Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Організація обчислювальних процесів

Лабораторна робота №3

**«Дослідження дисциплін обслуговування заявок при обмежених ресурсах»**

Виконала:

студентка групи ІВ-71

Молчанова В.С.

Номер залікової книжки: 7110

Перевірив:

Сімоненко А.В.

Київ

2020 р.

## Завдання на роботу

1. Згідно варіанту виконати потактову візуалізацію системи обслуговування заявок з розподіленим ресурсом, параметрами системи є діапазон ваг заявок, діапазон інтервалів виникнення наступної заявки (інтенсивність вхідного потоку), діапазон пріоритетів (для пріоритетних дисциплін обслуговування). В систему може надходити будь-яка кількість заявок.
2. Якщо система обслуговування з чергами, то кількість черг не більша 32. Якщо система обслуговування з пріоритетами, то число пріоритетів не більш ніж 32.
3. Побудувати графіки залежності середнього часу очікування від інтенсивності вхідного потоку заявок та залежність проценту простою ресурсу від інтенсивності вхідного потоку заявок.
4. Для пріоритетних систем побудувати графік залежності середнього часу очікування від пріоритету при фіксованій інтенсивності вхідного потоку заявок. Для систем без пріоритетів побудувати графік залежності кількості заявок від часу очікування при фіксованій інтенсивності вхідного потоку заявок.
5. Пояснити форму графіків.

## Варіант

1. 10 mod 16 + 1 = 11 – Алгоритм Корбато.

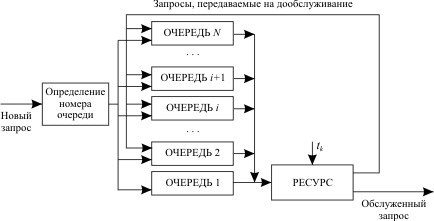
### Опис роботи дисципліни обслуговування

Квант часу, що виділяється програмі на обслуговування зростає зі збільшенням номера черги зазвичай за правилом:

*tki = 2i-1 · tk*, де *tk* - квант часу, що виділяється для програм з черги 1.

Вважається, що тривалість виконання програми приблизно пропорційна її довжині. Щонайменше, від довжини програми прямо залежить час, що витрачається на передачу програми між ОЗП і зовнішнім ЗП при її активізації.

Визначення номеру черги, в яку надходить програма при первинному завантаженні, здійснюється за алгоритмом планування Корбато: програма відразу надходить в чергу *i = [log2 lp / ltk + 1]*, де *lp* - довжина програми в байтах; *ltk* - число байт, які можуть бути передані між ОЗП і зовнішньою пам'яттю за час *tk* (рис. 1).

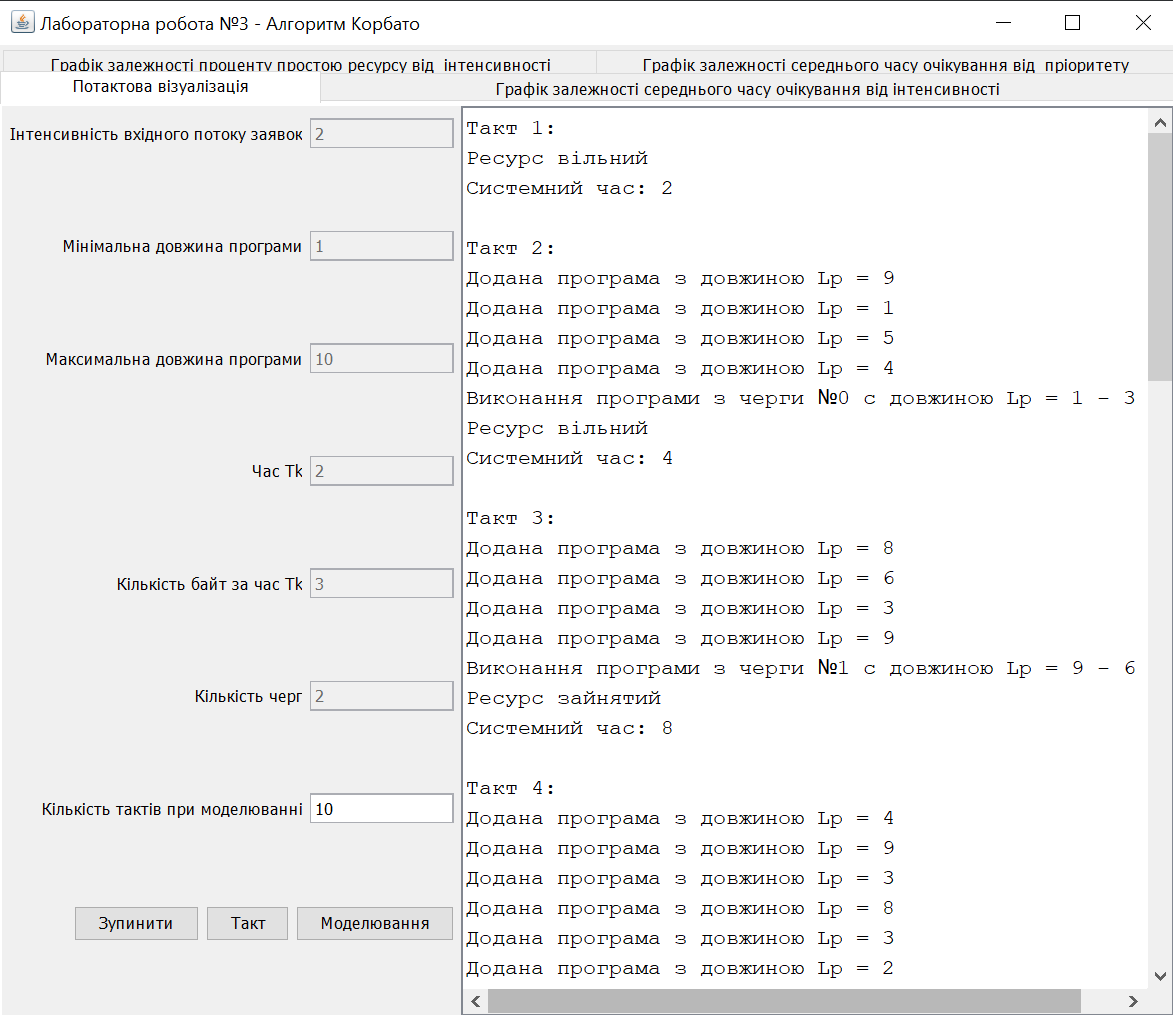


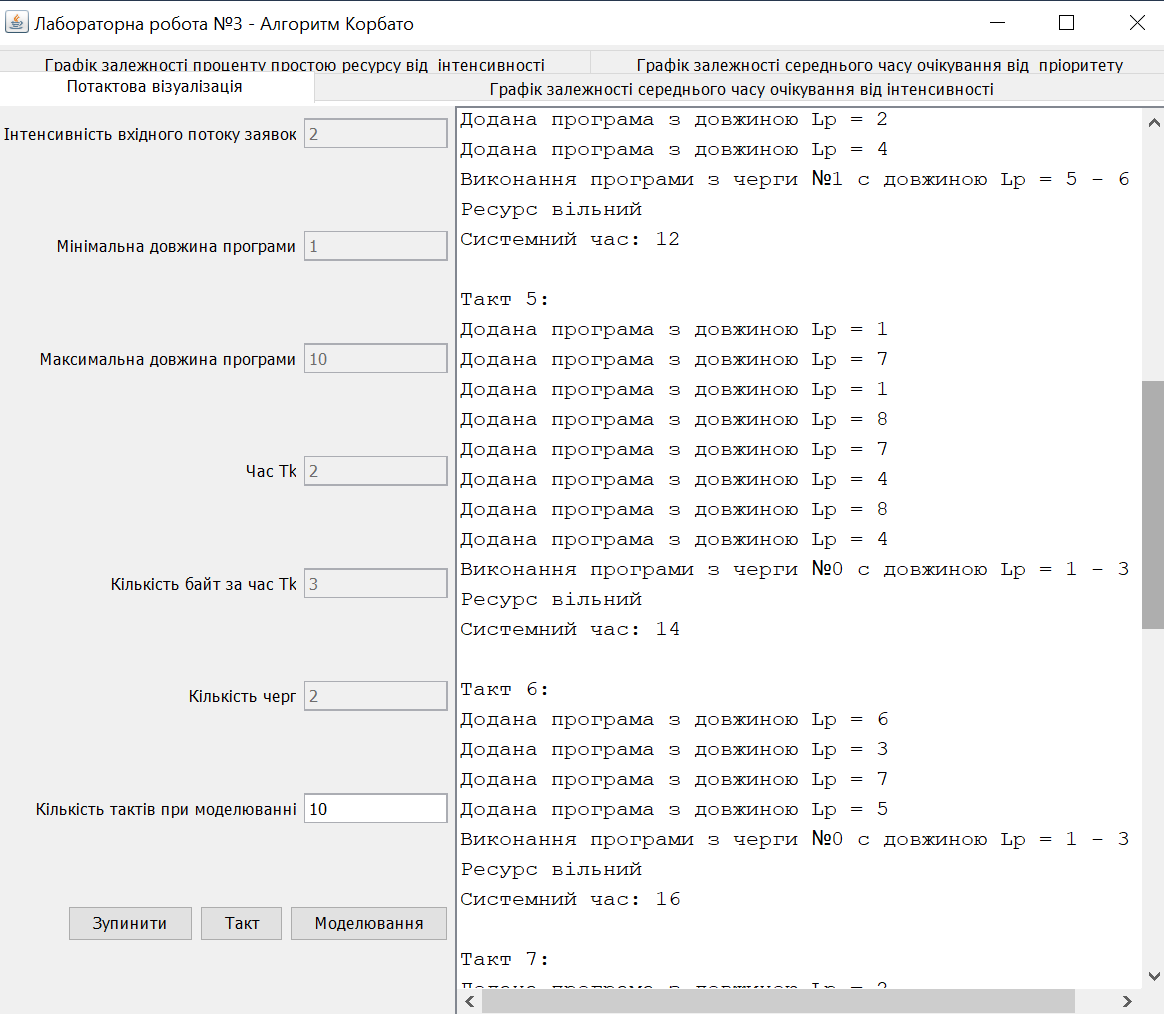
*Рис. 1.*  Схема розподілу ресурсу при багаточерговій дисципліні обслуговування з статичним зазначенням пріоритетів програм

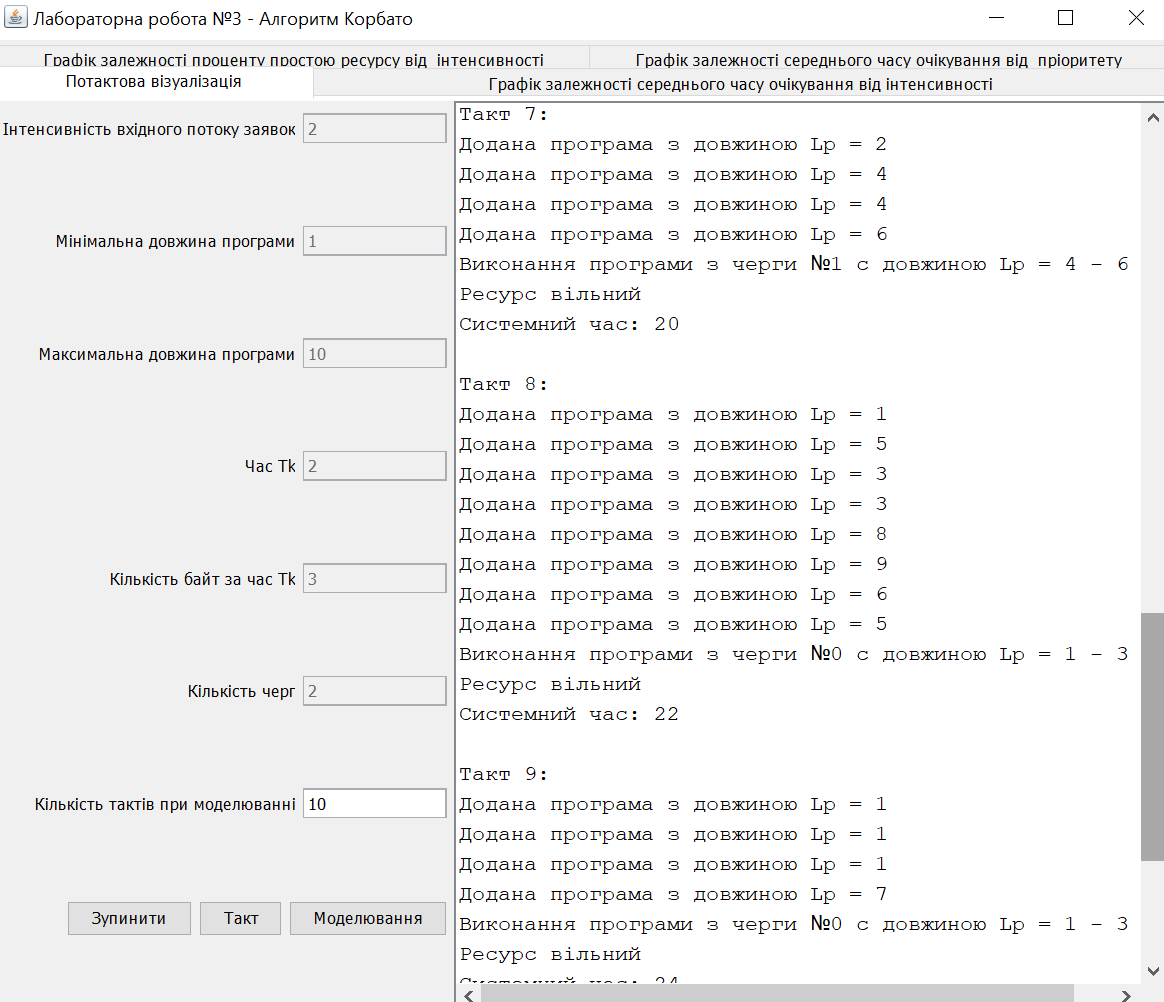
Ця дисципліна дозволяє скоротити кількість системних перемикань за рахунок того, що програмам, які вимагають більшого часу рішення, будуть надаватися чималі кванти часу вже при першому занятті ними ресурсу (нераціонально програмі, яка вимагає для свого рішення 1 годину часу, спочатку виділяти квант в 1 мс ).

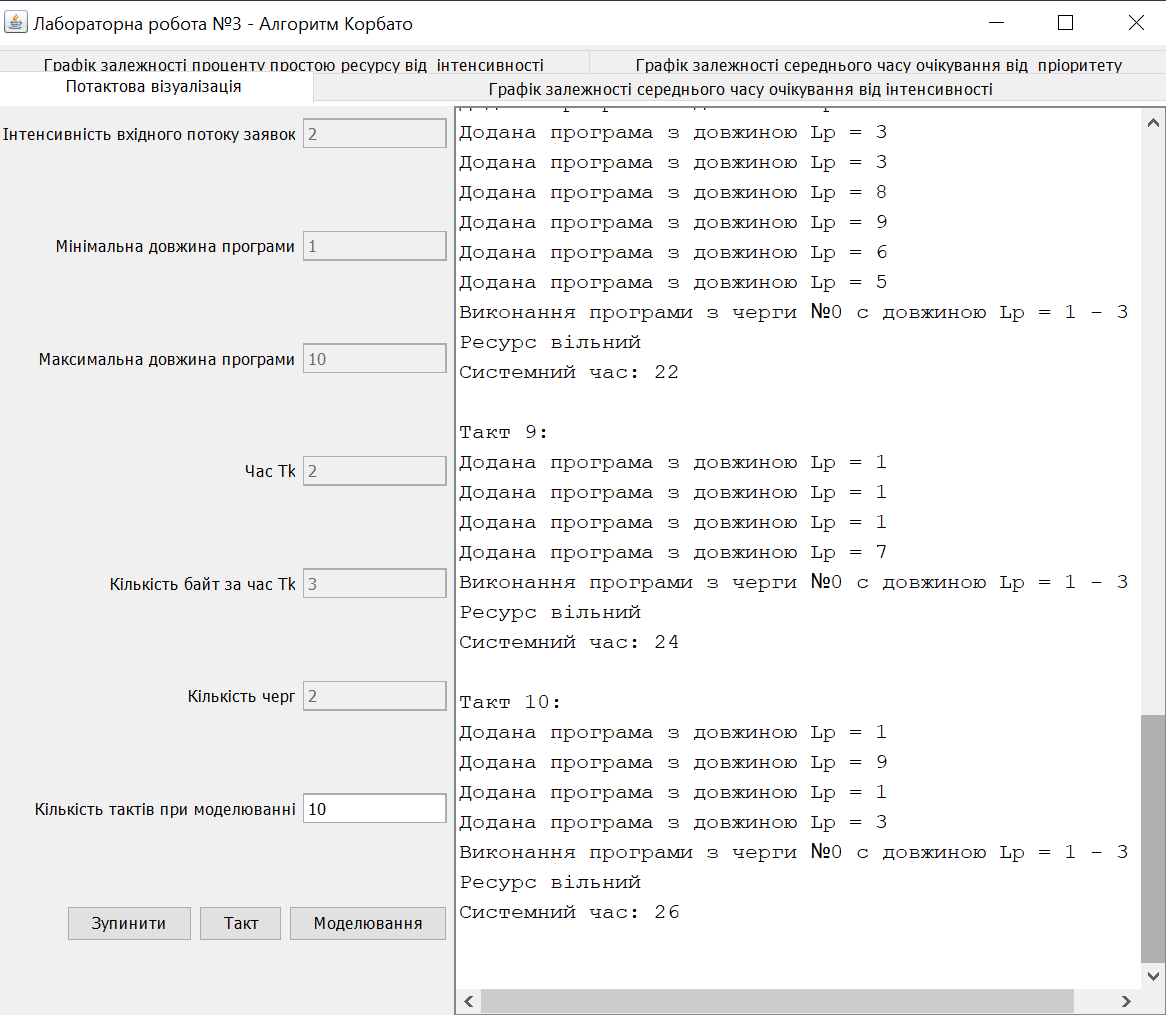
## Результати роботи програми

#### Потактова візуалізація





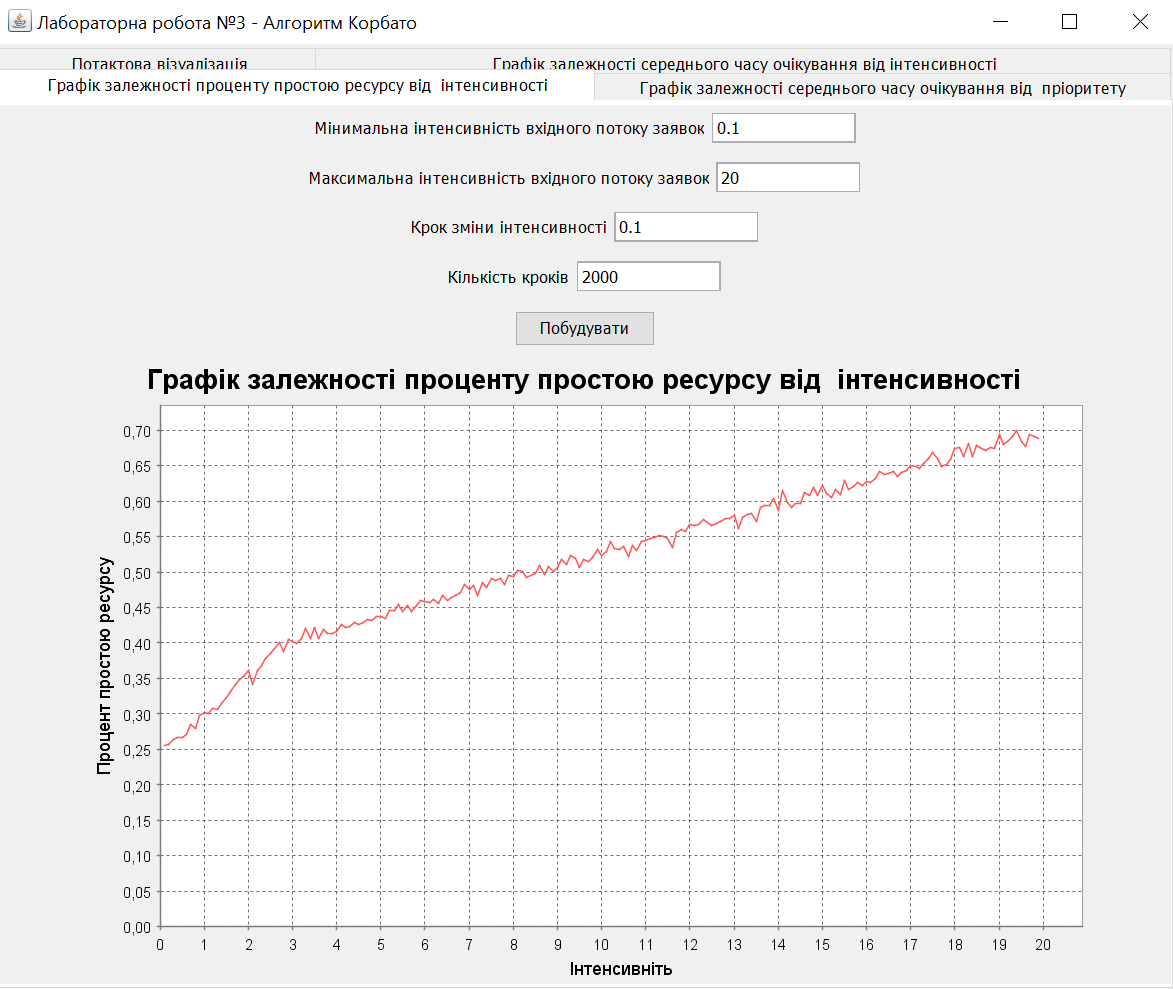




#### Графік залежності середнього часу очікування від інтенсивності вхідного потоку заявок



#### Графік залежності проценту простою ресурсу від інтенсивності вхідного потоку заявок



#### Графік залежності середнього часу очікування від пріоритету при фіксованій інтенсивності вхідного потоку заявок



## Лістинг програми

#### Файл CPO\_Lab3.java

import javax.swing.\*;

public class CPO\_Lab3 {

public static void main(String[] args) {

if (System.getProperty("os.name").contains("Windows")) {

try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

} catch (ClassNotFoundException e) {

} catch (InstantiationException e) {

} catch (IllegalAccessException e) {

} catch (UnsupportedLookAndFeelException e) {}

}

MainFrame frame = new MainFrame();

frame.setVisible(true);

}

}

#### Файл Corbato.java

import java.util.ArrayList;

public class Corbato {

private int tk;

private int ltk;

private ArrayList<ProgramQueue> queues;

private double lambda;

private int minLp;

private int maxLp;

private int t;

private int generateCount;

private double averageWaitTime;

private double sumWaitTime;

private int doneCount;

private int CPUdowntime;

public Corbato(int queuesCount, int tk, int ltk, double lambda, int minLp, int maxLp) {

queues = new ArrayList<ProgramQueue>();

for (int i = 0; i < queuesCount; i++) {

queues.add(new ProgramQueue());

}

this.tk = tk;

this.ltk = ltk;

this.lambda = lambda;

this.minLp = minLp;

this.maxLp = maxLp;

t = 0;

generateCount = 0;

doneCount = 0;

averageWaitTime = 0;

sumWaitTime = 0;

CPUdowntime = 0;

}

public double getAverageWaitTime() {

return averageWaitTime;

}

public double getCPUdowntimePercent() {

return ((double) CPUdowntime / (double) t);

}

public void addProgram(Program p) {

int i = (int) (Math.log((double) p.getLp() / ltk) / Math.log(2));

if (++i >= queues.size()) {

i = queues.size() - 1;

}

if (i < 0) {

i = 0;

}

queues.get(i).add(p);

}

public void tact() {

int i = 0;

while (((double) t \* lambda) > generateCount) {

Program p = Program.generateProgram(minLp, maxLp);

addProgram(p);

generateCount++;

}

while ((i < queues.size()) && (queues.get(i).getSize() == 0)) {

i++;

}

boolean isResourceInUse = false;

Program program = null;

if (i < queues.size()) {

for (int j = 0; j < queues.size(); j++) {

if (j != i) {

queues.get(j).waitAll((int) (Math.pow(2, (double) i) \* tk));

}

}

program = queues.get(i).waitAllExcept(0, (int) (Math.pow(2, (double) i) \* tk),

(int) (Math.pow(2, (double) i) \* ltk));

if (program.getLp() <= 0) {

sumWaitTime += program.getWaitTime();

averageWaitTime = sumWaitTime / ++doneCount;

}

if (program.getLp() >= 0) {

if ((i + 1) < queues.size()) {

queues.get(i + 1).add(program);

}

else {

queues.get(i).add(program);

}

isResourceInUse = true;

}

else {

isResourceInUse = false;

}

}

if (isResourceInUse) {

t += (int) (Math.pow(2, (double) i) \* tk);

}

else {

if (program == null) {

t += tk;

CPUdowntime += tk;

}

else {

t += (int) (Math.pow(2, (double) i) \* tk);

CPUdowntime += ((double) Math.abs(program.getLp()) / ltk) \* tk;

}

}

}

public String tactWithLog() {

String log = "";

int i = 0;

while (((double) t \* lambda) > generateCount) {

Program p = Program.generateProgram(minLp, maxLp);

addProgram(p);

generateCount++;

log += "Добавлена программа с длиной Lp = " + String.valueOf(p.getLp()) + "\n";

}

while ((i < queues.size()) && (queues.get(i).getSize() == 0)) {

i++;

}

boolean isResourceInUse = false;

Program program = null;

if (i < queues.size()) {

for (int j = 0; j < queues.size(); j++) {

if (j != i) {

queues.get(j).waitAll((int) (Math.pow(2, (double) i) \* tk));

}

}

log += "Выполнение программы из очереди №" + String.valueOf(i) + " с длиной Lp = " +

String.valueOf(queues.get(i).get(0).getLp()) + " - " +

String.valueOf((int) (Math.pow(2, (double) i) \* ltk));

program = queues.get(i).waitAllExcept(0, (int) (Math.pow(2, (double) i) \* tk),

(int) (Math.pow(2, (double) i) \* ltk));

if (program.getLp() <= 0) {

sumWaitTime += program.getWaitTime();

averageWaitTime = sumWaitTime / ++doneCount;

}

if (program.getLp() >= 0) {

if ((i + 1) < queues.size()) {

queues.get(i + 1).add(program);

log += " = " + String.valueOf(program.getLp()) + "\n";

}

else {

queues.get(i).add(program);

log += " = " + String.valueOf(program.getLp()) + "\n";

}

isResourceInUse = true;

}

else {

log += " = " + "0" + "\n";

isResourceInUse = false;

}

}

if (isResourceInUse) {

t += (int) (Math.pow(2, (double) i) \* tk);

log += "Ресурс занят\n";

}

else {

log += "Ресурс свободен\n";

if (program == null) {

CPUdowntime += tk;

t += tk;

}

else {

CPUdowntime += ((double) Math.abs(program.getLp()) / ltk) \* tk;

t += (int) (Math.pow(2, (double) i) \* tk);

}

}

log += "Системное время: " + String.valueOf(t) + "\n";

return log;

}

}

#### Файл ProgramQueue.java

import java.util.ArrayList;

public class ProgramQueue {

private ArrayList<Program> programs;

public ProgramQueue() {

programs = new ArrayList<Program>();

}

public int getSize() {

return programs.size();

}

public void add(Program p) {

programs.add(p);

}

public Program get(int i) {

return programs.get(i);

}

public void waitAll(int tk) {

if (programs.size() > 0) {

for (Program p : programs) {

p.wait(tk);

}

}

}

public Program waitAllExcept(int e, int tk, int ltk) {

Program program = null;

for (int i = 0; i < programs.size(); i++) {

if (i != e) {

programs.get(i).wait(tk);

}

else {

programs.get(i).execute(ltk);

program = programs.get(i).copy();

programs.remove(i);

}

}

return program;

}

}

#### Файл Program.java

public class Program {

private int startLp;

private int lp;

private int waitTime;

public Program(int lp) {

this.lp = lp;

startLp = lp;

waitTime = 0;

}

public static Program generateProgram(int minLp, int maxLp) {

return new Program(minLp + (int)(Math.random() \* (double)(maxLp - minLp)));

}

public Program copy() {

Program copyProgram = new Program(this.lp);

copyProgram.waitTime = this.waitTime;

copyProgram.startLp = startLp;

return copyProgram;

}

public int getStartLp() {

return startLp;

}

public int getLp() {

return lp;

}

public void execute(int ltk) {

lp -= ltk;

}

public void wait(int tk) {

waitTime += tk;

}

public int getWaitTime() {

return waitTime;

}

}

#### Файл MainFrame.java

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

public class MainFrame extends JFrame {

private JTabbedPane tabbedPane;

public MainFrame() {

super();

setTitle("Лабораторна робота №3 - Алгоритм Корбато");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setMinimumSize(new Dimension(800, 600));

setExtendedState(JFrame.MAXIMIZED\_BOTH);

tabbedPane = new JTabbedPane();

JPanel vPanel = new VisualizationPanel();

WaitTimePanel g1Panel = new WaitTimePanel();

CPUDowntimePanel g2Panel = new CPUDowntimePanel();

PriorityPanel g3Panel = new PriorityPanel();

tabbedPane.add("Потактова візуалізація", vPanel);

tabbedPane.add("Графік залежності середнього часу очікування від інтенсивності", g1Panel);

tabbedPane.add("Графік залежності проценту простою ресурсу від інтенсивності", g2Panel);

tabbedPane.add("Графік залежності середнього часу очікування від пріоритету", g3Panel);

tabbedPane.setSelectedIndex(0);

add(tabbedPane);

}

private class VisualizationPanel extends JPanel {

private JTextField lambdaField;

private JTextField minLpField;

private JTextField maxLpField;

private JTextField tkField;

private JTextField ltkField;

private JTextField queuesCountField;

private JTextField stepsField;

private JTextArea consoleArea;

private JButton clearButton;

private Corbato corbato;

private int step;

public VisualizationPanel() {

super();

setLayout(new BorderLayout());

JPanel parametersPanel = new JPanel();

parametersPanel.setLayout(new GridLayout(8, 1));

JPanel tempPanel = new JPanel();

tempPanel.setLayout(new FlowLayout(FlowLayout.RIGHT));

tempPanel.add(new JLabel("Інтенсивність вхідного потоку заявок"));

lambdaField = new JTextField();

lambdaField.setColumns(10);

tempPanel.add(lambdaField);

parametersPanel.add(tempPanel);

tempPanel = new JPanel();

tempPanel.setLayout(new FlowLayout(FlowLayout.RIGHT));

tempPanel.add(new JLabel("Мінімальна довжина програми"));

minLpField = new JTextField();

minLpField.setColumns(10);

tempPanel.add(minLpField);

parametersPanel.add(tempPanel);

tempPanel = new JPanel();

tempPanel.setLayout(new FlowLayout(FlowLayout.RIGHT));

tempPanel.add(new JLabel("Максимальна довжина програми"));

maxLpField = new JTextField();

maxLpField.setColumns(10);

tempPanel.add(maxLpField);

parametersPanel.add(tempPanel);

tempPanel = new JPanel();

tempPanel.setLayout(new FlowLayout(FlowLayout.RIGHT));

tempPanel.add(new JLabel("Час Tk"));

tkField = new JTextField();

tkField.setColumns(10);

tempPanel.add(tkField);

parametersPanel.add(tempPanel);

tempPanel = new JPanel();

tempPanel.setLayout(new FlowLayout(FlowLayout.RIGHT));

tempPanel.add(new JLabel("Кількість байт за час Tk"));

ltkField = new JTextField();

ltkField.setColumns(10);

tempPanel.add(ltkField);

parametersPanel.add(tempPanel);

tempPanel = new JPanel();

tempPanel.setLayout(new FlowLayout(FlowLayout.RIGHT));

tempPanel.add(new JLabel("Кількість черг"));

queuesCountField = new JTextField();

queuesCountField.setColumns(10);

tempPanel.add(queuesCountField);

parametersPanel.add(tempPanel);

tempPanel = new JPanel();

tempPanel.setLayout(new FlowLayout(FlowLayout.RIGHT));

tempPanel.add(new JLabel("Кількість тактів при моделюванні"));

stepsField = new JTextField();

stepsField.setColumns(10);

tempPanel.add(stepsField);

parametersPanel.add(tempPanel);

tempPanel = new JPanel();

tempPanel.setLayout(new FlowLayout(FlowLayout.RIGHT));

clearButton = new JButton(new AbstractAction() {

public void actionPerformed(ActionEvent event) {

lambdaField.setEnabled(true);

minLpField.setEnabled(true);

maxLpField.setEnabled(true);

tkField.setEnabled(true);

ltkField.setEnabled(true);

queuesCountField.setEnabled(true);

consoleArea.setText("");

((JButton)event.getSource()).setEnabled(false);

corbato = null;

}

});

clearButton.setText("Зупинити");

clearButton.setEnabled(false);

tempPanel.add(clearButton);

JButton stepButton = new JButton(new AbstractAction() {

public void actionPerformed(ActionEvent event) {

if (corbato == null) {

lambdaField.setEnabled(false);

minLpField.setEnabled(false);

maxLpField.setEnabled(false);

tkField.setEnabled(false);

ltkField.setEnabled(false);

queuesCountField.setEnabled(false);

clearButton.setEnabled(true);

step = 0;

corbato = new Corbato(

Integer.valueOf(queuesCountField.getText()),

Integer.valueOf(tkField.getText()),

Integer.valueOf(ltkField.getText()),

Double.valueOf(lambdaField.getText()),

Integer.valueOf(minLpField.getText()),

Integer.valueOf(maxLpField.getText()));

}

consoleArea.setText(consoleArea.getText() + "Такт " +

String.valueOf(step + 1) + ":\n" + corbato.tactWithLog() + "\n");

step++;

}

});

stepButton.setText("Такт");

tempPanel.add(stepButton);

JButton modellingButton = new JButton(new AbstractAction() {

public void actionPerformed(ActionEvent event) {

if (corbato == null) {

lambdaField.setEnabled(false);

minLpField.setEnabled(false);

maxLpField.setEnabled(false);

tkField.setEnabled(false);

ltkField.setEnabled(false);

queuesCountField.setEnabled(false);

clearButton.setEnabled(true);

step = 0;

corbato = new Corbato(

Integer.valueOf(queuesCountField.getText()),

Integer.valueOf(tkField.getText()),

Integer.valueOf(ltkField.getText()),

Double.valueOf(lambdaField.getText()),

Integer.valueOf(minLpField.getText()),

Integer.valueOf(maxLpField.getText()));

}

int prevStep = step;

for (step = step; step < (prevStep + Integer.valueOf(stepsField.getText())); step++) {

consoleArea.setText(consoleArea.getText() + "Такт " +

String.valueOf(step + 1) + ":\n" + corbato.tactWithLog() + "\n");

}

}

});

modellingButton.setText("Моделювання");

tempPanel.add(modellingButton);

parametersPanel.add(tempPanel);

add(parametersPanel, BorderLayout.WEST);

consoleArea = new JTextArea();

consoleArea.setBackground(Color.WHITE);

Font consoleFont = new Font("Monospaced", Font.PLAIN, 14);

consoleArea.setEditable(false);

consoleArea.setAutoscrolls(true);

consoleArea.setFont(consoleFont);

consoleArea.setForeground(Color.WHITE);

JScrollPane consolePane = new JScrollPane(consoleArea);

add(consolePane);

corbato = null;

step = 0;

}

}

private class WaitTimePanel extends JPanel {

private Corbato corbato;

private GraphPanel graph;

private JTextField maxLambdaField;

private JTextField minLambdaField;

private JTextField lambdaStepField;

private JTextField stepCountField;

public WaitTimePanel() {

setLayout(new BorderLayout());

JPanel optionsPanel = new JPanel();

optionsPanel.setLayout(new GridLayout(5, 1));

JPanel tempPanel = new JPanel();

minLambdaField = new JTextField();

minLambdaField.setColumns(10);

tempPanel.add(new JLabel("Мінімальна інтенсивність вхідного потоку заявок"));

tempPanel.add(minLambdaField);

optionsPanel.add(tempPanel);

tempPanel = new JPanel();

maxLambdaField = new JTextField();

maxLambdaField.setColumns(10);

tempPanel.add(new JLabel("Максимальна інтенсивність вхідного потоку заявок"));

tempPanel.add(maxLambdaField);

optionsPanel.add(tempPanel);

tempPanel = new JPanel();

lambdaStepField = new JTextField();

lambdaStepField.setColumns(10);

tempPanel.add(new JLabel("Крок зміни інтенсивності"));

tempPanel.add(lambdaStepField);

optionsPanel.add(tempPanel);

tempPanel = new JPanel();

stepCountField = new JTextField();

stepCountField.setColumns(10);

tempPanel.add(new JLabel("Кількість кроків"));

tempPanel.add(stepCountField);

optionsPanel.add(tempPanel);

JButton generateButton = new JButton();

generateButton.setAction(new AbstractAction() {

public void actionPerformed(ActionEvent event) {

double intensity = Double.valueOf(minLambdaField.getText());

graph.setModel(new GraphModel("Графік залежності середнього часу очікування від інтенсивності",

"Інтенсивність", "Середній час очікування"));

while (intensity <= Double.valueOf(maxLambdaField.getText())) {

corbato = new Corbato(10, 10, 100, intensity, 1, 10000);

for (int i = 0; i < Integer.valueOf(stepCountField.getText()); i++) {

corbato.tact();

}

graph.getModel().add(intensity, corbato.getAverageWaitTime());

intensity += Double.valueOf(lambdaStepField.getText());

}

}

});

generateButton.setText("Побудувати");

tempPanel = new JPanel();

tempPanel.add(generateButton);

optionsPanel.add(tempPanel);

add(optionsPanel, BorderLayout.NORTH);

graph = new GraphPanel();

graph.setModel(new GraphModel("Графік залежності середнього часу очікування від інтенсивності",

"Інтенсивність", "Середній час очікування"));

add(graph);

}

}

private class CPUDowntimePanel extends JPanel {

private Corbato corbato;

private GraphPanel graph;

private JTextField maxLambdaField;

private JTextField minLambdaField;

private JTextField lambdaStepField;

private JTextField stepCountField;

public CPUDowntimePanel() {

setLayout(new BorderLayout());

JPanel optionsPanel = new JPanel();

optionsPanel.setLayout(new GridLayout(5, 1));

JPanel tempPanel = new JPanel();

minLambdaField = new JTextField();

minLambdaField.setColumns(10);

tempPanel.add(new JLabel("Мінимальна інтенсивність вхідного потоку заявок"));

tempPanel.add(minLambdaField);

optionsPanel.add(tempPanel);

tempPanel = new JPanel();

maxLambdaField = new JTextField();

maxLambdaField.setColumns(10);

tempPanel.add(new JLabel("Максимальна інтенсивність вхідного потоку заявок"));

tempPanel.add(maxLambdaField);

optionsPanel.add(tempPanel);

tempPanel = new JPanel();

lambdaStepField = new JTextField();

lambdaStepField.setColumns(10);

tempPanel.add(new JLabel("Крок зміни інтенсивності"));

tempPanel.add(lambdaStepField);

optionsPanel.add(tempPanel);

tempPanel = new JPanel();

stepCountField = new JTextField();

stepCountField.setColumns(10);

tempPanel.add(new JLabel("Кількість кроків"));

tempPanel.add(stepCountField);

optionsPanel.add(tempPanel);

JButton generateButton = new JButton();

generateButton.setAction(new AbstractAction() {

public void actionPerformed(ActionEvent event) {

double intensity = Double.valueOf(minLambdaField.getText());

graph.setModel(new GraphModel("Графік залежності проценту простою ресурсу від інтенсивності",

"Інтенсивніть", "Процент простою ресурсу"));

while (intensity <= Double.valueOf(maxLambdaField.getText())) {

corbato = new Corbato(10, 10, 100, intensity, 1, 10000);

for (int i = 0; i < Integer.valueOf(stepCountField.getText()); i++) {

corbato.tact();

}

graph.getModel().add(intensity, corbato.getCPUdowntimePercent());

intensity += Double.valueOf(lambdaStepField.getText());

}

}

});

generateButton.setText("Побудувати");

tempPanel = new JPanel();

tempPanel.add(generateButton);

optionsPanel.add(tempPanel);

add(optionsPanel, BorderLayout.NORTH);

graph = new GraphPanel();

graph.setModel(new GraphModel("Графік залежності проценту простою ресурсу від інтенсивності",

"Інтенсивніть", "Процент простою ресурсу"));

add(graph);

}

}

private class PriorityPanel extends JPanel {

private Corbato corbato;

private GraphPanel graph;

private JTextField lambdaField;

private JTextField stepCountField;

public PriorityPanel() {

setLayout(new BorderLayout());

JPanel optionsPanel = new JPanel();

optionsPanel.setLayout(new GridLayout(3, 1));

JPanel tempPanel = new JPanel();

lambdaField = new JTextField();

lambdaField.setColumns(10);

tempPanel.add(new JLabel("Інтенсивність вхідного потоку заявок"));

tempPanel.add(lambdaField);

optionsPanel.add(tempPanel);

tempPanel = new JPanel();

stepCountField = new JTextField();

stepCountField.setColumns(10);

tempPanel.add(new JLabel("Кількість кроків"));

tempPanel.add(stepCountField);

optionsPanel.add(tempPanel);

JButton generateButton = new JButton();

generateButton.setAction(new AbstractAction() {

public void actionPerformed(ActionEvent event) {

double intensity = Double.valueOf(lambdaField.getText());

graph.setModel(new GraphModel("Графік залежності середнього часу очікування від пріоритету",

"Пріоритет заявки", "Середній час очікування"));

for (int priority = 1; priority < 10000; priority++) {

corbato = new Corbato(10, 10, 100, intensity, priority, priority);

for (int i = 0; i < Integer.valueOf(stepCountField.getText()); i++) {

corbato.tact();

}

graph.getModel().add(priority, corbato.getAverageWaitTime());

}

}

});

generateButton.setText("Побудувати");

tempPanel = new JPanel();

tempPanel.add(generateButton);

optionsPanel.add(tempPanel);

add(optionsPanel, BorderLayout.NORTH);

graph = new GraphPanel();

graph.setModel(new GraphModel("Графік залежності середнього часу очікування від пріоритету",

"Пріоритет заявки", "Середній час очікування"));

add(graph);

}

}

}

#### Файл GraphPanel.java

import org.jfree.chart.ChartFactory;

import org.jfree.chart.ChartPanel;

import org.jfree.chart.JFreeChart;

import org.jfree.chart.plot.PlotOrientation;

import org.jfree.chart.plot.XYPlot;

import javax.swing.\*;

import java.awt.\*;

public class GraphPanel extends JPanel {

private JFreeChart chart;

private GraphModel model;

public GraphPanel() {

super();

}

public void setModel(GraphModel model) {

this.model = model;

refresh();

}

public GraphModel getModel() {

return model;

}

public void refresh() {

chart = ChartFactory.createXYLineChart(

this.model.getName(),

this.model.getXName(),

this.model.getYName(),

this.model.getDataset(),

PlotOrientation.VERTICAL,

false,

false,

false

);

XYPlot plot = chart.getXYPlot();

plot.setBackgroundPaint(Color.WHITE);

plot.setDomainGridlinePaint(Color.DARK\_GRAY);

plot.setRangeGridlinePaint(Color.DARK\_GRAY);

this.removeAll();

add(new ChartPanel(chart));

revalidate();

repaint();

}

}

#### Файл GraphModel.java

import org.jfree.data.xy.XYSeries;

import org.jfree.data.xy.XYSeriesCollection;

import java.util.HashMap;

public class GraphModel {

private String name;

private String xName;

private String yName;

private XYSeries data;

private HashMap<Double, Integer> xCount;

public GraphModel(String name, String xName, String yName) {

this.name = name;

this.xName = xName;

this.yName = yName;

data = new XYSeries(this.name, true);

xCount = new HashMap<Double, Integer>();

}

public String getName() {

return name;

}

public String getXName() {

return xName;

}

public String getYName() {

return yName;

}

public XYSeriesCollection getDataset() {

return new XYSeriesCollection(data);

}

public void removeAll() {

data = new XYSeries(this.name, true);

xCount = new HashMap<Double, Integer>();

}

public void add(double x, double y) {

if (xCount.containsKey(Double.valueOf(x))) {

int i = 0;

while (data.getX(i).doubleValue() != x) {

i++;

}

int prevCount = xCount.get(Double.valueOf(x));

xCount.put(x, Integer.valueOf(++prevCount));

double temp = data.getY(i).doubleValue();

temp += y;

temp /= xCount.get(Double.valueOf(x).doubleValue());

data.remove(i);

data.add(x, temp);

}

else {

data.add(x, y);

xCount.put(Double.valueOf(x), 1);

}

}

}