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 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

Вариант 14

Название: Коллекции

Дисциплина: Языки программирования для работы с большими данными

Студент	ИУ6-23М		В.А. Елисеев
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			П.В. Степанов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2022

Цель работы: получение навыков работы с коллекциями в Java.

Задание 1:

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

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

Подзадача 1.

Код программы:

```
public class L1_6 {
    public static void main(String[] args) throws Exception {
        Random random = new Random();

        ArrayList<Integer> list = new ArrayList<Integer>();
        for (int i = 0; i < 16; i++) list.add(random.nextInt(11) - 5);

        System.out.println("Исходный список: " + list);

        for (Integer num : new ArrayList<Integer>(list)) {
            if (num < 0) {
                list.remove(num);
                list.add(num);
            }
        }

        System.out.println("Список после сортировки: " + list);
    }
}
```

Результат работы программы:

```
Исходный список: [0, -4, 2, -4, -2, 3, -5, -3, 3, -3, 0, -3, 3, -2, -2, 4]
Список после сортировки: [0, 2, 3, 3, 0, 3, 4, -4, -4, -2, -5, -3, -3, -3, -2, -2]
```

Подзадача 2.

Код программы:

```java
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Collections;

public class L1_7 {
    Run | Debug
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<String>();

        System.out.println();
        try (BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(new File(System.getProperty(key: "user.dir") + "\\src\\file.txt")), charsetName: "U
            for (String line; (line = br.readLine()) != null;) {
                list.add(line);
            }
        } catch (Exception e) {
            System.out.println(x: "Файл куда-то делся :(");
            System.exit(status: 0);
        }

        System.out.println("Список до сортировки: " + list);
        Collections.sort(list);
        System.out.println("Список после сортировки: " + list);
    }
}
```

Результат работы программы:

src > file.txt

```
1  Ноль
2  Целковый
3  Полушка
4  Четвертушка
5  Осьмушка
6  Подувичок
7  Медячок
8  Серебрячок
9  Золотничок
10 Девятичок
11 Десятичок
```

```
Список до сортировки: [Ноль, Целковый, Полушка, Четвертушка, Осьмушка, Подувичок, Медячок, Серебрячок, Золотничок, Девятичок, Десятичок, ]
Список после сортировки: [, Девятичок, Десятичок, Золотничок, Медячок, Ноль, Осьмушка, Подувичок, Полушка, Серебрячок, Целковый, Четвертушка]
```

Задание 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.

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

Подзадача 1.

Код программы:

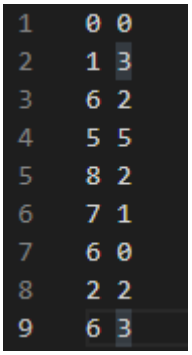
```java
public class Dot {
    double x;
    double y;

    Dot(double x, double y) {
        this.x = x;
        this.y = y;
    }

    @Override
    public boolean equals(Object obj) {
        Dot dot = (Dot) obj;

        return this.x == dot.x && this.y == dot.y;
    }

    @Override
    public int hashCode() {
        return Objects.hash(x, y);
    }

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

}
```

```java
public class Line {
    double k;
    double b;

    String axis;
    double axisValue;

    Line(double k, double b) {
        this.k = k;
        this.b = b;
    }

    Line(String axis, double value) {
        this.axis = axis;
        this.axisValue = value;
    }


    @Override
    public boolean equals(Object o) {
        if (o == this)
            return true;
        if (!(o instanceof Line)) {
            return false;
        }
        Line line = (Line) o;
        return k == line.k && b == line.b && Objects.equals(axis, line.axis) && axisValue == line.axisValue;
    }

    @Override
    public int hashCode() {
        return Objects.hash(k, b, axis, axisValue);
    }

    @Override
    public String toString() {
        return "{" +
            " k='" + this.k + "'" +
            ", b='" + this.b + "'" +
            ", axis='" + this.axis + "'" +
            ", axisValue='" + this.axis + "'" +
            "}";
    }

}
```

```java
static String parseLine(Line line) {
    return line.axis == null ? "y = " + line.k + "x + " + line.b : line.axis + " = " + line.axisValue;
}

Run | Debug
public static void main(String[] args) throws Exception {
    String filePath = System.getProperty(key: "user.dir") + "\\src\\dots.txt";

    ArrayList<Dot> dotsList = new ArrayList<Dot>();
    HashMap<Line, HashSet<Dot>> lines = new HashMap<>();

    try (BufferedReader br = new BufferedReader(new FileReader(new File(filePath)))) {
        for (String line; (line = br.readLine()) != null;) {
            String[] vals = line.split(regex: " ");

            Dot dot = new Dot(Integer.parseInt(vals[0]), Integer.parseInt(vals[1]));
            dotsList.add(dot);
        }
    } catch (Exception e) {
        System.out.println(x: "Файл куда-то делся :(");
        System.exit(status: 0);
    }

    for (Dot dot1 : dotsList.subList(fromIndex: 0, dotsList.size() - 1)) {
        for (Dot dot2 : dotsList.subList(dotsList.indexOf(dot1) + 1, dotsList.size())) {
            Line line;

            if (dot1.x == dot2.x) {
                line = new Line(axis: "x", dot1.x);
            }
            else if (dot1.y == dot2.y) {
                line = new Line(axis: "y", dot1.y);
            }
            else {
                double k = (dot2.y - dot1.y) / (dot2.x - dot1.x);
                double x = dot1.x * k * -1 + dot1.y;

                k = (double) Math.round(k * 100) / 100;
                x = (double) Math.round(x * 100) / 100;

                line = new Line(k, x);
            }
```

```java
            HashSet<Dot> dots = lines.get(line);
            if (dots == null) {
                lines.put(line, new HashSet<Dot>(Arrays.asList(dot1, dot2)));
            }
            else {
                dots.addAll(Arrays.asList(dot1, dot2));
            }
        }
    }

    lines.forEach((line, dots) -> {
        if (dots.size() > 2) {
            System.out.println("Прямая " + parseLine(line) + " пересекается с " + dots.size() + " точками");
        }
    });
}
```

Результат работы программы:




```
Прямая y = 1.0x + 0.0 пересекается с 3 точками
Прямая y = 2.0 пересекается с 3 точками
Прямая x = 6.0 пересекается с 3 точками
Прямая y = 1.0x + -6.0 пересекается с 3 точками
Прямая y = -2.0x + 15.0 пересекается с 3 точками
```

Подзадача 2.

Код программы:

```java
public class Segment {
    Dot dot1;
    Dot dot2;

    Line line;

    public Segment(Dot dot1, Dot dot2, Line line) {
        this.dot1 = dot1;
        this.dot2 = dot2;
        this.line = line;
    }

    Boolean contains(double x) {
        return x > dot1.x && x < dot2.x || x < dot1.x && x > dot2.x;
    }

    @Override
    public boolean equals(Object o) {
        if (o == this)
            return true;
        if (!(o instanceof Segment)) {
            return false;
        }
        Segment segment = (Segment) o;
        return Objects.equals(dot1, segment.dot1) && Objects.equals(dot2, segment.dot2) && Objects.equals(line, segment.line);
    }

    @Override
    public int hashCode() {
        return Objects.hash(dot1, dot2, line);
    }

    @Override
    public String toString() {
        return "{" +
            " dot1='" + this.dot1 + "'" +
            ", dot2='" + this.dot2 + "'" +
            ", line='" + this.line + "'" +
            "}";
    }

}
```

```java
public static void main(String[] args) {
    String filePath = System.getProperty(key: "user.dir") + "\\src\\segments.txt";

    ArrayList<Segment> segments = new ArrayList<Segment>();
    TreeMap<Double, Segment[]> map = new TreeMap<Double, Segment[]>();

    try (BufferedReader br = new BufferedReader(new FileReader(new File(filePath)))) {
        for (String lineStr; (lineStr = br.readLine()) != null;) {
            String[] dots = lineStr.split(regex: " - ");
            String[] dot1S = dots[0].split(regex: " ");
            String[] dot2S = dots[1].split(regex: " ");

            Dot dot1 = new Dot(Integer.parseInt(dot1S[0]), Integer.parseInt(dot1S[1]));
            Dot dot2 = new Dot(Integer.parseInt(dot2S[0]), Integer.parseInt(dot2S[1]));

            Line line;

            if (dot1.x == dot2.x) {
                line = new Line(axis: "x", dot1.x);
            }
            else if (dot1.y == dot2.y) {
                line = new Line(axis: "y", dot1.y);
            }
            else {
                double k = (dot2.y - dot1.y) / (dot2.x - dot1.x);
                double x = dot1.x * k * -1 + dot1.y;

                k = (double) Math.round(k * 100) / 100;
                x = (double) Math.round(x * 100) / 100;

                line = new Line(k, x);
            }

            segments.add(new Segment(dot1, dot2, line));
        }
    } catch (Exception e) {
        System.out.println(x: "Что-то пошло не так :(");
        System.exit(status: 0);
    }
```

```
    // Перебираем все пары отрезков
    for (Segment seg1 : segments.subList(fromIndex: 0, segments.size() - 1)) {
        for (Segment seg2 : segments.subList(segments.indexOf(seg1) + 1, segments.size())) {
            System.out.println(seg1.toString() + "\n" + seg2.toString());
            // Проверяем, что коэффициенты k не равны и прямые не параллельны одной и той же оси
            if (seg1.line.k != seg2.line.k || seg1.line.axis != seg2.line.axis) {
                Segment[] pair = {seg1, seg2};

                if (seg1.line.axis == null && seg2.line.axis == null) {         // Если прямые не параллельны OX и OY
                    double absc = (seg1.line.b - seg2.line.b) / (seg2.line.k - seg1.line.k);

                    Boolean inSeg1 = seg1.contains(absc);
                    Boolean inSeg2 = seg2.contains(absc);

                    if (inSeg1 && inSeg2) {
                        map.put(absc, pair);
                        System.out.println("Пересекаются на абсциссе " + absc);
                    }
                }
                else if (seg1.line.axis != null && seg2.line.axis == null) {    // Если прямая seg1 параллельна одной из осей
                    if (seg1.line.axis == "x") {
                        double absc = seg1.line.axisValue;

                        if (seg2.contains(absc)) {
                            map.put(absc, pair);
                            System.out.println("Пересекаются на абсциссе " + absc);
                        }
                    }
                    else if (seg1.line.axis == "y") {
                        double absc = (seg1.line.axisValue - seg2.line.b) / seg2.line.k;

                        if (seg2.contains(absc)) {
                            map.put(absc, pair);
                            System.out.println("Пересекаются на абсциссе " + absc);
                        }
                    }
                }
                else if (seg1.line.axis == null && seg2.line.axis != null) {    // Если прямая seg2 параллельна одной из осей
                    if (seg2.line.axis == "x") {
                        double absc = seg2.line.axisValue;

                        if (seg1.contains(absc)) {
                            map.put(absc, pair);
                            System.out.println("Пересекаются на абсциссе " + absc);
                        }
                    }
```

```
                    }
                    else if (seg2.line.axis == "y") {
                        double absc = (seg2.line.axisValue - seg1.line.b) / seg1.line.k;

                        if (seg1.contains(absc)) {
                            map.put(absc, pair);
                            System.out.println("Пересекаются на абсциссе " + absc);
                        }
                    }
                }
                else {  // Если прямые параллельны различным осям
                    if (seg1.line.axis == "x") {
                        map.put(seg1.line.axisValue, pair);
                        System.out.println("Пересекаются на абсциссе " + seg1.line.axisValue);
                    }
                    else if (seg2.line.axis == "x") {
                        map.put(seg2.line.axisValue, pair);
                        System.out.println("Пересекаются на абсциссе " + seg2.line.axisValue);
                    }
                }
            }
            else {
                System.out.println(x: "Параллельны друг другу");
            }

            System.out.println();
        }
    }

    System.out.println("Минимальная абсцисса: " + map.firstKey());
}
```

Результат работы программы:

src > segments.txt

```
0 0 - 10 10
5 5 - 10 5
0 4 - 10 4
2 3 - 6 4
5 4 - 8 1
1 8 - 8 1
```

```
{ dot1='{ x='0.0', y='0.0'}', dot2='{ x='10.0', y='10.0'}', line='{ k='1.0', b='0.0', axis='null', axisValue='null'}'}
{ dot1='{ x='5.0', y='5.0'}', dot2='{ x='10.0', y='5.0'}', line='{ k='0.0', b='0.0', axis='y', axisValue='y'}'}
Пересекаются на абсциссе 5.0

{ dot1='{ x='0.0', y='0.0'}', dot2='{ x='10.0', y='10.0'}', line='{ k='1.0', b='0.0', axis='null', axisValue='null'}'}
{ dot1='{ x='0.0', y='4.0'}', dot2='{ x='10.0', y='4.0'}', line='{ k='0.0', b='0.0', axis='y', axisValue='y'}'}
Пересекаются на абсциссе 4.0

{ dot1='{ x='0.0', y='0.0'}', dot2='{ x='10.0', y='10.0'}', line='{ k='1.0', b='0.0', axis='null', axisValue='null'}'}
{ dot1='{ x='2.0', y='3.0'}', dot2='{ x='6.0', y='4.0'}', line='{ k='0.25', b='2.5', axis='null', axisValue='null'}'}
Пересекаются на абсциссе 3.3333333333333335

{ dot1='{ x='0.0', y='0.0'}', dot2='{ x='10.0', y='10.0'}', line='{ k='1.0', b='0.0', axis='null', axisValue='null'}'}
{ dot1='{ x='5.0', y='4.0'}', dot2='{ x='8.0', y='1.0'}', line='{ k='-1.0', b='9.0', axis='null', axisValue='null'}'}

{ dot1='{ x='0.0', y='0.0'}', dot2='{ x='10.0', y='10.0'}', line='{ k='1.0', b='0.0', axis='null', axisValue='null'}'}
{ dot1='{ x='1.0', y='8.0'}', dot2='{ x='8.0', y='1.0'}', line='{ k='-1.0', b='9.0', axis='null', axisValue='null'}'}
Пересекаются на абсциссе 4.5

{ dot1='{ x='5.0', y='5.0'}', dot2='{ x='10.0', y='5.0'}', line='{ k='0.0', b='0.0', axis='y', axisValue='y'}'}
{ dot1='{ x='0.0', y='4.0'}', dot2='{ x='10.0', y='4.0'}', line='{ k='0.0', b='0.0', axis='y', axisValue='y'}'}
Параллельны друг другу

{ dot1='{ x='5.0', y='5.0'}', dot2='{ x='10.0', y='5.0'}', line='{ k='0.0', b='0.0', axis='y', axisValue='y'}'}
{ dot1='{ x='2.0', y='3.0'}', dot2='{ x='6.0', y='4.0'}', line='{ k='0.25', b='2.5', axis='null', axisValue='null'}'}

{ dot1='{ x='5.0', y='5.0'}', dot2='{ x='10.0', y='5.0'}', line='{ k='0.0', b='0.0', axis='y', axisValue='y'}'}
{ dot1='{ x='5.0', y='4.0'}', dot2='{ x='8.0', y='1.0'}', line='{ k='-1.0', b='9.0', axis='null', axisValue='null'}'}

{ dot1='{ x='5.0', y='5.0'}', dot2='{ x='10.0', y='5.0'}', line='{ k='0.0', b='0.0', axis='y', axisValue='y'}'}
{ dot1='{ x='1.0', y='8.0'}', dot2='{ x='8.0', y='1.0'}', line='{ k='-1.0', b='9.0', axis='null', axisValue='null'}'}
Пересекаются на абсциссе 4.0

{ dot1='{ x='0.0', y='4.0'}', dot2='{ x='10.0', y='4.0'}', line='{ k='0.0', b='0.0', axis='y', axisValue='y'}'}
{ dot1='{ x='2.0', y='3.0'}', dot2='{ x='6.0', y='4.0'}', line='{ k='0.25', b='2.5', axis='null', axisValue='null'}'}

{ dot1='{ x='0.0', y='4.0'}', dot2='{ x='10.0', y='4.0'}', line='{ k='0.0', b='0.0', axis='y', axisValue='y'}'}
{ dot1='{ x='5.0', y='4.0'}', dot2='{ x='8.0', y='1.0'}', line='{ k='-1.0', b='9.0', axis='null', axisValue='null'}'}

{ dot1='{ x='0.0', y='4.0'}', dot2='{ x='10.0', y='4.0'}', line='{ k='0.0', b='0.0', axis='y', axisValue='y'}'}
{ dot1='{ x='1.0', y='8.0'}', dot2='{ x='8.0', y='1.0'}', line='{ k='-1.0', b='9.0', axis='null', axisValue='null'}'}
Пересекаются на абсциссе 5.0

{ dot1='{ x='2.0', y='3.0'}', dot2='{ x='6.0', y='4.0'}', line='{ k='0.25', b='2.5', axis='null', axisValue='null'}'}
{ dot1='{ x='5.0', y='4.0'}', dot2='{ x='8.0', y='1.0'}', line='{ k='-1.0', b='9.0', axis='null', axisValue='null'}'}
Пересекаются на абсциссе 5.2

{ dot1='{ x='2.0', y='3.0'}', dot2='{ x='6.0', y='4.0'}', line='{ k='0.25', b='2.5', axis='null', axisValue='null'}'}
{ dot1='{ x='1.0', y='8.0'}', dot2='{ x='8.0', y='1.0'}', line='{ k='-1.0', b='9.0', axis='null', axisValue='null'}'}
Пересекаются на абсциссе 5.2

{ dot1='{ x='5.0', y='4.0'}', dot2='{ x='8.0', y='1.0'}', line='{ k='-1.0', b='9.0', axis='null', axisValue='null'}'}
{ dot1='{ x='1.0', y='8.0'}', dot2='{ x='8.0', y='1.0'}', line='{ k='-1.0', b='9.0', axis='null', axisValue='null'}'}
Параллельны друг другу
```

```
Параллельны друг другу

Минимальная абсцисса: 3.3333333333333335
```

Ссылка на программное решение:

https://github.com/ArMaxik/BigDataLanguages/tree/main/lr6

Вывод: в ходе лабораторной работы были получены навыки с коллекциями в Java.