + **numberTask** + **"; numberDevice:"** + **numberDevice** + **"; numberNextDevice:"** + **numberNextDevice** + **"; timeEntrance:"** + String.*format*(**" %.5f"**, **timeEntrance**)  
 + **"; timeSolution:"** + String.*format*(**" %.5f"**, **timeSolution**);  
 **return** result;  
  
 }  
}

**Block.java**

**package** km.lab2;  
*/\*  
Computer modelling.  
Lab 2  
Dolinniy Alexandr  
IO-31  
01.11.2015  
\*/*  
**public class** Block {  
 **private double timeIn**;  
 **private double timeOut**;  
 **private int numberDevice**;  
  
 **public** Block(**double** timeIn, **double** timeOut, **int** numberDevice) {  
 **this**.**timeIn** = timeIn;  
 **this**.**timeOut** = timeOut;  
 **this**.**numberDevice** = numberDevice;  
 }  
  
 **public double** getTimeIn() {  
 **return timeIn**;  
 }  
  
 **public void** setTimeIn(**double** timeIn) {  
 **this**.**timeIn** = timeIn;  
 }  
  
 **public double** getTimeOut() {  
 **return timeOut**;  
 }  
  
 **public void** setTimeOut(**double** timeOut) {  
 **this**.**timeOut** = timeOut;  
 }  
  
 **public int** getNumberDevice() {  
 **return numberDevice**;  
 }  
  
 **public void** setNumberDevice(**int** numberDevice) {  
 **this**.**numberDevice** = numberDevice;  
 }  
}