Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

по лабораторной работе №6

Название: Коллекции

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-22М (Группа) _____ (Подпись, дата) В.А. Трофимов (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) П.В. Степанов (И.О. Фамилия)

Москва, 2023

Задания:

1.6. Не используя вспомогательных объектов, переставить отрицательные элементы данного списка в конец, а положительные – в начало этого списка.

1.7. Ввести строки из файла, записать в список ArrayList. Выполнить сортировку строк, используя метод sort() из класса Collections.

2.6. На плоскости задано N точек. Вывести в файл описания всех прямых, которые проходят более чем через одну точку из заданных. Для каждой прямой указать, через сколько точек она проходит. Использовать класс HashMap.

2.7. На плоскости задано N отрезков. Найти точку пересечения двх отрезков, имеющую минимальную абсциссу. Использовать класс TreeMap.

Код для решения задания 1.6:

```
package bdjava.lab6;

import java.util.ArrayList;
import static java.lang.System.out;

public class Part1 {
    public static void main(String[] args) {
            ArrayList<Integer> list = new ArrayList<>();
            list.add(3);
            list.add(-2);
            list.add(4);
            list.add(-1);
            list.add(-5);
            list.add(0);
            list.add(2);

            int size = list.size();
            int j = 0;
            for (int i = 0; i < size; i++) {
                int element = list.get(j);
                if (element < 0) {
                    list.remove(j);
                    list.add(element);
                } else {
                    j++;
                }
            }
            out.println(list);
    }
}
```

Код для решения задания 1.7:

```
package bdjava.lab6;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

import static java.lang.System.out;


public class Part2 {
    public static void main(String[] args) {
        ArrayList<String> strings = new ArrayList<>();
        try {
            Scanner reader = new Scanner(new File("src/bdjava.lab6/input.txt"));
            while (reader.hasNextLine()) {
                strings.add(reader.nextLine());
            }
            reader.close();
        } catch (FileNotFoundException e) {
            out.println(e.getMessage());
            return;
        }

        Collections.sort(strings);
        for (String str : strings) {
            out.println(str);
        }
    }
}
```

Код для решения задания 2.6:

```
package bdjava.lab6;
import java.io.File;
import java.io.FileWriter;
import java.util.*;

public class Part3 {
    public static void main(String[] args) throws Exception {
        int n = 5;
        Point[] points = new Point[n];
        points[0] = new Point(1, 3);
        points[1] = new Point(2, 2);
        points[2] = new Point(3, 1);
        points[3] = new Point(4, 4);
        points[4] = new Point(5, 5);

        Map<String, List<Point>> map = new HashMap<>();
        // перебираем все точки
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
```

```
                if(i != j) {
                    String key = getLineKey(points[i], points[j]); // ключ для
прямой
                    if (!map.containsKey(key)) {
                        map.put(key, new ArrayList<>());
                    }
                    map.get(key).add(points[i]);
                    map.get(key).add(points[j]);
                }
            }
        }


        File file = new File("src/bdjava/lab6/lines.txt");
        FileWriter writer = new FileWriter(file);
        for (Map.Entry<String, List<Point>> entry : map.entrySet()) {
            List<Point> pointsLine = entry.getValue();
            getUniquePoints(pointsLine);
            int count = pointsLine.size();
            writer.write(entry.getKey() + " проходит через " + count + " точек: "
+ pointsLine.toString() + "\n");
        }
        writer.close();
    }

    public static String getLineKey(Point p1, Point p2) {
        double k = (double) (p1.y - p2.y) / (p1.x - p2.x);
        double b = p1.y - k * p1.x;
        return String.format("y = %.2f x + %.2f", k, b);
    }

    public static void getUniquePoints(List<Point> points) {
        Set<Point> set = new HashSet<>();
        for (Point point : points) {
            set.add(point);
        }
        points.clear();
        points.addAll(set);
    }

    public static class Point {
        public int x;
        public int y;

        public Point(int x, int y) {
            this.x = x;
            this.y = y;
        }

        @Override
        public String toString() {
            return "(" + x + ", " + y + ")";
        }
    }
}
```

Код для решения задания 2.7:

```
package bdjava.lab6;

import java.util.*;

import static java.lang.System.out;

public class Part4 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        out.print("Введите количество отрезков: ");
        int n = scanner.nextInt();
        TreeMap<Integer, List<Segment>> map = new TreeMap<>();

        for (int i = 1; i <= n; i++) {
            out.print("Введите координаты начала и конца отрезка №" + i + ": ");
            int a = scanner.nextInt();
            int b = scanner.nextInt();
            int c = scanner.nextInt();
            int d = scanner.nextInt();
            Segment s = new Segment(a, b, c, d);
            int minX = Math.min(a, c);
            if (!map.containsKey(minX)) {
                map.put(minX, new ArrayList<>());
            }
            map.get(minX).add(s);
        }

        Point minPoint = null;
        List<Segment> segments = null;
        Map.Entry<Integer, List<Segment>> entry = map.firstEntry();
        while (entry != null) {
            List<Segment> currentSegments = entry.getValue();
            if (currentSegments.size() > 1) {
                for (int i = 0; i < currentSegments.size(); i++) {
                    for (int j = i + 1; j < currentSegments.size(); j++) {
                        Point p =
currentSegments.get(i).intersect(currentSegments.get(j));
                        if (minPoint == null || p.x < minPoint.x) {
                            minPoint = p;
                            segments = new ArrayList<>();
                            segments.add(currentSegments.get(i));
                            segments.add(currentSegments.get(j));
                        }
                    }
                }
            }
            entry = map.higherEntry(entry.getKey());
        }

        if (minPoint != null) {
            out.println("Точка пересечения двух отрезков с минимальной абсциссой:
" + minPoint);
            out.println("Отрезки, содержащие данную точку:");
            for (Segment s : segments) {
```

```
                out.println(s);
            }
        } else {
            out.println("Таких отрезков нет.");
        }
    }

    static class Point {
        float x, y;

        public Point(float x, float y) {
            this.x = x;
            this.y = y;
        }

        @Override
        public String toString() {
            return "(" + x + ", " + y + ")";
        }
    }

    static class Segment {
        int x1, y1, x2, y2;

        public Segment(int x1, int y1, int x2, int y2) {
            this.x1 = x1;
            this.y1 = y1;
            this.x2 = x2;
            this.y2 = y2;
        }
        public Point intersect(Segment other) {
            ArrayList<Integer> lx = new
ArrayList<Integer>(Arrays.asList(x1,x2,other.x1,other.x2));
            ArrayList<Integer> ly = new
ArrayList<Integer>(Arrays.asList(y1,y2,other.y1,other.y2));
            float k1 = (y2-y1)/(x2-x1);
            float k2 = (other.y2-other.y1)/(other.x2-other.x1);
            float x = (y1 - other.y1 + k2 * other.x1 - k1 * x1) / (k2 - k1);
            float y = k1 * (x - x1) + y1;
            if ((x >= Collections.min(lx) || x <= Collections.max(lx)) && (y >=
Collections.min(ly) || y <= Collections.max(ly)))
                return new Point(x, y);
            else
                return null;
        }
        @Override
        public String toString() {
            return "(" + x1 + ", " + y1 + ") - (" + x2 + ", " + y2 + ")";
        }
    }
}
```

Вывод:

В ходе выполнения заданий были реализованы различные алгоритмы для работы с коллекциями данных. В задании 1.6 была реализована функция перестановки отрицательных элементов списка в конец, а положительных в начало. Для этого использовались методы работы со списками, такие как add(), remove() и size(). В задании 1.7 была реализована функция чтения строк из файла и записи их в список ArrayList. Для сортировки строк использовался метод sort() из класса Collections. Это позволило отсортировать строки в порядке возрастания или убывания. В задании 2.6 была реализована функция вывода в файл описаний всех прямых, которые проходят более чем через одну точку из заданных. Для этого использовался класс HashMap, который позволил хранить информацию о прямых и количестве точек, через которые они проходят. В задании 2.7 была реализована функция поиска точки пересечения двух отрезков, имеющей минимальную абсциссу. Для этого использовался класс TreeMap, который позволил хранить информацию об отрезках и их координатах. В результате выполнения всех заданий были получены функциональные программы, способные обрабатывать различные типы данных и выполнять различные операции с ними. Были использованы различные методы работы со списками и коллекциями, что позволило решить поставленные задачи.