



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 № 5

Название лабораторной работы: Коллекции

Дисциплина: Языки программирования для работы с большими данными

Студент гр. ИУ6-22М

(Подпись, дата)

Р.Г. Гаделия

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2023

Введение

Целью лабораторной работы является приобретение навыков работы с коллекциями на языке программирования Java.

Практическая часть

Задание 1

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

Код написанной программы представлен в листинге 1.

Листинг 1 – Программа для первого задания

```
package com.java.lab;

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

// 6. Не используя вспомогательных объектов, переставить отрицательные
элементы данного списка в конец, а положительные – в начало этого
списка.
public class Main {

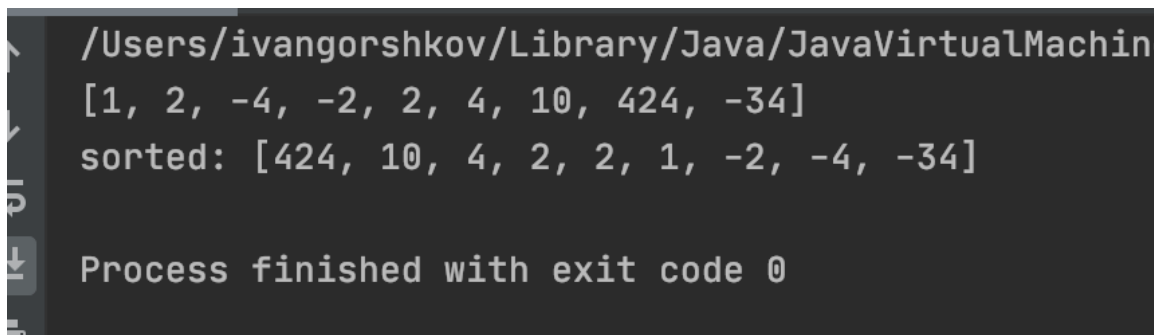
    public static void main(String[] args) {
        var list = new ArrayList<>(List.of(1, 2, -4, -2, 2, 4, 10, 424,
-34));

        System.out.println(list);

        list.sort(Collections.reverseOrder());

        System.out.println("sorted: " + list);
    }
}
```

Результат выполнения программы показан на рисунке 1.



```
/Users/ivangorshkov/Library/Java/JavaVirtualMachin
[1, 2, -4, -2, 2, 4, 10, 424, -34]
sorted: [424, 10, 4, 2, 2, 1, -2, -4, -34]

Process finished with exit code 0
```

Рисунок 1 – Результат выполнения программы

Задание 2

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

Код написанной программы представлен в листинге 2.

Листинг 2 – Программа для второго задания

```

package com.java.lab;

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

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

    public static void main(String[] args) throws
FileNotFoundException {
        var scan = new Scanner(new File("in.txt"));
        ArrayList<String> arrStr = new ArrayList<>();

        while (scan.hasNext()) {
            arrStr.add(scan.nextLine());
        }

        arrStr.sort(Collections.reverseOrder());
        System.out.println(arrStr);
    }
}

```

Результат выполнения программы показан на рисунке 2.

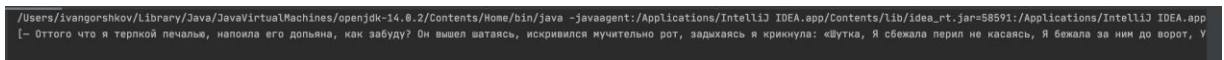


Рисунок 2 – Результат выполнения программы

Задание 3

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`.

Код написанной программы представлен в листинге 3.

Листинг 3 – Программа для третьего задания

```

package com.java.lab;

import java.awt.geom.Line2D;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.util.*;

// 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.

```
public class Main {

    public static void main(String[] args) throws IOException {
        var plane = new Plane();

        plane.addPoint(new Point(1,2));
        plane.addPoint(new Point(2,5));
        plane.addPoint(new Point(3,6));
        plane.addPoint(new Point(4,8));
        plane.addPoint(new Point(5,9));
        plane.addPoint(new Point(6,1));
        plane.addPoint(new Point(7,3));
        plane.addPoint(new Point(8,4));
        plane.addPoint(new Point(9,6));
        plane.addPoint(new Point(10,4));
        plane.addPoint(new Point(11,3));
        plane.addPoint(new Point(5,13));

        plane.addLine(new Line(new Point(1,1), new Point(9,8)));
        plane.addLine(new Line(new Point(2,0), new Point(3,4)));
        plane.addLine(new Line(new Point(10,10), new Point(3,8)));
        plane.addLine(new Line(new Point(1,2), new Point(2,1)));
        plane.addLine(new Line(new Point(5,2), new Point(5,1)));

        Map<Line, Integer> map = new HashMap<>();

        for (var line : plane.getLineList()) {
            int count = 0;

            for (var point : plane.getPointList()) {
                if (line.contains(point)) {
                    count++;
                }
            }

            map.put(line, count);
        }

        Path path = Paths.get("out.txt");
        var k = 0;
        for (Map.Entry<Line, Integer> item : map.entrySet()) {
            if (item.getValue() == 0) continue;
            if (k == 1) {
                Files.write(path, Collections.singleton(item.getKey()
+ ": " + item.getValue()), StandardCharsets.UTF_8);
            } else {
                Files.write(path, Collections.singleton(item.getKey()
+ ": " + item.getValue()), StandardCharsets.UTF_8,
StandardOpenOption.APPEND);
            }
            k += 1;
        }
    }
}
```

```

class Plane {

    private final Set<Point> pointList = new HashSet<>();
    private final Set<Line> lineList = new HashSet<>();

    Plane() { }

    void addPoint(Point point) {
        pointList.add(point);
    }

    void addLine(Line line) {
        lineList.add(line);
    }

    Set<Line> getLineList() {
        return lineList;
    }

    Set<Point> getPointList() {
        return pointList;
    }
}

class Point {
    private final float x;
    private final float y;

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

    float getX() {
        return x;
    }

    float getY() {
        return y;
    }

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

class Line {

    private final Point startPoint;
    private final Point endPoint;

    Line(Point startPoint, Point endPoint) {

```

```

        this.startPoint = startPoint;
        this.endPoint = endPoint;
    }

    Point getStartPoint() {
        return startPoint;
    }

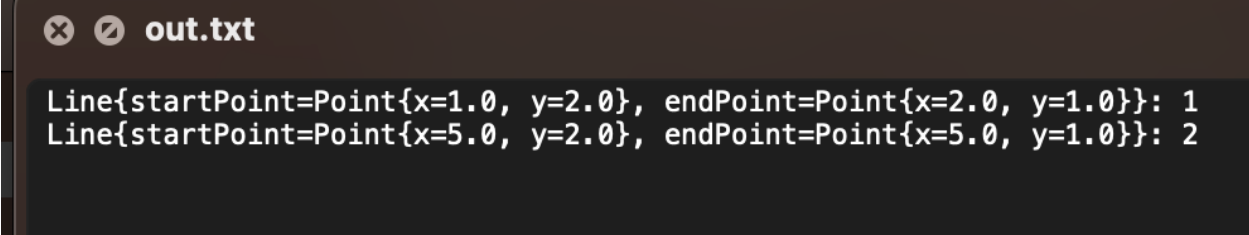
    Point getEndPoint() {
        return endPoint;
    }

    boolean contains(Point point) {
        var line = new Line2D.Double();
        line.setLine(getStartPoint().getX(), getStartPoint().getY(),
getEndPoint().getX(), getEndPoint().getY());
        return line.ptLineDist(point.getX(), point.getY()) == 0;
    }

    @Override
    public String toString() {
        return "Line{" +
            "startPoint=" + startPoint +
            ", endPoint=" + endPoint +
            '}';
    }
}

```

Результат выполнения программы показан на рисунке 3.



```

Line{startPoint=Point{x=1.0, y=2.0}, endPoint=Point{x=2.0, y=1.0}}: 1
Line{startPoint=Point{x=5.0, y=2.0}, endPoint=Point{x=5.0, y=1.0}}: 2

```

Рисунок 3 – Результат выполнения программы

Задание 4

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

Код написанной программы представлен в листинге 4.

Листинг 4 – Программа для четвертого задания

```

package com.java.lab;
import java.util.*;

// 7. На плоскости задано N отрезков. Найти точку пересечения двух
отрезков, имеющую минимальную абсциссу. Использовать класс TreeMap.
    public class Main {

        public static void main(String[] args) {
            LineSegment[] lineSegments = {

```

```

        new LineSegment(new Point(1, 8), new Point(9, 1)),
        new LineSegment(new Point(2, 7), new Point(8, 2)),
        new LineSegment(new Point(3, 6), new Point(7, 3)),
        new LineSegment(new Point(4, 5), new Point(6, 4)),
        new LineSegment(new Point(5, 4), new Point(5, 5)),
        new LineSegment(new Point(6, 3), new Point(4, 3)),
        new LineSegment(new Point(7, 2), new Point(3, 2)),
    };

    TreeMap<Double, Point> intersectionPoints = new
TreeMap<>();

    for (int i = 0; i < lineSegments.length; i++) { for (int j
= i + 1; j < lineSegments.length; j++) {
        if (lineSegments[i].intersects(lineSegments[j])) {
            Point intersectionPoint =
lineSegments[i].getIntersectionPoint(lineSegments[j]);
            intersectionPoints.put(intersectionPoint.getX(), intersectionPoint);
        }
    }

    Point minIntersectionPoint =
intersectionPoints.firstEntry().getValue();
    System.out.println("Min Point: " + minIntersectionPoint);

}

}

class LineSegment {
    Point startPoint;
    Point endPoint;

    LineSegment(Point startPoint, Point endPoint) {
        this.startPoint = startPoint;
        this.endPoint = endPoint;
    }

    double getSlope() {
        return (endPoint.getY() - startPoint.getY()) /
(endPoint.getX() - startPoint.getX());
    }

    double getYIntercept() {
        return startPoint.getY() - getSlope() * startPoint.getX();
    }

    boolean intersects(LineSegment other) {
        double common = (endPoint.getX() -
startPoint.getX())*(other.endPoint.getY() - other.startPoint.getY()) -
(endPoint.getY() - startPoint.getY())*(other.endPoint.getX() -
other.startPoint.getX());

        if (common == 0) return false;

        double rH = (startPoint.getY() -
other.startPoint.getY())*(other.endPoint.getX() -

```



```

other.startPoint.getX()) - (startPoint.getX() -
other.startPoint.getX())*(other.endPoint.getY() -
other.startPoint.getY());
        double sH = (startPoint.getY() -
other.startPoint.getY())*(endPoint.getX() - startPoint.getX()) -
(startPoint.getX() - other.startPoint.getX())*(endPoint.getY() -
startPoint.getY());

        double r = rH / common;
        double s = sH / common;

        return r >= 0 && r <= 1 && s >= 0 && s <= 1;
    }

    Point getIntersectionPoint(LineSegment other) {
        var x = (other.getYIntercept() - getYIntercept()) /
(getSlope() - other.getSlope());
        double y = getSlope() * x + getYIntercept();
        return new Point(x, y);
    }
}

class Point {
private final double x;
private final double y;

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

double getX() {
    return x;
}

double getY() {
    return y;
}

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

```

Результат выполнения программы показан на рисунке 4.

```

/Users/ivangorshkov/Library/Java/JavaVirtualMachines/openjdk-
Min Point: Point{x=4.999999999999991, y=4.5000000000000006}

```

Рисунок 4 – Результат выполнения программы

Вывод: В результате выполнения лабораторной работы были приобретены навыки работы с коллекциями на языке программирования Java.