ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)»

Кафедра МОЭВМ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к практическому заданию №2

на тему:

«Визуализация алгоритмов сортировки.

Сортировка вставками и слиянием»

по дисциплине «Построение и анализ алгоритмов»

Выполнили: студенты группы №3304

Лурье Роман Андреевич

Черепкова Юлия Сергеевна

Факультет КТИ

Преподаватель:

Фирсов М.А.

Cанкт-Петербург

2015 г.

**Оглавление**

[**Формулировка задания** 3](#_Toc423517372)

[**Спецификация** 3](#_Toc423517373)

[**1.** **Алгоритмы сортировки, выбранные для реализации.** 3](#_Toc423517374)

[**2.** **Формат входных данных.** 3](#_Toc423517375)

[**3.** **Формат выходных данных** 3](#_Toc423517376)

[**4.** **Режимы работы программы** 3](#_Toc423517377)

[**5.** **Описание интерфейса** 4](#_Toc423517378)

[**Использованные структуры данных** 5](#_Toc423517379)

[**План разработки и распределение зон ответственности** 5](#_Toc423517380)

[**Код программы** 8](#_Toc423517381)

[ColorInt.java 8](#_Toc423517382)

[Data.java 8](#_Toc423517383)

[FileWorker.java 8](#_Toc423517384)

[Generation.java 10](#_Toc423517385)

[GenerationParam.java 13](#_Toc423517386)

[LogicIns.java 18](#_Toc423517387)

[MainFunction.java 22](#_Toc423517388)

[Sort.java 22](#_Toc423517389)

[SortAlgorithms.java 26](#_Toc423517390)

[SortEngine.java 31](#_Toc423517391)

[**Тестирование** 37](#_Toc423517392)

[**1.** **План тестирования** 37](#_Toc423517393)

[**2.** **Тестирование** 37](#_Toc423517394)

[**Заключение** 43](#_Toc423517395)

# **Формулировка задания**

Разработать программу, визуализирующую алгоритмы сортировки массива, на языке Java с использованием библиотек “Swing”, “AVT”. Программа должна иметь простой в использовании, удобный интерфейс.

# **Спецификация**

1. **Алгоритмы сортировки, выбранные для реализации.**

Как известно, существует множество вариантов сортировки массивов, различающихся как логикой, так и быстродействием. На начальном этапе разработки было принято решение визуализировать работу следующих алгоритмов:

* Сортировка вставками;
* Сортировка слиянием;

1. **Формат входных данных.**

Алгоритмы сортировки в качестве входных данных требуют неупорядоченный массив целочисленных значений. Данный массив можно будет получить несколькими вариантами:

* Используя файл;
* С помощью строки ввода;
* Автоматическая генерация;

Ограничения, накладываемые на входные данные:

* Массив, полученный из файла или из строки ввода, должен представлять собой последовательность натуральных чисел, разделенных пробелами.
* Обязательно указывать размерность, т.е. количество элементов массива.
* Максимальное значение элемента массива – 400.
* Максимальные количество элементов массива – 1000.

1. **Формат выходных данных**

Алгоритмы сортировки на выходе дают отсортированный массив целочисленных значений. Данный массив по желанию пользователя можно сохранить в файл.

Кроме того, процесс и результат сортировки отображаются на экране в виде столбиковой диаграммы, в которой элементы массива представляются в виде прямоугольников, вытянутых по вертикали, высота которых соответствует значению элементу.

1. **Режимы работы программы**

Программа может функционировать в следующих режимах:

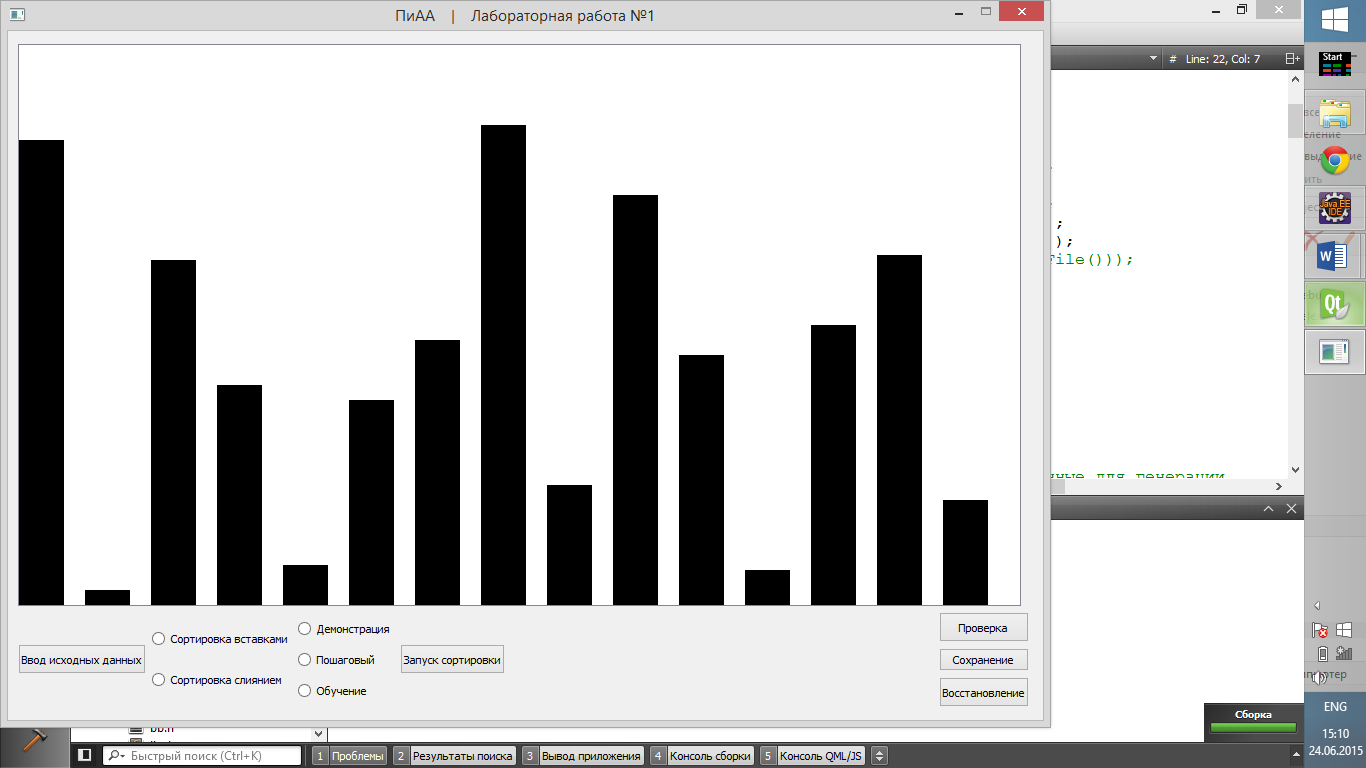
* Режим непрерывной демонстрации работы алгоритма;
* Режим пошаговая визуализации работы алгоритма.

1. **Описание интерфейса**

Разрабатываемый интерфейс должен учитывать требования, описанные в спецификации, а именно: поддержка всех указанных способов ввода, алгоритмов сортировки, режимов работы программы и сохранения результатов работы. Кроме того, визуализация должна предусматривать демонстрацию логики выполнения алгоритмов сортировки. Предполагаемый интерфейс изображен на рисунке 1.

Рисунок 1

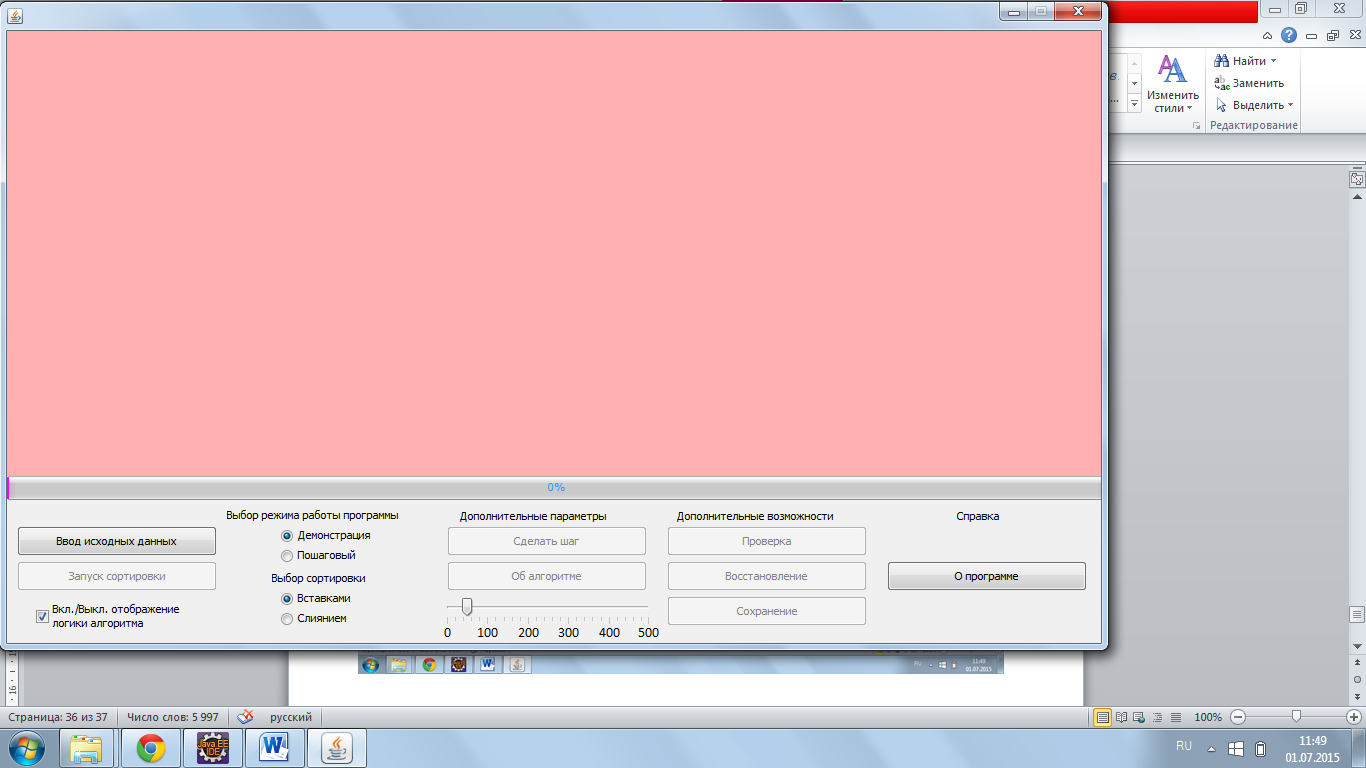
Предполагаемый интерфейс



Финальный интерфейс изображен на рисунке 2.

Рисунок 2

Финальный интерфейс



# **Использованные структуры данных**

Для хранения входных и выходных данных используется структура «массив». Память выделяется исходя из количества элементов входной последовательности, которое мы знаем заранее. Никаких дополнительных преобразований не производится.

# **План разработки и распределение зон ответственности**

|  |  |  |
| --- | --- | --- |
| Дата | Описание задачи | Исполнители |
| 24-25.06.2015 | Составление и согласование спецификации и плана разработки | Лурье Р.  Черепкова Ю. |
| Составление плана тестирования |
| Разработка прототипа программы, включающего в себя:  - генерация входных данных;  - ввод входных данных из файла;  - результаты корректно работающих алгоритмов сортировки (вывод в файл/экран);  - примитивный пользовательский интерфейс, обеспечивающий функционал прототипа. |
| Ведение промежуточной документации (отчет) |
| Реализация алгоритма сортировки слиянием | Лурье Р. |
| Визуализация (30%) |
| Генерация входных данных |
| Создание Git-репозитория, загрузка в него имеющихся спецификации и плана разработки |
| Реализация алгоритма сортировки вставками | Черепкова Ю. |
| Работа над интерфейсом (30%) |
| Ввод входных данных из файла |
| Загрузка разработанных материалов в Git-репозиторий |
| 26-28.04.2015 | Сдача прототипа программы | Лурье Р.  Черепкова Ю. |
| Корректировка плана тестирования |
| Разработка первой версии программы, включающей в себя:  - функционал прототипа;  - режим работы программы «Демонстраиця»;  - режим работы программы «Пошаговый»;  - примитивная визуализация входных данных;  - усовершенствованный интерфейс. |
| Ведение промежуточной документации (отчет) |
| Визуализация (50-70%) | Лурье Р. |
| Разработка режима программы «Пошаговый» |
| Загрузка разработанных материалов в Git-репозиторий |
| Разработка интерфейса (50-70%) | Черепкова Ю. |
| Разработка режима программы «Демонстрация» |
| Загрузка разработанных материалов в Git-репозиторий |
| 28-29.06.2015 | Сдача первой версии программы | Лурье Р.  Черепкова Ю. |
| Корректировка плана тестирования |
| Ведение промежуточной документации (отчет) |
| Интеграция разработанных интерфейса и визуализации |
| Разработка второй версии программы, включающей в себя:  - Функционал первой версии программы;  - Все способы ввода исходных данных, описанные в спецификации;  - Возможность сохранения результатов сортировки в файл;  - Конечный вариант визуализации алгоритмов сортировки; |
| «Ручной» ввод входных данных пользователем во время работы программы | Лурье Р. |
| Завершение работы над визуализацией |
| Добавление опции «Проверка на упорядоченность массива» |
| Загрузка разработанных материалов в Git-репозиторий |
| Добавление опции «Сохранить результаты сортировки в файл» | Черепкова Ю. |
| Завершение работы над интерфейсом |
| Загрузка разработанных материалов в Git-репозиторий |
| 30.06.2015-1.07.2015 | Корректировка плана тестирования | Лурье Р.  Черепкова Ю. |
| Окончательное тестирование |
| Завершение составления отчета |
| Сдача второй версии программы |
| Загрузка разработанных материалов в Git-репозиторий |

Резервные дни: **2.07.2015**, **3.07.15**

# **Код программы**

## ColorInt.java

**package** def;

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

**public** **class** ColorInt

{

Color InColor;

**public** **int** I;

ColorInt()

{

InColor = Color.***white***;

}

}

## Data.java

**package** def;

**public** **class** Data

{

**int**[] array;

String f\_Sort\_txt = "Сортировка";

String b\_input\_txt = "Ввод исходных данных";

String b\_start\_sort\_txt = "Запуск сортировки";

String b\_make\_step\_txt = "Сделать шаг";

String b\_check\_txt = "Проверка";

String b\_save\_txt = "Сохранение";

String b\_revival\_txt = "Восстановление";

String b\_opt\_for\_txt = "Выбрать";

String b\_file\_txt = "Файл";

String b\_string\_txt = "Строка ввода";

String b\_auto\_txt = "Автоматическая";

String b\_OK\_txt = "ОК";

String b\_cancel\_txt = "Отмена";

String b\_logic\_txt = "Об алгоритме";

String b\_help\_txt = "Помощь";

String b\_about\_prog\_txt = "О программе";

//окошко выбора сортировки

String b\_Insertion\_txt = "Сортировка вставками";

String b\_Merge\_txt = "Сортировка слиянием";

**static** **final** **int** ***const\_max*** = 1000;

Data()

{

}

}

## FileWorker.java

package def;

import java.io.File;

import java.io.PrintWriter;

import java.io.IOException;

import java.io.FileNotFoundException;

import java.io.BufferedReader;

import java.io.FileReader;

public class FileWorker {

public static File exists(String fileName) throws FileNotFoundException {

File file = new File(fileName);

if (!file.exists()){

throw new FileNotFoundException(file.getName());

}

else

{

return file;

}

}

public static File write(File file, String text) {

// Определяем файл

//File file = new File(fileName);

try {

// Проверяем, что если файл не существует, то создаем его

if(!file.exists()) {

file.createNewFile();

}

//PrintWriter обеспечит возможности записи в файл

PrintWriter out = new PrintWriter(file.getAbsoluteFile());

try {

//Записываем текст в файл

out.print(text);

out.close();

return file;

}

finally {

//После чего мы должны закрыть файл

//Иначе файл не запишется

out.close();

}

}

catch(IOException e) {

throw new RuntimeException(e);

}

}

public static String read(String fileName) throws FileNotFoundException {

//Этот спец. объект для построения строки

StringBuilder sb = new StringBuilder();

File file = exists(fileName);

try {

//Объект для чтения файла в буфер

BufferedReader in = new BufferedReader(new FileReader( file.getAbsoluteFile()));

try {

//В цикле построчно считываем файл

String s;

while ((s = in.readLine()) != null) {

sb.append(s);

sb.append("\n");

}

}

finally {

//Также не забываем закрыть файл

in.close();

}

} catch(IOException e) {

throw new RuntimeException(e);

}

//Возвращаем полученный текст с файла

return sb.toString();

}

public static void delete(String nameFile) throws FileNotFoundException {

exists(nameFile);

new File(nameFile).delete();

}

}

// В main

/\*import java.io.FileNotFoundException;

public class Main {

//private static String text = "This new text \nThis new text2\nThis new text3\nThis new text4\n";

//private static String fileName = "C:/Users/User/Workspace/Work1/a.txt";

public static void main(String[] args) throws FileNotFoundException{

//Чтение файла

//String textFromFile = FileWorker.read(fileName);

//System.out.println(textFromFile);

//Запись в файл

//FileWorker.write(fileName, text);

}

}\*/

## Generation.java

package def;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class Generation extends JDialog implements ActionListener {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

JPanel windowGeneration;

JButton button\_file;

JButton button\_string;

JButton button\_auto;

JButton button\_cancel;

JLabel label\_space;

Data dataGen;

private boolean result; // результат работы окна (true - нажатие кнопки OK, false - нажатие кнопки Отмена);

// А это для дочернего процесса дочернего

private GenerationParam dialog = null; //сслыка, пока пустая, на объект дочернего окна

public Generation(java.awt.Frame parent, Data data) {

super(parent, true);

dataGen = data;

// Создаем кнопочки

button\_file = new JButton(dataGen.b\_file\_txt);

button\_string = new JButton(dataGen.b\_string\_txt);

button\_auto = new JButton(dataGen.b\_auto\_txt);

button\_cancel = new JButton(dataGen.b\_cancel\_txt);

label\_space = new JLabel("- - - - - - - - - - - - - - - - - - - - -");

// Назначаем кнопочкам слушателей

button\_file.addActionListener(this);

button\_string.addActionListener(this);

button\_auto.addActionListener(this);

button\_cancel.addActionListener(this);

// Добавляем кнопочки на панель

windowGeneration = new JPanel();

windowGeneration.setLayout(null);

button\_file.setBounds(20, 20, 150, 30);

button\_string.setBounds(20, 60, 150, 30);

button\_auto.setBounds(20, 100, 150, 30);

label\_space.setBounds(20, 130, 150, 30);

button\_cancel.setBounds(20, 160, 150, 30);

windowGeneration.add(button\_file);

windowGeneration.add(button\_string);

windowGeneration.add(button\_auto);

windowGeneration.add(label\_space);

windowGeneration.add(button\_cancel);

getContentPane().add(windowGeneration);

setPreferredSize(new Dimension(190, 230));

this.add(windowGeneration);

// Сделать размер окна подходящим

this.pack();

// Запрет на изменение размера экрана

this.setResizable(false);

//Точка размещения экрана

Point center = GraphicsEnvironment.getLocalGraphicsEnvironment().getCenterPoint();

int w = this.getWidth();

int h = this.getHeight();

center.x = center.x - w/2;

center.y = center.y - h/2;

this.setLocation(center);

}

public boolean execute() { // метод для "общения" с родительским окном

this.setVisible(true); // прорисовать дочернее окно; в данном месте выполнение программы приостановится, ожидая окончания работы пользователя (в нашем случае - функции dispose())

return this.result; // вернуть в качестве результата условное значение нажатой кнопки

}

public boolean readArray(File file, int real\_length) {

try {

Scanner in = new Scanner(file);

real\_length = 0;

int size = 0;

if (in.hasNextInt())

size = in.nextInt();

else {

in.close();

return false;

}

dataGen.array = new int[size];

if (in.hasNextInt()) {

while(in.hasNextInt()){

dataGen.array[real\_length] = in.nextInt();

real\_length++;

}

in.close();

return true;

}

else {

in.close();

return false;

}

}

catch (FileNotFoundException ex) {

return false;

}

}

public void actionPerformed(ActionEvent e) {

// Получение источника события

JButton clickedButton = (JButton) e.getSource();

String actioncommand = clickedButton.getActionCommand();

// КНОПКА "ФАЙЛ"

if (actioncommand == dataGen.b\_file\_txt) {

// Создаем новый объект

// Отображаем диалог пользователю

// В ret заносится значение, по которому мы можем понять, что хочет пользователь

JFileChooser fileopen = new JFileChooser();

fileopen.setDialogTitle("Open file");

int ret = fileopen.showOpenDialog(this);

if (ret == JFileChooser.APPROVE\_OPTION) {

File file = fileopen.getSelectedFile();

int real\_length = 0;

// Считываем массив тут

boolean inforWasRead = readArray(file, real\_length);

if ((dataGen.array == null) || (!inforWasRead))

{ //если массив пустой, тогда

JOptionPane.showMessageDialog(null, "При чтении с файла произошла ошибка", " ", JOptionPane.ERROR\_MESSAGE);

return;

}

else {

//custom title, no icon

JOptionPane.showMessageDialog(this, "Данные считаны успешно!"," ", JOptionPane.PLAIN\_MESSAGE);

}

this.result = true;

this.dispose(); // уничтожить окно

}

}

// КНОПКА "СТРОКА ВВОДА"

if (actioncommand == dataGen.b\_string\_txt) {

this.dialog = new GenerationParam(this, dataGen, 1);

if (this.dialog.executeParam()) {

// действия при нажатии клавиши ОК

// здесь же прописывается считывание нужных результатов, введённых пользователем, используя функции get, заблаговременно прописанные вами в класе дочернего окна

JOptionPane.showMessageDialog(this, "Данные считаны успешно!"," ", JOptionPane.PLAIN\_MESSAGE);

this.result = true;

this.dispose(); // уничтожить окно

}

else {

// Ничего не происходит ^^

}

}

// КНОПКА "АВТОМАТИЧЕСКАЯ"

if (actioncommand == dataGen.b\_auto\_txt) {

this.dialog = new GenerationParam(this, dataGen, 2);

if (this.dialog.executeParam()) {

// действия при нажатии клавиши ОК

// здесь же прописывается считывание нужных результатов, введённых пользователем, используя функции get, заблаговременно прописанные вами в класе дочернего окна

JOptionPane.showMessageDialog(this, "Данные считаны успешно!"," ", JOptionPane.PLAIN\_MESSAGE);

this.result = true;

this.dispose(); // уничтожить окно

}

else {

// Ничего не происходит ^^

}

}

// КНОПКА "ОТМЕНА"

if (actioncommand == dataGen.b\_cancel\_txt) {

this.result = false; // пользователь нажал Отмена

this.dispose(); // уничтожить окно

}

}

}

## GenerationParam.java

package def;

import java.awt.\*;

import java.awt.event.\*;

import java.io.FileNotFoundException;

import java.util.Scanner;

import javax.swing.\*;

import javax.swing.event.ChangeEvent;

import javax.swing.event.ChangeListener;

import java.util.Random;

import java.io.File;

public class GenerationParam extends JDialog implements ActionListener {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private boolean resultParam; // результат работы окна (true - нажатие кнопки OK, false - нажатие кнопки Отмена);

int mode; // Режим работы вспомогательного окна

// Данные для режимов

JPanel p1;

JPanel p2;

JPanel allP;

JLabel label1;

JLabel label2;

JLabel label\_space;

JSpinner spin;

JSpinner spin1;

JSpinner spin2;

private JTextField displayField;

JButton button\_OK;

JButton button\_cancel;

Data dataGen;

String forArray;

public void setDisplayValue(String val){

displayField.setText(val);

}

public String getDisplayValue() {

return displayField.getText();

}

ChangeListener listener = new ChangeListener() {

public void stateChanged(ChangeEvent e) {

Object val = spin.getValue();

if (val instanceof Integer) {

int result = (int) val;

if (result > Data.const\_max) {

Component c = spin.getEditor().getComponent(0);

c.setBackground(Color.RED);

}

else {

Component c = spin.getEditor().getComponent(0);

c.setBackground(Color.WHITE);

}

}

}

};

public GenerationParam(Generation parent, Data data, int md) {

super(parent, true);

dataGen = data;

mode= md;

// РЕЖИМ 1

if (mode == 1) {

// Лэйбл и спик

label1 = new JLabel("Задайте размер массива");

SpinnerModel model =

new SpinnerNumberModel(1, //initial value

1, //min

Data.const\_max, //max

1); //step

spin = new JSpinner(model);

spin.addChangeListener(listener);

// Лейбл + строка ввода + 2 кнопки

label2 = new JLabel("Введите элементы массива через пробел");

displayField = new JTextField(100);

button\_OK = new JButton(data.b\_OK\_txt);

button\_cancel = new JButton(data.b\_cancel\_txt);

label\_space = new JLabel("- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -");

// Назначаем кнопочкам слушателей

button\_cancel.addActionListener(this);

button\_OK .addActionListener(this);

// Добавляем кнопочки на панель

p1 = new JPanel();

p1.setLayout(null);

label1.setBounds(20, 20, 150, 20);

spin.setBounds(200, 20, 50, 30);

label2.setBounds(20, 50, 250, 30);

displayField.setBounds(20, 90, 400, 30);

label\_space.setBounds(20, 125, 400, 20);

button\_OK.setBounds(40, 150, 150, 30);

button\_cancel.setBounds(250, 150, 150, 30);

p1.add(label1);

p1.add(spin);

p1.add(label2);

p1.add(displayField);

p1.add(label\_space);

p1.add(button\_OK);

p1.add(button\_cancel);

getContentPane().add(p1);

setPreferredSize(new Dimension(440, 220));

this.add(p1);

// Сделать размер окна подходящим

this.pack();

// Запрет на изменение размера экрана

this.setResizable(false);

//Точка размещения экрана

Point center = GraphicsEnvironment.getLocalGraphicsEnvironment().getCenterPoint();

int w = this.getWidth();

int h = this.getHeight();

center.x = center.x - w/2;

center.y = center.y - h/2;

this.setLocation(center);

}

// РЕЖИМ 2

if (mode == 2) {

label\_space = new JLabel(" ");

// Создаем лейбл + спин

label1 = new JLabel("Задайте размер массива");

SpinnerModel model =

new SpinnerNumberModel(100, //initial value

1, //min

Data.const\_max, //max

1); //step

spin = new JSpinner(model);

spin.addChangeListener(listener);

// Создаем лейбл + спин + спин

label2 = new JLabel("Задайте минимальное и максимальное значение элементов");

SpinnerModel model1 =

new SpinnerNumberModel(1, //initial value

1, //min

Integer.MAX\_VALUE, //max

1); //step

spin1 = new JSpinner(model1);

//spin.addChangeListener(listener);

SpinnerModel model2 =

new SpinnerNumberModel(300, //initial value

1, //min

Integer.MAX\_VALUE, //max

1); //step

spin2 = new JSpinner(model2);

//spin.addChangeListener(listener);

// Создаем 2 кнопочки

button\_OK = new JButton(data.b\_OK\_txt);

button\_cancel = new JButton(data.b\_cancel\_txt);

// Назначаем кнопочкам слушателей

button\_cancel.addActionListener(this);

button\_OK .addActionListener(this);

label\_space = new JLabel("- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -");

// Добавляем кнопочки на панель

p1 = new JPanel();

p1.setLayout(null);

label1.setBounds(20, 20, 150, 20);

spin.setBounds(200, 20, 50, 30);

label2.setBounds(20, 50, 400, 30);

spin1.setBounds(200, 90, 50, 30);

spin2.setBounds(270, 90, 50, 30);

label\_space.setBounds(20, 130, 400, 20);

button\_OK.setBounds(40, 155, 150, 30);

button\_cancel.setBounds(250, 155, 150, 30);

p1.add(label1);

p1.add(spin);

p1.add(label2);

p1.add(spin1);

p1.add(spin2);

p1.add(label\_space);

p1.add(button\_OK);

p1.add(button\_cancel);

getContentPane().add(p1);

setPreferredSize(new Dimension(440, 225));

this.add(p1);

// Сделать размер окна подходящим

this.pack();

// Запрет на изменение размера экрана

this.setResizable(false);

//Точка размещения экрана

Point center = GraphicsEnvironment.getLocalGraphicsEnvironment().getCenterPoint();

int w = this.getWidth();

int h = this.getHeight();

center.x = center.x - w/2;

center.y = center.y - h/2;

this.setLocation(center);

}

}

public boolean executeParam() { // метод для "общения" с родительским окном

this.setVisible(true); // прорисовать дочернее окно; в данном месте выполнение программы приостановится, ожидая окончания работы пользователя (в нашем случае - функции dispose())

return this.resultParam; // вернуть в качестве результата условное значение нажатой кнопки

}

public void actionPerformed(ActionEvent e) {

// Получение источника события

JButton clickedButton = (JButton) e.getSource();

String actioncommand = clickedButton.getActionCommand();

// КНОПКА "ОК"

if (actioncommand == dataGen.b\_OK\_txt) {

// РЕЖИМ 1

if (mode == 1) {

// Восстановление цвета

displayField.setBackground(Color.WHITE);

// Считываем размер массива

Object val = spin.getValue();

int result = 0;

if (val instanceof Integer) {

result = (int) val;

dataGen.array = new int[result];

// А теперь поехали читать массив

forArray = displayField.getText();

File file = new File ("forArray.txt");

file = FileWorker.write(file, forArray);

try {

Scanner in = new Scanner(file);

int real\_length = 0;

if (in.hasNextInt()) {

while ((in.hasNextInt()) && (dataGen.array.length > real\_length)) {

dataGen.array[real\_length] = in.nextInt();

real\_length++;

}

if ((dataGen.array.length == real\_length) && (!in.hasNextInt())) {

in.close();

this.resultParam = true;

this.dispose(); // уничтожить окно

}

else {

in.close();

JOptionPane.showMessageDialog(this, "Размер массива не соответствует количеству элементов, заданных в строке",

"",

JOptionPane.WARNING\_MESSAGE);

Component c = spin.getEditor().getComponent(0);

c.setBackground(Color.RED);

}

}

else {

in.close();

JOptionPane.showMessageDialog(this, "Строка пуста!", " ", JOptionPane.ERROR\_MESSAGE);

displayField.setBackground(Color.RED);

}

}

catch (FileNotFoundException ex) {

JOptionPane.showMessageDialog(this, "При чтении исходных данных произошла ошибка", " ", JOptionPane.ERROR\_MESSAGE);

}

}

else

JOptionPane.showMessageDialog(this, "Некорректный размер массива", " ", JOptionPane.ERROR\_MESSAGE);

}

// РЕЖИМ 2

if (mode == 2) {

// Считываем размер массива

Object val = spin.getValue();

int result = 0;

if (val instanceof Integer) {

result = (int) val;

dataGen.array = new int[result];

// Считываем минимальное значение

val = spin1.getValue();

int param\_min = 1;

if (val instanceof Integer) {

param\_min = (int) val;

// Считываем максимальное значение

val = spin2.getValue();

int param\_max = Integer.MAX\_VALUE;

if (val instanceof Integer) {

param\_max = (int) val;

// А теперь поехали генерировать значения в массив

Random r = new Random();

for(int j = 0; j < dataGen.array.length; j++)

dataGen.array[j] = r.nextInt(param\_max)+param\_min;

this.resultParam = true; // пользователь нажал ОК

this.dispose(); // уничтожить окно

}

}

}

}

}

// КНОПКА "ОТМЕНА"

if (actioncommand == dataGen.b\_cancel\_txt) {

this.resultParam = false; // пользователь нажал Отмена

this.dispose(); // уничтожить окно

}

}

}

## LogicIns.java

package def;

import java.awt.\*;

import javax.swing.\*;

import javax.swing.text.DefaultCaret;

public class LogicIns extends JFrame {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

JPanel pl;

static JLabel label\_general;

static JLabel label\_general\_text;

static JLabel label\_legend;

static JLabel label\_legend\_text;

static JTextField tf\_red;

static JTextField tf\_yellow;

static JTextField tf\_green;

static JTextField tf\_blue;

JTextArea ta;

String strLog = "";

Sort parentSE;

LogicIns(Sort se) {

parentSE = se;

// ЕСЛИ СОРТИРОВКА ВСТАВКАМИ

if (se.typeOfSort == true) {

label\_general = new JLabel("Общие сведения");

label\_general\_text = new JLabel("<HTML>Сортировка вставками (англ. Insertion sort) — алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов. Вычислительная сложность - O(n^2).");

label\_legend = new JLabel("Условные обозначения");

label\_legend\_text = new JLabel("<HTML> - следующий элемент <br> - текущий элемент <br> - элемент меньше текущего <br> - элемент меньше текущегоэлемент больше или равен текущему ");

Font font = new Font("Calibri", Font.BOLD , 14);

label\_general.setFont(font);

label\_legend.setFont(font);

Font fontAD = new Font("Calibri", Font.PLAIN , 14);

label\_general\_text.setFont(fontAD);

label\_legend\_text.setFont(fontAD);

tf\_red = new JTextField();

tf\_yellow = new JTextField();

tf\_green = new JTextField();

tf\_blue = new JTextField();

tf\_red.setBackground(Color.RED);

tf\_yellow.setBackground(Color.YELLOW);

tf\_green.setBackground(Color.GREEN);

tf\_blue.setBackground(Color.CYAN);

ta = new JTextArea("Пояснения\n", 450, 150);

// Добавляем на панель

pl = new JPanel();

pl .setLayout(null);

label\_general.setBounds(20, 20, 150, 15);

label\_general\_text.setBounds(20, 40, 450, 85);

label\_legend.setBounds(20, 135, 150, 15);

label\_legend\_text.setBounds(40, 155, 450, 70);

tf\_red.setBounds(20, 160, 15, 15);

tf\_yellow.setBounds(20, 177, 15, 15);

tf\_blue.setBounds(20, 195, 15, 15);

tf\_green.setBounds(20, 212, 15, 15);

tf\_red.setEnabled(false);

tf\_yellow.setEnabled(false);

tf\_green.setEnabled(false);

tf\_blue.setEnabled(false);

DefaultCaret caret = (DefaultCaret) ta.getCaret();

caret.setUpdatePolicy(DefaultCaret.ALWAYS\_UPDATE);

JScrollPane scrollingArea = new JScrollPane(ta,JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS, JScrollPane.HORIZONTAL\_SCROLLBAR\_NEVER);

scrollingArea.setBounds(20, 237, 450, 150);

ta.setCaret(caret);

scrollingArea.setViewportView(ta);

pl.add(label\_general);

pl.add(label\_legend);

pl.add(label\_general\_text);

pl.add(label\_legend\_text);

pl.add(tf\_red);

pl.add(tf\_yellow);

pl.add(tf\_green);

pl.add(tf\_blue);

pl.add(scrollingArea);

getContentPane().add(pl);

setPreferredSize(new Dimension(500, 430));

// Устанавливаем размер окна достаточно большим, чтобы разместить на нем все нужные объекты

this.pack();

// Запрет на изменение размера экрана

this.setResizable(false);

//Точка размещения экрана

Point center = GraphicsEnvironment.getLocalGraphicsEnvironment().getCenterPoint();

int w = this.getWidth();

int h = this.getHeight();

center.x = center.x - w/2;

center.y = center.y - h/2;

this.setLocation(center);

}

// ЕСЛИ СОРТИРОВКА СЛИЯНИЕМ

else

{

label\_general = new JLabel("Общие сведения");

label\_general\_text = new JLabel("<HTML>Сортировка слиянием (англ. Merge sort) — алгоритм сортировки, который упорядочивает последовательность в определённом порядке. Эта сортировка — хороший пример использования принципа «разделяй и властвуй». Сначала задача разбивается на несколько подзадач меньшего размера. Затем эти задачи решаются с помощью рекурсивного вызова или непосредственно, если их размер достаточно мал. Наконец, их решения комбинируются, и получается решение исходной задачи. <br>Вычислительная сложность - O(n log n)");

label\_legend = new JLabel("Условные обозначения");

label\_legend\_text = new JLabel("<HTML> - правая часть в текущем разбиении <br> - левая часть в предыдущем разбиении <br> - левая часть в текущем разбиении <br> Разбиением называется часть массива, сортируемая на данном этапе");

Font font = new Font("Calibri", Font.BOLD , 14);

label\_general.setFont(font);

label\_legend.setFont(font);

Font fontAD = new Font("Calibri", Font.PLAIN , 14);

label\_general\_text.setFont(fontAD);

label\_legend\_text.setFont(fontAD);

tf\_yellow = new JTextField();

tf\_green = new JTextField();

tf\_blue = new JTextField();

tf\_yellow.setBackground(Color.YELLOW);

tf\_green.setBackground(Color.GREEN);

tf\_blue.setBackground(Color.CYAN);

ta = new JTextArea("Пояснения\n", 450, 90);

// Добавляем на панель

pl = new JPanel();

pl .setLayout(null);

label\_general.setBounds(20, 20, 150, 15);

label\_general\_text.setBounds(20, 40, 450, 140);

label\_legend.setBounds(20, 190, 150, 15);

label\_legend\_text.setBounds(40, 210, 450, 70);

tf\_yellow.setBounds(20, 215, 15, 15);

tf\_green.setBounds(20, 232, 15, 15);

tf\_blue.setBounds(20, 250, 15, 15);

tf\_yellow.setEnabled(false);

tf\_green.setEnabled(false);

tf\_blue.setEnabled(false);

DefaultCaret caret = (DefaultCaret) ta.getCaret();

caret.setUpdatePolicy(DefaultCaret.ALWAYS\_UPDATE);

JScrollPane scrollingArea = new JScrollPane(ta,JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS, JScrollPane.HORIZONTAL\_SCROLLBAR\_NEVER);

scrollingArea.setBounds(20, 295, 450, 90);

ta.setCaret(caret);

scrollingArea.setViewportView(ta);

pl.add(label\_general);

pl.add(label\_legend);

pl.add(label\_general\_text);

pl.add(label\_legend\_text);

pl.add(tf\_yellow);

pl.add(tf\_green);

pl.add(tf\_blue);

pl.add(scrollingArea);

getContentPane().add(pl);

setPreferredSize(new Dimension(500, 430));

// Устанавливаем размер окна достаточно большим, чтобы разместить на нем все нужные объекты

this.pack();

// Запрет на изменение размера экрана

this.setResizable(false);

//Точка размещения экрана

Point center = GraphicsEnvironment.getLocalGraphicsEnvironment().getCenterPoint();

int w = this.getWidth();

int h = this.getHeight();

center.x = center.x - w/2;

center.y = center.y - h/2;

this.setLocation(center);

}

}

}

## MainFunction.java

**package** def;

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

**public** **class** MainFunction {

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

{

**try** {

// Set cross-platform Java L&F (also called "Metal")

UIManager.*setLookAndFeel*(

UIManager.*getSystemLookAndFeelClassName*());

}

**catch** (UnsupportedLookAndFeelException e) {

// handle exception

}

**catch** (ClassNotFoundException e) {

// handle exception

}

**catch** (InstantiationException e) {

// handle exception

}

**catch** (IllegalAccessException e) {

// handle exception

}

Data data = **new** Data();

**new** Sort(data);

}

}

## Sort.java

**package** def;

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

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

**public** **class** Sort **extends** JFrame {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**static** **final** **int** ***FPS\_MIN*** = 0;

**static** **final** **int** ***FPS\_MAX*** = 500;

**static** **final** **int** ***FPS\_INIT*** = 50; //initial frames per second

// Объявление компонент

JPanel p1;

VisPanel vpanel;

**static** JButton *button\_input*;

**static** JButton *button\_start\_sort*;

**static** JButton *button\_make\_step*;

**static** JButton *button\_check*;

**static** JButton *button\_save*;

**static** JButton *button\_revival*;

**static** JButton *button\_logic*;

**static** JButton *button\_help*;

**static** JButton *button\_about\_prog*;

**static** JRadioButton *r\_demo\_mode*;

**static** JRadioButton *r\_step\_mode*;

**static** JLabel *label\_opt\_for\_mode*;

**static** JRadioButton *r\_ins\_s*;

**static** JRadioButton *r\_mer\_s*;

**static** JLabel *label\_opt\_for\_sort*;

**static** JLabel *label\_posib*;

**static** JLabel *label\_option\_step*;

**static** JLabel *label\_help*;

JProgressBar progressBar;

JSlider slider;

JCheckBox needToPaint;

**public** LogicIns L = **null**;

**public** **boolean** typeOfSort;

**public** **int** SleepTime = 50;

// Конструктор нам все нарисует как надо

@SuppressWarnings("deprecation")

Sort(Data data) {

vpanel = **new** VisPanel(**this**);

vpanel.setPreferredSize(**new** Dimension(1100, 445));

vpanel.setBackground(Color.***PINK***);

// Создаем кнопочки (с помощью конструктора им можно задать имя)

*button\_input* = **new** JButton("Cursor.HAND\_CURSOR");

*button\_start\_sort* = **new** JButton("Cursor.HAND\_CURSOR");

*button\_make\_step* = **new** JButton("Cursor.HAND\_CURSOR");

*button\_check* = **new** JButton("Cursor.HAND\_CURSOR");

*button\_save* = **new** JButton("Cursor.HAND\_CURSOR");

*button\_revival* = **new** JButton("Cursor.HAND\_CURSOR");

*button\_logic* = **new** JButton("Cursor.HAND\_CURSOR");

*button\_help* = **new** JButton("Cursor.HAND\_CURSOR");

*button\_about\_prog* = **new** JButton("Cursor.HAND\_CURSOR");

*button\_input*.~~setLabel~~(data.b\_input\_txt);

*button\_input*.setCursor(Cursor.*getPredefinedCursor*(Cursor.***HAND\_CURSOR***));

*button\_start\_sort*.~~setLabel~~(data.b\_start\_sort\_txt);

*button\_start\_sort*.setCursor(Cursor.*getPredefinedCursor*(Cursor.***HAND\_CURSOR***));

*button\_make\_step*.~~setLabel~~(data.b\_make\_step\_txt);

*button\_make\_step*.setCursor(Cursor.*getPredefinedCursor*(Cursor.***HAND\_CURSOR***));

*button\_check*.~~setLabel~~(data.b\_check\_txt);

*button\_check*.setCursor(Cursor.*getPredefinedCursor*(Cursor.***HAND\_CURSOR***));

*button\_save*.~~setLabel~~(data.b\_save\_txt);

*button\_save*.setCursor(Cursor.*getPredefinedCursor*(Cursor.***HAND\_CURSOR***));

*button\_revival*.~~setLabel~~(data.b\_revival\_txt);

*button\_revival*.setCursor(Cursor.*getPredefinedCursor*(Cursor.***HAND\_CURSOR***));

*button\_logic*.~~setLabel~~(data.b\_logic\_txt);

*button\_logic*.setCursor(Cursor.*getPredefinedCursor*(Cursor.***HAND\_CURSOR***));

*button\_help*.~~setLabel~~(data.b\_help\_txt);

*button\_help*.setCursor(Cursor.*getPredefinedCursor*(Cursor.***HAND\_CURSOR***));

*button\_about\_prog*.~~setLabel~~(data.b\_about\_prog\_txt);

*button\_about\_prog*.setCursor(Cursor.*getPredefinedCursor*(Cursor.***HAND\_CURSOR***));

*r\_demo\_mode* = **new** JRadioButton("Демонстрация");

*r\_demo\_mode*.setActionCommand("Демонстрация");

*r\_demo\_mode*.setSelected(**true**);

*r\_step\_mode* = **new** JRadioButton("Пошаговый");

*r\_step\_mode*.setActionCommand("Пошаговый");

ButtonGroup group = **new** ButtonGroup();

group.add(*r\_demo\_mode*);

group.add(*r\_step\_mode*);

*label\_opt\_for\_mode* = **new** JLabel("Выбор режима работы программы");

*r\_ins\_s* = **new** JRadioButton("Вставками");

*r\_ins\_s*.setActionCommand("Вставками");

*r\_ins\_s*.setSelected(**true**);

*r\_mer\_s* = **new** JRadioButton("Слиянием");

*r\_mer\_s*.setActionCommand("Слиянием");

ButtonGroup group\_s = **new** ButtonGroup();

group\_s.add(*r\_ins\_s*);

group\_s.add(*r\_mer\_s*);

*label\_opt\_for\_sort* = **new** JLabel("Выбор сортировки");

*label\_posib* = **new** JLabel("Дополнительные возможности");

*label\_option\_step* = **new** JLabel("Дополнительные параметры");

*label\_help* = **new** JLabel("Справка");

progressBar = **new** JProgressBar();

progressBar.setBackground(Color.***MAGENTA***);

progressBar.setForeground(Color.***MAGENTA***);

progressBar.setMinimum(0);

progressBar.setMaximum(100);

progressBar.setStringPainted(**true**);

slider = **new** JSlider(JSlider.***HORIZONTAL***, ***FPS\_MIN***, ***FPS\_MAX***, ***FPS\_INIT***);

slider.setMajorTickSpacing(100);

slider.setMinorTickSpacing(20);

slider.setPaintTicks(**true**);

slider.setPaintLabels(**true**);

Font font = **new** Font("Calibri", Font.***PLAIN***, 14);

slider.setFont(font);

needToPaint = **new** JCheckBox("<HTML>Вкл./Выкл. отображение <br>логики алгоритма");

needToPaint.setSelected(**true**);

// Добавляем кнопочки на панель

p1 = **new** JPanel();

p1 .setLayout(**null**);

vpanel.setBounds(0, 0, 1100, 445);

progressBar.setBounds(0, 445, 1100, 25);

*button\_input*.setBounds(10, 495, 200,30);

*label\_opt\_for\_mode*.setBounds(220, 472, 250, 25);

*r\_demo\_mode*.setBounds(270, 492, 150, 25);

*r\_step\_mode*.setBounds(270, 512, 150, 25);

*label\_opt\_for\_sort*.setBounds(265, 535, 250, 25);

*r\_ins\_s*.setBounds(270, 555, 150, 25);

*r\_mer\_s*.setBounds(270, 575, 150, 25);

*button\_start\_sort*.setBounds(10, 530, 200,30);

*button\_make\_step*.setBounds(440, 495, 200, 30);

*button\_logic*.setBounds(440, 530, 200, 30);

slider.setBounds(428, 565, 226, 50);

*button\_check*.setBounds(660, 495, 200, 30);

*button\_save*.setBounds(660, 565, 200, 30);

*button\_revival*.setBounds(660, 530, 200, 30);

*label\_posib*.setBounds(670, 470, 200, 30);

*label\_option\_step*.setBounds(453, 470, 200, 30);

*label\_help*.setBounds(950, 470, 200, 30);

needToPaint.setBounds(25, 570, 180,30);

//button\_help.setBounds(880, 495, 200, 30);

*button\_about\_prog*.setBounds(880, 530, 200, 30);

p1.add(vpanel);

p1.add(progressBar);

p1.add(*button\_input*);

p1.add(*label\_opt\_for\_mode*);

p1.add(*r\_demo\_mode*);

p1.add(*r\_step\_mode*);

p1.add(*label\_opt\_for\_sort*);

p1.add(*r\_ins\_s*);

p1.add(*r\_mer\_s*);

p1.add(*button\_start\_sort*);

p1.add(*button\_make\_step*);

p1.add(*button\_logic*);

p1.add(slider);

p1.add(*button\_check*);

p1.add(*button\_save*);

p1.add(*button\_revival*);

p1.add(*label\_posib*);

p1.add(*label\_option\_step*);

p1.add(*label\_help*);

p1.add(needToPaint);

p1.add(*button\_help*);

p1.add(*button\_about\_prog*);

// В начале блокируем все кнопки, кроме одной

*button\_start\_sort*.setEnabled(**false**);

*button\_make\_step*.setEnabled(**false**);

*button\_check*.setEnabled(**false**);

*button\_save*.setEnabled(**false**);

*button\_revival*.setEnabled(**false**);

*button\_logic*.setEnabled(**false**);

getContentPane().add(p1);

setPreferredSize(**new** Dimension(1100, 640));

// Тут подключаем движок

SortEngine sEngine = **new** SortEngine(**this**, data);

*button\_input*.addActionListener(sEngine);

*button\_start\_sort*.addActionListener(sEngine);

*button\_check*.addActionListener(sEngine);

*button\_make\_step*.addActionListener(sEngine);

*button\_revival*.addActionListener(sEngine);

*button\_save*.addActionListener(sEngine);

*button\_logic*.addActionListener(sEngine);

slider.addChangeListener(sEngine);

*button\_help*.addActionListener(sEngine);

*button\_about\_prog*.addActionListener(sEngine);

// Подключаем всплывающие подсказки

vpanel.setToolTipText("Визуализация сортировки");

progressBar.setToolTipText("Прогресс сортировки");

slider.setToolTipText("Задержка отрисовки");

*button\_input*.setToolTipText("<HTML>Нажми для ввода исходных данных <br>из файла, строки ввода или автоматически <br>с заданием параметров генерации");

*button\_start\_sort*.setToolTipText("<HTML>Нажми для запуска сортировки (кнопка <br>доступна при наличии исходных данных)");

*button\_make\_step*.setToolTipText("<HTML>Нажми, чтобы сделать шаг алгоритма (кнопка <br>доступна только в пошаговом режиме)");

*button\_check*.setToolTipText("<HTML>Нажми для проверки массива <br>на упорядоченность по неубыванию");

*button\_save*.setToolTipText("<HTML>Нажми для сохранения отсортированного <br>массива ");

*button\_revival*.setToolTipText("<HTML>Нажми для восстановления <br>исходных данных");

*button\_logic*.setToolTipText("<HTML>Нажми для получения объяснения <br>работы алгоритма");

*button\_help*.setToolTipText("Нажми для получения помощи");

*button\_about\_prog*.setToolTipText("<HTML>Нажми для получения общей информации <br>о программе");

ToolTipManager.*sharedInstance*().setInitialDelay(0);

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

// Устанавливаем размер окна достаточно большим, чтобы разместить на нем все нужные объекты

**this**.pack();

// Запрет на изменение размера экрана

**this**.setResizable(**false**);

//Точка размещения экрана

Point center = GraphicsEnvironment.*getLocalGraphicsEnvironment*().getCenterPoint();

**int** w = **this**.getWidth();

**int** h = **this**.getHeight();

center.x = center.x - w/2;

center.y = center.y - h/2;

**this**.setLocation(center);

// Отображаю окно

**this**.setVisible(**true**);

L = **new** LogicIns(**this**);

L.setDefaultCloseOperation(JFrame.***HIDE\_ON\_CLOSE***);

}

}

## SortAlgorithms.java

package def;

import java.awt.Color;

import javax.swing.JOptionPane;

public class SortAlgorithms

{

public static class Check extends Thread

{

Sort parent;

Data dataEngine;

Check(Sort S, Data D)

{

parent = S;

dataEngine = D;

}

public void run()

{

if(parent.vpanel.PanelArray == null)

{

parent.vpanel.PanelArray = new ColorInt[dataEngine.array.length];

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

{

parent.vpanel.PanelArray[i] = new ColorInt();

parent.vpanel.PanelArray[i].I = dataEngine.array[i];

}

}

boolean checkbool = true;

//сохраняем старые цвета: вдруг пригодятся.

Color[] ColorArray = new Color[parent.vpanel.PanelArray.length];

for (int i = 0; i < parent.vpanel.PanelArray.length; i++)

{

ColorArray[i] = new Color(0);

ColorArray[i] = parent.vpanel.PanelArray[i].InColor;

}

//

parent.vpanel.PanelArray[0].InColor = Color.GREEN;

for (int i = 1; i < parent.vpanel.PanelArray.length; i++)

{

if(parent.vpanel.PanelArray[i-1].I > parent.vpanel.PanelArray[i].I)

{

JOptionPane.showMessageDialog(parent, "Массив НЕ упорядочен!"," ", JOptionPane.INFORMATION\_MESSAGE);

checkbool = false;

break;

}

parent.vpanel.PanelArray[i].InColor = Color.GREEN;

try {

sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

parent.vpanel.repaint();

}

if(checkbool)

JOptionPane.showMessageDialog(parent, "Массив упорядочен!"," ", JOptionPane.INFORMATION\_MESSAGE);

//возвращаем потерянные цвета.

for (int i = 0; i < parent.vpanel.PanelArray.length; i++)

parent.vpanel.PanelArray[i].InColor = ColorArray[i];

//

parent.vpanel.repaint();

}

}

public static class InsertionSort extends Thread

{

Sort parent;

boolean isForce;

public int SleepTime = 20;

InsertionSort(Sort S, boolean method)

{

parent = S;

isForce = method;

}

public void run()

{

//обновляем данные о сортировке.

parent.vpanel.reInitComponents(); //обнуляем значения

parent.vpanel.SortName = " вставками"; //даем сортировке название.

InsSort();

//действия после работы потока.

parent.vpanel.reInitComponents(); //обнуляем значения

parent.vpanel.SortName = null; //убираем у сортировки название

JOptionPane.showMessageDialog(parent, "Сортировка вставками завершена!"," ", JOptionPane.INFORMATION\_MESSAGE);

parent.progressBar.setValue(0);; //обнуляем значения

Sort.button\_check.setEnabled(true);

Sort.button\_save.setEnabled(true);

Sort.button\_revival.setEnabled(true);

Sort.button\_input.setEnabled(true);

Sort.button\_start\_sort.setEnabled(false);

Sort.button\_make\_step.setEnabled(false);

}

private void InsSort()

{

int i, j;

ColorInt temp;

for (i = 1; i < parent.vpanel.PanelArray.length; i++)

{

float y = ((i+1)\*100/parent.vpanel.PanelArray.length);

parent.progressBar.setValue((int) y);

synchronized (this)

{

for (int k=0; k < i; k++)

{

parent.vpanel.PanelArray[k].InColor = Color.WHITE;

}

temp = parent.vpanel.PanelArray[i]; //текущий элемент это temp

parent.vpanel.PanelArray[i].InColor = Color.yellow; // мы рассматриваем его, значит меняем на красный цвет.

if(i != parent.vpanel.PanelArray.length-1)

parent.vpanel.PanelArray[i+1].InColor = Color.red;

boolean swapyet = false;

for (j = i - 1; j >= 0; j--)

{ //будем сравнивать со всеми предыдущими.

parent.vpanel.ACInt+=1; //Доступ к массиву

parent.vpanel.CompInt+=1; //одно сравнение

if (parent.vpanel.PanelArray[j].I < temp.I) //сравниваем предыдущий с текущим.

{

for (int k=0; k <= j; k++)

{

parent.vpanel.PanelArray[k].InColor = Color.CYAN;

}

break; //и выходим, если temp больше.

}

if(!swapyet)

swapyet = true;

parent.vpanel.PanelArray[j + 1] = parent.vpanel.PanelArray[j];

parent.vpanel.PanelArray[j] = temp;

parent.vpanel.ACInt+=3; //еще 3 доступа.

}

for(int k = i; k > j+1; k--)

parent.vpanel.PanelArray[k].InColor = Color.GREEN;

parent.vpanel.ACInt+=1; //Доступ к массиву

if(!parent.needToPaint.isSelected())

{ //если показ логики выключен на интерфейсе

for (int k=0; k < parent.vpanel.PanelArray.length; k++) //"обнуляем все"

parent.vpanel.PanelArray[k].InColor = Color.WHITE; // красим белым

}

parent.vpanel.repaint();

if (parent.L != null) {

parent.L.strLog = parent.L.strLog + "Шаг " + i + "\nТекущий элемент, лежащий в ячейке с индексом " + i + ", был помещен в ячейку\nмассива с индексом " + j + ";\n";

parent.L.strLog = parent.L.strLog + "При этом произошел сдвиг всех элементов, больших текущего элемента\nи лежащих в ячейках с индексом, меньшим, чем индекс текущего\n элемента;\n";

parent.L.ta.setText(parent.L.strLog);

}

try {

sleep(parent.SleepTime);

} catch (InterruptedException e) {

e.printStackTrace();

}

if(!isForce)

{

sleepSort();

}

}

}

}

public void sleepSort()

{

synchronized (this) {

try {

wait();

} catch (InterruptedException e) {

}

}

}

public void wakeSort()

{

synchronized (this) {

this.notify();

}

}

}

public static class MergeSort extends Thread

{

Sort parent;

boolean isForce;

int test = 0;

MergeSort(Sort S, boolean method)

{

parent = S;

isForce = method;

}

public void run()

{

//обновляем данные о сортировке.

parent.vpanel.reInitComponents(); //обнуляем значения

parent.vpanel.SortName = " слиянием"; //даем сортировке название.

parent.vpanel.repaint();

MergeSorting(parent.vpanel.PanelArray,0,parent.vpanel.PanelArray.length-1,0);

parent.progressBar.setValue(100); //обнуляем значения

//действия после работы потока.

parent.vpanel.reInitComponents(); //обнуляем значения

parent.vpanel.SortName = null; //убираем у сортировки название

JOptionPane.showMessageDialog(parent, "Сортировка слиянием завершена!"," ", JOptionPane.INFORMATION\_MESSAGE);

parent.progressBar.setValue(0);; //обнуляем значения

Sort.button\_check.setEnabled(true);

Sort.button\_save.setEnabled(true);

Sort.button\_revival.setEnabled(true);

Sort.button\_input.setEnabled(true);

Sort.button\_start\_sort.setEnabled(false);

Sort.button\_make\_step.setEnabled(false);

}

private void MergeSorting(ColorInt[] arr, int l, int r, int lforprint)

{ //lforprint - преременная, передаваемая в рекурсию для отрисовки левой части слияния.

synchronized (this)

{

//! Условие выхода из рекурсии

if(l >= r)

{

test++;

float y = (test)\*100/(2\*parent.vpanel.PanelArray.length);

parent.progressBar.setValue((int) y);

return;

}

int m = (l + r)/2;

//! Рекурсивная сортировка полученных массивов

MergeSorting(arr, l, m,lforprint);

MergeSorting(arr, m+1, r,l);

test++;

float y = (test)\*100/(2\*parent.vpanel.PanelArray.length);

parent.progressBar.setValue((int) y);

MergeS(arr, l, r, m, lforprint);

if(!isForce)

{

sleepSort();

}

}

}

private void MergeS(ColorInt[] arr, int l, int r, int m, int lforprint)

{

int j, start, fin;

int[] mas = new int[1001];

start=l; //начало левой части

fin=m+1; //начало правой части

for(j=l; j<=r; j++) //выполнять от начала до конца

if ((start<=m) && ((fin>r) || (arr[start].I<arr[fin].I)))

mas[j]=arr[start++].I;

else

mas[j]=arr[fin++].I;

//возвращение результата в список

for(j=l; j<=r; j++)

{

arr[j].I=mas[j];

parent.vpanel.ACInt++; //доступ к массиву

}

for (int k=0; k < lforprint; k++) //"обнуляем все", до левой части слияния

parent.vpanel.PanelArray[k].InColor = Color.WHITE;

for (int k=lforprint; k < l; k++) //левую часть слияния, если она есть, конечно

parent.vpanel.PanelArray[k].InColor = Color.GREEN; //красим в зеленый

for(int i = l; i <= r; i++)

{

if(i <= m) //левую часть в голубой

parent.vpanel.PanelArray[i].InColor = Color.CYAN;

else //правую в желтый.

parent.vpanel.PanelArray[i].InColor = Color.YELLOW;

}

if (parent.L != null) {

/\*parent.L.strLog = parent.L.strLog +\*/

parent.L.strLog = parent.L.strLog + "Шаг\nСоединены два упорядоченных участка массива (левая часть с " + l + " до " + m + " , \nправая с " + (m+1) + " до " + r + ");\n";

parent.L.ta.setText(parent.L.strLog);

}

parent.vpanel.ACInt+=3; //3 доступа к массиву

parent.vpanel.CompInt+=3; // 3 сравнения

if(!parent.needToPaint.isSelected())

{ //если показ логики выключен на интерфейсе

for (int k=0; k < parent.vpanel.PanelArray.length; k++) //"обнуляем все"

parent.vpanel.PanelArray[k].InColor = Color.WHITE; // красим белым

}

parent.vpanel.repaint(); //отрисовка изменений на панели.

try {

sleep(parent.SleepTime);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

public void sleepSort()

{

synchronized (this)

{

try

{

wait();

}

catch (InterruptedException e)

{

}

}

}

public void wakeSort()

{

synchronized (this)

{

this.notify();

}

}

}

}

## SortEngine.java

package def;

import def.SortAlgorithms.InsertionSort;

import def.SortAlgorithms.MergeSort;

import def.SortAlgorithms.Check;

import java.awt.event.\*;

import javax.swing.\*;

import javax.swing.event.\*;

import java.io.File;

public class SortEngine extends JFrame implements ActionListener, ChangeListener {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

Sort parent; // ссылка на Sort

private Generation dialogGen = null; //сслыка, пока пустая, на объект дочернего окна

//private ChooseSorting dialogChSort = null; //сслыка, пока пустая, на объект дочернего окна

Data dataEngine;

boolean haveBeenLoad;

InsertionSort IS;

MergeSort MS;

Check Ch;

// Конструктор сохраняет ссылку к Sort window в переменной класса parent

SortEngine(Sort parent, Data data){

this.parent = parent;

dataEngine = data;

}

public void stateChanged(ChangeEvent e) {

JSlider source = (JSlider)e.getSource();

if (!source.getValueIsAdjusting()) {

parent.SleepTime = (int)source.getValue();

}

}

public void actionPerformed(ActionEvent e) {

// Получение источника события

JButton clickedButton = (JButton) e.getSource();

// Если это кнопка "Ввод исходных данных"

String actioncommand = clickedButton.getActionCommand();

if (actioncommand == dataEngine.b\_input\_txt) {

// обработчик нажатия на кнопку, которая по логике должна вызывать дочернее окно

// создаём объект дочернего окна, передавая в конструктор this-ссылку - т.о., именно это окно становится для нового родительским

this.dialogGen = new Generation(this, dataEngine);

if (this.dialogGen.execute()) {

// действия при нажатии клавиши ОК

// здесь же прописывается считывание нужных результатов, введённых пользователем, используя функции get, заблаговременно прописанные вами в класе дочернего окна

haveBeenLoad = true;

if ((dataEngine.array == null) || (dataEngine.array.length == 0))

{

haveBeenLoad = false;

JOptionPane.showMessageDialog(null, "Некорректные исходные данные", " ", JOptionPane.ERROR\_MESSAGE);

return;

}

else

{

Sort.button\_start\_sort.setEnabled(true);

Sort.button\_check.setEnabled(true);

Sort.button\_save.setEnabled(true);

parent.vpanel.PanelArray = new ColorInt[dataEngine.array.length];

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

{

parent.vpanel.PanelArray[i] = new ColorInt();

parent.vpanel.PanelArray[i].I = dataEngine.array[i];

}

parent.vpanel.SortName = null; //главное - не забыть, что сортировки нет, и убрать ее имя.

parent.vpanel.repaint();

}

}

else {

// действия при нажатии на клавишу отмены

haveBeenLoad = false;

}

}

if(actioncommand == dataEngine.b\_start\_sort\_txt)

{ //если это кнопка "Запуск сортировки"

if (haveBeenLoad)

{

Sort.button\_start\_sort.setEnabled(false);

Sort.button\_check.setEnabled(false);

Sort.button\_save.setEnabled(false);

Sort.button\_revival.setEnabled(false);

Sort.button\_input.setEnabled(false);

parent.vpanel.PanelArray = new ColorInt[dataEngine.array.length];

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

{

parent.vpanel.PanelArray[i] = new ColorInt();

parent.vpanel.PanelArray[i].I = dataEngine.array[i];

}

int x = 0;

parent.L.ta.setText(null);

parent.L.strLog = "";

Sort.button\_logic.setEnabled(true);

if ((Sort.r\_demo\_mode.isSelected()) && (Sort.r\_ins\_s.isSelected()))

x = 1;

if ((Sort.r\_step\_mode.isSelected()) && (Sort.r\_ins\_s.isSelected())) {

x = 11;

Sort.button\_make\_step.setEnabled(true);

Sort.button\_start\_sort.setEnabled(false);

parent.typeOfSort = true;

}

if ((Sort.r\_demo\_mode.isSelected()) && (Sort.r\_mer\_s.isSelected()))

x = 2;

if ((Sort.r\_step\_mode.isSelected()) && (Sort.r\_mer\_s.isSelected())) {

x = 22;

Sort.button\_make\_step.setEnabled(true);

Sort.button\_start\_sort.setEnabled(false);

parent.typeOfSort = false;

}

switch(x)

{

case 1:

IS = new InsertionSort(parent, true);

IS.start();

break;

case 11:

IS = new InsertionSort(parent, false);

IS.start();

break;

case 2:

MS = new MergeSort(parent, true);

MS.start();

break;

case 22:

MS = new MergeSort(parent, false);

MS.start();

break;

case 0:

break;

}

}

else

{

JOptionPane.showMessageDialog(null, "Исходные данные не загружены", " ", JOptionPane.WARNING\_MESSAGE);

return;

}

}

if(actioncommand == dataEngine.b\_revival\_txt)

{ //если это кнопка "Восстановить"

if (haveBeenLoad)

{ //если данные были загружены

parent.vpanel.PanelArray = new ColorInt[dataEngine.array.length];

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

{

parent.vpanel.PanelArray[i] = new ColorInt();

parent.vpanel.PanelArray[i].I = dataEngine.array[i];

}

parent.vpanel.SortName = null; //главное - не забыть, что сортировки нет, и убрать ее имя.

parent.vpanel.repaint();

JOptionPane.showMessageDialog(this, "Исходный массив восстановлен!"," ", JOptionPane.INFORMATION\_MESSAGE);

Sort.button\_start\_sort.setEnabled(true);

}

else

{

JOptionPane.showMessageDialog(null, "Исходные данные не загружены", " ", JOptionPane.WARNING\_MESSAGE);

return;

}

}

if(actioncommand == dataEngine.b\_check\_txt)

{ //если это кнопка "Проверка"

if (haveBeenLoad)

{ //если данные были загружены

Ch = new Check(parent, dataEngine);

Ch.start();

}

else

{

JOptionPane.showMessageDialog(null, "Исходные данные не загружены", " ", JOptionPane.WARNING\_MESSAGE);

return;

}

}

if(actioncommand == dataEngine.b\_make\_step\_txt)

{

if(IS != null)

IS.wakeSort();

if(MS != null)

MS.wakeSort();

}

if(actioncommand == dataEngine.b\_save\_txt)

{

JFileChooser fileChooser = new JFileChooser();

fileChooser.setDialogTitle("Save file");

int userSelection = fileChooser.showSaveDialog(this);

if (userSelection == JFileChooser.APPROVE\_OPTION) {

File fileToSave = fileChooser.getSelectedFile();

String str\_temp = "";

for (int schet = 0; schet < parent.vpanel.PanelArray.length; schet++) {

str\_temp = str\_temp + parent.vpanel.PanelArray[schet].I + " ";

}

str\_temp = parent.vpanel.PanelArray.length + "\n" + str\_temp;

fileToSave = FileWorker.write(fileToSave, str\_temp);

}

}

if(actioncommand == dataEngine.b\_logic\_txt)

{

parent.L.setVisible(true);

}

if(actioncommand == dataEngine.b\_about\_prog\_txt)

{

JOptionPane.showMessageDialog(null, "<HTML><h2>Проект \"Визуализация сортировки\"</h2>"

+ "<p>Практика, 2й курс, 4й семестр. "

+ "<p>\"ЛЭТИ\", Факультет КТИ, кафедра МОЭВМ"

+ "<p><p>Над проектом работали студенты группы 3304:"

+ "<p> Лурье Роман, Черепкова Юлия."

+ "<p>Контакты: john.lurye@gmail.com , 166\_yulia@mail.ru"

+ "<p><p>Copyright &copy; 01.07.2015", "О программе...", JOptionPane.INFORMATION\_MESSAGE);

}

}

}

**VisPanel.java**

package def;

import javax.swing.\*;

import java.awt.\*;

public class VisPanel extends JPanel

{

private static final long serialVersionUID = 1L;

private final int fontsize = 14; //размер шрифта.

private int WidthCoef;

private int HeigthCoef;

public ColorInt[] PanelArray;

public Sort parent = null; //указатель на родителя

private String SortDef = "Сортировка: ";

String SortName = null;

private String Comp = "Сравнений: ";

int CompInt;

private String AC = "Доступов к массиву: ";

int ACInt;

private String Delay = "Задержка : ";

private String ms = " мс";

private String Delim = " | ";

private String NumberofElements = " элем.";

public VisPanel(Sort p)

{

setOpaque(true);

parent = p;

}

protected void paintComponent(Graphics g)

{

super.paintComponent(g);

if(PanelArray != null)

{

WidthCoef = getWidth()/PanelArray.length;

if (WidthCoef > 1)

HeigthCoef=WidthCoef-WidthCoef/3;

else

HeigthCoef=WidthCoef=1;

Graphics2D g2d = (Graphics2D) g;

g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

// drawing finction

g2d.setFont(new Font("Calibri", Font.PLAIN, fontsize));

if (SortName != null)

{ //выводим на печать всю информацию о сортировке.

String S = SortDef+SortName+Delim+Comp+CompInt+Delim+AC+ACInt+Delim+Delay+parent.SleepTime+ms;

g2d.drawString(S, 5, 12);

}

//печать сообщения с количеством элементов.

g2d.drawString(PanelArray.length+NumberofElements, (int) (getWidth()-fontsize\*5.54), fontsize-2); //6,54 - коэфициент ширины символа от размера шрифта (экспериментально)

//поиск максимума

int max=PanelArray[0].I; //начальное условие для поиска максимального элемента в массиве (чтобы определить масштаб)

for(int i = 1;i < PanelArray.length; i++)

if(PanelArray[i].I > max)

max = PanelArray[i].I; //поиск максимального элемента

int k=(getHeight()-45)/max; //вычисляем масштаб - высота сцены минус запас 5px и делим это все на максимальное значение

for(int i=0;i<PanelArray.length;i++)

{

// drawing axis

g2d.setPaint(PanelArray[i].InColor);

g2d.fillRect(i\*WidthCoef,getHeight()-k\*PanelArray[i].I,HeigthCoef,k\*PanelArray[i].I); //рисуем прямоугольник

}

}

}

void reInitComponents()

{

CompInt = 0;

ACInt = 0;

}

}

# **Тестирование**

1. **План тестирования**

В ходе тестирования разработанного программного продукта необходимо проверить корректность работы следующих компонент программы:

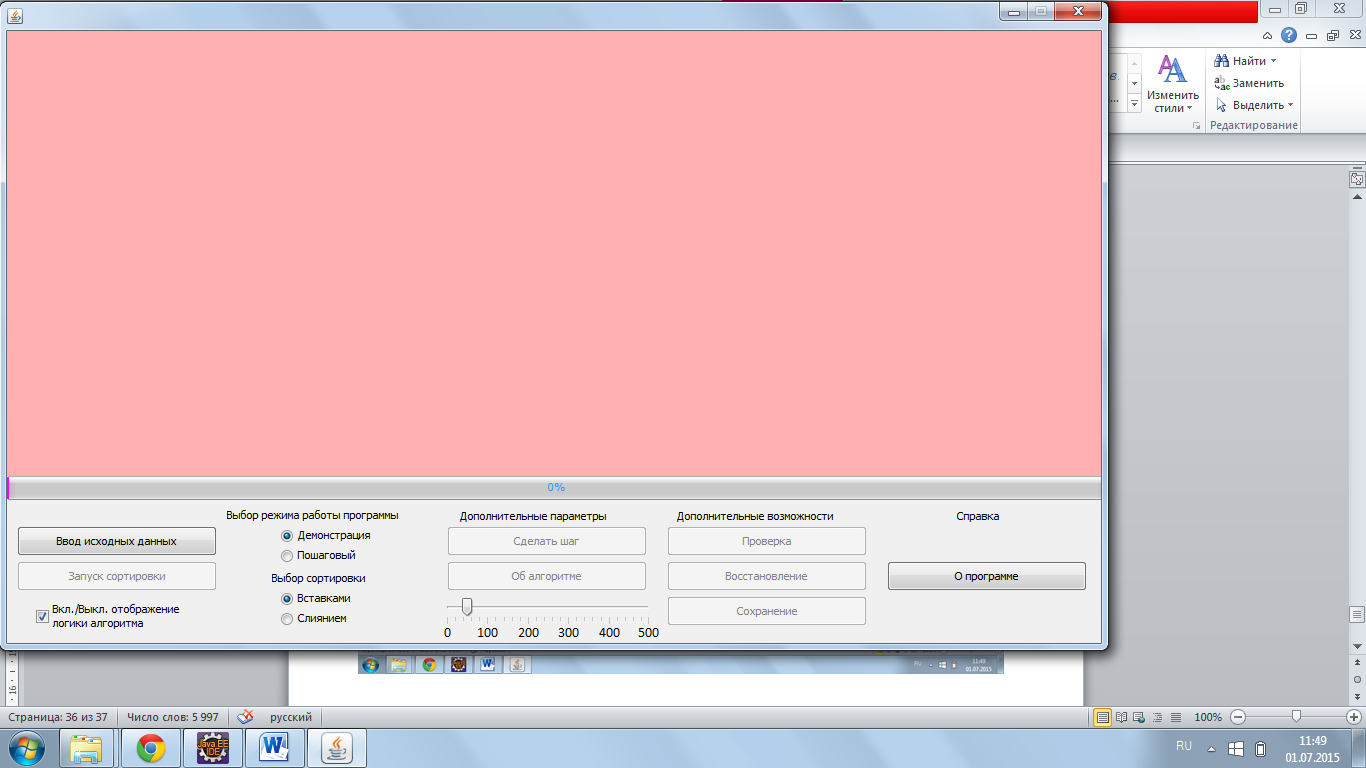
* ввод/генерация исходных данных: файл, строка ввода, автоматический с заданием параметров;
* визуализация алгоритмов сортировки в режиме «Демонстрация»;
* визуализация алгоритмов сортировки в режиме «Пошаговый»;
* отображение логики работы алгоритмов сортировки;
* опция «Проверка массива на упорядоченность»;
* опция «Восстановление исходного массива»;
* сохранение результата сортировки в текстовый файл;
* содержимого репозитория Git ;

1. **Тестирование**

При запуске программы пользователь увидит окно (далее: главное окно), изображенное на Рисунке 1.

*Рисунок 1*

*Окно при запуске программы*



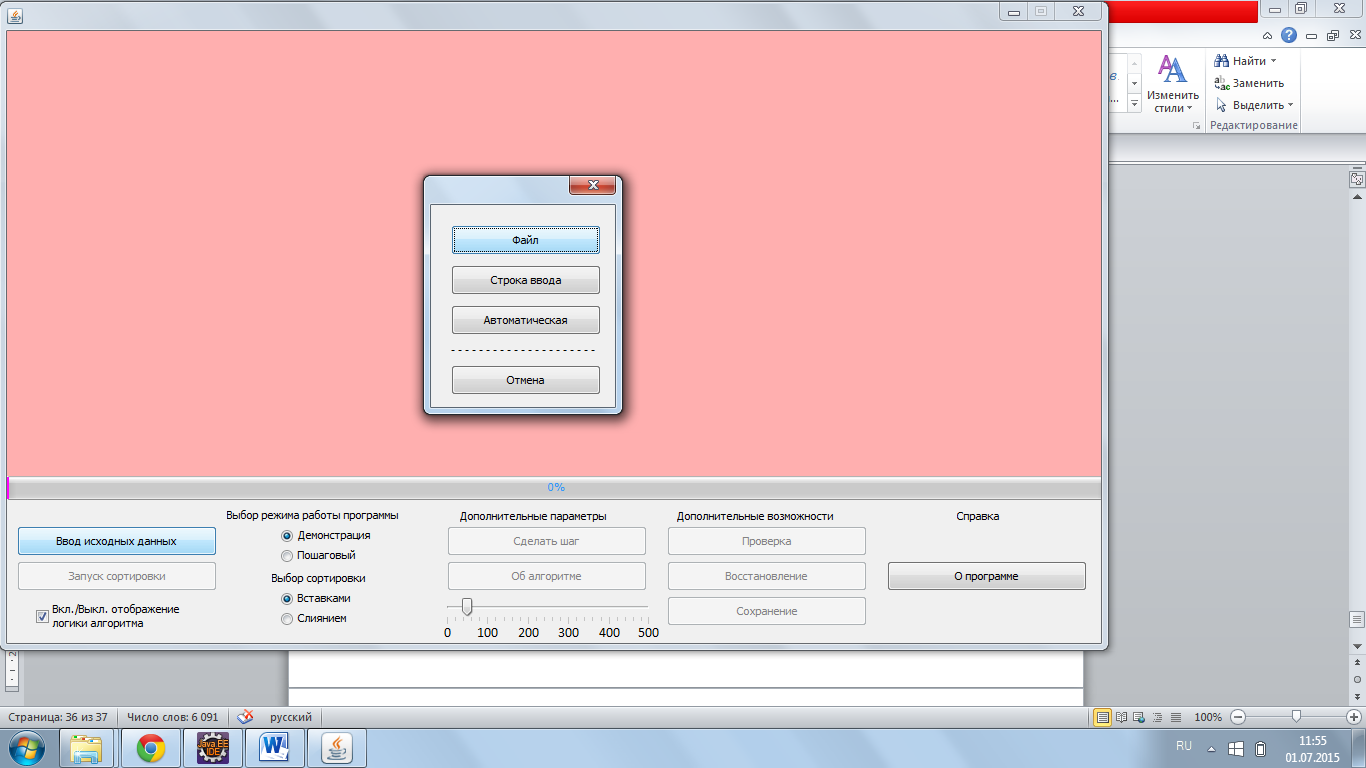
При нажатии на кнопку «Ввод исходных данных» появляется еще одно окно (далее: окно 1, см. Рисунок 2), из которого можно выбрать режим ввода данных. Корректность работы всех режимов данных была протестирована. Исходные данные должны удовлетворять следующим условиям:

* Количество элементов массива должно варьироваться от 1 до 1000;
* Значения элементов массива должны лежать в диапазоне от 1 до 400;
* Если данные считываются в файле, то первое значение в файле – количество элементов в массиве, а далее следует, собственно, сама последовательность элементов;

При тестировании данного компонента программы ошибок не обнаружено.

*Рисунок 2*

*Окно ввода данных*



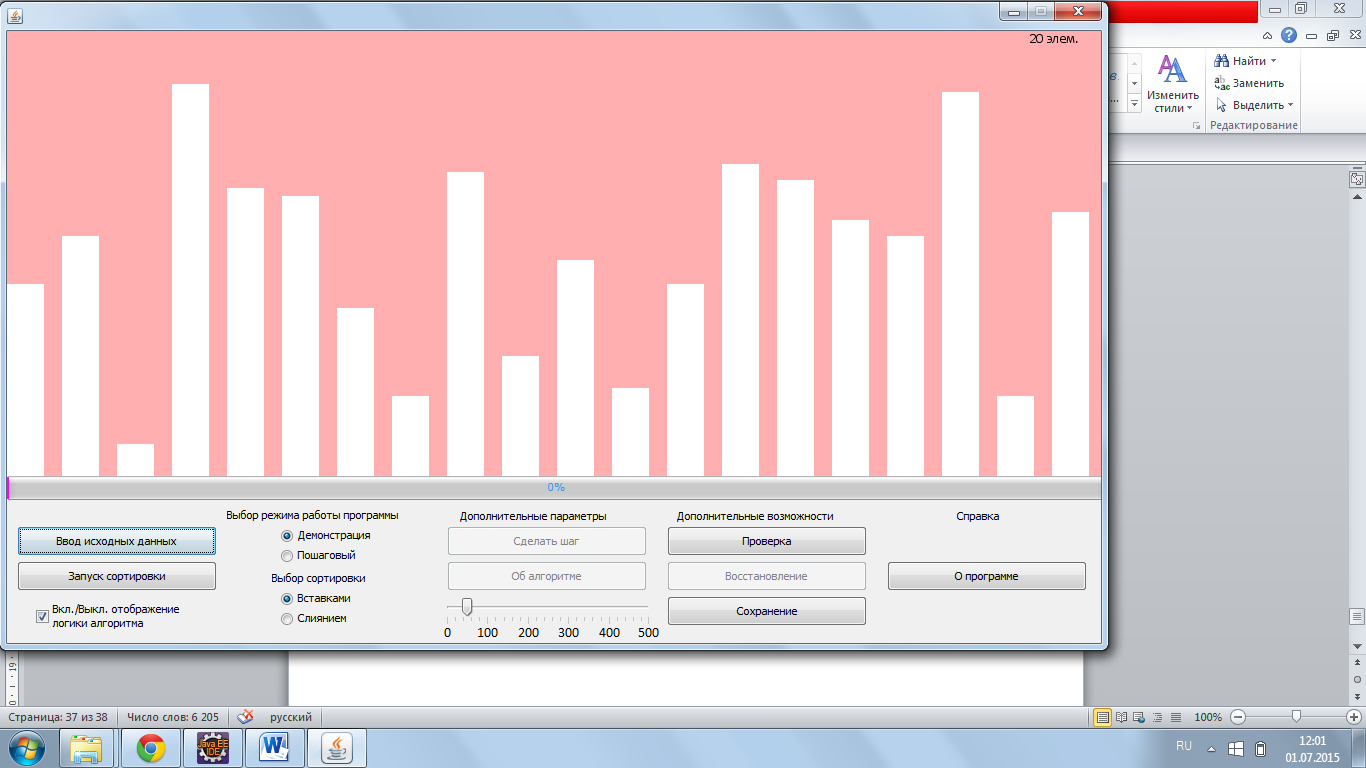
Только после ввода исходных данных пользователь может запустить программу, нажав кнопку «Запуск сортировки» и предварительно установив необходимые настройки. К настройкам относятся:

* Режим работы программы;
* Выбор сортировки;
* Включение/Выключение отображения сортировки;
* Задержка отрисовки;

При выборе режима демонстрации кнопка «Сделать шаг» останется блокированной, а при выборе режима демонстрации данная кнопка становится активной. Кнопка «Об алгоритме» активна в обоих режимах работы программы. Работа каждой из описанных выше программных компонент, а также их взаимодействия были протестированы. При тестировании ошибок не обнаружено.

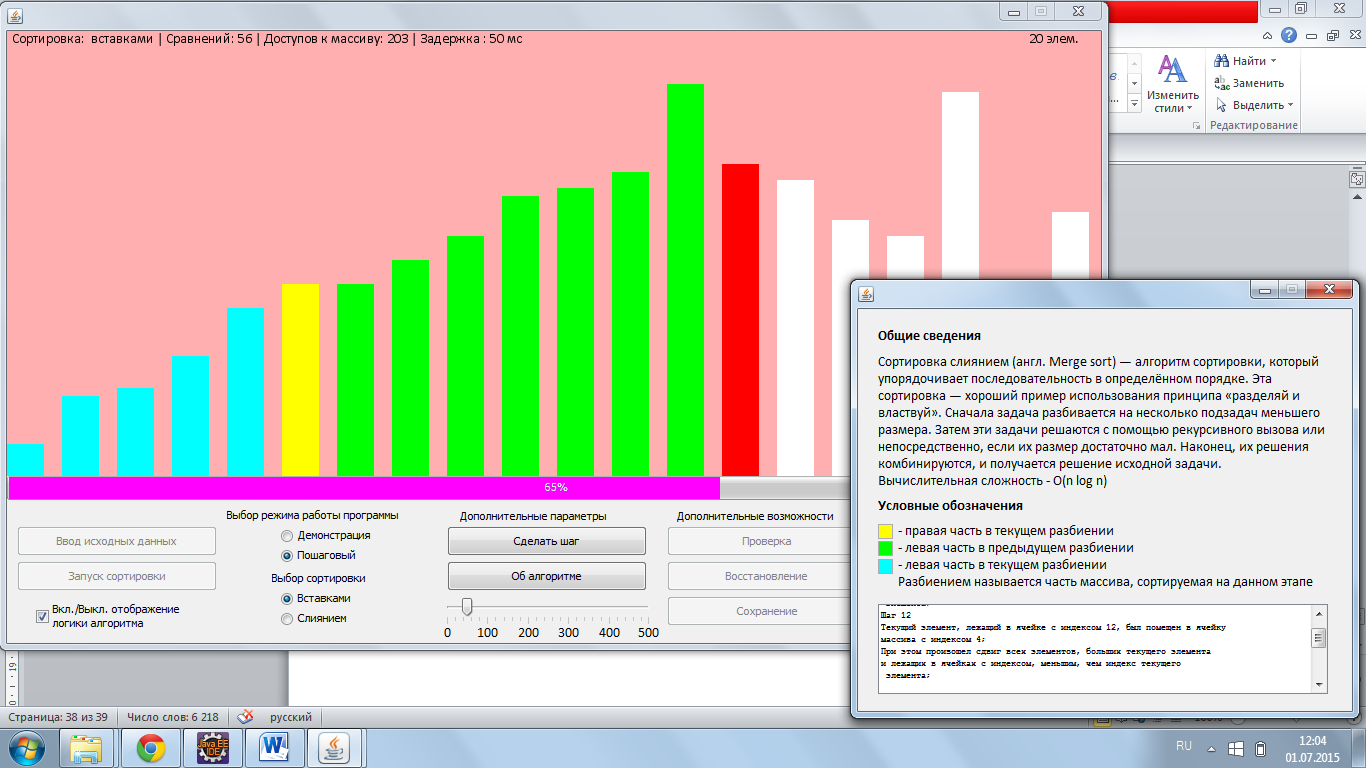
*Рисунок 3*

*Окно с загруженными данными*



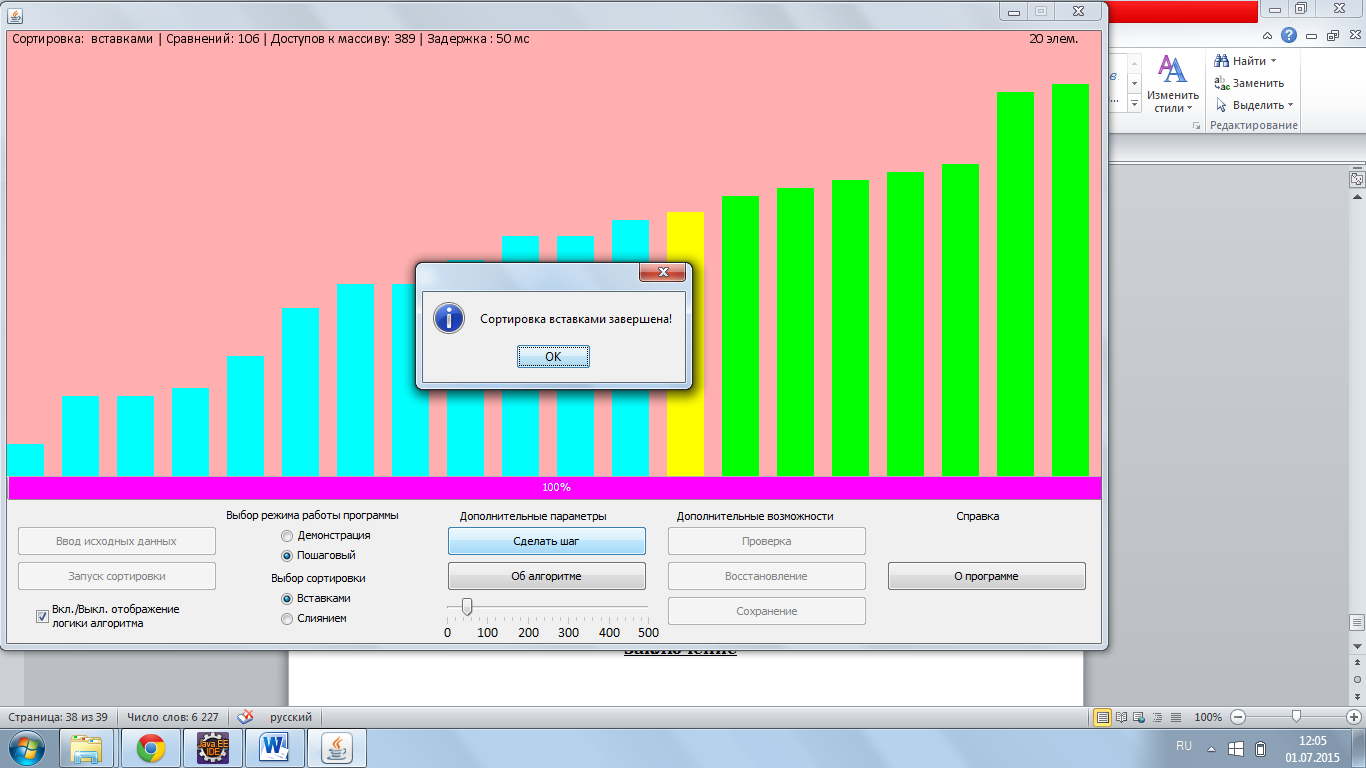
*Рисунок 4*

*Окно с отображением логики алгоритма (сортировка вставками)*



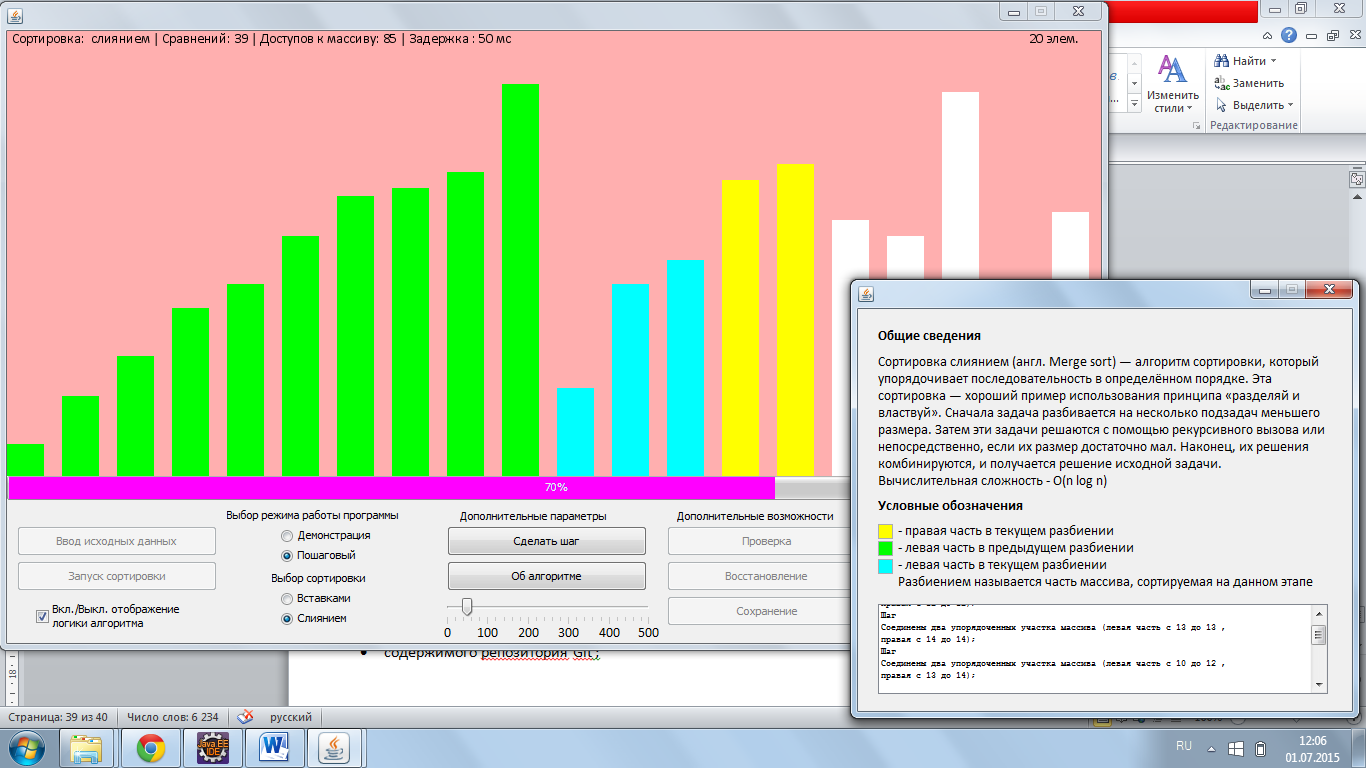
*Рисунок 5*

*Сообщение об окончании сортировки вставками*



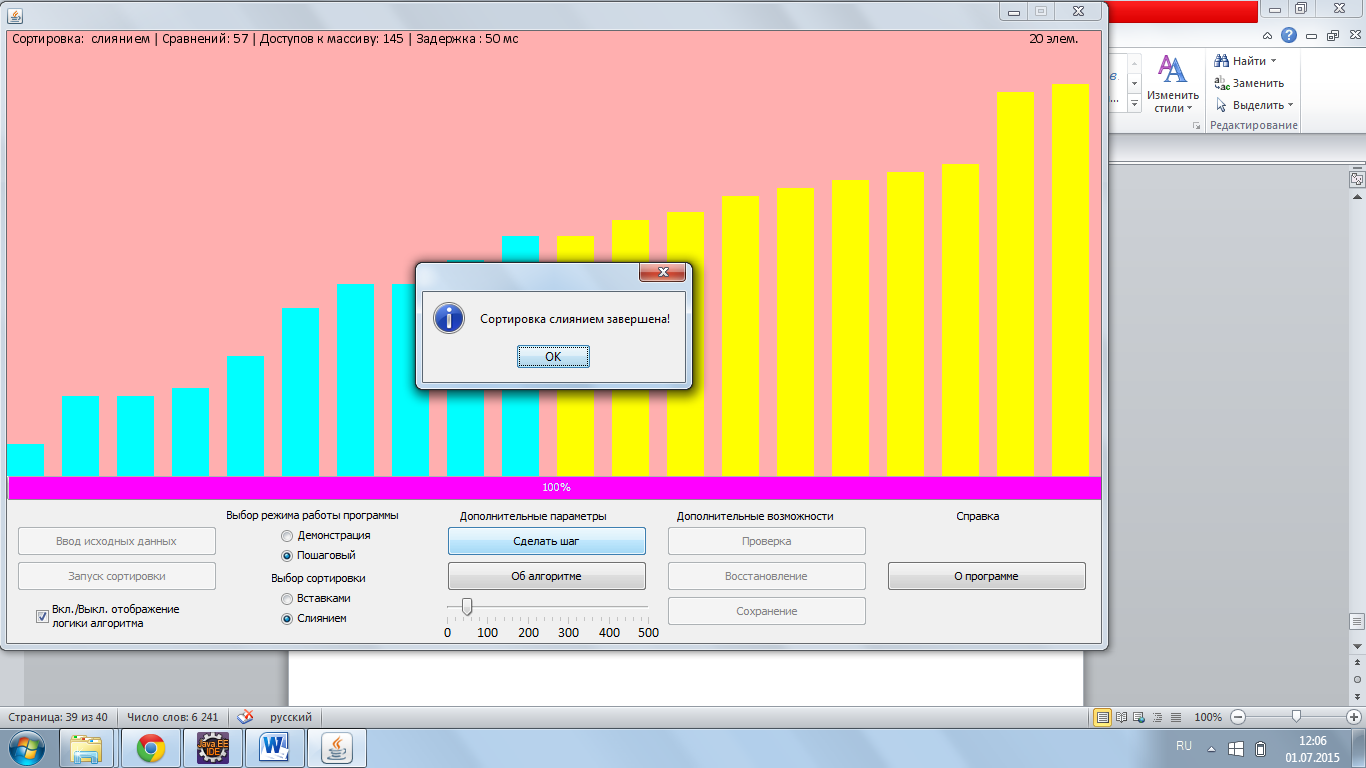
*Рисунок 6*

*Окно с отображением логики алгоритма (сортировка слиянием)*



*Рисунок 7*

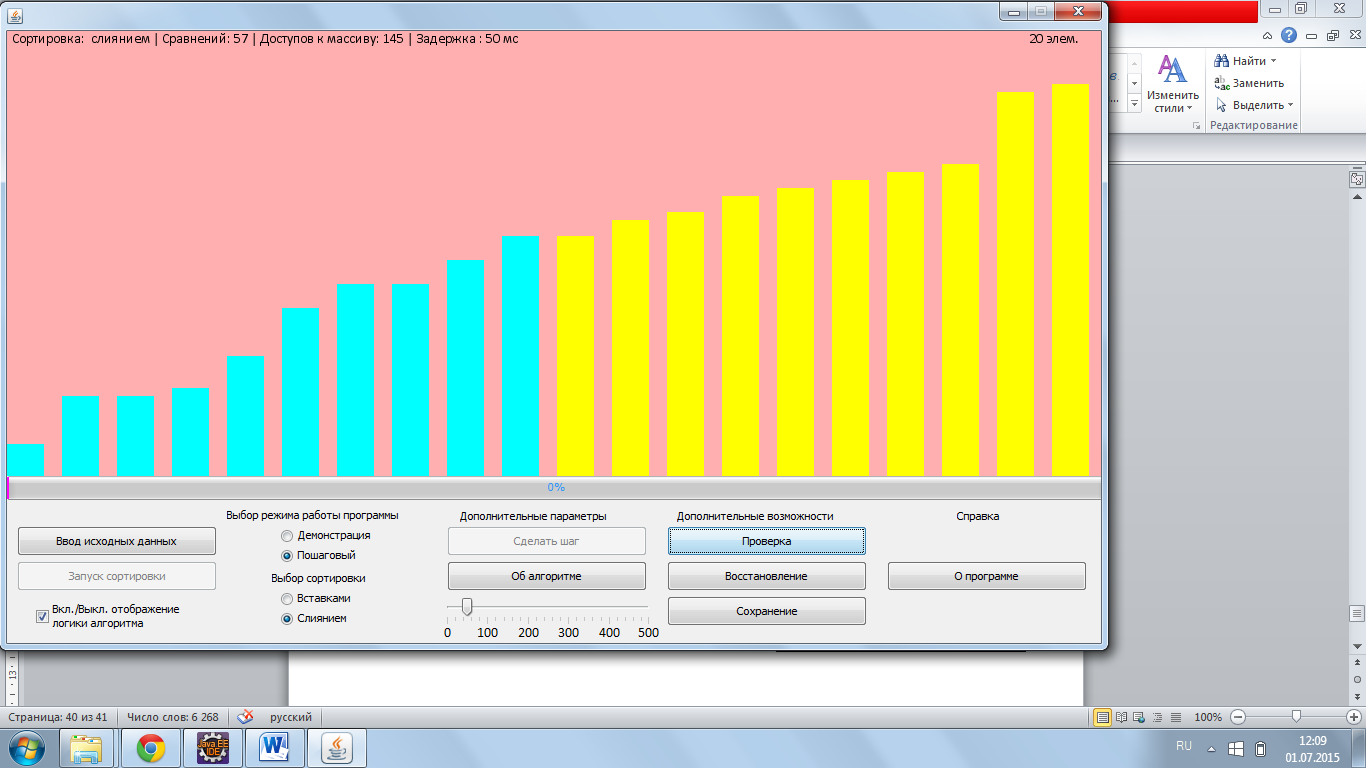
*Сообщение об окончании сортировки слиянием*



После работы алгоритма становятся доступными кнопки «Проверка», «Восстановление», «Сохранение». При тестировании данных компонент программы ошибок не обнаружено.

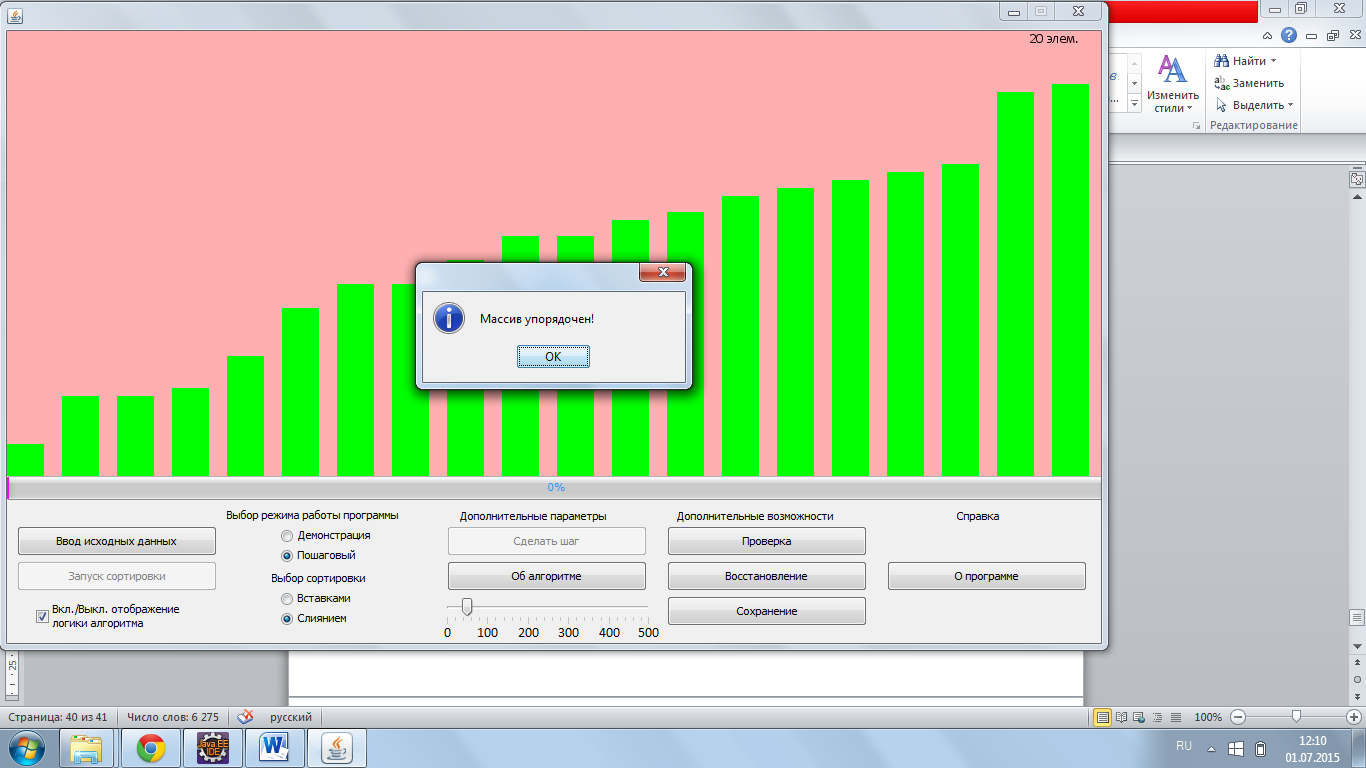
*Рисунок 8*

*Активация кнопок «Восстановление», «Проверка», Сохранение»*



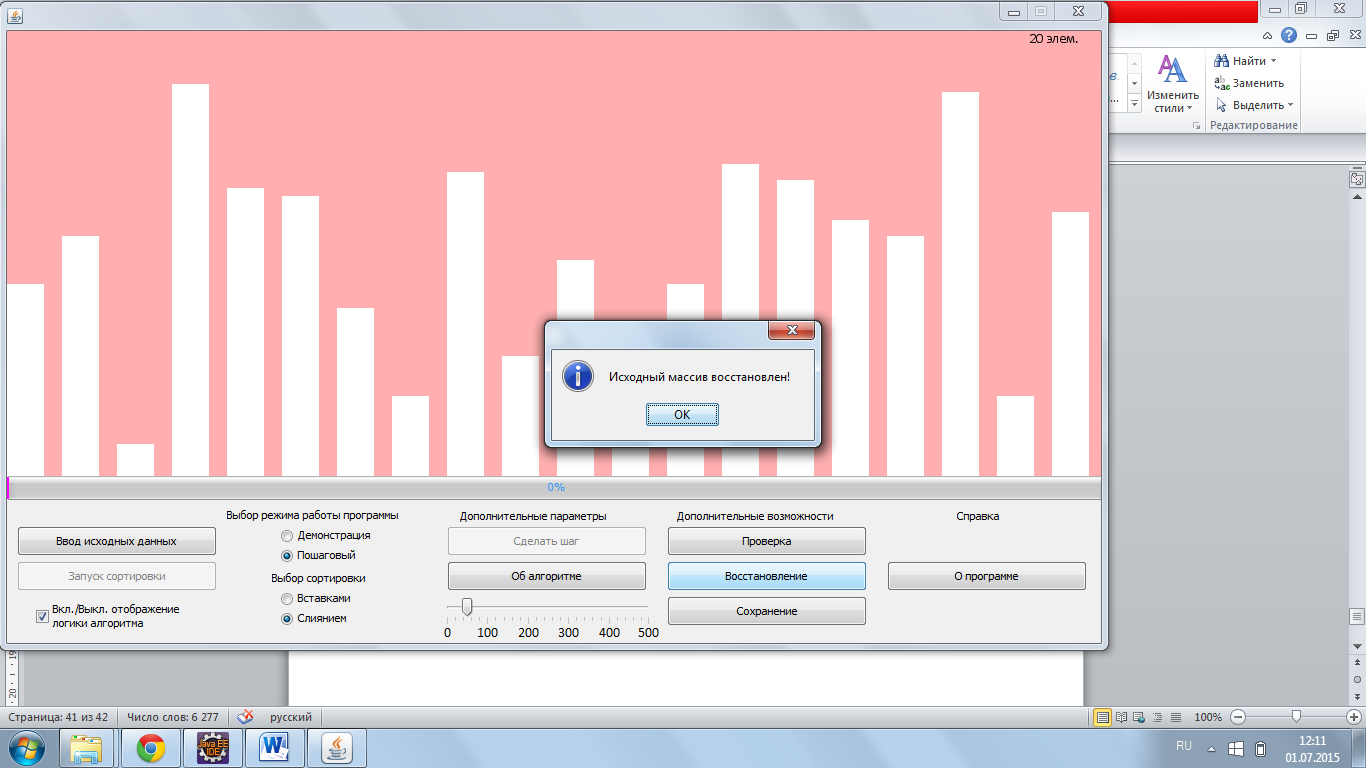
*Рисунок 9*

*Проверка исходного массива на упорядоченность*



*Рисунок 10*

*Восстановление исходного массива*



Также для обеспечения дружелюбного пользовательского интерфейса при наведении мышки на различные части окна появляются «всплывающие» подсказки. Также для улучшения понимания работы алгоритма существует опция подцветки логики алгоритма и кнопка «Об алгоритме».

# **Заключение**

В ходе выполнения практического проекта была изучена тема: «Алгоритмы сортировки: сортировка вставками и слиянием. Была осуществлена визуализация алгоритма двумя разными способами, а именно:

Способ 1: визуализация в режиме демонстрации, непрерывно показывает работу алгоритмов сортировки

Способ 2: пошаговая визуализация, процесс работы алгоритмов контролирует пользователь.

Был продуман и реализован алгоритм генерации необходимых для программы текстовых данных. Полученные в программе результаты совпали с ожидаемыми, ошибок обнаружено не было.

Были улучшен навык владения языком Java.

Проект был выполнен успешно.