Министерство образования и науки Российской Федерации Федеральное агентство по образованию

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №3 дисциплины

«Разработка программных систем»

Выполнил студент группы ИВТ-31 /Крючков И. С./ Проверил /Чистяков Г. А./

Киров 2023

1. Задание

Разработать графическое приложение с использованием библиотеки Swing. Для выполнения лабораторной работы необходимо решить следующие задачи.

Выбрать и согласовать с преподавателем задачу, для решения которой может быть использована программа, разработанная в ходе предыдущей лабораторной работы.

Разработать программу для решения выбранной задачи (взаимодействие с пользователем должно осуществляться с применением графического интерфейса).

1. Листинг программы

Исходный код программы приведен в приложении А.

1. Экранные формы

Экранные формы приведены в приложении Б.

1. Вывод

В ходе выполнения лабораторной работы были получены навыки разработки графического пользовательского интерфейса с применением технологии Swing, изучены ее основные компоненты. Написано приложения с графическим пользовательским интерфейсом, предназначенное для выполнения интервальных операция посредством sqrt-декомпозиции.

Приложение А.

Листинг программы

**Main.java**

**package rpslab;**

**import java.io.FileNotFoundException;**

**import java.util.InputMismatchException;**

**public class Main{**

**private static Decomposition dn;**

**public static void main(String args[]) {**

**try {**

**dn = new Decomposition("input.txt");**

**} catch (DecompositionException e) {**

**System.out.println(e.getMessage());**

**return;**

**} catch (FileNotFoundException e) {**

**System.out.println("input.txt не найден");**

**return;**

**} catch (InputMismatchException e) {**

**System.out.println("input.txt имеет неверный формат");**

**return;**

**}**

**GUI app = new GUI(dn);**

**app.setVisible(true);**

**}**

**}**

**GUI.java**

**package rpslab;**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**public class GUI extends JFrame {**

**private JButton openChangeValueDlgBtn = new JButton("Изниметь значение в точке");**

**private JButton openChangeValuesBtn = new JButton("Изменить значения на интервале");**

**private JButton sumValuesBtn = new JButton("Сумма значений на интервале");**

**private JButton exitBtn = new JButton("Выход");**

**private ChangeValueDialog changeValDlg;**

**private ChangeValuesDialog changeValsDlg;**

**private GetSumDialog sumDlg;**

**public GUI(Decomposition dn) {**

**super("Lab");**

**this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**this.setSize(300, 200);**

**this.setResizable(false);**

**this.getRootPane().setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));**

**this.setLocationRelativeTo(null);**

**Container content = this.getContentPane();**

**content.setLayout(new GridLayout(4, 1, 10, 10));**

**content.add(openChangeValueDlgBtn);**

**content.add(openChangeValuesBtn);**

**content.add(sumValuesBtn);**

**content.add(exitBtn);**

**changeValDlg = new ChangeValueDialog(this, dn);**

**changeValsDlg = new ChangeValuesDialog(this, dn);**

**sumDlg = new GetSumDialog(this, dn);**

**openChangeValueDlgBtn.addActionListener(new OpenChangeValueDlgHandler());**

**openChangeValuesBtn.addActionListener(new OpenChangeValuesDlgHandler());**

**sumValuesBtn.addActionListener(new OpenSumValuesDlgHandler());**

**exitBtn.addActionListener(new ExitHandler());**

**};**

**class ExitHandler implements ActionListener {**

**@Override**

**public void actionPerformed(ActionEvent event) {**

**System.exit(0);**

**}**

**}**

**class OpenChangeValueDlgHandler implements ActionListener {**

**@Override**

**public void actionPerformed(ActionEvent event) {**

**changeValDlg.setVisible(true);**

**}**

**}**

**class OpenChangeValuesDlgHandler implements ActionListener {**

**@Override**

**public void actionPerformed(ActionEvent event) {**

**changeValsDlg.setVisible(true);**

**}**

**}**

**class OpenSumValuesDlgHandler implements ActionListener {**

**@Override**

**public void actionPerformed(ActionEvent event) {**

**sumDlg.setVisible(true);**

**}**

**}**

**}**

**GetSumDialog.java**

**package rpslab;**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**import java.awt.event.WindowAdapter;**

**import java.awt.event.WindowListener;**

**import java.awt.event.WindowEvent;**

**public class GetSumDialog extends JDialog {**

**private JButton exBtn = new JButton("Закрыть");**

**private JButton getSumBtn = new JButton("Получить сумму");**

**private JTextField pointInput, endPointInput;**

**private Decomposition dn;**

**public GetSumDialog(GUI parent, Decomposition dn) {**

**super(parent, "Сумма значений", Dialog.ModalityType.DOCUMENT\_MODAL);**

**this.dn = dn;**

**this.setBounds(232, 232, 300, 200);**

**getContentPane().add(createGUI());**

**pack();**

**this.addWindowListener(closeWindow);**

**exBtn.addActionListener(new CloseHandler());**

**getSumBtn.addActionListener(new ChangeHandler());**

**}**

**private JPanel createGUI() {**

**int n = dn.getLen();**

**JPanel panel = this.createVerticalPanel();**

**panel.setBorder (BorderFactory.createEmptyBorder(12,12,12,12));**

**JPanel pointPanel = this.createHorizontalPanel();**

**JLabel pointLabel = new JLabel(String.format("Начальная точка [0-%d]:", n-1));**

**pointPanel.add(pointLabel);**

**pointPanel.add(Box.createHorizontalStrut(12));**

**pointInput = new JTextField(15);**

**pointPanel.add(pointInput);**

**JPanel endPointPanel = this.createHorizontalPanel();**

**JLabel endPointLabel = new JLabel(String.format("Конечная точка [0-%d]:", n-1));**

**endPointPanel.add(endPointLabel);**

**endPointPanel.add(Box.createHorizontalStrut(12));**

**endPointInput = new JTextField(15);**

**endPointPanel.add(endPointInput);**

**JPanel grid = new JPanel( new GridLayout( 2,1, 0,7) );**

**grid.add(getSumBtn);**

**grid.add(exBtn);**

**this.makeSameSize(new JComponent[] { pointLabel, endPointLabel } );**

**panel.add(pointPanel);**

**panel.add(Box.createVerticalStrut(12));**

**panel.add(endPointPanel);**

**panel.add(Box.createVerticalStrut(17));**

**panel.add(grid);**

**return panel;**

**}**

**public JPanel createVerticalPanel() {**

**JPanel panel = new JPanel();**

**panel.setLayout(new BoxLayout(panel, BoxLayout.Y\_AXIS));**

**return panel;**

**}**

**public JPanel createHorizontalPanel() {**

**JPanel panel = new JPanel();**

**panel.setLayout(new BoxLayout(panel, BoxLayout.X\_AXIS));**

**return panel;**

**}**

**private static WindowListener closeWindow = new WindowAdapter() {**

**public void windowClosing(WindowEvent e) {**

**e.getWindow().dispose();**

**}**

**};**

**public void makeSameSize(JComponent[] components) {**

**int[] array = new int[components.length];**

**for (int i = 0; i < array.length; i++) {**

**array[i] = components[i].getPreferredSize().width;**

**}**

**int maxSizePos = maximumElementPosition(array);**

**Dimension maxSize = components[maxSizePos].getPreferredSize();**

**for (int i=0; i<components.length; i++) {**

**components[i].setPreferredSize(maxSize);**

**components[i].setMinimumSize(maxSize);**

**components[i].setMaximumSize(maxSize);**

**}**

**}**

**private int maximumElementPosition(int[] array) {**

**int maxPos = 0;**

**for (int i = 1; i < array.length; i++) {**

**if (array[i] > array [maxPos])**

**maxPos = i;**

**}**

**return maxPos;**

**}**

**class CloseHandler implements ActionListener {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**dispose();**

**}**

**}**

**class ChangeHandler implements ActionListener {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**int point = 0;**

**try {**

**point = Integer.parseInt(pointInput.getText());**

**} catch (NumberFormatException ex) {**

**JOptionPane.showMessageDialog(null, "Начальная точка указана неверно");**

**return;**

**}**

**int n = dn.getLen();**

**if (point < 0 || point > n-1) {**

**JOptionPane.showMessageDialog(null, "Начальная точка указана неверно");**

**return;**

**}**

**int endPoint = 0;**

**try {**

**endPoint = Integer.parseInt(endPointInput.getText());**

**} catch (NumberFormatException ex) {**

**JOptionPane.showMessageDialog(null, "Конечная точка точка указана неверно");**

**return;**

**}**

**if (endPoint < point || endPoint > n-1) {**

**JOptionPane.showMessageDialog(null, "Конечная точка указана неверно");**

**return;**

**}**

**try {**

**double s = dn.getSum(point, endPoint);**

**JOptionPane.showMessageDialog(null, String.format("Сумма %s", s));**

**} catch (DecompositionException exc) {**

**JOptionPane.showMessageDialog(null, exc.getMessage());**

**}**

**}**

**}**

**}**

**Decomposition.java**

**package rpslab;**

**import java.io.File;**

**import java.io.FileNotFoundException;**

**import java.util.Scanner;**

**import java.util.InputMismatchException;**

**import java.util.ArrayList;**

**import java.util.Locale;**

**/\*\***

**\* Execution of interval operations by means of sqrt decomposition**

**\***

**\* @author Ilya Kryuchkov**

**\*/**

**public class Decomposition {**

**/\*\* Array of elements \*/**

**private ArrayList<Number> data;**

**/\*\* Number of elements \*/**

**private int n;**

**/\*\* Maximum number of elements \*/**

**private final int MAX\_DATA\_SIZE = 1000;**

**/\*\* Maximum value of the element \*/**

**private final long MAX\_VALUE = 10\_000\_000\_000L;**

**/\*\* Array of blocks \*/**

**private ArrayList<Number> blocks;**

**/\*\* Root Value \*/**

**private int rt;**

**/\*\***

**\* Constructs new sqrt decomposition class**

**\* @param filename Name input file**

**\* @throws DecompositionException Decomposition error**

**\* @throws FileNotFoundException File not found**

**\* @throws InputMismatchException File has an incorrect format**

**\*/**

**public Decomposition(String filename) throws FileNotFoundException, DecompositionException, InputMismatchException {**

**readData(filename);**

**calcBlocks();**

**}**

**/\*\***

**\* Reading data from a file**

**\* @param filename Name input file**

**\* @throws DecompositionException Decomposition error**

**\* @throws FileNotFoundException File not found**

**\* @throws InputMismatchException File has an incorrect format**

**\*/**

**private void readData(String filename) throws FileNotFoundException, DecompositionException, InputMismatchException {**

**Scanner in = new Scanner(new File(filename)).useLocale(Locale.US);**

**n = in.nextInt();**

**if (n > MAX\_DATA\_SIZE) {**

**throw new DecompositionException(String.format("Максимальное количество элементов - %s", MAX\_DATA\_SIZE));**

**}**

**data = new ArrayList<Number>(n);**

**int i = 0;**

**while ( (in.hasNextLong() || in.hasNextDouble()) && i < n) {**

**if (in.hasNextLong()) {**

**long t = in.nextLong();**

**if (t > MAX\_VALUE) {**

**throw new DecompositionException(String.format("Максимальное значение элемента - %s", MAX\_VALUE));**

**}**

**if (t < -MAX\_VALUE) {**

**throw new DecompositionException(String.format("Минимальное значение элемента - %s", -MAX\_VALUE));**

**}**

**data.add(t);**

**} else {**

**double t = in.nextDouble();**

**if (t > MAX\_VALUE) {**

**throw new DecompositionException(String.format("Максимальное значение элемента - %s", MAX\_VALUE));**

**}**

**if (t < -MAX\_VALUE) {**

**throw new DecompositionException(String.format("Минимальное значение элемента - %s", -MAX\_VALUE));**

**}**

**data.add(t);**

**}**

**i++;**

**}**

**if (i == 0) {**

**throw new InputMismatchException();**

**}**

**data.trimToSize();**

**n = data.size();**

**}**

**/\*\***

**\* Splitting into intervals and calculating the sum on each interval**

**\*/**

**private void calcBlocks() {**

**rt = (int) Math.ceil(Math.sqrt(n));**

**blocks = new ArrayList<Number>(rt);**

**for (int i = 0; i < rt - 1; ++i) {**

**blocks.add(0);**

**final int idx = i \* rt;**

**int j = 0;**

**while (j < rt && idx + j < n){**

**Number v = blocks.get(i);**

**v = v.doubleValue() + data.get(idx + j).doubleValue();**

**blocks.set(i, v);**

**++j;**

**}**

**}**

**}**

**/\*\***

**\* Calculating the sum at a given interval**

**\* @param a Starting point of the interval**

**\* @param b The end point of the interval**

**\* @return The sum of the values in the interval from a to b**

**\* @throws DecompositionException Decomposition error**

**\*/**

**public double getSum(int a, int b) throws DecompositionException {**

**if (a < 0 || a > b || a >= n || b < 0 || b >= n) {**

**throw new DecompositionException("Интервал некорректный");**

**}**

**double sum = 0;**

**final int startBlock = a/rt;**

**final int endBlock = b/rt;**

**if (startBlock == endBlock) {**

**for (int i = a; i <= b; ++i) {**

**sum += data.get(i).doubleValue();**

**}**

**} else {**

**for (int i = startBlock+1; i < endBlock; ++i) {**

**sum += blocks.get(i).doubleValue();**

**}**

**final int aIdx = a % rt;**

**for (int i = aIdx; i < rt; ++i) {**

**sum += data.get(startBlock\*rt + i).doubleValue();**

**}**

**final int bIdx = b % rt;**

**for (int i = 0; i <= bIdx; ++i) {**

**sum += data.get(endBlock \* rt + i).doubleValue();**

**}**

**}**

**return sum;**

**}**

**/\*\***

**\* Changing the value at a given point**

**\* @param id Index of the item to change**

**\* @param x New value**

**\* @throws DecompositionException Decomposition error**

**\*/**

**public void updateValue(int id, Number x) throws DecompositionException {**

**if (id < 0 || id >= n) {**

**throw new DecompositionException("Индекс некорректный");**

**}**

**int bid = id / rt;**

**double v = data.get(id).doubleValue();**

**double bv = blocks.get(bid).doubleValue();**

**data.set(id, x);**

**blocks.set(bid, x.doubleValue() - v + bv);**

**}**

**/\*\***

**\* Changing values at a given interval**

**\* @param a Starting point of the interval**

**\* @param b The end point of the interval**

**\* @param x New value**

**\* @throws DecompositionException Decomposition error**

**\*/**

**public void updateValues(int a, int b, Number x) throws DecompositionException {**

**if (a < 0 || a > b || a >= n || b < 0 || b >= n) {**

**throw new DecompositionException("Интервал некорректный");**

**}**

**for(int i = a; i <= b; ++i) {**

**updateValue(i, x);**

**}**

**}**

**/\*\***

**\* Getting the number of elements**

**\* @return Number of elements**

**\*/**

**public int getLen() {**

**return n;**

**}**

**/\*\***

**\* Getting the maximum value of an element**

**\* @return Maximum value of the element**

**\*/**

**public long getMaxValue() {**

**return MAX\_VALUE;**

**}**

**}**

**DecompositionException.java**

**package rpslab;**

**public class DecompositionException extends Exception{**

**public DecompositionException(String message){**

**super(message);**

**}**

**}**

**ChangeValuesDialog.java**

**package rpslab;**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**import java.awt.event.WindowAdapter;**

**import java.awt.event.WindowListener;**

**import java.awt.event.WindowEvent;**

**public class ChangeValuesDialog extends JDialog {**

**private JButton exBtn = new JButton("Закрыть");**

**private JButton saveBtn = new JButton("Изменить");**

**private JTextField pointInput, endPointInput, valueInput;**

**private Decomposition dn;**

**public ChangeValuesDialog(GUI parent, Decomposition dn) {**

**super(parent, "Изменить значения", Dialog.ModalityType.DOCUMENT\_MODAL);**

**this.dn = dn;**

**this.setBounds(232, 232, 300, 240);**

**getContentPane().add(createGUI());**

**pack();**

**this.addWindowListener(closeWindow);**

**exBtn.addActionListener(new CloseHandler());**

**saveBtn.addActionListener(new ChangeHandler());**

**}**

**private JPanel createGUI() {**

**int n = dn.getLen();**

**JPanel panel = this.createVerticalPanel();**

**panel.setBorder (BorderFactory.createEmptyBorder(12,12,12,12));**

**JPanel pointPanel = this.createHorizontalPanel();**

**JLabel pointLabel = new JLabel(String.format("Начальная точка [0-%d]:", n-1));**

**pointPanel.add(pointLabel);**

**pointPanel.add(Box.createHorizontalStrut(12));**

**pointInput = new JTextField(15);**

**pointPanel.add(pointInput);**

**JPanel endPointPanel = this.createHorizontalPanel();**

**JLabel endPointLabel = new JLabel(String.format("Конечная точка [0-%d]:", n-1));**

**endPointPanel.add(endPointLabel);**

**endPointPanel.add(Box.createHorizontalStrut(12));**

**endPointInput = new JTextField(15);**

**endPointPanel.add(endPointInput);**

**JPanel valuePanel = this.createHorizontalPanel();**

**JLabel valueLabel = new JLabel("Значение:");**

**valuePanel.add(valueLabel);**

**valuePanel.add(Box.createHorizontalStrut(12));**

**valueInput = new JTextField(15);**

**valuePanel.add(valueInput);**

**JPanel grid = new JPanel( new GridLayout( 2,1, 0,7) );**

**grid.add(saveBtn);**

**grid.add(exBtn);**

**this.makeSameSize(new JComponent[] { pointLabel, valueLabel, endPointLabel } );**

**panel.add(pointPanel);**

**panel.add(Box.createVerticalStrut(12));**

**panel.add(endPointPanel);**

**panel.add(Box.createVerticalStrut(12));**

**panel.add(valuePanel);**

**panel.add(Box.createVerticalStrut(17));**

**panel.add(grid);**

**return panel;**

**}**

**public JPanel createVerticalPanel() {**

**JPanel panel = new JPanel();**

**panel.setLayout(new BoxLayout(panel, BoxLayout.Y\_AXIS));**

**return panel;**

**}**

**public JPanel createHorizontalPanel() {**

**JPanel panel = new JPanel();**

**panel.setLayout(new BoxLayout(panel, BoxLayout.X\_AXIS));**

**return panel;**

**}**

**private static WindowListener closeWindow = new WindowAdapter() {**

**public void windowClosing(WindowEvent e) {**

**e.getWindow().dispose();**

**}**

**};**

**public void makeSameSize(JComponent[] components) {**

**int[] array = new int[components.length];**

**for (int i = 0; i < array.length; i++) {**

**array[i] = components[i].getPreferredSize().width;**

**}**

**int maxSizePos = maximumElementPosition(array);**

**Dimension maxSize = components[maxSizePos].getPreferredSize();**

**for (int i=0; i<components.length; i++) {**

**components[i].setPreferredSize(maxSize);**

**components[i].setMinimumSize(maxSize);**

**components[i].setMaximumSize(maxSize);**

**}**

**}**

**private int maximumElementPosition(int[] array) {**

**int maxPos = 0;**

**for (int i = 1; i < array.length; i++) {**

**if (array[i] > array [maxPos])**

**maxPos = i;**

**}**

**return maxPos;**

**}**

**class CloseHandler implements ActionListener {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**dispose();**

**}**

**}**

**class ChangeHandler implements ActionListener {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**int point = 0;**

**try {**

**point = Integer.parseInt(pointInput.getText());**

**} catch (NumberFormatException ex) {**

**JOptionPane.showMessageDialog(null, "Начальная точка указана неверно");**

**return;**

**}**

**int n = dn.getLen();**

**if (point < 0 || point > n-1) {**

**JOptionPane.showMessageDialog(null, "Начальная точка указана неверно");**

**return;**

**}**

**int endPoint = 0;**

**try {**

**endPoint = Integer.parseInt(endPointInput.getText());**

**} catch (NumberFormatException ex) {**

**JOptionPane.showMessageDialog(null, "Конечная точка точка указана неверно");**

**return;**

**}**

**if (endPoint < point || endPoint > n-1) {**

**JOptionPane.showMessageDialog(null, "Конечная точка указана неверно");**

**return;**

**}**

**double val = 0.0;**

**try {**

**val = Double.parseDouble(valueInput.getText());**

**} catch (NumberFormatException ex) {**

**JOptionPane.showMessageDialog(null, "Значение указано неверно");**

**return;**

**}**

**long max\_value = dn.getMaxValue();**

**if (val > max\_value) {**

**JOptionPane.showMessageDialog(null, String.format("Максимальное значение элемента: {%s}", max\_value));**

**return;**

**}**

**if (val < -max\_value) {**

**JOptionPane.showMessageDialog(null, String.format("Минимальное значение элемента: {%s}", -max\_value));**

**return;**

**}**

**try {**

**dn.updateValues(point, endPoint, val);**

**JOptionPane.showMessageDialog(null, "Значения изменены");**

**} catch (DecompositionException exc) {**

**JOptionPane.showMessageDialog(null, exc.getMessage());**

**}**

**}**

**}**

**}**

**ChangeValueDialog.java**

**package rpslab;**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**import java.awt.event.WindowAdapter;**

**import java.awt.event.WindowListener;**

**import java.awt.event.WindowEvent;**

**public class ChangeValueDialog extends JDialog {**

**private JButton exBtn = new JButton("Закрыть");**

**private JButton saveBtn = new JButton("Изменить");**

**private JTextField pointInput, valueInput;**

**private Decomposition dn;**

**public ChangeValueDialog(GUI parent, Decomposition dn) {**

**super(parent, "Изменить значение в точке", Dialog.ModalityType.DOCUMENT\_MODAL);**

**this.dn = dn;**

**this.setBounds(232, 232, 300, 200);**

**getContentPane().add(createGUI());**

**pack();**

**this.addWindowListener(closeWindow);**

**exBtn.addActionListener(new CloseHandler());**

**saveBtn.addActionListener(new ChangeHandler());**

**}**

**private JPanel createGUI() {**

**int n = dn.getLen();**

**JPanel panel = this.createVerticalPanel();**

**panel.setBorder (BorderFactory.createEmptyBorder(12,12,12,12));**

**JPanel pointPanel = this.createHorizontalPanel();**

**JLabel pointLabel = new JLabel(String.format("Точка [0-%d]:", n-1));**

**pointPanel.add(pointLabel);**

**pointPanel.add(Box.createHorizontalStrut(12));**

**pointInput = new JTextField(15);**

**pointPanel.add(pointInput);**

**JPanel valuePanel = this.createHorizontalPanel();**

**JLabel valueLabel = new JLabel("Значение:");**

**valuePanel.add(valueLabel);**

**valuePanel.add(Box.createHorizontalStrut(12));**

**valueInput = new JTextField(15);**

**valuePanel.add(valueInput);**

**JPanel grid = new JPanel( new GridLayout( 2,1, 0,7) );**

**grid.add(saveBtn);**

**grid.add(exBtn);**

**this.makeSameSize(new JComponent[] { pointLabel, valueLabel } );**

**panel.add(pointPanel);**

**panel.add(Box.createVerticalStrut(12));**

**panel.add(valuePanel);**

**panel.add(Box.createVerticalStrut(17));**

**panel.add(grid);**

**return panel;**

**}**

**public JPanel createVerticalPanel() {**

**JPanel panel = new JPanel();**

**panel.setLayout(new BoxLayout(panel, BoxLayout.Y\_AXIS));**

**return panel;**

**}**

**public JPanel createHorizontalPanel() {**

**JPanel panel = new JPanel();**

**panel.setLayout(new BoxLayout(panel, BoxLayout.X\_AXIS));**

**return panel;**

**}**

**private static WindowListener closeWindow = new WindowAdapter() {**

**public void windowClosing(WindowEvent e) {**

**e.getWindow().dispose();**

**}**

**};**

**public void makeSameSize(JComponent[] components) {**

**int[] array = new int[components.length];**

**for (int i = 0; i < array.length; i++) {**

**array[i] = components[i].getPreferredSize().width;**

**}**

**int maxSizePos = maximumElementPosition(array);**

**Dimension maxSize = components[maxSizePos].getPreferredSize();**

**for (int i=0; i<components.length; i++) {**

**components[i].setPreferredSize(maxSize);**

**components[i].setMinimumSize(maxSize);**

**components[i].setMaximumSize(maxSize);**

**}**

**}**

**private int maximumElementPosition(int[] array) {**

**int maxPos = 0;**

**for (int i = 1; i < array.length; i++) {**

**if (array[i] > array [maxPos])**

**maxPos = i;**

**}**

**return maxPos;**

**}**

**class CloseHandler implements ActionListener {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**dispose();**

**}**

**}**

**class ChangeHandler implements ActionListener {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**int point = 0;**

**try {**

**point = Integer.parseInt(pointInput.getText());**

**} catch (NumberFormatException ex) {**

**JOptionPane.showMessageDialog(null, "Точка указана неверно");**

**return;**

**}**

**int n = dn.getLen();**

**if (point < 0 || point > n-1) {**

**JOptionPane.showMessageDialog(null, "Точка указана неверно");**

**return;**

**}**

**double val = 0.0;**

**try {**

**val = Double.parseDouble(valueInput.getText());**

**} catch (NumberFormatException ex) {**

**JOptionPane.showMessageDialog(null, "Значение указано неверно");**

**return;**

**}**

**long max\_value = dn.getMaxValue();**

**if (val > max\_value) {**

**JOptionPane.showMessageDialog(null, String.format("Максимальное значение элемента: {%s}", max\_value));**

**return;**

**}**

**if (val < -max\_value) {**

**JOptionPane.showMessageDialog(null, String.format("Минимальное значение элемента: {%s}", -max\_value));**

**return;**

**}**

**try {**

**dn.updateValue(point, val);**

**JOptionPane.showMessageDialog(null, "Значение изменено");**

**} catch (DecompositionException exc) {**

**JOptionPane.showMessageDialog(null, exc.getMessage());**

**}**

**}**

**}**

**}**

Приложение Б.

Экранные формы

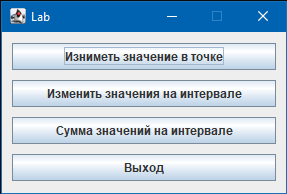


Рисунок 1 – Главное окно

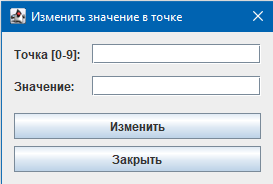


Рисунок 2 – Окно изменения значения в точке

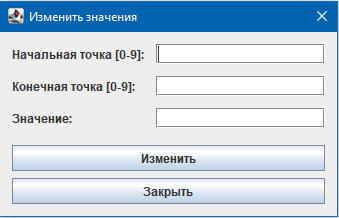


Рисунок 3 – Окно изменения значения на интервале

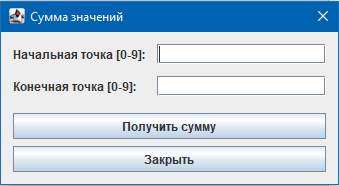


Рисунок 4 – Окно получения суммы значений на интервале