|  |  |
| --- | --- |
| 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**

**Отчет**

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-21М |  |  | В.В. Ерискин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Задание**

Задача 1.

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

Задача 2.

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

**Решение**:

Программа 1.

|  |
| --- |
|  |
|  | 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) throws FileNotFoundException { |
|  | Scanner scan = new Scanner(new File("/Users/vitaly/Desktop/testFile.txt")); |
|  | ArrayList<String> arrStr = new ArrayList<>(); |
|  |  |
|  | while (scan.hasNext()) { |
|  | arrStr.add(scan.nextLine()); |
|  | } |
|  | scan.close(); |
|  | Collections.sort(arrStr); |
|  |  |
|  | for (String anArrStr : arrStr) { |
|  | System.out.println(anArrStr); |
|  | } |
|  | } |
|  | } |

Программа 2.

|  |
| --- |
|  |
| import java.util.ArrayList; | |
|  | import java.util.List; |
|  | import java.util.Map; |
|  | import java.util.TreeMap; |
|  |  |
|  | public class Main { |
|  | private static Map<Double, List<Point>> map = new TreeMap<>(); |
|  | private static Point point = new Point(Math.random()\*10, Math.random()\*10); |
|  | private static Point point1 = new Point(Math.random()\*10, Math.random()\*10); |
|  | private static Point point2 = new Point(Math.random()\*10, Math.random()\*10); |
|  | private static Point point3 = new Point(Math.random()\*10, Math.random()\*10); |
|  | private static Point point4 = new Point(Math.random()\*10, Math.random()\*10);; |
|  | private static Point point5 = new Point(Math.random()\*10, Math.random()\*10); |
|  | private static Point point6 = new Point(Math.random()\*10, Math.random()\*10); |
|  | private static Point point7 = new Point(Math.random()\*10, Math.random()\*10); |
|  | private static Point point8 = new Point(Math.random()\*10, Math.random()\*10); |
|  | private static Point point9 = new Point(Math.random()\*10, Math.random()\*10); |
|  | private static Point point10 = new Point(Math.random()\*10, Math.random()\*10); |
|  | private static Point point11 = new Point(Math.random()\*10, Math.random()\*10); |
|  |  |
|  | /\*\* |
|  | \* @param args |
|  | \*/ |
|  | public static void main(String[] args) { |
|  | System.out.println(checkSegments(point, point1, point2, point3)); |
|  | System.out.println(checkSegments(point4, point5, point6, point7)); |
|  | System.out.println(checkSegments(point8, point9, point10, point11)); |
|  | System.out.println(map); |
|  | } |
|  |  |
|  | //проверим пересечение [p1, p2] and [p3, p4] |
|  | private static boolean checkSegments(Point p1, Point p2, Point p3, Point p4) { |
|  | List<Point> list = new ArrayList<>(); |
|  | List<Point> buff = new ArrayList<>(); |
|  | //сортируем, p1.x <= p2.x |
|  | if (p2.x < p1.x) { |
|  | Point tmp = p1; |
|  | p1 = p2; |
|  | p2 = tmp; |
|  | } |
|  | // p3.x <= p4.x |
|  | if (p4.x < p3.x) { |
|  | Point tmp = p3; |
|  | p3 = p4; |
|  | p4 = tmp; |
|  | } |
|  | //проверяем потенциальную возможность пересечения |
|  | if (p2.x < p3.x) { |
|  | return false; //если не пересекаются |
|  | } |
|  | //проверка на вертикальность |
|  | if ((p1.x - p2.x == 0) && (p3.x - p4.x == 0)) { |
|  | //проверка на параллельность |
|  | if (p1.x == p3.x) { |
|  | //проверка имеют ли общее y (пересекаются ли) |
|  | //для этого проверяем на обратное условие (нет пересечения) |
|  | if (!((Math.max(p1.y, p2.y) < Math.min(p3.y, p4.y)) || |
|  | (Math.min(p1.y, p2.y) > Math.max(p3.y, p4.y)))) { |
|  | return true; |
|  | } |
|  | } |
|  | return false; |
|  | } |
|  | //ищем коэффициенты в уравнении координат, содержащих отрезки |
|  | //f1(x) = A1\*x + b1 = y |
|  | //f2(x) = A2\*x + b2 = y |
|  | //для вертикали |
|  | if (p1.x - p2.x == 0) { |
|  | //find Xa, Ya - точки пересечения двух линий |
|  | double Xa = p1.x; |
|  | double A2 = (p3.y - p4.y) / (p3.x - p4.x); |
|  | double b2 = p3.y - A2 \* p3.x; |
|  | double Ya = A2 \* Xa + b2; |
|  | if (p3.x <= Xa && p4.x >= Xa && Math.min(p1.y, p2.y) <= Ya && |
|  | Math.max(p1.y, p2.y) >= Ya) { |
|  | list.add(p1); |
|  | list.add(p2); |
|  | list.add(p3); |
|  | list.add(p4); |
|  | map.put(Xa, list); |
|  | return true; |
|  | } |
|  | return false; |
|  | } |
|  |  |
|  | //для второго сегмента |
|  | if (p3.x - p4.x == 0) { |
|  | // Xa, Ya - точки переченния двух линий |
|  | double Xa = p3.x; |
|  | System.out.println(Xa); |
|  | double A1 = (p1.y - p2.y) / (p1.x - p2.x); |
|  | double b1 = p1.y - A1 \* p1.x; |
|  | double Ya = A1 \* Xa + b1; |
|  | if (p1.x <= Xa && p2.x >= Xa && Math.min(p3.y, p4.y) <= Ya && |
|  | Math.max(p3.y, p4.y) >= Ya) { |
|  | list.add(p1); |
|  | list.add(p2); |
|  | list.add(p3); |
|  | list.add(p4); |
|  | map.put(Xa, list); |
|  | return true; |
|  | } |
|  | return false; |
|  | } |
|  |  |
|  | //оба сегмента не вертикальны |
|  | double A1 = (p1.y - p2.y) / (p1.x - p2.x); |
|  | double A2 = (p3.y - p4.y) / (p3.x - p4.x); |
|  | double b1 = p1.y - A1 \* p1.x; |
|  | double b2 = p3.y - A2 \* p3.x; |
|  | if (A1 == A2) { |
|  | return false; //если параллельны |
|  | } |
|  |  |
|  | //Xa абсциса точки пересечения двух прямых |
|  | double Xa = (b2 - b1) / (A1 - A2); |
|  |  |
|  | if ((Xa < Math.max(p1.x, p3.x)) || (Xa > Math.min(p2.x, p4.x))) { |
|  | return false; //Xa находится вне пересечения проекций отрезков на оси X |
|  | } else { |
|  | list.add(p1); |
|  | list.add(p2); |
|  | list.add(p3); |
|  | list.add(p4); |
|  | map.put(Xa, list); |
|  | return true; |
|  | } |
|  | } |
|  |  |
|  | } |

|  |  |
| --- | --- |
| public class Point { | |
|  | double x, y; |
|  |  |
|  | public Point(double newX, double newY) { |
|  | x = newX; |
|  | y = newY; |
|  | } |
|  |  |
|  | @Override |
|  | public String toString() { |
|  | return "Point{x=" + x + ", y=" + y + "}"; |
|  | } |
|  | } |

**Вывод:**

Задание выполнено, программы компилируются и успешно выполняются.