



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения

**ОТЧЁТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №6
по дисциплине
«Программирование на языке Java»**

Тема: Техники сортировки в Java

Выполнил студент группы ИКБО-16-20

Пак С.А.

Принял ассистент кафедры ИиППО

Русляков А.А.

Практические работы выполнены «_____» 2021г.

«Зачтено» «_____» 2021г

Москва 2021

СОДЕРЖАНИЕ

ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ.....	3
1. Алгоритм сортировки вставками.....	3
2. Алгоритм быстрой сортировки (Quick Sort).....	3
3. Алгоритм сортировка слиянием (Merge Sort).....	3
РЕШЕНИЕ ЗАДАЧИ.....	5
1. Постановка задачи.....	5
2. Программный код.....	5
3. Вывод программы.....	12
ВЫВОД.....	15

ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

Сортировка — это процесс упорядочивания списка элементов (организация в определенном порядке) исходного списка элементов, который возможно организован в виде контейнера или храниться в виде коллекции. Процесс сортировки основан на упорядочивании конкретных значений, например:

- сортировка списка результатов экзаменов баллов в порядке возрастания результата;
- сортировка списка людей в алфавитном порядке по фамилии.

Есть много алгоритмов для сортировки списка элементов, которые различаются по эффективности.

1. Алгоритм сортировки вставками

Работа метода сортировки состоит из следующих шагов:

- выбрать любой элемент из списка элементов и вставить его в надлежащее место в отсортированный подсписок;
- повторять предыдущий шаг, до тех пор, пока все элементы не будут вставлены.

Более детально:

- рассматриваем первый элемент списка как отсортированный подсписок (то есть первый элемент списка);
- вставим второй элемент в отсортированный подсписок, сдвигая первый элемент по мере необходимости, чтобы освободить место для вставки нового элемента;
- вставим третий элемент в отсортированный подсписок (из двух элементов), сдвигая элементы по мере необходимости;
- повторяем до тех пор, пока все значения не будут вставлены на свои соответствующие позиции.

2. Алгоритм быстрой сортировки (Quick Sort)

Состоит из последовательного выполнения двух шагов:

- массив $A[1..n]$ разбивается на два непустых подмассивов по отношению к "опорному элементу";
- два подмассива сортируются рекурсивно посредством Quick Sort.

3. Алгоритм сортировка слиянием (Merge Sort)

- Состоит из последовательного выполнения трех шагов:
- разделить массив $A[1..n]$ на 2 равные части;
- провести сортировку слиянием двух подмассивов (рекурсивно);

- объединить (соединить) два отсортированных подмассива

РЕШЕНИЕ ЗАДАЧИ

1. Постановка задачи

Задания:

1. Написать тестовый класс, который создает массив класса Student и сортирует массив iDNumber и сортирует его вставками;
2. Напишите класс SortingStudentsByGPA который реализует интерфейс Comparator таким образом, чтобы сортировать список студентов по их итоговым баллам в порядке убывания с использованием алгоритма быстрой сортировки.
3. Напишите программу, которая объединяет два списка данных о студентах в один отсортированный списках с использованием алгоритма сортировки слиянием.

2. Программный код

Файл MergeSort.java:

```
package ru.mirea;

import java.util.*;

public class App {
    private static final Scanner IN = new Scanner(System.in);

    private static String[] helper;

    public static void main(String[] args) {
        String[] list1 = new String[5];           // первый список имён
        студентов
        String[] list2 = new String[5];           // второй список имён
        студентов

        System.out.print("Введите 5 имён студентов: ");
        for (int i = 0; i < list1.length; ++i) {
            list1[i] = IN.next();
        }

        System.out.print("Введите ещё 5 имён студентов: ");
        for (int i = 0; i < list2.length; ++i) {
            list2[i] = IN.next();
        }

        String[] list = new String[list1.length + list2.length];

        System.arraycopy(list1, 0, list, 0, list1.length);
        System.arraycopy(list2, 0, list, list1.length, list2.length);
    }
}
```

```

    sort(list);

    System.out.println();
    System.out.println("Список студентов:");

    for (String stud : list) {
        System.out.println("    " + stud);
    }
}

/**
 * Сравнивает лексикографически 2 строки
 * @param val1        первая сравниваемая строка
 * @param val2        вторая сравниваемая строка
 */
private static boolean less(String val1, String val2) {
    return val1.compareTo(val2) < 0;
}

/**
 * Объединяет подмассивы array[low...mid], array[mid+1...high]
 * @param array        массив
 * @param low          наименьший индекс
 * @param mid          индекс среднего элемента
 * @param high         наибольший индекс
 */
private static void merge(String[] array, int low, int mid, int
high) {
    int i = low;
    int j = mid + 1;

    helper = array.clone();

    for (int k = low; k <= high; ++k) {
        if (i > mid) {
            array[k] = helper[j++];
        }
        else if (j > high) {
            array[k] = helper[i++];
        }
        else if (less(helper[j], helper[i])) {
            array[k] = helper[j++];
        }
        else {
            array[k] = helper[i++];
        }
    }
}

```

```

    }
}

/**
 * Проводит сортировку слиянием
 * @param array      сортируемый массив
 */
private static void sort(String[] array) {
    helper = new String[array.length];
    sort(array, 0, array.length - 1);
}

/**
 * Проводит сортировку слиянием
 * @param array      сортируемый массив
 * @param low        наименьший индекс массива
 * @param high       наибольший индекс массива
 */
private static void sort(String[] array, int low, int high) {
    if (high <= low) {
        return;
    }

    int mid = low + (high - low) / 2;

    sort(array, low, mid);
    sort(array, mid + 1, high);
    merge(array, low, mid, high);
}
}

```

Файл Student.java:

```

package ru.mirea.classes;

import java.util.Comparator;

public class Student implements Comparator<Student>,
Comparable<Student> {
    private int id;
    private double gpa;
    private String name;

    /**
     * Конструктор не по умолчанию
     * @param id      id студента
     * @param gpa     итоговый балл
     * @param name    имя студента
     */
}

```

```

    */
    public Student(int id, double gpa, String name) {
        this.id = id;
        this.gpa = gpa;
        this.name = name;
    }

    /**
     * Геттер для поля id
     * @return      id студента
     */
    public int getId() {
        return this.id;
    }

    /**
     * Геттер для поля gpa
     * @return      итоговый балл
     */
    public double getGpa() {
        return this.gpa;
    }

    /**
     * Геттер для поля name
     * @return      имя студента
     */
    public String getName() {
        return this.name;
    }

    /**
     * Возвращает число < 0, если id студента меньше id другого
студента
     *              число = 0, если id студента равно id другого
студента
     *              число > 0, если id студента больше id другого
студента
     * @param o1      первый студент
     * @param o2      другой студент
     */
    public int compare(Student o1, Student o2) {
        double res = o1.getGpa() - o2.getGpa();
        return Double.compare(res, 0.0);
    }
}

```



```

/**
 * Возвращает число < 0, если id студента меньше id другого
 студента
 *           число = 0, если id студента равно id другого
 студента
 *           число > 0, если id студента больше id другого
 студента
 * @param other          другой студент
 */
public int compareTo(Student other) {
    return this.id - other.getId();
}
}

```

Файл InsertSort.java:

```

package ru.mirea;

import ru.mirea.classes.*;

public class InsertSort {
    public static void main(String[] args) {
        Student[] studs = new Student[5];

        studs[0] = new Student(5, 4.5, "Student #5");
        studs[1] = new Student(2, 4.0, "Student #2");
        studs[2] = new Student(3, 3.0, "Student #3");
        studs[3] = new Student(1, 3.5, "Student #1");
        studs[4] = new Student(4, 3.6, "Student #4");

        System.out.println("Список студентов:");
        for (Student stud : studs) {
            System.out.println(" * " + stud.getName());
        }

        insertSort(studs);

        System.out.println("Список студентов:");
        for (Student stud : studs) {
            System.out.println(" * " + stud.getName());
        }
    }

/**
 * Реализует сортировку вставками
 * @param array          сортируемый массив
 */
private static void insertSort(Comparable[] array) {

```

```

        for (int i = 1; i < array.length; ++i) {
            for (int j = i; j > 0 && less(array[j], array[j - 1]); --j)
            {
                swap(array, j, j - 1);
            }
        }
    }

    /**
     * Определяет, меньше ли val1, чем val2
     * @param val1      значение 1
     * @param val2      значение 2
     */
    private static boolean less(Comparable val1, Comparable val2) {
        return val1.compareTo(val2) < 0;
    }

    /**
     * Меняет местами заданные элементы массива
     * @param array      массив
     * @param i           индекс одного из элементов
     * @param j           индекс другого элемента
     */
    private static void swap(Comparable[] array, int i, int j) {
        Comparable temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }
}

```

Файл QuickSort.java:

```

package ru.mirea;

import java.util.Comparator;
import ru.mirea.classes.*;

public class QuickSort {
    public static void main(String[] args) {
        Student[] studs = new Student[5];

        studs[0] = new Student(5, 4.5, "Student #5");
        studs[1] = new Student(2, 4.0, "Student #2");
        studs[2] = new Student(3, 3.0, "Student #3");
        studs[3] = new Student(1, 3.5, "Student #1");
        studs[4] = new Student(4, 3.6, "Student #4");

        System.out.println("Список студентов:");
        for (Student stud : studs) {

```

```

        System.out.println("    * " + stud.getName() + ": " +
stud.getGpa());
    }

    sort(studs);

    System.out.println("Список студентов:");
    for (Student stud : studs) {
        System.out.println("    * " + stud.getName() + ": " +
stud.getGpa());
    }
}

/**
 * Определяет, меньше ли val1, чем val2
 * @param val1      значение 1
 * @param val2      значение 2
 */
private static boolean less(Comparator val1, Comparator val2) {
    return val1.compare(val1, val2) < 0;
}

/**
 * Меняет местами заданные элементы массива
 * @param array      массив
 * @param i          индекс одного из элементов
 * @param j          индекс другого элемента
 */
private static void swap(Comparator[] array, int i, int j) {
    Comparator temp = array[i];
    array[i] = array[j];
    array[j] = temp;
}

/**
 * Выполняет разбиение массива
 */
private static int divide(Comparator[] array, int low, int high)
{
    int i = low;
    int j = high + 1;

    Comparator pivot = array[low];

    while (true) {
        while (less(array[++i], pivot)) {
            if (i == high)

```

```

        break;
    }

    while (less(pivot, array[--j])) {
        if (j == low)
            break;
    }

    if (i >= j)
        break;

    swap(array, i, j);
}

swap(array, low, j);
return j;
}

private static void sort(Comparator[] array) {
    sort(array, 0, array.length - 1);
}

private static void sort(Comparator[] array, int low, int high)
{
    if (high <= low)
        return;

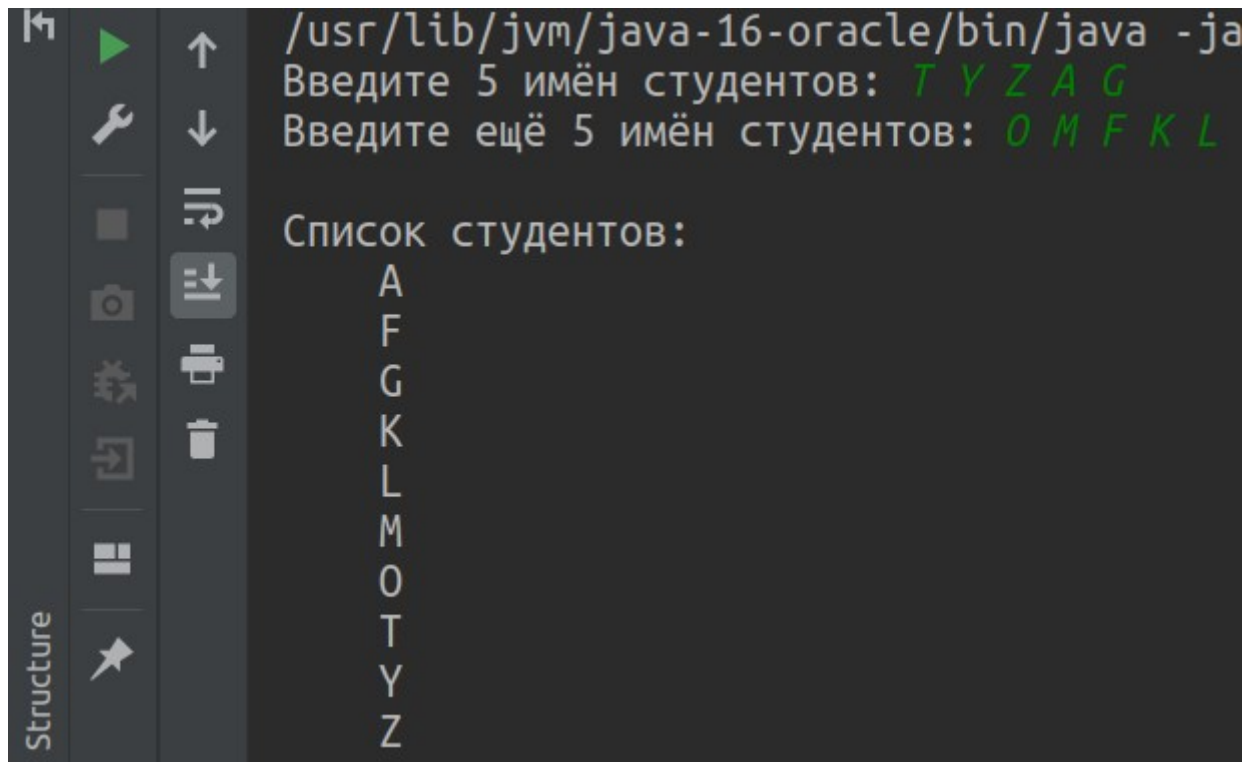
    int j = divide(array, low, high);

    sort(array, low, j - 1);
    sort(array, j + 1, high);
}
}

```

3. Вывод программы

На рис.1, 2, 3 показан результат сортировки слиянием, сортировки вставками и быстрой сортировки соответственно.

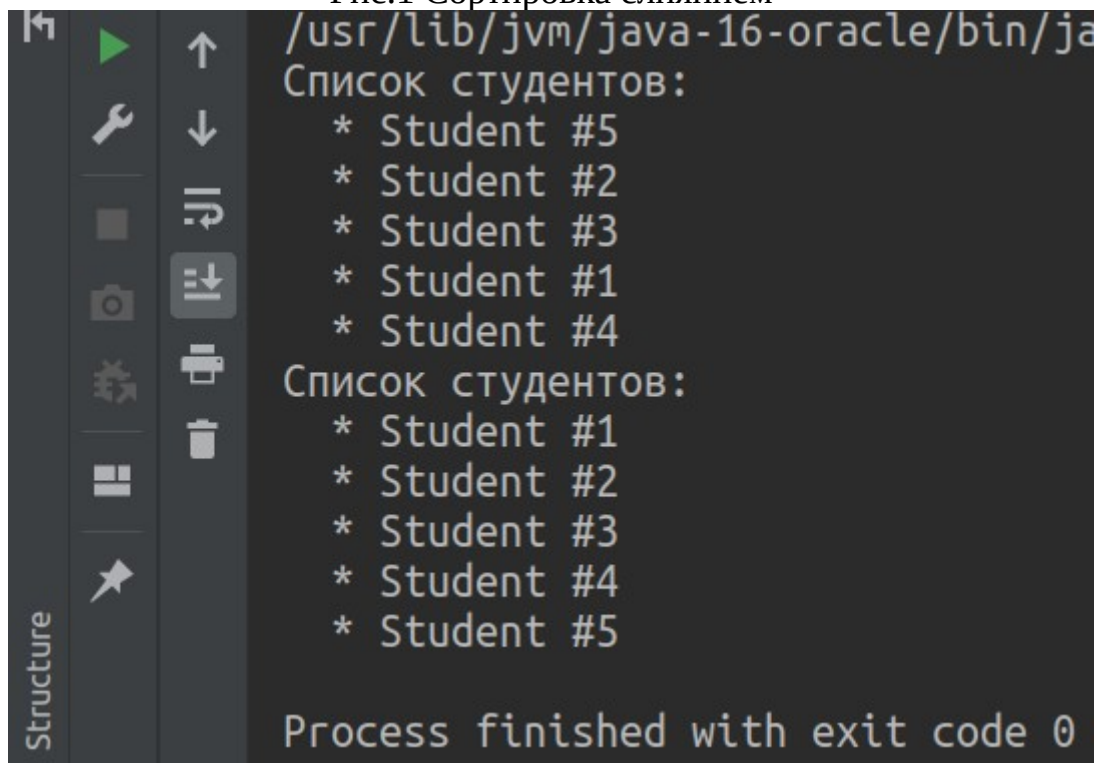


The screenshot shows a Java IDE with a dark theme. The left sidebar contains icons for running, debugging, and other IDE functions. The main window displays the output of a Java program. The first two lines are prompts for student names, followed by a list of names sorted alphabetically.

```
/usr/lib/jvm/java-16-oracle/bin/java -ja
Введите 5 имён студентов: T Y Z A G
Введите ещё 5 имён студентов: O M F K L

Список студентов:
A
F
G
K
L
M
O
T
Y
Z
```

Рис.1 Сортировка слиянием

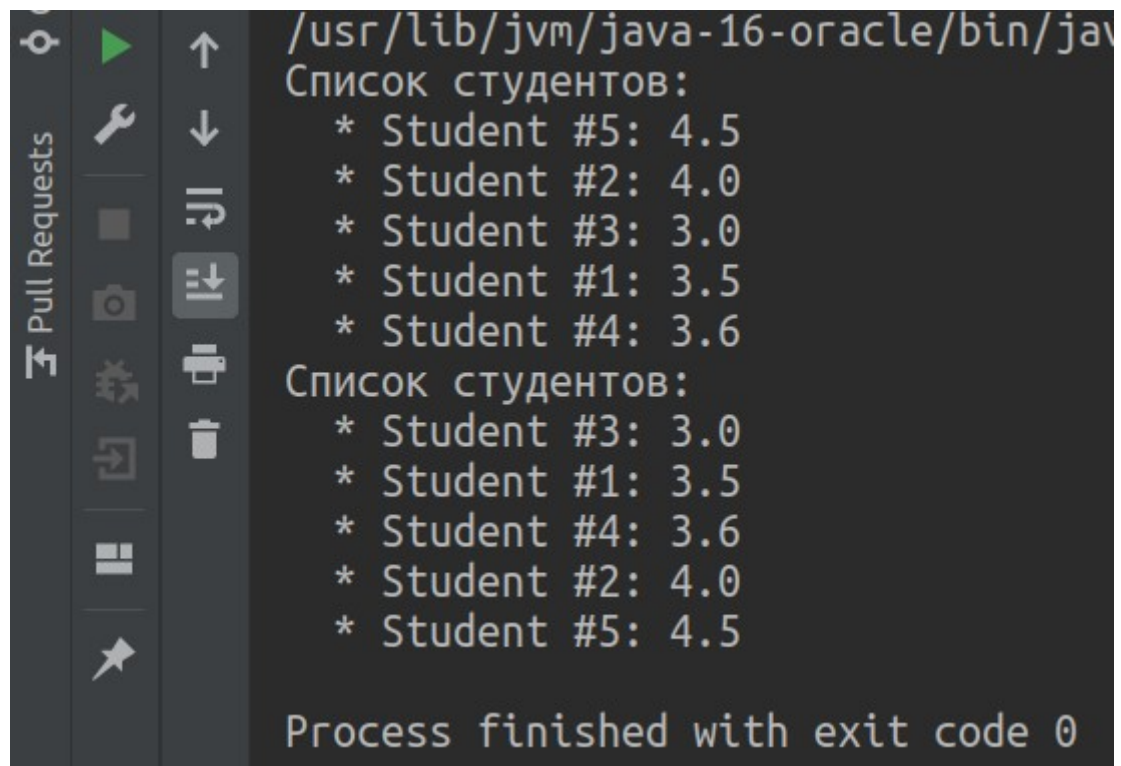


The screenshot shows a Java IDE with a dark theme. The main window displays the output of a Java program. It shows two lists of student IDs, followed by a message indicating the process finished successfully.

```
/usr/lib/jvm/java-16-oracle/bin/ja
Список студентов:
* Student #5
* Student #2
* Student #3
* Student #1
* Student #4
Список студентов:
* Student #1
* Student #2
* Student #3
* Student #4
* Student #5

Process finished with exit code 0
```

Рис.2 Сортировка вставками



```
/usr/lib/jvm/java-16-oracle/bin/jav
Список студентов:
* Student #5: 4.5
* Student #2: 4.0
* Student #3: 3.0
* Student #1: 3.5
* Student #4: 3.6
Список студентов:
* Student #3: 3.0
* Student #1: 3.5
* Student #4: 3.6
* Student #2: 4.0
* Student #5: 4.5

Process finished with exit code 0
```

Рис.3 Сортировка вставками

ВЫВОД

В ходе выполнения работы освоил на практике методы сортировки с использованием приемов программирования на объектно-ориентированном языке Java.