МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по учебной практике

Тема: Сортировка: макс-куча

Студент гр. 1383 _____ Богданов Е.М.

Студентка гр. 1383 _____ Манучарова А.С.

Студент гр. 1383 _____ Харитонов Н.М.

Руководитель _____ Токарев А.П.

Санкт-Петербург
2023

ЗАДАНИЕ
НА УЧЕБНУЮ ПРАКТИКУ

Студент Богданов Е.М. группы 1383

Студентка Манучарова А.С. группы 1383

Студент Харитонов Н.М. группы 1383

Тема практики: Сортировка: макс-куча

Задание на практику:

Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.

Алгоритм: Макс-куча.

Сроки прохождения практики: 30.06.2023 – 13.07.2023

Дата сдачи отчета: 12.07.2020

Дата защиты отчета: 12.07.2020

Студент	_____	Богданов Е.М.
Студентка	_____	Манучарова А.С.
Студент	_____	Харитонов Н.М.
Руководитель	_____	Токарев А.П.

АННОТАЦИЯ

В ходе выполнения мини-проекта требуется представить алгоритм сортировки "макс-куча" в виде графического интерфейса. Программа должна посредством понятного и удобного интерфейса давать пользователю возможность выбрать способ добавления элементов в дерево: через файл, указанный пользователем, или ввод каждого элемента вручную. По мере добавления элементов, они должны отражаться на пользовательском экране. Должен быть реализован следующий функционал: окно для добавления нового элемента, кнопка для вставки этого элемента, кнопка для одного шага просеивания, кнопка для удаления последнего добавленного элемента (программа возвращается к состоянию за шаг до добавления этого элемента), кнопка начала заново. После каждого нажатия на экран должно выводиться сообщение о работе программы. В конечном итоге программа должна выводить на экран отсортированный массив.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы состоит в реализации алгоритма сортировки «макс-куча» в виде графического интерфейса. Основными задачами работы являются: возможность считывания данных из файла и при вводе пользователем непосредственно, наглядное представление каждого этапа сортировки на экране посредством нажатия на соответствующие кнопки, вывод отсортированного массива.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

1.1.1 Требования к вводу исходных данных

Должна быть реализована возможность выбора пользователем способа ввода данных: через файл или самим пользователем через графический интерфейс.

1.1.2 Требования к визуализации

Представление алгоритма в виде дерева (макс-кучи) – каждый элемент в своей ячейке, окно для добавления элементов и соответствующая кнопка, кнопка для шага вперед, кнопка для удаления последнего введенного элемента, кнопка для начала алгоритма заново, панель для вывода отсортированного массива.

1.2 Уточнение требований после сдачи прототипа

Вывод сообщений о работе алгоритма на каждом шаге.

1.3 Уточнение требований после сдачи 1-ой версии

1) Вывод значений из считанного файла на экран
2) Более подробные комментарии к шагам алгоритма
3) Адаптивная вёрстка – возможность расширять экран
4) Изменение ограничений на кол-во элементов
5) Возможность добавлять новые элементы после считывания из файла

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

1. Составление спецификации и плана разработки.
2. MVP (minimum viable product).
3. Стартовый экран приложения.
4. Экран редактирования.
5. Тесты

2.2. Распределение ролей в бригаде

1. Составление спецификации и плана разработки. Исполнители: Богданов Е.М., Манучарова А.С., Харитонов Н.М.
2. MVP (minimum viable product). Исполнители: Харитонов Н.М., Богданов Е.М., Манучарова А.С.
3. Стартовый экран приложения.
   - По указанному полному пути к файлу считать входные данные в массив. Исполнитель: Манучарова А.С.
   - Сортировка (реализация макс-кучи), тестирование. Исполнитель: Богданов Е.М.
   - Верстка стартового экрана. Исполнитель: Харитонов Н.М.
4. Экран редактирования.
   - Верстка. Исполнитель: Харитонов Н.М.
   - Выделение элементов при просеивании. Исполнитель: Манучарова А.С.
5. Тесты
   - Чтение из файла. Исполнитель: Манучарова А.С.
   - Сортировка. Исполнитель: Богданов Е.М.
   - Графический интерфейс. Исполнитель: Харитонов Н.М.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Структуры данных

1) Классический массив “[]”;

2) List – интерфейс, его реализация в виде ArrayList;

3) ObservableList (у визуальных элементов-контейнеров, типа Pane, AnchorPane и т.п.).

3.2. Основные методы

Считывание из файла:

setFileName() – метод, принимающий строку – полный путь к файлу. Присваивает имя полю file и возвращает проверку файла на существование.

makeArray() – в случае несуществования файла выбрасывается исключение. Затем производится чтение данных файла, элементы разделяются по пробельному символу. Запись в массив осуществляется, выполняя требование о наибольшем количестве элементов, равном 30. Метод возвращает массив целых чисел.

Сортировка:

sift_up_step() — метод, для сравнение текущей вершины с её родителем и «обмена» значений, в случае, если родитель меньше.

sift_down_step() — метод, для сравнение текущей вершины с её потомками(при наличии) и «обмена» значений, если текущее меньше чем значения кого-либо из потомка.

add_no_sift() — метод, для добавления нового элемента в дерево без «просеивания».

heap_sort_start() — метод, который меняет местами корень дерева и последний «лист» дерева.

Графический интерфейс:

onButtonStepClick() – метод, который при нажатии на кнопку “Сделать шаг” делает шаг алгоритма при просеивании вверх или вниз (вниз – после

нажатия кнопки “Извлечь массив”). Необходима для пошагового выполнения операций просеивания с визуализацией сравниваемых элементов.

onButtonAddClick() – метод, который при нажатии на кнопку “Добавить элемент” добавляет элемент из textField или из файла, если его считывание не закончилось.

onDeleteLastClick() – метод, который при нажатии на кнопку “Удалить последний элемент” удаляет последний добавленный элемент, а на экран выводит предыдущий вариант дерева.

onMakeArrayClick() – метод, который при нажатии на кнопку “Извлечь массив” делает переход к стадии получения отсортированного массива из дерева.

onRestartClick() – метод, который при нажатии на кнопку “Начать заново” начинает программу заново.

onFileReadClick() – метод, который при нажатии на кнопку “Чтение из файла” открывает окно с выбором файла в системе для извлечения из него массива.

onInteractiveClick() – метод, который при нажатии на кнопку “Интерактивный ввод” позволяет пользователю добавлять элементы с помощью поля для ввода текста.

createElements() – метод, делающий из массива элементы пользовательского интерфейса, которые составляют бинарное дерево, отображаемое на экране.

calcWidth() – метод, который по порядковому номеру элемента и по их количеству выдает абсциссу элемента (ось x).

calcHeight() – метод, который по порядковому номеру элемента выдает ординату элемента (ось y).

makeReverseArray() – метод, который из массива элементов (точнее, списка List<Integer>), являющимся готовым для ответа, но отсортированным по убыванию, делает массив строкой, отсортированной по возрастанию.

arrayToString() – метод, который массив элементов (классического)

превращает в строку.

isNumber() – метод, который проверяет, является ли данная строка целым числом (он вызывается, если не получилось преобразование Integer.parseInt()).

4. ТЕСТИРОВАНИЕ

4.1. Тестирование считывания файла

Таблица 1 – Проверка файла на существование

№	Входные данные	Выходные данные	Комментарий
1	temp.txt	true	Корректно
2	temp2.txt	true	Корректно
3	temp100.txt	false	Корректно

4.2. Тестирование сортировки

Таблица 2 – Тестирование алгоритма сортировки

№	Входные данные	Выходные данные	Комментарий
1	6 5 4 3 2 1	1 2 3 4 5 6	Корректно
2	0 4 -2 -3 5 -1	-3 -2 -1 0 4 5	Корректно
3	0	0	Корректно
4	0 0 0 0	0 0 0 0	Корректно
5	-1 0 1	-1 0 1	Корректно

4.3. Тестирование визуальной части

Таблица 3 – Получение абсциссы элемента по порядковому номеру

№	Входные данные	Выходные данные	Комментарий
1	0, 20	580	Корректно
2	5, 20	730	Корректно
3	14, 34	2230	Корректно

Таблица 4 – Получение ординаты элемента по порядковому номеру

№	Входные данные	Выходные данные	Комментарий
1	0	10	Корректно

2	5	110	Корректно
3	23	210	Корректно

Таблица 5 – Получение из массива элементов по убыванию строки элементов по возрастанию

№	Входные данные	Выходные данные	Комментарий
1	[5, 7, 3]	“3, 7, 5”	Корректно
2	[5, 7, 3, 97]	“97, 3, 7, 5”	Корректно
3	[]	“”	Корректно

Таблица 6 – Проверка элемента на число

№	Входные данные	Выходные данные	Комментарий
1	"-34565754654390"	true	Корректно
2	"2-345534534569"	false	Корректно
3	"234553453456910"	true	Корректно

ЗАКЛЮЧЕНИЕ

В ходе выполнения проекта была реализована сортировка «макс-куча» в виде графического интерфейса. Выполнена возможность считывания данных из файла и при введении пользователем. Наглядно представлены все этапы сортировки с выводом сообщений о работе программы. Выполнена адаптивная верстка.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шилдт Г. - Java 8. Руководство для начинающих - 2015
2. https://stackoverflow.com
3. https://javarush.com

ПРИЛОЖЕНИЕ А

СЧИТЫВАНИЕ ИЗ ФАЙЛА

ReadFile.java

```
import java.io.*;
import java.util.Scanner;

public class ReadFile {
    private File file;

    public boolean setFileName(String fileName) {
        this.file = new File(fileName);
        return file.exists();
    }

    public File getFileName() {
        return this.file;
    }

    public int[] makeArray() throws IOException {
        if (!file.exists()) {
            throw new IOException("File doesn't exist.");
        }
        FileReader fileReader = new FileReader(file);
        BufferedReader bufferedReader = new BufferedReader(fileReader);
        String line = bufferedReader.readLine();
        String[] strArr = line.split(" ");
        int len = Math.min(strArr.length, 100);
        int[] numArr = new int[len];
        for (int i = 0; i < strArr.length; i++) {
            if (i < 100){
                numArr[i] = Integer.parseInt(strArr[i]);
            }
            else break;
        }
        return numArr;
    }
}
```

ПРИЛОЖЕНИЕ В

ТЕСТИРОВАНИЕ ЧТЕНИЯ ИЗ ФАЙЛА

Heap.java

```
import org.junit.Test;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

import static org.junit.Assert.*;

public class ReadFileTest {

    @Test
    public void makeArr() throws IOException {
        File file1 = new File("temp.txt");
        File file2 = new File("temp2.txt");
        FileWriter fileWriter1 = new FileWriter(file1);
        FileWriter fileWriter2 = new FileWriter(file2);
        if (!file1.exists()){
            file1.createNewFile();
        }
        fileWriter1.write("2 35 -1 0 72 -15 8 100");
        fileWriter1.flush();
        fileWriter1.close();
        if (!file2.exists()){
            file2.createNewFile();
        }
        fileWriter2.write("55 63 -88 -77 -20 49 50 -58 23 -91 51 69 22 -
36 -39 -83 55 -81 -57 -98 13 11 -52 -8 -82" +
                " -30 -30 -46 1 6 -78 -88 91 -62 7 52 -92 13 90 -73 15 -
51 -60 2 29 -1 -80 -74 26 20 44 11 -14 -1 23 -26 -87 " +
                "-54 -37 72 1 59 -17 59 -16 75 -96 -79 26 -19 -10 42 -60
1 -78 -31 -59 -73 8 -56 -36 55 56 -37 91 9 -77 -64 35 " +
                "-83 -54 -27 61 4 30 -6 0 48 -51 27 -37 75 -13 31 74 93 -
3 62 10 -19");
        fileWriter2.flush();
        fileWriter2.close();
        ReadFile newFile1 = new ReadFile();
        newFile1.setFileName("temp.txt");
        ReadFile newFile2 = new ReadFile();
        newFile2.setFileName("temp2.txt");
        assertArrayEquals(newFile1.makeArray(), new int[] {2, 35, -1, 0,
72, -15, 8, 100});
        assertArrayEquals(newFile2.makeArray(), new int[] {55, 63, -88, -
77, -20, 49, 50, -58, 23, -91, 51, 69, 22, -36, -39, -83, 55,
                -81, -57, -98, 13, 11, -52, -8, -82, -30, -30, -46, 1, 6,
-78, -88, 91, -62, 7, 52, -92, 13, 90, -73, 15, -51, -60, 2,
                29, -1, -80, -74, 26, 20, 44, 11, -14, -1, 23, -26, -87,
-54, -37, 72, 1, 59, -17, 59, -16, 75, -96, -79, 26, -19, -10,
                42, -60, 1, -78, -31, -59, -73, 8, -56, -36, 55, 56, -37,
91, 9, -77, -64, 35, -83, -54, -27, 61, 4, 30, -6, 0, 48, -51, 27});
    }

    @Test
    public void setName() {
```

```
        ReadFile file = new ReadFile();
        assertTrue(file.setFileName("temp.txt"));
        assertTrue(file.setFileName("temp2.txt"));
        assertFalse(file.setFileName("temp100.txt"));
    }
}
```

ПРИЛОЖЕНИЕ С

СОРТИРОВКА

HeapForTests.java

```
public class Heap {
    enum States{
        COMPARE,
        CHANGE,
        DEFAULT
    }
    private States state = States.DEFAULT;
    private int[] heap=new int[0];
    private int last;
    private int index1 = -1;
    private int index2 = -1;
    private int index3 = -1;

    public int getIndex3() {
        return index3;
    }

    public States getState() {
        return state;
    }

    public int getIndex1() {
        return index1;
    }

    public int getIndex2() {
        return index2;
    }

    public int getLast() {
        return last;
    }

    public Heap()
    {
        last=0;
    }
    public static int swap(int a,int b)
    {
        return a;
    }

    public int[] getHeap() {
        return heap;
    }


    public void sift_up(int current_i)
    {
        int next=current_i;
        while(next>-1)
        {
```

```
            current_i=next;
            next=-1;
            if(current_i==0)
                return;
            int parent_i=current_i/2;
//            System.out.println("Сравнение");
            if(heap[parent_i]<heap[current_i])//Сравниваем текущую
вершину с её
родителем. Если текущая>родитель,то меняем местами
            {
                heap[current_i]=swap(heap[parent_i],
heap[parent_i]=heap[current_i]);
                next=parent_i;
                continue;
            }
        }
    }
    public void add(int current)
    {
        int []arr=new int[heap.length+1];
        for(int i=0;i<heap.length;++i)
            arr[i]=heap[i];
        arr[last]=current;
        heap=arr;
        sift_up(last);
        last++;
    }
    public void add_no_sift(int current){

        int []arr=new int[heap.length+1];
        for(int i=0;i<heap.length;++i)
            arr[i]=heap[i];
        arr[last]=current;
        heap=arr;
        System.out.println("aue"+heap+"\n");
        last++;
    }

    public int sift_up_step(int current_i){
        int next = current_i;
        if (next>-1) {
            next = -1;
            if (current_i == 0) {
                return next;
            }
            int parent_i = (current_i-1) / 2;
            index1 = parent_i;
            index2 = current_i;
            System.out.println("Сравнение " + index1 + "  " + index2);
            state = States.COMPARE;
            if (heap[parent_i] < heap[current_i])//Сравниваем текущую
вершину с
её родителем. Если текущая>родитель,то меняем местами
            {
                state = States.CHANGE;
                heap[current_i] = swap(heap[parent_i], heap[parent_i] =
heap[current_i]);
```

```
                next = parent_i;
            }
        }
        return next;
    }
    public int sift_down_step(int current_i){
        int next=current_i;
        if(next>-1)
        {
            state = States.DEFAULT;
            next=-1;
            int left;//Значение левого ребёнка текущей
вершины
            int rigth=-2147483648;//Значение правого ребёнка
текущей вершины
            int left_i=2*current_i+1;//Индекс левого ребёнка
            int rigth_i=2*current_i+2;//Индекс правого ребёнка
            int current=heap[current_i];//Значение текущей вершины

            if(left_i>=last)//Если нет детей,то не просееваем
                return next;

            left=heap[left_i];

            if(rigth_i<last)
                rigth=heap[rigth_i];
            state = States.COMPARE;
            index1 = current_i;
            index2 = left_i;
            index3 = rigth_i;
            if(current>left && current>rigth)//Сравнение текущей вершины
со
всеми остальными
                return next;

            if(left>current && left>=rigth)//Сравнение левого ребёнка со
всеми остальными
            {
                state = States.CHANGE;
                heap[left_i]=swap(current,heap[current_i]=left);
                next=left_i;
                return next;
            }
            if(rigth>current && rigth>=left)//Сравнение правого ребёнка с
остальными
            {
                state = States.CHANGE;
                heap[rigth_i]=swap(current,heap[current_i]=rigth);
                next=rigth_i;
                index2 = rigth_i;
                index3 = left_i;
                return next;
            }
        }
        return next;
    }
    public void sift_down(int current_i)
```

```
    {
        int next=current_i;
        while(next>-1)
        {
            current_i=next;//Индекс текущей вершины
            next=-1;
            int left;//Значение левого ребёнка текущей
вершины
            int rigth=-1000000;//Значение правого ребёнка
текущей вершины
            int left_i=2*current_i+1;//Индекс левого ребёнка
            int rigth_i=2*current_i+2;//Индекс правого ребёнка
            int current=heap[current_i];//Значение текущей вершины

            if(left_i>=last)//Если нет детей,то не просеиваем
                return;

            left=heap[left_i];

            if(rigth_i<last)
                rigth=heap[rigth_i];

            if(current>left && current>rigth)//Сравнение текущей вершины
со
всеми остальными
                return;

            if(left>current && left>=rigth)//Сравнение левого ребёнка со
всеми остальными
            {
                heap[left_i]=swap(current,heap[current_i]=left);
                next=left_i;
                continue;
            }
            if(rigth>current && rigth>=left)//Сравнение правого ребёнка с
остальными
            {
                heap[rigth_i]=swap(current,heap[current_i]=rigth);
                next=rigth_i;
                continue;
            }
        }

    }

    public int heap_sort_start(){
        heap[last-1]=swap(heap[0],heap[0]=heap[last-1]);//добавление
наибольшего
элемента в конец массива и замена его на "последнего" листа
        --last;
        return heap[last];
    }
    public void heap_sort()
    {
        int temp=last;
        while(last>0)
        {
```

```
            --last;
            sift_down(0);

        }
        last=temp;
    }
    public void print()
    {
        for(int i=0;i<last;++i)
        {
            System.out.printf("%d ",heap[i]);
        }
        System.out.printf("\n%d\n",last);
    }
}
```

ПРИЛОЖЕНИЕ D

ТЕСТИРОВАНИЕ СОРТИРОВКИ

HeapForTests.java

```
package testing;
public class HeapForTests {
    enum States{
        COMPARE,
        CHANGE,
        DEFAULT
    }
    private States state = States.DEFAULT;
    private int[] heap=new int[0];
    private int last;
    private int index1 = -1;
    private int index2 = -1;
    private int index3 = -1;

    public int getIndex3() {
        return index3;
    }
    public States getState() {
        return state;
    }
    public int getIndex1() {
        return index1;
    }
    public int getIndex2() {
        return index2;
    }

    public int getLast() {
        return last;
    }

    public HeapForTests()
    {
        last=0;
    }
    public HeapForTests(int siz)
    {
        int []arr=new int [siz];
        heap=arr;
        last=0;
    }
    public HeapForTests(int[] arr)
    {
        heap=arr;
        last=arr.length;
    }
    public void reHeap()
    {
        int []arr=new int[0];
        heap=arr;
        last=0;
    }
```

```
    public static int swap(int a,int b)
    {
        return a;
    }

    public int[] getHeap() {
        return heap;
    }
    public void add_no_sift(int current){

        int []arr=new int[heap.length+1];
        for(int i=0;i<heap.length;++i)
            arr[i]=heap[i];
        arr[last]=current;
        heap=arr;
        last++;
    }
    public int sift_up_step(int current_i){
        int next = current_i;
        if (next>-1) {
            next = -1;
            if (current_i == 0) {
                return next;
            }
            int parent_i = (current_i-1) / 2;
            index1 = parent_i;
            index2 = current_i;
            //System.out.println("Сравнение " + index1 + "  " + index2);
            state = States.COMPARE;
            if (heap[parent_i] < heap[current_i])//Сравниваем текущую
вершину с
её родителем. Если текущая>родитель,то меняем местами
            {
                state = States.CHANGE;
                heap[current_i] = swap(heap[parent_i], heap[parent_i] =
heap[current_i]);
                next = parent_i;
            }
        }
        return next;
    }
    public int sift_down_step(int current_i){
        int next=current_i;
        if(next>-1)
        {
            state = States.DEFAULT;
            next=-1;
            int left;//Значение левого ребёнка текущей
вершины
            int rigth=-2147483648;//Значение правого ребёнка
текущей вершины
            int left_i=2*current_i+1;//Индекс левого ребёнка
            int rigth_i=2*current_i+2;//Индекс правого ребёнка
            int current=heap[current_i];//Значение текущей вершины

            if(left_i>=last)//Если нет детей,то не просееваем
                return next;
```

```
            left=heap[left_i];

            if(rigth_i<last)
                rigth=heap[rigth_i];
            state = States.COMPARE;
            index1 = current_i;
            index2 = left_i;
            index3 = rigth_i;
            if(current>left && current>rigth)//Сравнение текущей вершины
со
всеми остальными
                return next;

            if(left>current && left>=rigth)//Сравнение левого ребёнка со
всеми остальными
            {
                state = States.CHANGE;
                heap[left_i]=swap(current,heap[current_i]=left);
                next=left_i;
                return next;
            }
            if(rigth>current && rigth>=left)//Сравнение правого ребёнка с
остальными
            {
                state = States.CHANGE;
                heap[rigth_i]=swap(current,heap[current_i]=rigth);
                next=rigth_i;
                index2 = rigth_i;
                index3 = left_i;
                return next;
            }
        }
        return next;
    }
    public int heap_sort_start(){
        heap[last-1]=swap(heap[0],heap[0]=heap[last-1]);//добавление
наибольшего
элемента в конец массива и замена его на "последнего" листа
        --last;
        return heap[last];
    }
    public void print()
    {
        for(int i=0;i<last;++i)
        {
            System.out.printf("%d ",heap[i]);
        }
        System.out.println("");
    }
    public void sift_up()
    {
        int next=getLast()-1;
        while(next>-1)
            next= sift_up_step(next);
    }
    public void sift_down()
```

```
    {
        int next=0;
        while(next>-1)
            next=sift_down_step(next);
        //System.out.println(last);
    }
    public void Add(int key)
    {
        add_no_sift(key);
        sift_up();
    }
    public void make_heap()
    {
        int []temp=heap;
        heap=new int[0];
        last=0;
        for(int i=0;i<temp.length;++i)
        {
            Add(temp[i]);
        }
    }
    public void heapSort()
    {
        int temp=last;
        for(;last>0;)
        {
            heap_sort_start();
            sift_down();
        }
        last=temp;
    }

}
```

Tests.java

```
package testing;
import static org.junit.Assert.*;
public class Tests {
    public static void main(String[] args)
    {
        int [][]tests={{6,5,4,3,2,1},{0,4,-2,-3,5,-1},{0},{0,0,0,0},{-
1,0,1}};
        int [][]ans={{1,2,3,4,5,6},{-3,-2,-1,0,4,5},{0},{0,0,0,0},{-
1,0,1}};
        HeapForTests a=new HeapForTests();
        for(int i=0;i<5;++i)
        {
            a.reHeap();
            for(int j=0;j<tests[i].length;++j)
            {
                a.Add(tests[i][j]);
            }
            a.heapSort();
            assertArrayEquals(a.getHeap(),ans[i]);
        }
    }
}
```

ПРИЛОЖЕНИЕ Е

ГРАФИЧЕСКИЙ ИНТЕРФЕЙС

Файл разметки hello-view.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox alignment="CENTER" minHeight="200" minWidth="300"
xmlns:fx="http://javafx.com/fxml/1"
xmlns="http://javafx.com/javafx/17.0.2-ea"
fx:controller="com.example.demo1.HelloController">

   <ScrollPane id="" prefWidth="Infinity" prefHeight="400" >
      <AnchorPane fx:id="graph">
         <children>

         </children>
      </AnchorPane>

   </ScrollPane>
   <VBox prefWidth="Infinity" alignment="CENTER" prefHeight="50" style="
-fx-border-color:#ccc; -fx-border-width: 1; -fx-border-style: solid;">
      <Label fx:id="labelFromFile" wrapText="true"/>
   </VBox>
   <VBox prefWidth="Infinity" alignment="CENTER" prefHeight="50" style="
-fx-border-color:#ccc; -fx-border-width: 1; -fx-border-style: solid;" >
      <Label fx:id="labelArray" wrapText="true"/>
   </VBox>
   <VBox prefWidth="Infinity" alignment="CENTER" prefHeight="50" style="
-fx-border-color:#ccc; -fx-border-width: 1; -fx-border-style: solid;">
      <Label fx:id="labelForUser" text="Выберите тип ввода"
wrapText="true"/>
   </VBox>

   <HBox alignment="CENTER" prefWidth="Infinity" prefHeight="50">
      <Button onAction="#onButtonStepClick" text="Сделать шаг" />
      <Button onAction="#onButtonAddClick" text="Добавить элемент" />
      <TextField fx:id="textField" promptText="Введите число"/>
      <Button onAction="#onDeleteLastClick" text="Удалить последний
жлемент"/>
      <Button onAction="#onMakeArrayClick"  text="Извлечь массив" />
      <Button onAction="#onRestartClick"  text="Начать заново" />
      <Button onAction="#onFileReadClick"  text="Чтение из файла" />
      <Button onAction="#onInteractiveClick"  text="Интерактивный ввод"
/>
   </HBox>
</VBox>
```

HelloController.java

```
package com.example.demo1;


import javafx.fxml.FXML;
import javafx.geometry.Pos;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.stage.FileChooser;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class HelloController {



    enum Type{
        FILE,
        UI,
        NOT_SELECTED
    }
    private Type type = Type.NOT_SELECTED;
    private String path = "";
    private List<Integer> arr = new ArrayList<>();

    private boolean isFirst = true;
    private boolean isSorting = false;
    private boolean changeElements = true;
    private List<Integer> resultArray = new ArrayList<>();
    private boolean canAdd = false;
    private int[] arrayFile;
    private int current;
    private Heap heap = new Heap();
    private int next;
    private FileChooser fileChooser;
    public static final int WIDTH = 1200;
    private static final List<List<Integer>> historyOfArrays = new
ArrayList<>();



    public static String makeReverseArray(List<Integer> list){
        if (list.isEmpty()) return "";
        StringBuilder stringBuilder = new StringBuilder();
        for (int i = 0; i < list.size()-1; i++){
            stringBuilder.append(list.get(list.size()-i-1));
            stringBuilder.append(", ");
        }
        stringBuilder.append(list.get(0));
        return stringBuilder.toString();
```

```
    }
    private static void copyForHistory(List<Integer> array){
        List<Integer> tempArr = new ArrayList<>(array);
        historyOfArrays.add(tempArr);
    }
    public String arrayToString(int[] arr){
        if (arr.length == 0) return "";
        StringBuilder stringBuilder = new StringBuilder();
        for (int i = 0; i < arr.length-1; i++){
            stringBuilder.append(arr[i] + ", ");
        }
        stringBuilder.append(arr[arr.length-1]);
        return stringBuilder.toString();
    }
    public boolean isNumber(String str){
        boolean flag = false;
        if (str.equals("-0")) return false;
        for (int i = 0; i < str.length(); i++){
            if (i == 0){
                if (str.charAt(i) == '-')
                    continue;
            }
            if (str.charAt(i) >= '0' && str.charAt(i) <= '9'){
                flag = true;
                continue;
            }else{
                return false;
            }
        }
        return flag;
    }
    static void copyToAL(List<Integer> list, int[] arr, int len){
        list.clear();
        for (int i = 0; i < len; i++){
            list.add(arr[i]);
        }
    }

    public int calcWidth(int pos, int count){
        pos++;
        int oldPos = pos;
        int i = 0;
        for (; pos>1; i++){
            pos /=2;
        }
        int thisWidth = WIDTH;

        int oldCount = count;
        int j = 0;
        for (; count > 1; j++){
            count/=2;
        }
        if (j > 4){
            thisWidth *= Math.pow(2, j-4);
        }
//        System.out.println(thisWidth);
```

```
//        System.out.println(j);
        return (int)
(thisWidth/2/Math.pow(2,i)+thisWidth/Math.pow(2,i)*(oldPos%Math.pow(2,i))
- 20);
    }

    public int calcHeight(int pos){
        pos++;
        int i = 0;
        for (; pos>1; i++){
            pos /=2;
        }
        return 10 + 50*i;
    }

    @FXML
    private Label labelForUser;
    @FXML
    private Label labelArray;
    @FXML
    private Label labelFromFile;
    @FXML
    private AnchorPane graph;
    @FXML
    private TextField textField;
    @FXML
    public void onButtonStepClick() {
        if (type == Type.NOT_SELECTED) return;
        if (arr.size() == 0){
            graph.getChildren().clear();
            createElements(arr, arr.size());
            if (resultArray.size() != 0){
                labelForUser.setText("Массив построен");
                labelForUser.setStyle("-fx-text-fill: black");
            }
            return;
        }
        if (isSorting){
            if (changeElements || arr.size() == 1){
                resultArray.add(heap.heap_sort_start());
                copyToAL(arr, heap.getHeap(), heap.getLast());
                graph.getChildren().clear(); // Удаление всех элементов
                createElements(arr, arr.size()); // Создание новых
                labelForUser.setText("Извлечен максимальный элемент.
Сделайте шаг");  // Смена текста
                labelForUser.setStyle("-fx-text-fill: black");
// Смена цвета текста
                labelArray.setText(makeReverseArray(resultArray));
                changeElements = false;
            }else{
                if (isFirst){
                    next = 0;
                    isFirst = false;
                }
                next = heap.sift_down_step(next);
                if (next == -1){
                    isFirst = true;
```

```
                changeElements = true;
                graph.getChildren().clear(); // Удаление всех
элементов
                createElements(arr, arr.size()); // Создание новых
                labelForUser.setStyle("-fx-text-fill: black");
                if (heap.getState() == Heap.States.COMPARE){

graph.getChildren().get(heap.getIndex1()).setStyle("-fx-text-fill: red");
                    ((Label)
graph.getChildren().get(heap.getIndex1())).setBorder(getBorder(Color.RED)
);
                    if (heap.getIndex2() < heap.getLast()){

graph.getChildren().get(heap.getIndex2()).setStyle("-fx-text-fill: red");
                        ((Label)
graph.getChildren().get(heap.getIndex2())).setBorder(getBorder(Color.RED)
);
                        labelForUser.setText("Сравниваются 2
элемента:" +heap.getHeap()[heap.getIndex1()] + ", "
+heap.getHeap()[heap.getIndex2()] + " .Элементы не поменялись, т.к. "  +
heap.getHeap()[heap.getIndex2()] +
                                " не больше " +
heap.getHeap()[heap.getIndex1()] + ". Сделайте шаг");
                    }
                    if (heap.getIndex3() < heap.getLast()){

graph.getChildren().get(heap.getIndex3()).setStyle("-fx-text-fill: red");
                        ((Label)
graph.getChildren().get(heap.getIndex3())).setBorder(getBorder(Color.RED)
);
                    }
                }else{
                    labelForUser.setText("Просеивание вниз завершено.
Сделайте шаг.");
                }
                return;
            }
            copyToAL(arr, heap.getHeap(), heap.getLast());
            graph.getChildren().clear(); // Удаление всех элементов
            createElements(arr, arr.size()); // Создание новых
            labelForUser.setStyle("-fx-text-fill: black");
            if (heap.getState() == Heap.States.CHANGE){
                graph.getChildren().get(heap.getIndex1()).setStyle("-
fx-text-fill: blue");
                ((Label)
graph.getChildren().get(heap.getIndex1())).setBorder(getBorder(Color.BLUE
));
                if (heap.getIndex2() < heap.getLast()){

graph.getChildren().get(heap.getIndex2()).setStyle("-fx-text-fill:
blue");
                    ((Label)
graph.getChildren().get(heap.getIndex2())).setBorder(getBorder(Color.BLUE
));
                    labelForUser.setText("Элементы "  +
heap.getHeap()[heap.getIndex1()] + " и " +
heap.getHeap()[heap.getIndex2()] + " поменялись местами, т.к. " +
```

```
heap.getHeap()[heap.getIndex1()] +
                        " больше " +
heap.getHeap()[heap.getIndex2()] + ". Сделайте шаг");
                }
                if (heap.getIndex3() < heap.getLast()) {

graph.getChildren().get(heap.getIndex3()).setStyle("-fx-text-fill: red");
                    ((Label)
graph.getChildren().get(heap.getIndex3())).setBorder(getBorder(Color.RED)
);
                    labelForUser.setText("Элемент "
+heap.getHeap()[heap.getIndex1()] + " сравнивается с элементами "
+heap.getHeap()[heap.getIndex2()]  + ", " +
heap.getHeap()[heap.getIndex3()]+ ". Элементы "  +
heap.getHeap()[heap.getIndex1()] + " и " +
heap.getHeap()[heap.getIndex2()] + " поменялись местами, т.к. " +
heap.getHeap()[heap.getIndex1()] +
                        " не меньше " +
heap.getHeap()[heap.getIndex3()] + " и " +
heap.getHeap()[heap.getIndex1()] +
                        " больше " +
heap.getHeap()[heap.getIndex2()] + ". Сделайте шаг");
                }
            }
        }
    }else{
        if (!canAdd){
            if (isFirst){
                next = heap.getLast()-1;
                isFirst = false;
            }
            next = heap.sift_up_step(next);
            if (next == -1){
                canAdd = true;
                isFirst = true;
                graph.getChildren().clear(); // Удаление всех
элементов
                createElements(arr, arr.size()); // Создание новых
                labelForUser.setStyle("-fx-text-fill: black");
// Смена цвета текста
                if (heap.getState() == Heap.States.DEFAULT){
                    labelForUser.setText("Просеивание вверх окончено.
Добавьте новый элемент");
                }else{
                    if (heap.getState() == Heap.States.COMPARE){

graph.getChildren().get(heap.getIndex1()).setStyle("-fx-text-fill: red");
                        ((Label)
graph.getChildren().get(heap.getIndex1())).setBorder(getBorder(Color.RED)
);

graph.getChildren().get(heap.getIndex2()).setStyle("-fx-text-fill: red");
                        ((Label)
graph.getChildren().get(heap.getIndex2())).setBorder(getBorder(Color.RED)
);
                    }
                    if (type == Type.FILE && current >=
```

```
arrayFile.length){
                    labelForUser.setText("Элементы не поменялись,
т.к. " + heap.getHeap()[heap.getIndex2()] +
                            " не больше " +
heap.getHeap()[heap.getIndex1()] + ". Просеивание вверх окончено. Файл
считан. Добавьте новый элемент вручную или извлеките массив");
                    type = Type.UI;
                }else{
                    labelForUser.setText("Элементы не поменялись,
т.к. " + heap.getHeap()[heap.getIndex2()] +
                            " не больше " +
heap.getHeap()[heap.getIndex1()] + ". Просеивание вверх окончено.
Добавьте новый элемент");
                }
            }
            return;
        }
        copyToAL(arr, heap.getHeap(), heap.getLast());
        graph.getChildren().clear(); // Удаление всех элементов
        createElements(arr, arr.size()); // Создание новых
        if (heap.getState() == Heap.States.CHANGE){
            graph.getChildren().get(heap.getIndex1()).setStyle("-
fx-text-fill: blue");
            ((Label)
graph.getChildren().get(heap.getIndex1())).setBorder(getBorder(Color.BLUE
));
            graph.getChildren().get(heap.getIndex2()).setStyle("-
fx-text-fill: blue");
            ((Label)
graph.getChildren().get(heap.getIndex2())).setBorder(getBorder(Color.BLUE
));
            labelForUser.setStyle("-fx-text-fill: black");
// Смена цвета текста
            labelForUser.setText("Элементы " +
heap.getHeap()[heap.getIndex1()] + " и " +
heap.getHeap()[heap.getIndex2()] + " поменялись местами, т.к. " +
heap.getHeap()[heap.getIndex1()] +
                    " больше " + heap.getHeap()[heap.getIndex2()]
+  ". Сделайте шаг");
        }
      }
    }
  }
  @FXML
  public void onButtonAddClick() {
    if (type == Type.NOT_SELECTED) return;
    if (isSorting) return;
    if (canAdd){
      try{
        int value;
        if (type == Type.UI){
          value = Integer.parseInt(textField.getText());
        }else{
          if (current < arrayFile.length){
            value = arrayFile[current];
            current++;
          }else{
```

```
                    labelForUser.setText("Весь файл считан");
                    labelForUser.setStyle("-fx-text-fill: red");
                    return;
                }
            }
            if (value >= -2147483647 && value <=2147483647){
                copyForHistory(arr);
                arr.add(value);
                heap.add_no_sift(value);
                graph.getChildren().clear();
                createElements(arr, arr.size());
                labelForUser.setText("Элемент добавлен. Сделайте
шаг");
                labelForUser.setStyle("-fx-text-fill: black");
                canAdd = false;
            }else{
                if (textField.getText().equals("")){
                    labelForUser.setStyle("-fx-text-fill: red");
                    labelForUser.setText("Введена пустая строка");
                }else{
                    labelForUser.setStyle("-fx-text-fill: red");
                    labelForUser.setText("Число по модулю должно быть
не больше 2'147'483'647");
                }
            }
        }catch(Exception exception){
            if (isNumber(textField.getText())) {
                labelForUser.setStyle("-fx-text-fill: red");
                labelForUser.setText("Число по модулю должно быть не
больше 2'147'483'647");
            } else {
                labelForUser.setStyle("-fx-text-fill: red");
                labelForUser.setText("Введено не число (или не
целое)");
            }
        }
    }
    public void createElements(List<Integer> list, int size){
        for (int i = 0; i < size; i++) {
            graph.getChildren().add(createElement(list.get(i), i, size));
        }
    }

    public Label createElement(int element, int number, int count){
        Label label = new Label();
        label.setText(String.valueOf(element));
        label.setLayoutX(calcWidth(number, count));
        label.setLayoutY(calcHeight(number));
        label.setPrefWidth(73);
        label.setBorder(getBorder(Color.BLACK));
        label.setAlignment(Pos.CENTER);
        return label;
    }
    public Border getBorder(Color color){
        return new Border(new BorderStroke(color,
BorderStrokeStyle.SOLID, new CornerRadii(2), new BorderWidths(2)));
```

```
    }

    public void onDeleteLastClick() {
        if (type == Type.NOT_SELECTED) return;

        if (canAdd && !isSorting){
            if (arr.size() > 0){
                if (type == Type.FILE){
                    current--;
                }
                arr = historyOfArrays.get(historyOfArrays.size()-1);
                historyOfArrays.remove(historyOfArrays.size()-1);
                heap = new Heap();
                for (int i: arr){
                    heap.add(i);
                }
                graph.getChildren().clear(); // Удаление всех элементов
                createElements(arr, arr.size()); // Создание новых
            }
        }
    }

    public void onMakeArrayClick() {
        if (type == Type.NOT_SELECTED) return;
        if (canAdd && arr.size() > 0){
            if (!isSorting){
                isSorting = true;
                labelForUser.setStyle("-fx-text-fill: black");
                labelForUser.setText("Начинается построение массива.
Сделайте шаг");
                graph.getChildren().clear(); // Удаление всех элементов
                createElements(arr, arr.size()); // Создание новых
            }
        }
    }

    public void onRestartClick() {
        if (type == Type.NOT_SELECTED) return;
        graph.getChildren().clear(); // Удаление всех элементов
        heap = new Heap();
        canAdd = true;
        next = -1;
        arr.clear();
        isFirst = true;
        isSorting = false;
        changeElements = true;
        resultArray = new ArrayList<>();
        labelForUser.setStyle("-fx-text-fill: black");
        labelForUser.setText("Выберите тип ввода");
        labelArray.setText("");
        labelFromFile.setText("");
        type = Type.NOT_SELECTED;
    }

    public void onFileReadClick() {
        if (type == Type.NOT_SELECTED){
            fileChooser = new FileChooser();
```

```
            fileChooser.getExtensionFilters().addAll( new
FileChooser.ExtensionFilter("Text files", "*.txt"));
            List<File> selectedFiles =
fileChooser.showOpenMultipleDialog(null);
            try{
                path = selectedFiles.get(0).getAbsolutePath();
//
System.out.println(selectedFiles.get(0).getAbsolutePath());
                ReadFile readFile = new ReadFile();
                readFile.setFileName(path);
                arrayFile = readFile.makeArray();
                current = 0;
                type = Type.FILE;
                labelForUser.setStyle("-fx-text-fill: black");
                labelForUser.setText("Добавьте новый элемент");
                labelFromFile.setText("Файл: " +
arrayToString(arrayFile));
                canAdd = true;
            }catch (IOException|NumberFormatException exception){
                labelForUser.setStyle("-fx-text-fill: red");
                labelForUser.setText("Ошибка чтения файла");
            }catch(Exception exception){
                labelForUser.setStyle("-fx-text-fill: red");
                labelForUser.setText("Файл не выбран");
            }
        }
    }

    public void onInteractiveClick() {
        if (type == Type.NOT_SELECTED){
            type = Type.UI;
            labelForUser.setStyle("-fx-text-fill: black");
            labelForUser.setText("Добавьте новый элемент");
            canAdd = true;
        }
    }

}
```

HelloApplication.java

```
package com.example.demo1;


import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new
FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 1200, 600);
        stage.setTitle("Макс. куча (heapsort)");
        stage.setScene(scene);
        stage.setMinWidth(1100);
        stage.setMinHeight(500);
        stage.show();
    }



    public static void main(String[] args) {
        launch();
    }
}
```

ПРИЛОЖЕНИЕ F

ТЕСТИРОВАНИЕ ГРФИЧЕСКОГО ИНТЕРФЕЙСА

TestMain.java

```
package com.example.demo1;


import java.util.ArrayList;
import java.util.List;

import static org.junit.Assert.*;

public class TestMain {

    @org.junit.Test
    public void test(){
        List<Integer> lst = new ArrayList<>();
        lst.add(5);
        lst.add(7);
        lst.add(3);

        HelloController helloController = new HelloController();
        // calcWidth
        assertEquals(helloController.calcWidth(0,20), 580);
        assertEquals(helloController.calcWidth(1,20), 280);
        assertEquals(helloController.calcWidth(5,20), 730);
        assertEquals(helloController.calcWidth(8,34), 430);
        assertEquals(helloController.calcWidth(14,34), 2230);

        //calcHeight
        assertEquals(helloController.calcHeight(0), 10);
        assertEquals(helloController.calcHeight(1), 60);
        assertEquals(helloController.calcHeight(5), 110);
        assertEquals(helloController.calcHeight(23), 210);

        //isNumber
        assertEquals(helloController.makeReverseArray(lst),"3, 7, 5");
        lst.add(97);
        assertEquals(helloController.makeReverseArray(lst), "97, 3, 7,
5");
        lst.add(2);
        assertEquals(helloController.makeReverseArray(lst), "2, 97, 3, 7,
5");
        assertEquals(helloController.makeReverseArray(new
ArrayList<Integer>()), "");

        //arrayToString
        int[] arr1 = {4, 7, 2, 87, 23, 76, -72, 0};
        int[] arr2 = {5,7,2,87,23,76,-72,0};
        int[] arr3 = {};
        assertEquals(helloController.arrayToString(arr1), "4, 7, 2, 87,
23, 76, -72, 0");
        assertEquals(helloController.arrayToString(arr2), "5, 7, 2, 87,
23, 76, -72, 0");
        assertEquals(helloController.arrayToString(arr3), "");
```

```
        //isNumber
        assertTrue(helloController.isNumber("23455"));
        assertTrue(helloController.isNumber("-34565754654390"));
        assertTrue(helloController.isNumber("234553453456910"));
        assertFalse(helloController.isNumber("2-345534534569"));
        assertFalse(helloController.isNumber("4350ы"));
    }
}
```