**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**отчет**

**По учебной практике**

**Тема: Визуализация алгоритма QuickSort на Java**

|  |  |  |
| --- | --- | --- |
| Студент гр. 5383 |  | Нуреев Р.А |
| Студент гр. 5383 |  | Еськов А.Ю. |
| Студент гр. 5383 |  | Ришко С.Е. |
| Руководитель |  |  |

Санкт-Петербург

2017

**ЗАДАНИЕ**

**на Учебную практику**

|  |  |  |
| --- | --- | --- |
| Студент Нуреев Р.А. группы 5383 | | |
| Студент Еськов А.Ю. группы 5383 | | |
| Студент Ришко С.Е. группы 5383  Тема практики: Визуализация алгоритма на Java | | |
| Задание на практику:  Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.  Алгоритм: Быстрая сортировка. | | |
| Сроки прохождения практики: 21.06.2017 – 04.07.2017 | | |
| Дата сдачи отчета: \_\_.\_\_.2017 | | |
| Дата защиты отчета: \_\_.\_\_.2017 | | |
|  | | |
| Студент |  | Нуреев Р.А |
| Студент |  | Еськов А.Ю. |
| Студент |  | Ришко С.Е. |
| Руководитель |  |  |

**АННОТАЦИЯ**

В ходе работы над проектом учебной практики был рассмотрен алгоритм быстрой сортировки массива. Разработана программа для демонстрации работы этого алгоритма на языке программирования Java. Визуализация алгоритма должна быть доступной для понимания пользователем, а интерфейс программы удобным.

Проект выполняется в команде, что способствует развитию навыков групповой работы. Базовым условием реализации программы в команде является наличие плана разработки и разделения обязанностей.

**SUMMARY**

In the course of work on the draft training practice, the algorithm quick sorting of the array was considered. A program has been developed to demonstrate the operation of this algorithm in the Java programming language. Visualization of the algorithm should be accessible for understanding the user, and the program interface is convenient.

Project implementation in a team that promotes the development of group work skills. The basic condition for the implementation of the program in the team is the availability of a development plan and a section of duties.

Содержание

[ВВЕДЕНИЕ 5](#_Toc486585431)

[1. ОПИСАНИЕ АЛГОРИТМА 7](#_Toc486585432)

[2. ТРЕБОВАНИЯ К ПРОГРАММЕ 8](#_Toc486585433)

[2.1. Исходные требования к программе 8](#_Toc486585434)

[2.1.1. Требования к входным данным 8](#_Toc486585435)

[1.1.2. Требования к визуализации массива 8](#_Toc486585436)

[1.1.3. Требования к демонстрации решения 9](#_Toc486585437)

[1.1.4. Требования к интерфейсу программы 9](#_Toc486585438)

[2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ 10](#_Toc486585439)

[2.1. План разработки 10](#_Toc486585440)

[2.2. Распределение ролей в бригаде 10](#_Toc486585441)

[3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ 11](#_Toc486585442)

[3.1. Основной метод 11](#_Toc486585443)

[3.2. Структуры данных 11](#_Toc486585444)

[3.3. Диаграмма классов 11](#_Toc486585445)

[3.4. Exception в программе 12](#_Toc486585446)

[4. ТЕСТИРОВАНИЕ 13](#_Toc486585447)

[4.1. Тестирование интерфейса программы 13](#_Toc486585448)

[4.2. Тестирование алгоритма программы 16](#_Toc486585449)

[1.2.1. Тест №1 16](#_Toc486585450)

[1.2.2. Тест №2 16](#_Toc486585451)

[1.2.3. Тест №3 17](#_Toc486585452)

[ЗАКЛЮЧЕНИЕ 18](#_Toc486585453)

[Список использованных источников 19](#_Toc486585454)

[Приложение А. Код программы 20](#_Toc486585455)

# ВВЕДЕНИЕ

QuickSort является существенно улучшенным вариантом алгоритма сортировки с помощью прямого обмена (его варианты известны как «[Пузырьковая сортировка](https://ru.wikipedia.org/wiki/%D0%9F%D1%83%D0%B7%D1%8B%D1%80%D1%8C%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0)» и «[Шейкерная сортировка](https://ru.wikipedia.org/wiki/%D0%A8%D0%B5%D0%B9%D0%BA%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0)»), известного, в том числе, своей низкой эффективностью. Принципиальное отличие состоит в том, что в первую очередь производятся перестановки на наибольшем возможном расстоянии и после каждого прохода элементы делятся на две независимые группы. Любопытный факт: улучшение самого неэффективного прямого метода сортировки дало в результате один из наиболее эффективных улучшенных методов.

Общая идея алгоритма состоит в следующем:

* Выбрать из массива элемент, называемый опорным. Это может быть любой из элементов массива. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность.
* Сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующие друг за другом: «меньшие опорного», «равные» и «большие».
* Для отрезков «меньших» и «больших» значений выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.

На практике массив обычно делят не на три, а на две части: например, «меньшие опорного» и «равные и большие»; такой подход в общем случае эффективнее, так как упрощает алгоритм разделения (см. ниже).

Хоар разработал этот метод применительно к [машинному переводу](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%B4); словарь хранился на [магнитной ленте](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B3%D0%BD%D0%B8%D1%82%D0%BD%D0%B0%D1%8F_%D0%BB%D0%B5%D0%BD%D1%82%D0%B0), и сортировка слов обрабатываемого текста позволяла получить их переводы за один прогон ленты, без перемотки её назад. Алгоритм был придуман Хоаром во время его пребывания в [Советском Союзе](https://ru.wikipedia.org/wiki/%D0%A1%D0%A1%D0%A1%D0%A0), где он обучался в [Московском университете](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D1%81%D0%BA%D0%BE%D0%B2%D1%81%D0%BA%D0%B8%D0%B9_%D0%B3%D0%BE%D1%81%D1%83%D0%B4%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82_%D0%B8%D0%BC%D0%B5%D0%BD%D0%B8_%D0%9C._%D0%92._%D0%9B%D0%BE%D0%BC%D0%BE%D0%BD%D0%BE%D1%81%D0%BE%D0%B2%D0%B0) компьютерному переводу и занимался разработкой русско-английского разговорника.

В процессе выполнения работы была разработана программа на языке Java, предназначенную для осуществления сортировки массива с помощью быстрой сортировки.

Требовалось реализовать пользовательский интерфейс и визуализацию алгоритма. Предусмотрена возможность ходить по решению вперед-назад, чтобы подробнее проследить ход решения.

# ОПИСАНИЕ АЛГОРИТМА

Алгоритм состоит из трёх шагов:

1. Выбрать элемент из массива. Назовём его опорным. Опорным является середина рассматриваемого массива.
2. *Разбиение*: перераспределение элементов в массиве таким образом, что элементы меньше опорного помещаются перед ним, а больше или равные после.
3. Рекурсивно применить первые два шага к двум подмассивам слева и справа от опорного элемента. Рекурсия не применяется к массиву, в котором отсутствуют элементы.

# ТРЕБОВАНИЯ К ПРОГРАММЕ

## Исходные требования к программе

### Требования к входным данным

Массив должен состоять только из целых чисел (от -99999 до 99999) количеством не более 100, элементы вводятся через пробел в специальное верхнее поле в окне программы (см. начальный прототип), либо через текстовый файл.

Пользователь выбирает, каким именно способом он будет вводить массив.

Ввод массива можно осуществить двумя способами:

1. Массив задается из файла .txt (рисунок 1) (для ввода с файла необходимо нажать )



Рисунок 1 – массив в файле

2. С помощью активного поля



Для запуска решения необходимо нажать “GO”.

### Требования к визуализации массива

Массив визуализируется в окне в виде (рисунок 2):



Рисунок 2 – Пример визуализации массива

Требования к демонстрации поиска решения

При демонстрации поиска решения рассматривается каждый шаг, при этом элементы, которые поменяются местами выделены синим цветом. Опорный элемент выделен красным.

После того, как вершина рассмотрена, она окрашивается в красный цвет.

### Требования к демонстрации решения

При демонстрации решения между 2 состояниями стрелками показаны в какие места перейдут обмениваемые элементы.

### Требования к интерфейсу программы

Интерфейс программы представлен на рисунке 3:

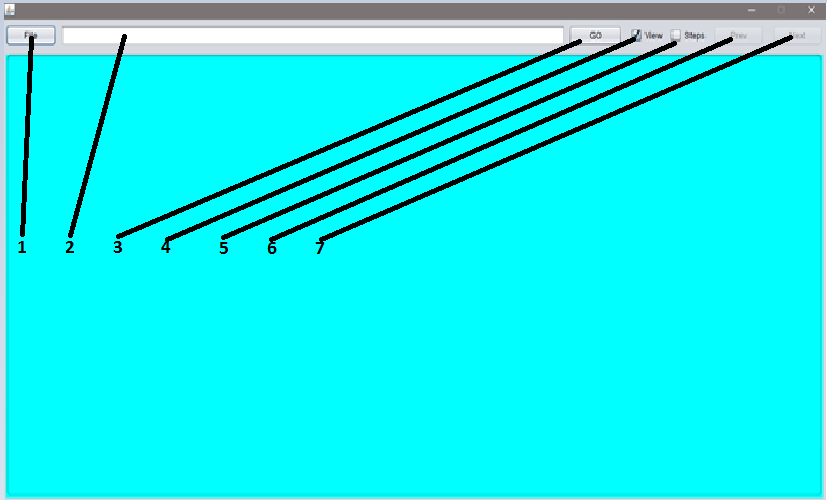


Рисунок 3 - Интерфейс программы

Описание интерфейса программы:

1. Кнопка для ввода данных с файла
2. Поле для ввода входных данных
3. Кнопка для запуска программы
4. Метка для скрытия/показа визуализации алгоритма
5. Метка для перехода к пошаговой демонстрации алгоритма
6. Кнопка для перехода к следующему шагу при пошаговой демонстрации алгоритма
7. Кнопка для перехода к предыдущему шагу при пошаговой демонстрации алгоритма

# ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

## План разработки

26.06: Программа должна иметь рабочий алгоритм, а также улучшенный прототип интерфейса (с вводом/выводом элементов массива с промежуточными данными, выводом промежуточных подмассивов, выделением опорных элементов для них цветом, соблюдением отступов при рекурсии).

28.06: Доработка алгоритма визуализации, добавление визуализации обмена значений при сортировке.

30.06: Доработка алгоритма визуализации, обработка различных исключительных ситуаций (например, ввода в качестве элементов массива не целых чисел, выбор пустого текстового файла), предоставление конечного варианта проекта.

## Распределение ролей в бригаде

Распределение ролей в бригаде представлено в Таблице 1.

Таблица 1 – Распределение ролей в бригаде

|  |  |
| --- | --- |
| ФИО | Обязанности |
| Ришко С.Е. | * Реализация алгоритма * Оформление спецификации |
| Еськов А.Ю. | * Визуализация алгоритма * Разработка архитектуры проекта * Оформление пояснительной записки |
| Нуреев Р.А. | * Реализация окончательного варианта визуализации * Разработка тестов программы * Оформление пояснительной записки |

# ОСОБЕННОСТИ РЕАЛИЗАЦИИ

## Основной метод

Программа имеет единственное окно для ввода/вывода массива и визуализации сортировки. Управление осуществляется с помощью библиотечных методов для компонентов jFrame.

## Структуры данных

**numArr -**  массив для сортировки

**mainList –** список массивов состояний на этапах сортировки

**linesList –** список массивов индексов меняющихся элементов

## Диаграмма классов

В программе используется 2 класса. Код, относительно маленького объёма, не имеет смысла разбивать на множество классов.

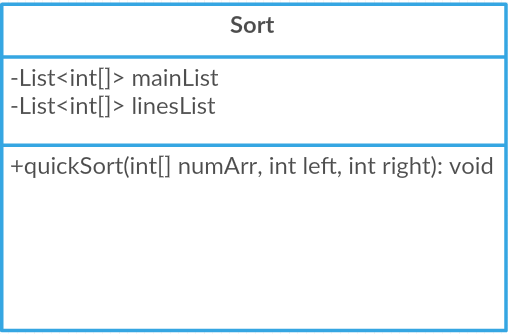
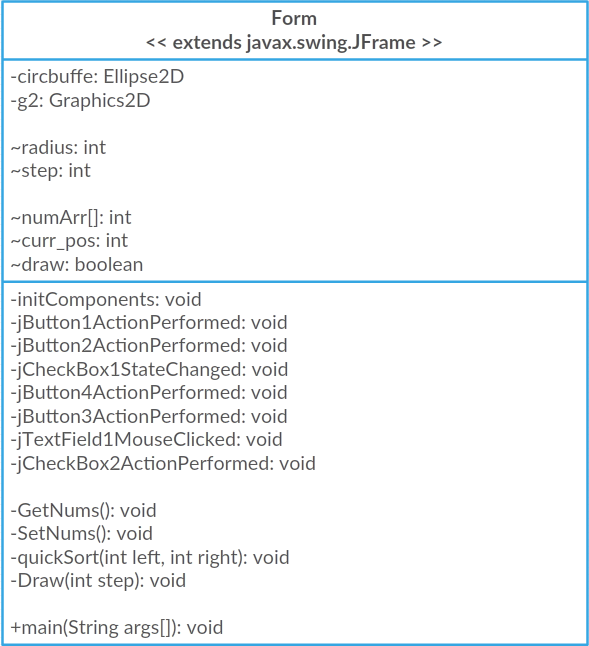


Рисунок 7 - UML диаграмма классов

## Exception в программе

Исключение — это событие при выполнении программы, которое приводит к её ненормальному или неправильному поведению.

В нашей программе такие события связаны с вводом данных пользователем. При вводе некорректного данного под сценой будет выводится сообщение об ошибке. В таблице 6 представлены такие события и выводимые сообщения.

Таблица 6 – Вывод сообщения при вводе некорректных данных

|  |  |
| --- | --- |
| Ввод некорректного данного | Сообщение |
| Введены отличные от цифр символы | Некорректный массив! |
| Введена пустая строка | Введите массив! |

# ТЕСТИРОВАНИЕ

## Тестирование интерфейса программы

При тестировании интерфейса программы будет проверяться корректная работа программы при работе с пользователем. Также в тестировании программы можно увидеть описание доработанного интерфейса программа.

При запуске программы открывается окно, представленное на рисунке 13.

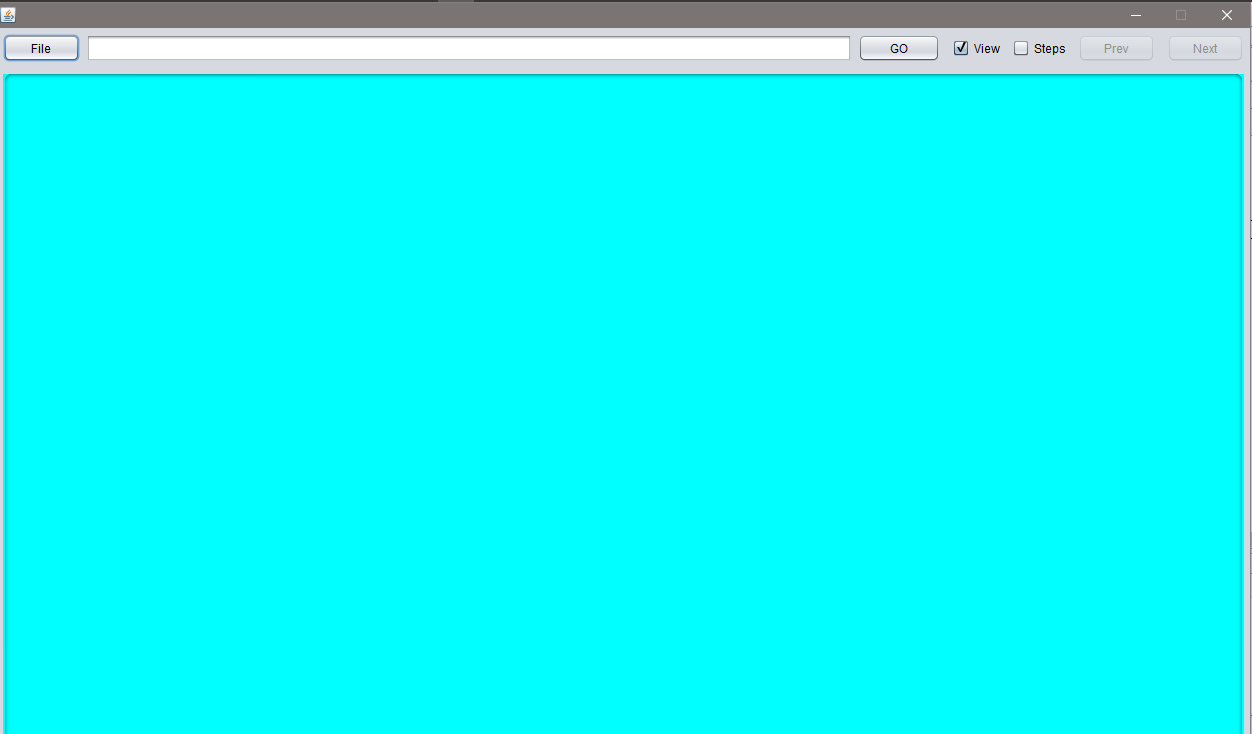


Рисунок 13 – Запуск программы

Как видно на рисунке 13, несколько кнопок заблокировано, потому что в данный момент Пользователь не может просмотреть пошаговый алгоритм.

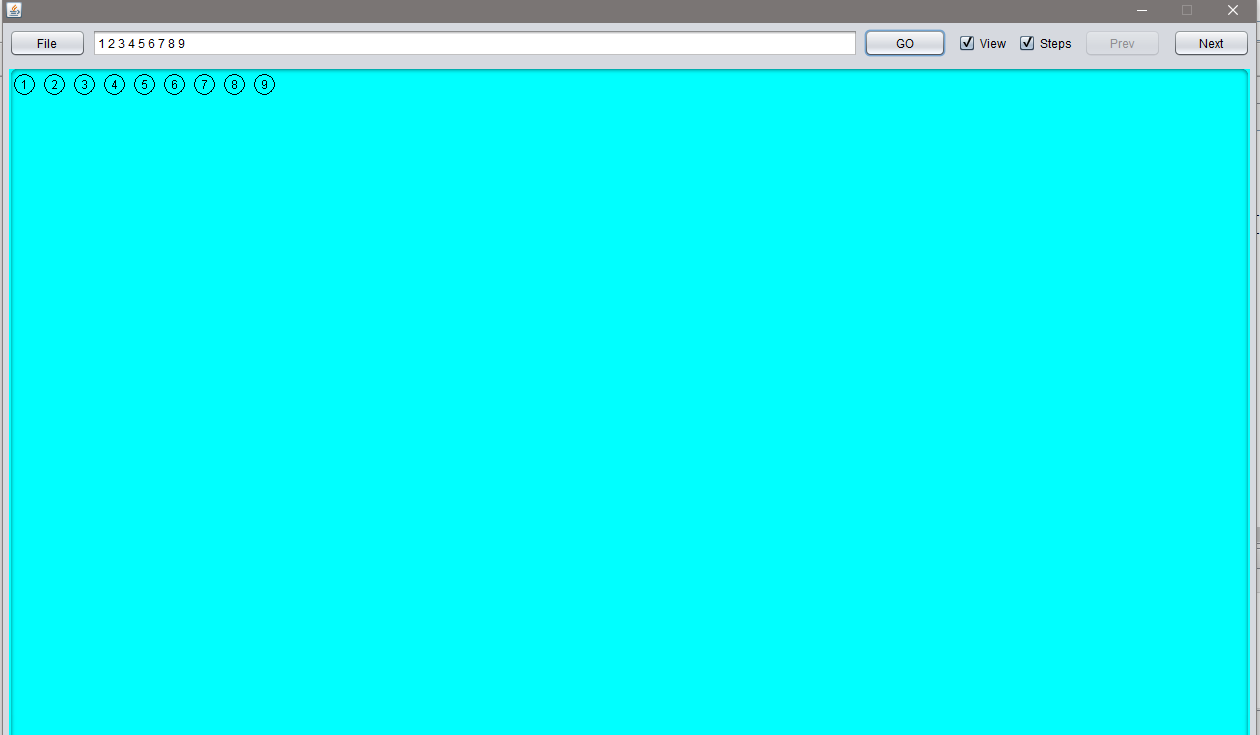


Рисунок 14 – Добавление массива

В данном окне надо ввести массив целых чисел.

Если пользователь введет некорректные данные, то программа обработает данное исключение и выведет соответствующее сообщение на экран.

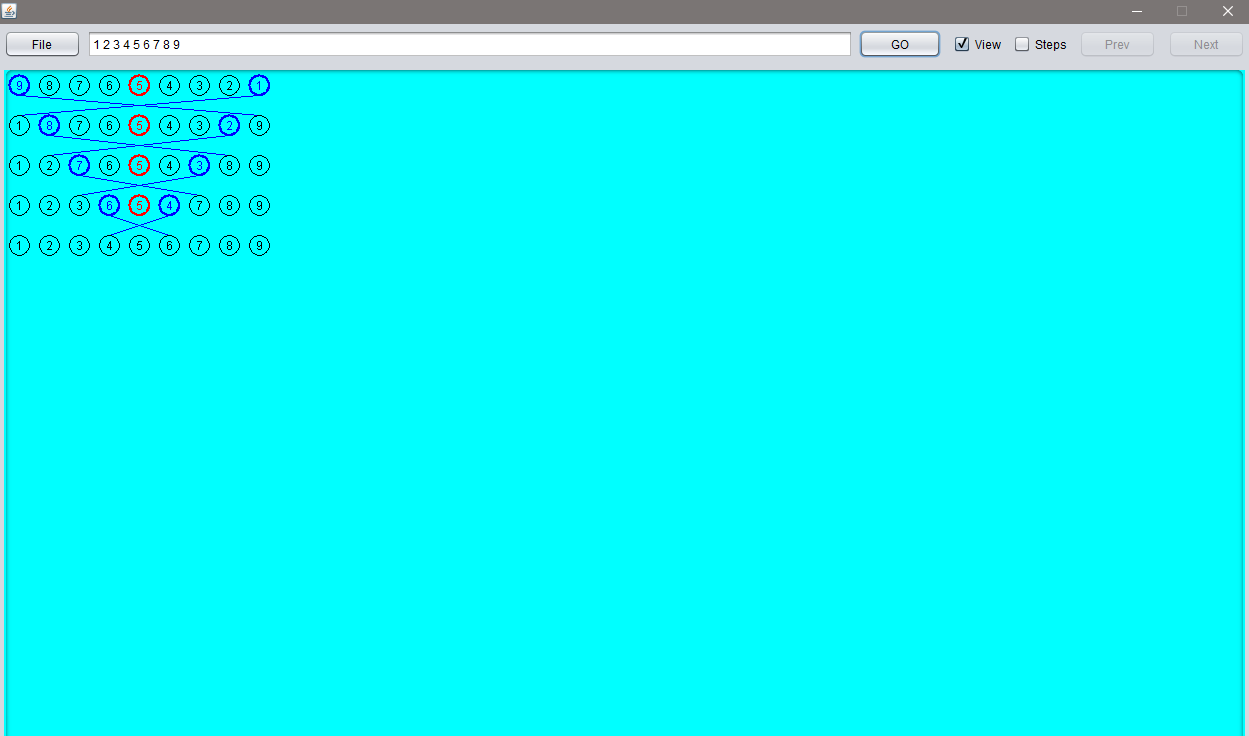


Рисунок 15 – полное решение

На рисунке 15 можно увидеть все шаги сортировки массива, т.к. не была помечена метка “Steps”

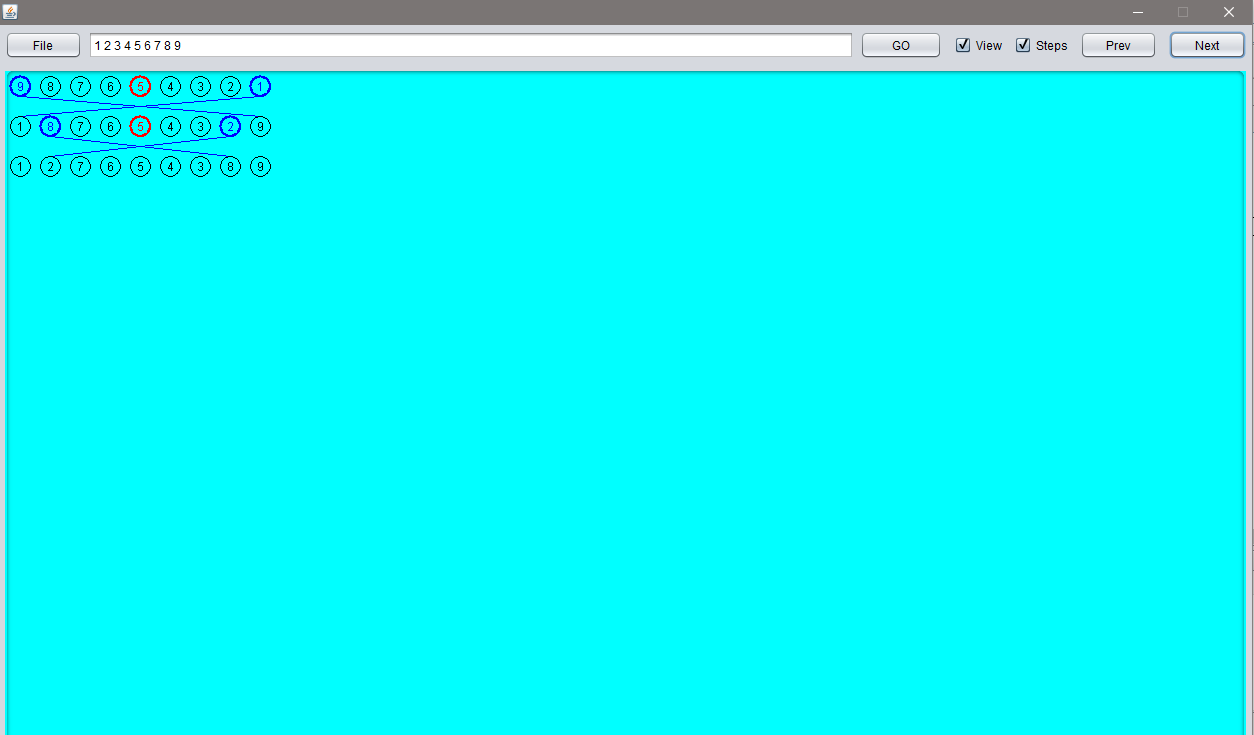


Рисунок 17 несколько шагов решения

Была отмечена метка “Steps”

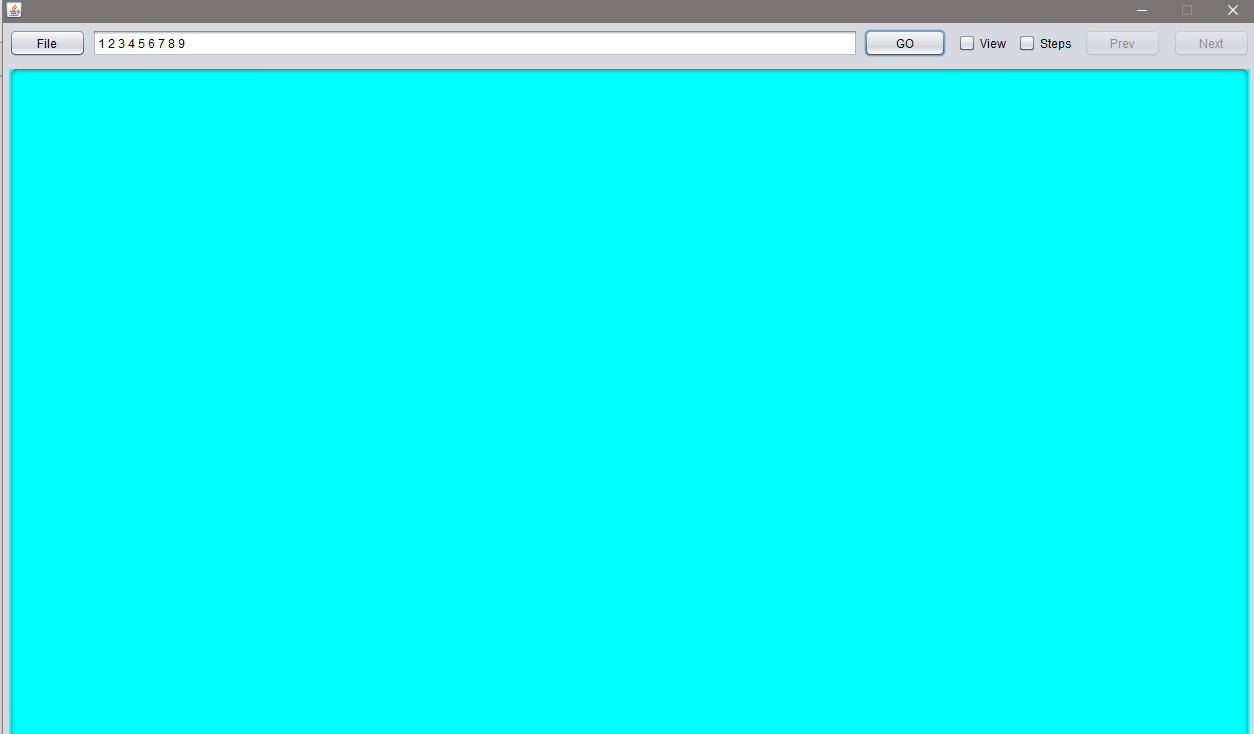
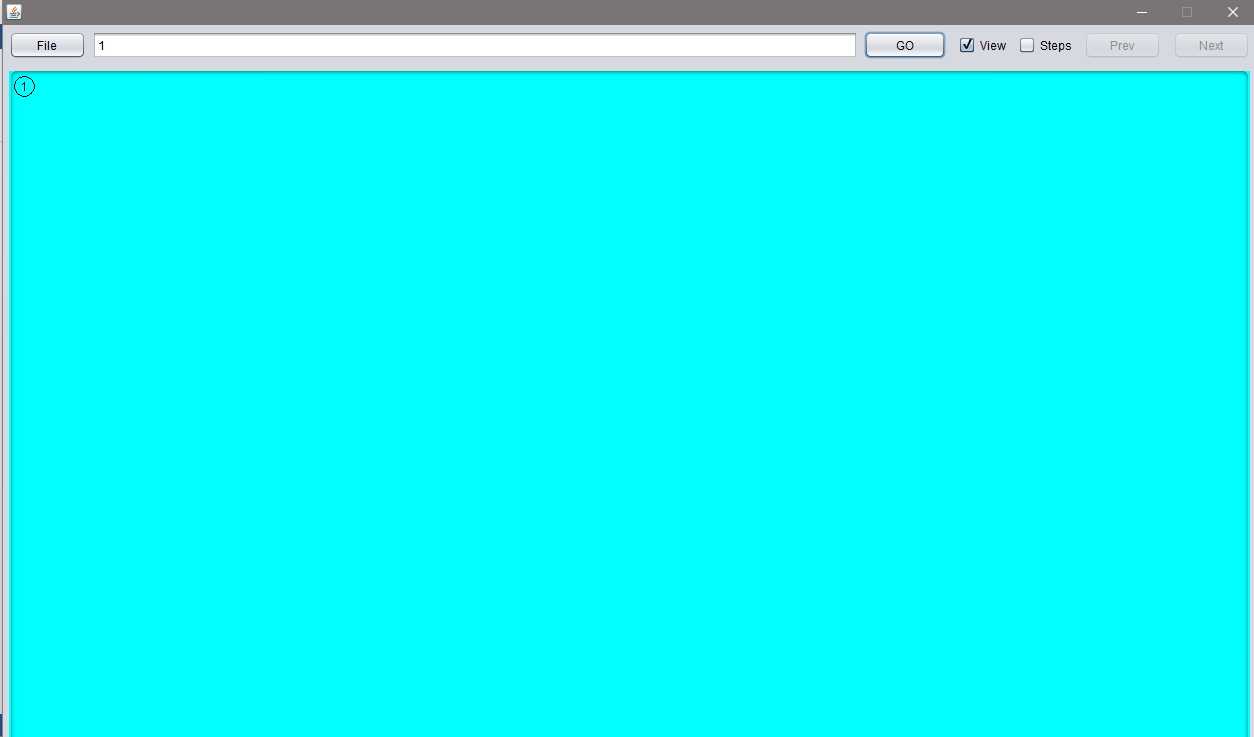


Рисунок 18 – Выполнен алгоритм без визуализации(не была отмечена метка View)

## Тестирование алгоритма программы

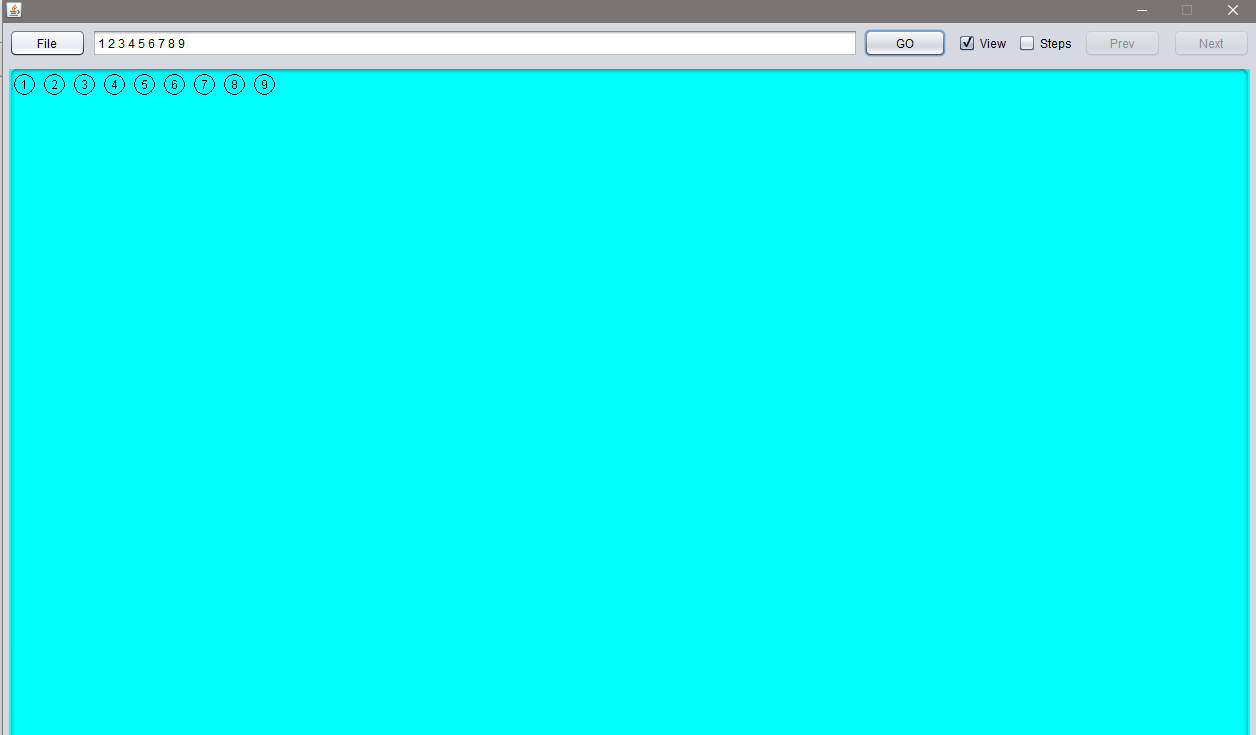
Проведем несколько тестов программы, причем массивы будут с некоторыми особенностями.

### Тест №1

1 элемент

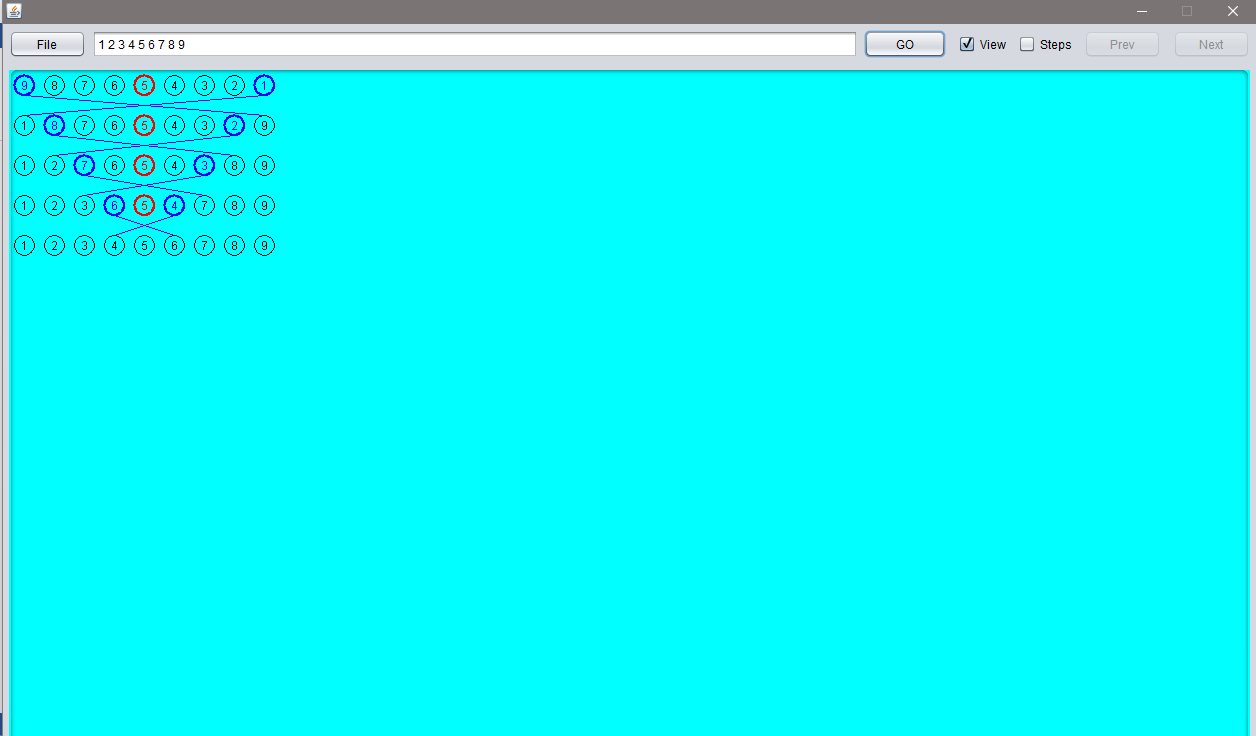
### Тест №2

Отсортированный массив



### Тест №3

Обычный неотсортированный массив



# ЗАКЛЮЧЕНИЕ

В результате работы над данным проектом была написана программа, которая успешно справляется со своей основной задачей: помочь понять пользователю принцип работы алгоритма быстрой сортровки.

Было проведено тестирование на работоспособность полученной программы. В ходе тестирования было выявлено, что полученная программа работает корректно и предупреждает ввод некорректных данных со стороны пользователя.

При реализации этой программы, в течение отведенного времени, несколько раз был перестроен графический интерфейс, поскольку первоначально поставленные требования оказались избыточны и были изменены (при этом программа сохранила свой функционал). Также были добавлены другие функции, которые помогут пользователю лучше разобраться в алгоритме поиска наикратчайших путей.

На данном этапе программа, т.е. реализация алгоритма и интерфейс, оптимальна и корректно работающая.

Работа по созданию программы позволила приобрести практические навыки не только по использованию методов программирования при решении задачи на сортировку массивов, но и навыки совместной работы и разделения обязанностей.

# Список использованных источников

1. «Алгоритмы. Построение и анализ» / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн, Издательство: Вильямс, 2013 год, 1324 стр.
2. Документация // URL: <http://docs.oracle.com/javase/8/javase-clienttechnologies.htm> (даты обращения: 21.06.2017-26.06.2017)
3. Паттерны для новичков: MVC vs MVP vs MVVM // Хабрахабр. URL: <https://habrahabr.ru/post/215605/> (дата обращения: 21.06.2017)
4. Эккель Брюс Философия Java [4-е издание] Издательство: Питер, 2015 год, 1170 стр.

# Приложение А. Код программы

**Main**

**import java.awt.BasicStroke;**

**import java.awt.Color;**

**import javax.swing.JOptionPane;**

**import java.util.ArrayList;**

**import java.util.List;**

**import java.awt.Graphics2D;**

**import java.awt.geom.Ellipse2D;**

**import java.io.File;**

**import java.io.IOException;**

**import java.util.Scanner;**

**import javax.swing.JFileChooser;**

**import myPack.Sort;**

**public class Form extends javax.swing.JFrame {**

**public Form() {**

**initComponents();**

**if (g2 == null)**

**g2 = (Graphics2D) jPanel1.getGraphics();**

**jPanel1.update(g2);**

**}**

**private Ellipse2D circbuffer;**

**private Graphics2D g2;**

**final int radius = 20;**

**final int step = 25;**

**int numArr[], curr\_pos = 0;**

**boolean draw = true;**

**//Список состояний массива**

**private List<int[]> mainList = new ArrayList<int[]>();**

**//Обводка**

**private List<int[]> linesList = new ArrayList<int[]>();**

**//Сортировка**

**private Sort srt;**

**//@SuppressWarnings("unchecked");**

**// <editor-fold defaultstate="collapsed" desc="Generated Code">**

**private void initComponents() {**

**jTextField1 = new javax.swing.JTextField();**

**jButton1 = new javax.swing.JButton();**

**jPanel1 = new javax.swing.JPanel();**

**jButton2 = new javax.swing.JButton();**

**jCheckBox1 = new javax.swing.JCheckBox();**

**jButton3 = new javax.swing.JButton();**

**jButton4 = new javax.swing.JButton();**

**jCheckBox2 = new javax.swing.JCheckBox();**

**setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);**

**setResizable(false);**

**jTextField1.addMouseListener(new java.awt.event.MouseAdapter() {**

**public void mouseClicked(java.awt.event.MouseEvent evt) {**

**jTextField1MouseClicked(evt);**

**}**

**});**

**jButton1.setText("GO");**

**jButton1.addActionListener(new java.awt.event.ActionListener() {**

**public void actionPerformed(java.awt.event.ActionEvent evt) {**

**jButton1ActionPerformed(evt);**

**}**

**});**

**jPanel1.setBackground(new java.awt.Color(0, 255, 255));**

**jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(""));**

**javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);**

**jPanel1.setLayout(jPanel1Layout);**

**jPanel1Layout.setHorizontalGroup(**

**jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)**

**.addGap(0, 0, Short.MAX\_VALUE)**

**);**

**jPanel1Layout.setVerticalGroup(**

**jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)**

**.addGap(0, 707, Short.MAX\_VALUE)**

**);**

**jButton2.setText("File");**

**jButton2.addActionListener(new java.awt.event.ActionListener() {**

**public void actionPerformed(java.awt.event.ActionEvent evt) {**

**jButton2ActionPerformed(evt);**

**}**

**});**

**jCheckBox1.setText("Steps");**

**jCheckBox1.addChangeListener(new javax.swing.event.ChangeListener() {**

**public void stateChanged(javax.swing.event.ChangeEvent evt) {**

**jCheckBox1StateChanged(evt);**

**}**

**});**

**jButton3.setText("Prev");**

**jButton3.setEnabled(false);**

**jButton3.addActionListener(new java.awt.event.ActionListener() {**

**public void actionPerformed(java.awt.event.ActionEvent evt) {**

**jButton3ActionPerformed(evt);**

**}**

**});**

**jButton4.setText("Next");**

**jButton4.setEnabled(false);**

**jButton4.addActionListener(new java.awt.event.ActionListener() {**

**public void actionPerformed(java.awt.event.ActionEvent evt) {**

**jButton4ActionPerformed(evt);**

**}**

**});**

**jCheckBox2.setSelected(true);**

**jCheckBox2.setText("View");**

**jCheckBox2.addActionListener(new java.awt.event.ActionListener() {**

**public void actionPerformed(java.awt.event.ActionEvent evt) {**

**jCheckBox2ActionPerformed(evt);**

**}**

**});**

**javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());**

**getContentPane().setLayout(layout);**

**layout.setHorizontalGroup(**

**layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)**

**.addGroup(layout.createSequentialGroup()**

**.addContainerGap()**

**.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)**

**.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)**

**.addGroup(layout.createSequentialGroup()**

**.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED\_SIZE, 77, javax.swing.GroupLayout.PREFERRED\_SIZE)**

**.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)**

**.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, 766, javax.swing.GroupLayout.PREFERRED\_SIZE)**

**.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)**

**.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE, 82, javax.swing.GroupLayout.PREFERRED\_SIZE)**

**.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)**

**.addComponent(jCheckBox2)**

**.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)**

**.addComponent(jCheckBox1)**

**.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)**

**.addComponent(jButton3, javax.swing.GroupLayout.PREFERRED\_SIZE, 77, javax.swing.GroupLayout.PREFERRED\_SIZE)**

**.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)**

**.addComponent(jButton4, javax.swing.GroupLayout.PREFERRED\_SIZE, 77, javax.swing.GroupLayout.PREFERRED\_SIZE)))**

**.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))**

**);**

**layout.setVerticalGroup(**

**layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)**

**.addGroup(layout.createSequentialGroup()**

**.addContainerGap()**

**.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)**

**.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)**

**.addComponent(jButton1)**

**.addComponent(jButton2)**

**.addComponent(jCheckBox1)**

**.addComponent(jButton3)**

**.addComponent(jButton4)**

**.addComponent(jCheckBox2))**

**.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)**

**.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)**

**.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))**

**);**

**pack();**

**}// </editor-fold>**

**private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {**

**if(jTextField1.getText().length() == 0){**

**JOptionPane.showMessageDialog(null, "Введите массив");**

**return;**

**}**

**mainList.clear();**

**linesList.clear();**

**jPanel1.removeAll();**

**jPanel1.update(g2);**

**jPanel1.revalidate();**

**GetNums();**

**mainList.add(numArr.clone());**

**srt = new Sort(mainList, linesList);**

**srt.quickSort(numArr, 0, numArr.length - 1);**

**//quickSort(0, numArr.length - 1);**

**if(draw) Draw(mainList.size());**

**SetNums();**

**}**

**private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {**

**JFileChooser chooser = new JFileChooser();**

**chooser.showOpenDialog(null);**

**File f = chooser.getSelectedFile();**

**try(Scanner in = new Scanner(f))**

**{**

**jTextField1.setText(in.nextLine());**

**}**

**catch(IOException ex){**

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

**}**

**}**

**private void jCheckBox1StateChanged(javax.swing.event.ChangeEvent evt) {**

**if(jCheckBox1.isSelected()){**

**//jButton3.setEnabled(true);**

**jButton4.setEnabled(true);**

**}else{**

**jButton3.setEnabled(false);**

**jButton4.setEnabled(false);**

**}**

**}**

**private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {**

**if(curr\_pos == 0){**

**draw = false;**

**jButton1ActionPerformed(evt);**

**}else{**

**jButton3.setEnabled(true);**

**}**

**//JOptionPane.showMessageDialog(null, mainList.size());**

**if(curr\_pos + 1> mainList.size())**

**{**

**jButton4.setEnabled(false);**

**jButton3.setEnabled(true);**

**SetNums();**

**return;**

**}**

**//Нужно удалить всё и отрисовать заного**

**jPanel1.removeAll();**

**jPanel1.revalidate();**

**jPanel1.update(g2);**

**//jPanel1.repaint();**

**curr\_pos++;**

**Draw(curr\_pos);**

**}**

**private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {**

**if(curr\_pos - 2 < 0)**

**{**

**jButton3.setEnabled(false);**

**return;**

**}**

**jButton4.setEnabled(true);**

**//Нужно удалить всё и отрисовать заного**

**jPanel1.removeAll();**

**jPanel1.revalidate();**

**jPanel1.update(g2);**

**//jPanel1.repaint();**

**curr\_pos--;**

**Draw(curr\_pos);**

**}**

**private void jTextField1MouseClicked(java.awt.event.MouseEvent evt) {**

**curr\_pos = 0;**

**}**

**private void jCheckBox2ActionPerformed(java.awt.event.ActionEvent evt) {**

**draw = jCheckBox2.isSelected();**

**}**

**private void GetNums(){**

**String strArr[] = jTextField1.getText().split(" ");**

**numArr = new int[strArr.length];**

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

**try{**

**numArr[i] = Integer.parseInt(strArr[i]);**

**}catch(NumberFormatException a){**

**JOptionPane.showMessageDialog(null, "Некорректный массив");**

**System.exit(1);**

**}**

**}**

**}**

**private void SetNums(){**

**String result = "";**

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

**result += Integer.toString(numArr[i]) + " ";**

**jTextField1.setText(result);**

**//JOptionPane.showMessageDialog(null, "Массив отсортирован, наслаждайся!");**

**}**

**/\*private void quickSort(int left, int right) {**

**int i = left, j = right;**

**int tmp, tmp\_line[] = new int[3];**

**int pivot = numArr[(left + right) / 2];**

**while (i <= j) {**

**while (numArr[i] < pivot)**

**i++;**

**while (numArr[j] > pivot)**

**j--;**

**if (i <= j) {**

**tmp = numArr[i];**

**numArr[i] = numArr[j];**

**numArr[j] = tmp;**

**if(i != j){**

**tmp\_line[0] = i;**

**tmp\_line[1] = j;**

**tmp\_line[2] = (left + right) / 2;**

**linesList.add(tmp\_line.clone());**

**mainList.add(numArr.clone()); //ADD**

**}**

**i++;**

**j--;**

**}**

**}**

**if (left < j){**

**quickSort(left, j);**

**}**

**if (i < right){**

**quickSort(i, right);**

**}**

**}\*/**

**private void Draw (int step) {**

**if(!draw && !jCheckBox2.isSelected()) return;**

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

**for(int q = 0; q < numArr.length; q++){**

**if(i < step - 1 && (q == linesList.get(i)[0] || q == linesList.get(i)[1] || q == linesList.get(i)[2])){**

**if(q == linesList.get(i)[2]){**

**g2.setColor(Color.red);**

**g2.setStroke(new BasicStroke(2));**

**}else{**

**g2.setColor(Color.blue);**

**g2.drawLine(-5 + linesList.get(i)[0]\*(radius+10) + radius, 5 + 40\*i + radius, -5 + linesList.get(i)[1]\*(radius+10) + radius, 5 + (i+1)\*40);**

**g2.drawLine(-5 + linesList.get(i)[1]\*(radius+10) + radius, 5 + 40\*i + radius, -5 + linesList.get(i)[0]\*(radius+10) + radius, 5 + (i+1)\*40);**

**g2.setStroke(new BasicStroke(2));**

**}**

**}**

**g2.drawString(Integer.toString(mainList.get(i)[q]), 12 + 30\*q, 20 + 40\*i);**

**g2.draw(new Ellipse2D.Float(5 + q\*(radius+10), 5 + 40\*i , radius, radius));**

**g2.setStroke(new BasicStroke(1));**

**g2.setColor(Color.BLACK);**

**}**

**}**

**jPanel1.revalidate();**

**}**

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

**try {**

**for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {**

**if ("Nimbus".equals(info.getName())) {**

**javax.swing.UIManager.setLookAndFeel(info.getClassName());**

**break;**

**}**

**}**

**} catch (ClassNotFoundException ex) {**

**java.util.logging.Logger.getLogger(Form.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);**

**} catch (InstantiationException ex) {**

**java.util.logging.Logger.getLogger(Form.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);**

**} catch (IllegalAccessException ex) {**

**java.util.logging.Logger.getLogger(Form.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);**

**} catch (javax.swing.UnsupportedLookAndFeelException ex) {**

**java.util.logging.Logger.getLogger(Form.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);**

**}**

**java.awt.EventQueue.invokeLater(new Runnable() {**

**public void run() {**

**new Form().setVisible(true);**

**}**

**});**

**}**

**// Variables declaration - do not modify**

**private javax.swing.JButton jButton1;**

**private javax.swing.JButton jButton2;**

**private javax.swing.JButton jButton3;**

**private javax.swing.JButton jButton4;**

**private javax.swing.JCheckBox jCheckBox1;**

**private javax.swing.JCheckBox jCheckBox2;**

**private javax.swing.JPanel jPanel1;**

**private javax.swing.JTextField jTextField1;**

**// End of variables declaration**

**}**

**Sort**

package myPack;

import java.util.ArrayList;

import java.util.List;

public class Sort {

//Список состояний массива

private List<int[]> mainList = new ArrayList<int[]>();

//Обводка

private List<int[]> linesList = new ArrayList<int[]>();

public Sort(List<int[]> a, List<int[]> b){

mainList = a;

linesList = b;

}

public void quickSort(int[] numArr, int left, int right) {

int i = left, j = right;

int tmp, tmp\_line[] = new int[3];

int pivot = numArr[(left + right) / 2];

/\* partition \*/

while (i <= j) {

while (numArr[i] < pivot)

i++;

while (numArr[j] > pivot)

j--;

if (i <= j) {

tmp = numArr[i];

numArr[i] = numArr[j];

numArr[j] = tmp;

if(i != j){

tmp\_line[0] = i;

tmp\_line[1] = j;

tmp\_line[2] = (left + right) / 2;

linesList.add(tmp\_line.clone());

mainList.add(numArr.clone()); //ADD

}

i++;

j--;

}

}

/\* recursion \*/

if (left < j){

quickSort(numArr, left, j);

}

if (i < right){

quickSort(numArr, i, right);

}

}

}