|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 **09.04.01/12 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений**

**Отчет**

|  |
| --- |
| **по лабораторной работе № 6** |

Вариант 6

**Название:** Коллекции

**Дисциплина:** Языки программирования для работы с большими данными

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

Москва, 2023

**ВВЕДЕНИЕ**

**Задание:**

* 1. Не используя вспомогательных объектов, переставить отрицательные элементы данного списка в конец, а положительные – в начало этого списка.
  2. Ввести строки из файла, записать в список ArrayList. Выполнить сортировку строк, используя метод sort() из класса Collections.
  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.
  4. На плоскости задано N отрезков. Найти точку пересечения двух отрезков, имеющую минимальную абсциссу. Использовать класс TreeMap.

**ОСНОВНАЯ ЧАСТЬ**

**Задание 1.1**

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

Код программы представлен в листинге 1. Результат работы программы показан на рисунке 1.

Листинг 1 – Код класса Main

import java.util.\*;  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Введите элементы списка через пробел: ");  
 String input = scanner.nextLine();  
 String[] items = input.split(" ");  
 List<Integer> listArray = new ArrayList<>();  
 for (String item : items) {  
 listArray.add(Integer.*parseInt*(item));  
 }  
 *rearrangeList*(listArray);  
 System.*out*.println("Переставленный список: " + listArray);  
 }  
  
 public static void rearrangeList(List<Integer> listArray) {  
 int j = 0;  
 for (int i = 0; i < listArray.size(); i++) {  
 if (listArray.get(i) < 0) {  
 if (i != j) {  
 int temp = listArray.get(i);  
 listArray.set(i, listArray.get(j));  
 listArray.set(j, temp);  
 }  
 j++;  
 }  
 }  
 }  
}

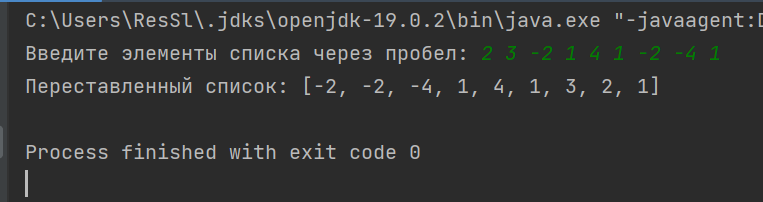


Рисунок 1 – Результат работы программы 1.1

**Задание 1.2**

Вводятся строки из файла, записываются в список ArrayList. Выполняется сортировка строк, используя метод sort() из класса Collections.

Код программы представлен в листингах 2, 3. Результат работы программы показан на рисунке 2.

Листинг 2 – Код класса Main

// Ввести строки из файла, записать в список ArrayList.  
// Выполнить сортировку строк, используя метод sort() из класса Collections.  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 ArrayList<String> lines = new ArrayList<>();  
 try {  
 Scanner scanner = new Scanner(new File("rows.txt"));  
 while (scanner.hasNextLine()) {  
 String line = scanner.nextLine();  
 lines.add(line);  
 }  
 scanner.close();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 Collections.*sort*(lines);  
 System.*out*.println("Отсортированные строки:");  
 for (String line : lines) {  
 System.*out*.println(line);  
 }  
 }  
}

Листинг 3 – Код файла rows.txt

asd  
fds  
few  
kkfd  
dfk  
reiw  
asd  
fdk  
fsfsd  
dad

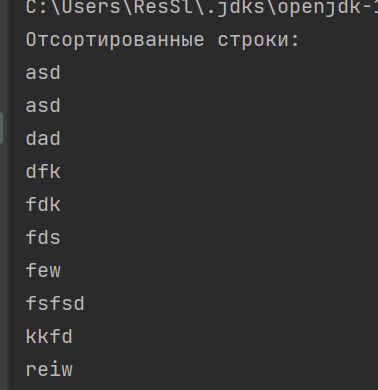


Рисунок 2 – Результат работы программы 1.2

**Задание 1.3**

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

Код программы представлен в листингах 4, 5. Результат работы программы показан на рисунке 3.

Листинг 4 – Код класса Main

import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 ArrayList<Point> points = new ArrayList<>();  
 try {  
 Scanner scanner = new Scanner(new File("points.txt"));  
 while (scanner.hasNextLine()) {  
 String line = scanner.nextLine();  
 String[] parts = line.split(" ");  
 int x = Integer.*parseInt*(parts[0]);  
 int y = Integer.*parseInt*(parts[1]);  
 points.add(new Point(x, y));  
 }  
 scanner.close();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 HashMap<Line, Integer> lines = new HashMap<>();  
 for (int i = 0; i < points.size(); i++) {  
 Point p1 = points.get(i);  
 for (int j = i + 1; j < points.size(); j++) {  
 Point p2 = points.get(j);  
 Line line = new Line(p1, p2);  
 if (lines.containsKey(line)) {  
 lines.put(line, lines.get(line) + 1);  
 } else {  
 lines.put(line, 2);  
 }  
 }  
 }  
  
 try {  
 PrintWriter writer = new PrintWriter(new File("lines.txt"));  
 for (Line line : lines.keySet()) {  
 int count = lines.get(line);  
 if (count > 2) {  
 writer.println(line.toString() + " - " + count + " points");  
 }  
 }  
 writer.close();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private static class Point {  
 public int x;  
 public int y;  
  
 public Point(int x, int y) {  
 this.x = x;  
 this.y = y;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (obj instanceof Point) {  
 Point other = (Point) obj;  
 return this.x == other.x && this.y == other.y;  
 }  
 return false;  
 }  
  
 @Override  
 public int hashCode() {  
 return this.x \* 31 + this.y \* 17;  
 }  
 }  
  
 private static class Line {  
 public double a;  
 public double b;  
 public double c;  
  
 public Line(Point p1, Point p2) {  
 this.a = p1.y - p2.y;  
 this.b = p2.x - p1.x;  
 this.c = p1.x \* p2.y - p2.x \* p1.y;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (obj instanceof Line) {  
 Line other = (Line) obj;  
 return this.a == other.a && this.b == other.b && this.c == other.c;  
 }  
 return false;  
 }  
  
 @Override  
 public int hashCode() {  
 return (int) (this.a \* 31 + this.b \* 17 + this.c);  
 }  
  
 @Override  
 public String toString() {  
 return this.a + "a + " + this.b + "b" + " = - (" + this.c + ")";  
 }  
 }  
}

Листинг 5 – Код файла points.txt

0 0  
1 1  
3 4  
2 2

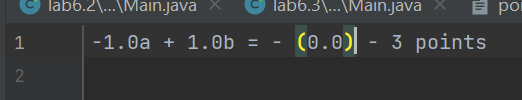


Рисунок 3 – Результат работы программы 1.3

**Задание 1.4**

На плоскости задано N отрезков. Находится точка пересечения двух отрезков, имеющую минимальную абсциссу. Используется класс TreeMap.

Код программы представлен в листингах 6, 7, 8. Результат работы программы показан на рисунке 4.

Листинг 6 – Код класса Point

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

Листинг 7 – Код класса Section

// На плоскости задано N отрезков.  
// Найти точку пересечения двух отрезков, имеющую минимальную абсциссу.  
// Использовать класс TreeMap.  
public class Section {  
 Point start;  
 Point end;  
 public Section(Point start, Point end) {  
 this.start = start;  
 this.end = end;  
 }  
  
 // возвращает значение коэффициента наклона (углового коэффициента) прямой,  
 // заданной отрезком, который вычисляется по формуле  
 // (y2 - y1) / (x2 - x1), где (x1, y1) и (x2, y2) - координаты начальной и конечной точек отрезка.  
 public double getSlope() {  
 return (end.y - start.y) / (end.x - start.x);  
 }  
  
 // возвращает значение y-пересечения (смещения) прямой,  
 // заданной отрезком, который вычисляется по формуле y1 - k \* x1,  
 // где k - значение коэффициента наклона, а (x1, y1) - координаты начальной точки отрезка.  
 public double getYIntercept() {  
 return start.y - getSlope() \* start.x;  
 }  
  
 public boolean intersects(Section other) {  
 double thisMinX = Math.*min*(start.x, end.x);  
 double thisMaxX = Math.*max*(start.x, end.x);  
 double otherMinX = Math.*min*(other.start.x, other.end.x);  
 double otherMaxX = Math.*max*(other.start.x, other.end.x);  
 if (thisMaxX < otherMinX || thisMinX > otherMaxX) {  
 return false;  
 }  
  
 double thisMinY = Math.*min*(start.y, end.y);  
 double thisMaxY = Math.*max*(start.y, end.y);  
 double otherMinY = Math.*min*(other.start.y, other.end.y);  
 double otherMaxY = Math.*max*(other.start.y, other.end.y);  
 if (thisMaxY < otherMinY || thisMinY > otherMaxY) {  
 return false;  
 }  
  
 double thisSlope = getSlope();  
 double otherSlope = other.getSlope();  
 if (thisSlope == otherSlope) {  
 return false;  
 }  
  
 double x = (other.getYIntercept() - getYIntercept()) / (thisSlope - otherSlope);  
 if (x < thisMinX || x > thisMaxX || x < otherMinX || x > otherMaxX) {  
 return false;  
 }  
  
 return true;  
 }  
  
 public Point getIntersectionPoint(Section other) {  
 double thisSlope = getSlope();  
 double otherSlope = other.getSlope();  
 double thisYIntercept = getYIntercept();  
 double otherYIntercept = other.getYIntercept();  
 double x = (otherYIntercept - thisYIntercept) / (thisSlope - otherSlope);  
 double y = thisSlope \* x + thisYIntercept;  
 return new Point(x, y);  
 }  
}

Листинг 8 – Код класса Main

import java.util.TreeMap;  
public class Main {  
 public static void main(String[] args) {  
 Section[] lineSegments = {  
 new Section(new Point(1, 1), new Point(4, 4)),  
 new Section(new Point(2, 3), new Point(5, 0)),  
 new Section(new Point(-1, 0), new Point(2, 2)),  
 new Section(new Point(-3, -1), new Point(0, 4)),  
 new Section(new Point(0, 2), new Point(3, -1))  
 };  
  
 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.x, intersectionPoint);  
 }  
 }  
 }  
  
 Point minIntersectionPoint = intersectionPoints.firstEntry().getValue();  
 System.*out*.println("Минимальная точка пересечения: (" + minIntersectionPoint.x + ", " + minIntersectionPoint.y + ")");  
 }  
}

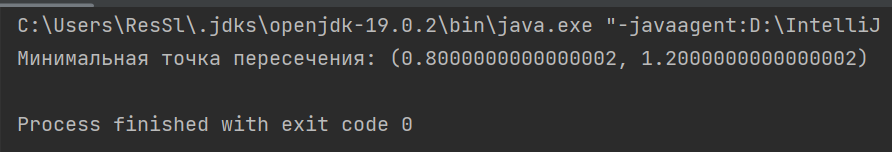


Рисунок 4 – Результат работы программы 1.4

**Вывод**: В результате выполнения лабораторной работы были получены практические навыки работы с коллекциями, а в особенности с классами Map, Set, HashMap, TreeMap.