**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по учебной практике**

Тема: Пирамидальная сортировка

|  |  |  |
| --- | --- | --- |
| Студент гр. 6382 |  | Костин С.В. |
| Студентка гр. 6382 |  | Черкасова Е.И. |
| Студентка гр. 6383 |  | Терещенко В.Н. |
| Руководитель |  | Фирсов М.А. |

Санкт-Петербург

2018

**ЗАДАНИЕ**

**на учебную практику**

Студент Костин С.В. группы 6382

Студентка Черкасова Е.И. группы 6382

Студентка Терещенко В.Н. группы 6383

Тема практики: Пирамидальная сортировка

Задание на практику:

Командная итеративная разработка визуализатора алгоритма(ов) на Java с графическим интерфейсом.

Алгоритм: Пирамидальная сортировка.

Сроки прохождения практики: 27.06.2018 – 10.07.2018

Дата сдачи отчета: 09.07.2018

Дата защиты отчета: 09.07.2018

|  |  |  |
| --- | --- | --- |
|  | | |
| Студент |  | Костин С.В. |
| Студентка |  | Черкасова Е.И. |
| Студентка |  | Терещенко В.Н. |
| Руководитель |  | Фирсов М.А. |

**Аннотация**

В данной работе планируется изучить язык java и закрепить знания в ходе написания проектной работы, темой которой является «Пирамидальная сортировка». В ходе написания проекта должна быть написана программа, сортирующая массив целых чисел с визуализацией и пользовательский интерфейсом.

**Содержание**

**[введение](#_Toc518978722)** [5](#_Toc518978722)

**[1. требования к программе](#_Toc518978723)** [6](#_Toc518978723)

**[1.1. Исходные Требования к программе](#_Toc518978724)** [6](#_Toc518978724)

**[1.2. Уточнение требований после сдачи прототипа](#_Toc518978725)** [6](#_Toc518978725)

**[2. План разработки и распределение ролей в бригаде](#_Toc518978726)** [7](#_Toc518978726)

**[2.1. План разработки](#_Toc518978727)** [7](#_Toc518978727)

**[2.2. Распределение ролей в бригаде](#_Toc518978728)** [7](#_Toc518978728)

**[3.](#_Toc518978729)****[ОСОБЕННОСТИ РЕАЛИЗАЦИИ](#_Toc518978729)** [8](#_Toc518978729)

**[3.1.](#_Toc518978730)****[Основные методы](#_Toc518978730)** [8](#_Toc518978730)

**[4.](#_Toc518978798)****[Тестирование](#_Toc518978798)** [10](#_Toc518978798)

**[4.1](#_Toc518978799)****[Тестирование кода алгоритма](#_Toc518978799)** [10](#_Toc518978799)

**[4.2](#_Toc518978800)****[Тестирование графического интерфейса](#_Toc518978800)** [10](#_Toc518978800)

**[заключение](#_Toc518978801)** [14](#_Toc518978801)

**[список использованных источников](#_Toc518978802)** [15](#_Toc518978802)

**[приложение а](#_Toc518978803)** [16](#_Toc518978803)

# **введение**

Цель проектной работы: познакомиться и изучить объектно-ориентированный язык программирования Java; параллельно со знакомством, для закрепления полученных знаний, разработать программу, осуществляющую алгоритм пирамидальной сортировки, с ее визуализацией и созданием пользовательского интерфейса для более удобного использования программы.

Задачи:

1. Написать алгоритм пирамидальной сортировки.
2. Визуализировать алгоритм.
3. Разработать пользовательский интерфейс.
4. Протестировать работу программы.
5. Получить опыт командной работы.

# **1. требования к программе**

## **1.1. Исходные Требования к программе**

Входные данные: на вход подается число элементов, целое число, соответствующее реальному количеству элементов введенной ниже строки, и сама строка, состоящая из целых чисел, разделенных пробелами. Строка не должна содержать равные числа. Ввод данных осуществляется из файла или с окна запущенной программы.

Выходные данные: на экран выводится бинарное дерево (на каждой итерации).

Интерфейс приложения. До начала работы алгоритма на экран выводится бинарное дерево, соответствующее массиву из элементов исходной строки, и кнопка “Next step”. В процессе работы программы строится бинарное дерево, соответствующее каждому шагу сортировки (построение каждого следующего изображения осуществляется после нажатия пользователем на кнопку “Next step”). На финальном шаге работы с программой помимо дерева выводится сообщение, объявляющее успешное окончание работы.

Алгоритм сортировки прототипа реализован на основе массива. Визуализация бинарного дерева (частичная ее реализация) основана на переопределении функций графической библиотеки Swing.

## **1.2. Уточнение требований после сдачи прототипа**

Алгоритм изменен, теперь пирамидальная сортировка реализована на дереве. Визуализация дерева адаптирована под библиотеку GraphStream.

**1.3 Уточнение требований после сдачи второй версии проекта**

Интерфейс приложения. При запуске программы на экран выводится поле, где позже будет рисоваться бинарное дерево, кнопки “Open file”, “Start sort”, “Next step”, “Previous step” и текстовое поле для ручного ввода массива. В процессе работы программы строится бинарное дерево, соответствующее каждому шагу сортировки (построение каждого следующего изображения осуществляется после нажатия пользователем на кнопку “Next step”).

# **2. План разработки и распределение ролей в бригаде**

## **2.1. План разработки**

Прототип программы написан к 07.07. Далее происходило усовершенствование проекта и исправление недочетов.

Для написания проекта используется система контроля версий GitHub: https://github.com/cherowl/java-project.

## **2.2. Распределение ролей в бригаде**

Костин С.В. ответственный за интерфейс.

Черкасова Е.И. – за реализацию алгоритма.

Терещенко В.Н. – тестирование.

1. **ОСОБЕННОСТИ РЕАЛИЗАЦИИ**
   1. **Основные методы**

Таблица 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Название метода | Входные данные | Выходные данные | Назначение |
| Controller.java | | | |
| private void checkTextField() | – | – | Проверяет, что введено в текстовом поле |
| public void startSort() | – | – | Содержит в себе действия кнопки “Start sort” |
| public void nextStep() | – | – | Содержит в себе действия кнопки “Next step” |
| public void previousStep() | – | – | Содержит в себе действия кнопки “Previous step” |
| BinTree.java | | | |
| private NodeTree addRecursive(NodeTree current, int value, NodeTree parent) | Принимает корневой узел, новое значение, и родителя | Возвращает корневой узел | Рекурсивный метод добавления нового узла со значением value в дерево |
| public BinTree createBinTree() | – | Возвращает дерево | Метод создания дерева. |
| public static int size(NodeTree node) | Принимает корневой узел | Возвращает размер дерева | Определения размера дерева |
| public void printTree(NodeTree root) | Принимает корневой узел | – | Печать дерева на консоль |
| NodeTree.java | | | |
| public void setLevel() | – | – | Рассчитывает уровень |
| public int getValue() | – | Возвращает значение узла | Получает значение узла |
| public int getLevel() | – | Возвращает уровень | Получает уровень узла |
| public NodeTree getLeft() | – | Возвращает левого потомка | – |
| public NodeTree getRight() | – | Возвращает правого потомка | – |
| ButtonsPanel.java | | | |
| private JButton createButton(String text, int x, int y) | Текст и координаты расположения кнопки | Созданная кнопка | Вызывает создателя кнопки |
| static JButton getjButton(String text, int x, int y) | Текст и координаты расположения кнопки | Созданная кнопка | Создает кнопку |

1. **Тестирование**
   1. **Тестирование кода алгоритма**

В ходе написания проекта был протестирован самый главный класс алгоритма – HeapSort с помощью JUnit5, ошибок найдено не было (рис. 1).

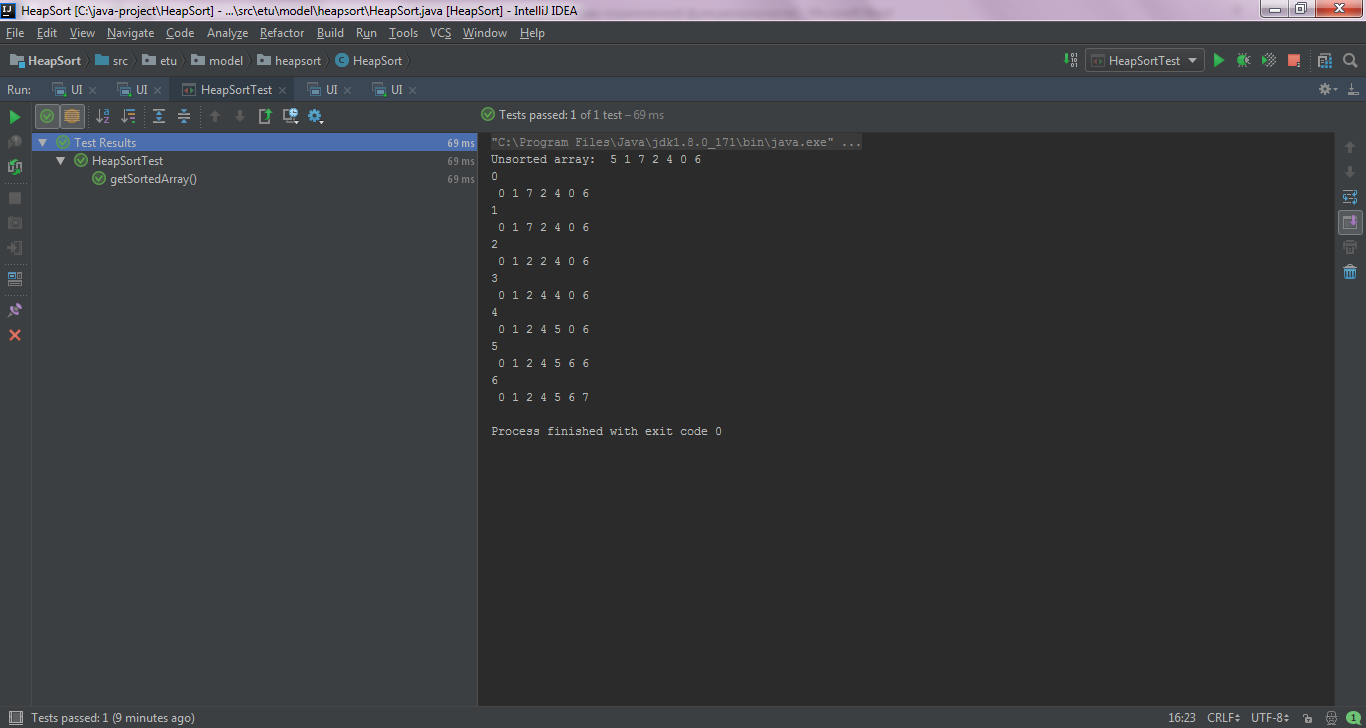


Рисунок 1

Код теста представлен в Приложении А (HeapSortTest.java).

* 1. **Тестирование графического интерфейса**

При входе в программу пользователю представляется стартовый экран (рис. 2).

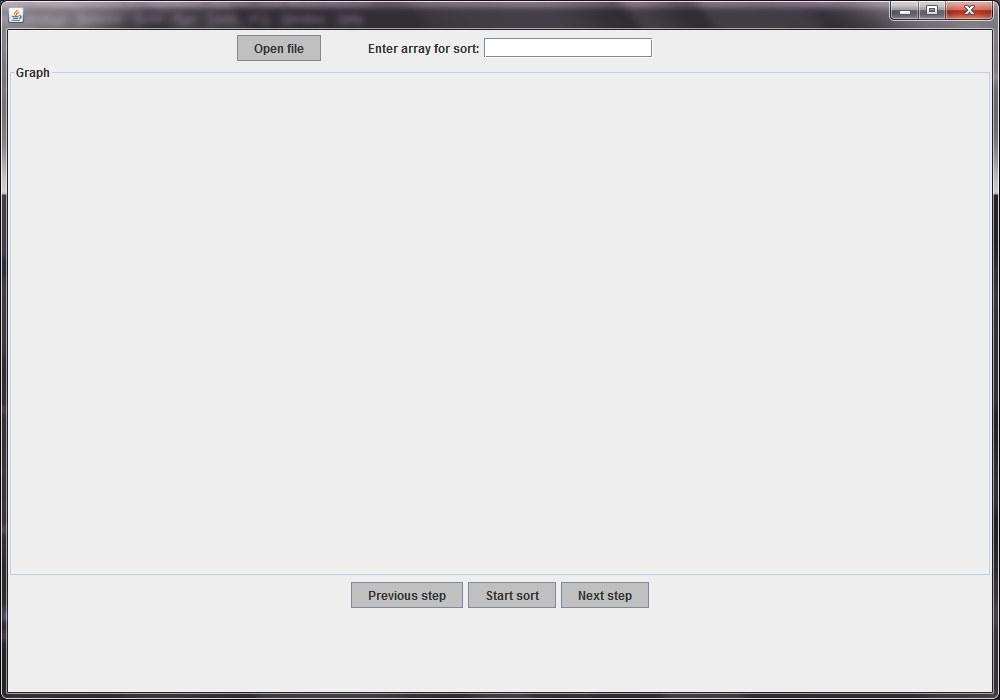


Рисунок 2

Пользователь выбирает, как ввести исходную информацию в программу: из файла (“File open”) или из текстового поля. При нажатии кнопки “File open” открывается окно, в котором необходимо выбрать файл, который содержит исходные данные (рис. 3).

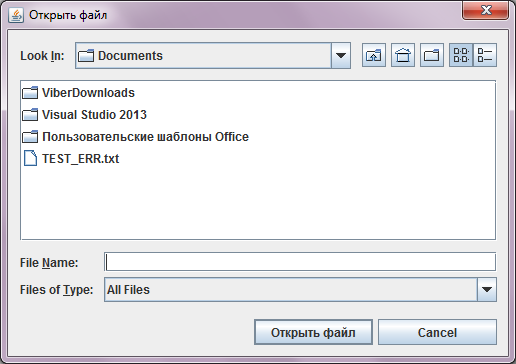


Рисунок 3

После считывания информации следует нажать кнопку “Start sort” – на экране появится изображение бинарного дерева, соответствующего введенной информации (рис. 4).

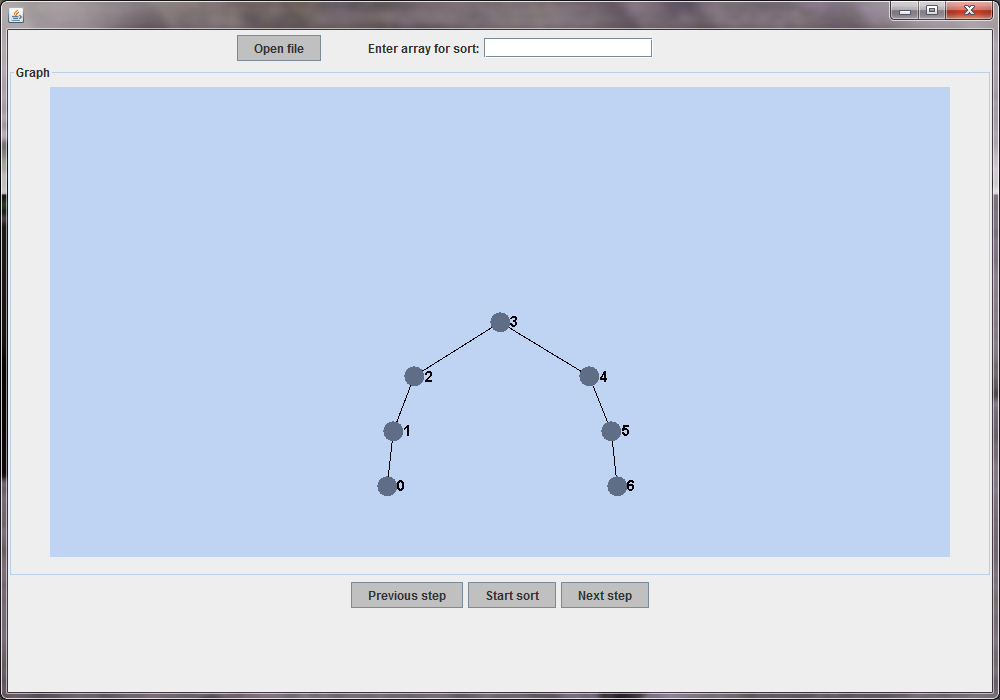


Рисунок 4

Если попытаться нажать на кнопку “Next step” или “Previous step”, на экране появится сообщение для пользователя, что так делать не следует (рис. 5).

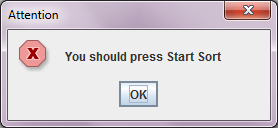


Рисунок 5

Нажатием кнопок “Next step” и “Previous step” пользователь может просмотреть каждый шаг итерации.

При попытке нажать кнопку “Next step” после просмотра последнего шага алгоритма появится соответствующее сообщение (рис. 6).

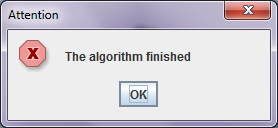


Рисунок 6

Аналогично и при попытке нажать кнопку “Previous step” на самом первом шаге алгоритма (рис. 7).

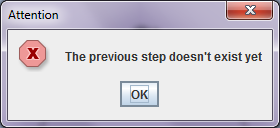


Рисунок 7

# **заключение**

В ходе выполнения проектной работы были получены знания о языке Java, необходимые для создания программы, были изучены: библиотека для создания графического интерфейса Swing при создании визуализации прототипа и пользовательского интерфейса, библиотека динамического графа GraphStream для построения бинарного дерева, класс для выполнения основных числовых операций Math и другие сопутствующие созданию проекта библиотеки. Программа была протестирована, ошибок в алгоритме найдено не было. Был получен опыт командной работы. Вышеперечисленная информация свидетельствует о выполнении поставленных задач.

# **список использованных источников**

1. Пирамидальная сортировка. [Электронный ресурс] URL: https://prog-cpp.ru/sort-pyramid
2. Википедия «Пирамидальная сортировка». [Электронный ресурс] Дата обновления: 10 июня 2018. URL: https://goo.gl/MqoTis
3. Библиотека GraphStream. 2010-2015, [GraphStream Team.](http://graphstream-project.org/doc/About-Us/)  [Электронный ресурс] URL: http://graphstream-project.org

# **приложение а**

**код теста**

--HeapSortTest.java--

import etu.model.heapsort.HeapSort;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.junit.jupiter.api.Assertions.assertTrue;

class HeapSortTest {

private HeapSort heapSort;

@Test

void getSortedArray() {

int[] testArr = {5, 1, 1, 2, 4, 0, 6};

heapSort = new HeapSort(testArr);

assertTrue(heapSort.getArray().length > 0);

assertEquals(0, heapSort.getArray()[0], "Cell №0");

assertEquals(1, heapSort.getArray()[1], "Cell №1");

assertEquals(1, heapSort.getArray()[2], "Cell №2");

assertEquals(2, heapSort.getArray()[3], "Cell №3");

assertEquals(4, heapSort.getArray()[4], "Cell №4");

assertEquals(5, heapSort.getArray()[5], "Cell №5");

assertEquals(6, heapSort.getArray()[6], "Cell №6");

}

}