Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Омский государственный технический университет»

Кафедра «Информатика и вычислительная техника»

Отчёт по лабораторной работе № 4
по дисциплине
«Проектирование и тестирование программного обеспечения»

Выполнил:
Студент гр. ПИН-211
Сероухов Е.С. _____
(подп., дата)

Проверил:
Старший преподаватель каф. ИВТ
Карабцов Р.Д. _____
(подп., дата)

Омск, 2023

ВВЕДЕНИЕ

Цель работы: научиться применять алгоритмы сортировки данных. Решить поставленную задачу.

Задача работы: Задача 4.3 (Задача сапожника)

PC/UVaIDs: 110405/10026

У сапожника имеется N заказов от покупателей, которые он должен выполнить. Сапожник может заниматься в день только одним заказом, и заказы обычно требуют на выполнение несколько дней. Для i-го заказа целое число Ti ($1 \leq T_i \leq 1000$) означает число дней, необходимых сапожнику для завершения заказа.

Но за популярность нужно платить. За каждый день задержки перед тем, как он приступит к работе над i-м заказом, сапожник согласился платить штраф в размере S_i ($1 \leq S_i \leq 1000$) центов в день. Помогите сапожнику, написав программу, находящую последовательность работ, ведущую к минимальному штрафу.

Входные данные

Входные данные начинаются со строки, содержащей одно положительное целое число, которое означает количество тестовых блоков, за которой следует пустая строка. Между двумя последовательными тестовыми блоками также находится пустая строка.

Первая строка каждого блока содержит целое число, задающее число заказов N, причем $1 \leq N \leq 1000$. i-я последующая строка содержит время завершения T_i и ежедневный штраф S_i для i-го заказа.

Выходные данные

Для каждого тестового блока ваша программа должна вывести последовательность заказов, ведущую к минимальному штрафу. Каждый заказ представляется своей позицией во входных данных. Все целые числа должны находиться на одной строке выходных данных, и каждая пара чисел должна быть разделена одним пробелом. Если возможны несколько решений, выведите первое в лексикографическом порядке.

Выходные данные для двух последовательных блоков должны быть разделены пустой строкой.

Пример входных данных

1
4
3 4
1 1000
2 2
5 5

Соответствующие выходные данные

2 1 3 4

РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Для решения задачи был написан код на ЯП Java в среде IntelliJ IDEA 2022.1.2.

Входные данные загружаются из файла “Orders.txt” (рисунок 1).

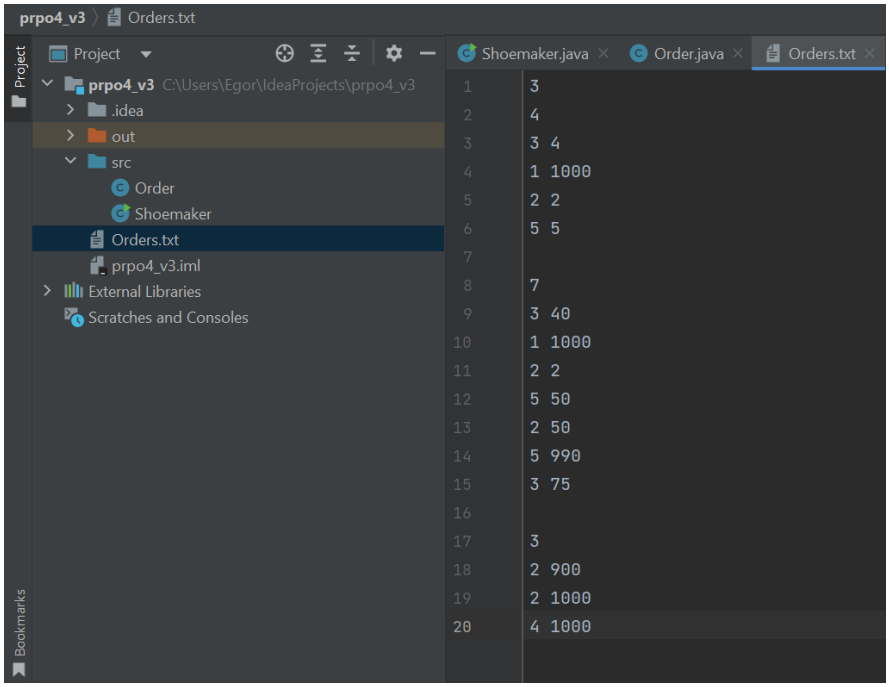

Рисунок 1 – Входные данные

Код программы:

```
import java.io.FileReader;
import java.io.IOException;
import java.util.*;

public class Shoemaker {
    final static List<String> Data = new ArrayList<>();
    public static void dataUploading(String path){
        try (FileReader fr= new FileReader(path)){
            Scanner scan = new Scanner(fr);
            while (scan.hasNextLine())
                Data.add(scan.nextLine());
        }
        catch(IOException ex){
            System.out.println(ex.getMessage());
        }
    }   //  загрузка данных

    public static void main(String[] args) {
        String file = "Orders.txt";
        dataUploading(file);
        System.out.println("Входные данные загружены из файла: " + file);
        System.out.print("Кол-во тестов: ");
        int numTests = Integer.parseInt(Data.get(0));
System.out.println(numTests);
        int numStr = 1;

        for (int t = 0; t < numTests; t++) {
```

```
            System.out.printf ("_________________Тест
№%d_________________\n",t+1);
            System.out.print("Кол-во заказов: ");
            int numOrders = Integer.parseInt(Data.get(numStr));
System.out.println(numOrders);
            numStr++;

            System.out.println("ДНИ   |   ШТРАФ");
            Order[] orders = new Order[numOrders];
            for (int j = 0; j < numOrders; j++) {
                String[] time_penalty;
                time_penalty = Data.get(numStr).split(" ", 2);
                numStr++;

                int time = Integer.parseInt(time_penalty[0]);
System.out.printf("%2s",time);
                int penalty = Integer.parseInt(time_penalty[1]);
System.out.printf("    |   %3s\n",penalty);
                orders[j] = new Order(time, penalty, j);
            }
            numStr++;

            // сортируем заказы по убыванию прибыли за день
            Arrays.sort(orders);

            // определяем порядок выполнения заказов
            int[] result = new int[numOrders];
            int index = 0;
            for (Order order : orders) {
                result[index] = order.index;
                index++;
            }

            // выводим порядок выполнения заказов
            System.out.println();
            System.out.println("Последовательность выполнения
заказов,\nвидущая к минимальному штрафу:");
            for (int j = 0; j < numOrders; j++)
                System.out.print(" " + (result[j]+1));
            System.out.println();
            System.out.println("_____________________________________________");
        }
    }
}
```

В классе Order.java прописано как будут сортироваться заказы:

```
class Order implements Comparable<Order> {
    int time;
    int penalty;
    int index;
    public Order(int time, int penalty, int index) {
        this.time = time;
        this.penalty = penalty;
        this.index = index;
    }
    public int compareTo(Order other) {

        int profit1 = this.penalty/this.time;

        int profit2 = other.penalty/other.time;
```

```
        if (profit1 != profit2)
            return Integer.compare(profit2, profit1);
        else
            return Integer.compare(this.index, other.index);
    }
}
/*
    Сортировка по убыванию прибыли за день происходит в методе compareTo,
который реализует интерфейс Comparable.

    1.  Вычисляется прибыль за день для текущего заказа (this) и для другого
заказа (other).
            Прибыль за день вычисляется как разница между штрафом и временем
выполнения заказа.
    2.  Происходит сравнение прибыли за день для текущего и другого заказов.
            Сравнение:
            Если они не равны, то возвращается результат сравнения, который
определяет порядок сортировки (
            - значение > 0 - текущий заказ должен быть после другого,
            - значение < 0 - текущий заказ должен быть перед другим,
            - значение = 0 - порядок не имеет значения).
    3.  Если прибыль за день для текущего и другого заказов равны, то
происходит сравнение по индексу заказа.
            Это нужно для того, чтобы при одинаковой прибыли за день заказы
сортировались по возрастанию индекса
                (то есть в порядке, в котором они были заданы во входных
данных).
*/
```

2. Было проведено тестирование работы программы (рисунок 2).

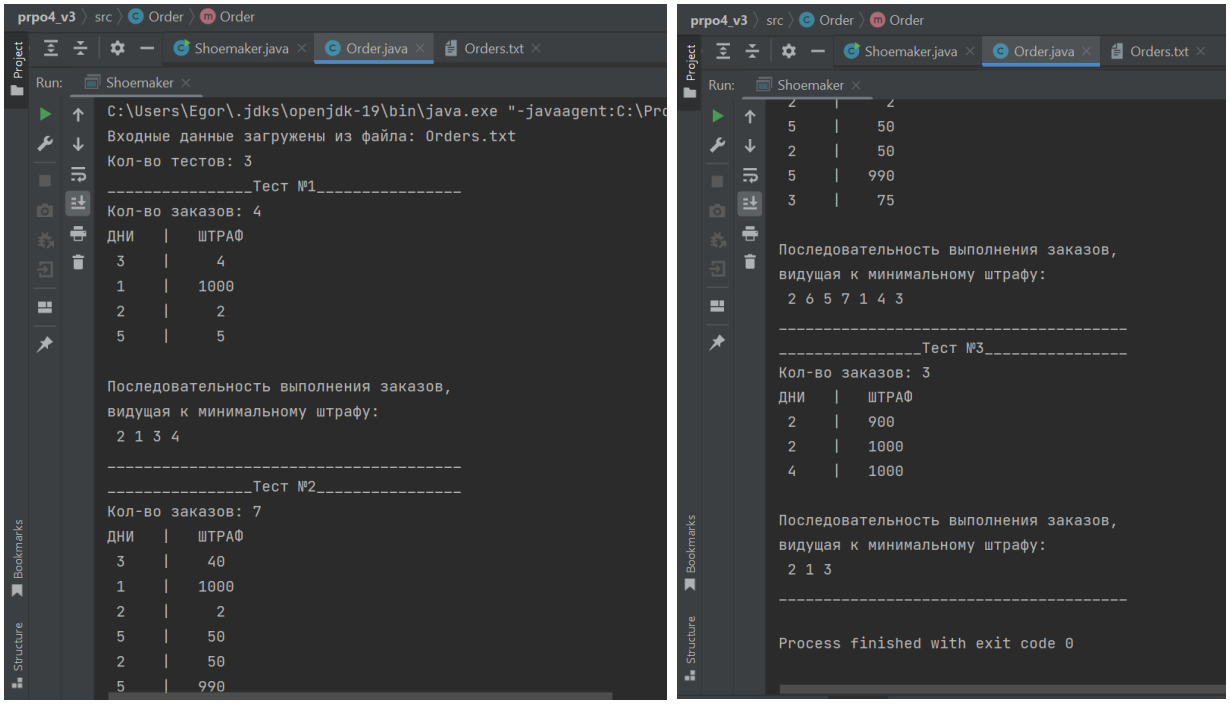

Рисунок 2 – Тестирование работы программы.

ЗАКЛЮЧЕНИЕ

Вывод: в ходе работы была написана программа на ЯП Java. Программа решила поставленную задачу и прошла проверку при тестировании.