|  |  |
| --- | --- |
| 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 **09.04.01/07 Интеллектуальные системы анализа,**

**обработки и интерпретации больших данных.**

**Отчет**

|  |
| --- |
| **по лабораторной работе № 6** |
|  |
| **Вариант 16** |

**Название:**

Коллекции

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-23М | |  |  | В.И. Тарасов |
|  | | (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 языке Java.

**Ход работы**

**Задание 1:**

1. Ввести строки из файла, записать в список ArrayList. Выполнить сортировку строк, используя метод sort() из класса Collections.
2. Задана строка, состоящая из символов '(', ')', '[', ']', '{', '}'. Проверить правильность расстановки скобок. Использовать стек.

Листинг 1 – Код подзадачи 1

package src.lab6\_1;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.Collections;

public class Program {

public static void main(String[] args) throws Exception {

try {

ArrayList<String> lines = new ArrayList<>(Files.readAllLines(

Paths.get("src/lab6\_1/data/input.txt"), StandardCharsets.UTF\_8));

System.out.println("List before sorting: \n" + lines);

Collections.sort(lines);

System.out.println("List after sorting: \n" + lines);

} catch (FileNotFoundException e) {

System.out.println("Файл не найден");

e.printStackTrace();

} catch (IOException e) {

System.out.println("Произошла ошибка I/O");

e.printStackTrace();

}

}

}

Результаты выполнения подзадачи 1 представлены на рисунке 1.

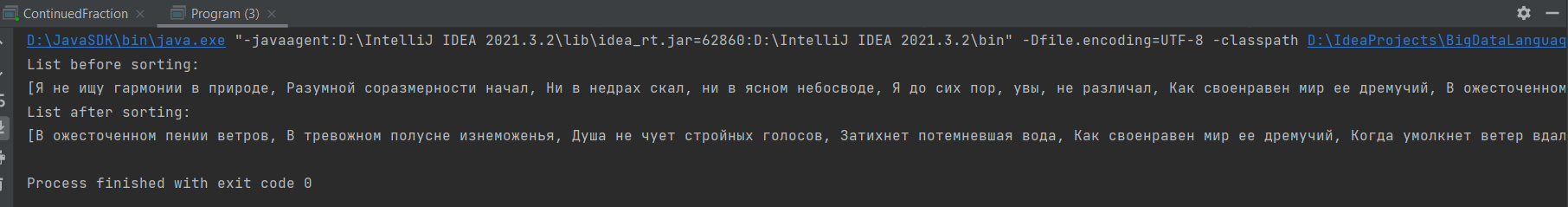


Рисунок 1 – Результат выполнения задания 1

Листинг 2 – Код подзадачи 2

package src.lab6\_1;

import java.util.EmptyStackException;

import java.util.HashMap;

import java.util.Scanner;

import java.util.Stack;

public class Program1 {

public static void main(String[] args) throws Exception {

Scanner console = new Scanner(System.in);

HashMap<Character, Character> parenthesis = new HashMap<>();

parenthesis.put('{', '}');

parenthesis.put('(', ')');

parenthesis.put('[', ']');

System.out.println("Введите строку из скобок");

String str = console.nextLine();

Stack<Character> stack = new Stack<>();

try {

for (char c : str.toCharArray()) {

if (c == '[' || c == '(' || c == '{') {

stack.push(c);

} else if (c == ']' || c == ')' || c == '}') {

if (parenthesis.get(stack.peek()) == c) {

stack.pop();

} else {

throw new Exception("Неверный порядок скобок");

}

}

}

if (stack.isEmpty()) {

System.out.println("Скобки расставлены правильно");

}

else {

System.out.println("Открывающих скобок больше");

}

} catch (EmptyStackException e) {

System.out.println("Закрывающая скобка без открывающей");

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

}

Результаты выполнения подзадачи 2 представлены на рисунке 2.

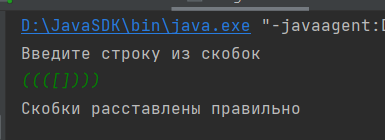


Рисунок 2 – Результаты выполнения подзадачи 2

**Задание 2:**

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

Листинг 3 – Код класса Point

package src.lab6\_2;

import java.util.Objects;

public class Point {

float x;

float y;

public Point(float x, float y){

this.x = x;

this.y = y;

}

public float getX() {

return x;

}

public float getY() {

return y;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Point point = (Point) o;

return Float.compare(point.x, x) == 0 && Float.compare(point.y, y) == 0;

}

@Override

public int hashCode() {

return Objects.hash(x, y);

}

@Override

public String toString() {

return "Point{" +

"x=" + x +

", y=" + y +

'}';

}

}

Листинг 4 – Код класса Line

package src.lab6\_2;

import java.util.ArrayList;

public class Line {

float slope;

ArrayList<Point> points;

boolean duplicate = false;

public Line(float slope) {

this.slope = slope;

this.points = new ArrayList<>();

}

public void addPoint(Point point) {

this.points.add(point);

}

public float getSlope() {

return slope;

}

public boolean getDuplicate() {

return duplicate;

}

public void setDuplicate(boolean duplicate) {

this.duplicate = duplicate;

}

@Override

public String toString() {

return "Line{" +

"slope=" + slope +

", points=" + points.toString() +

'}';

}

}

Листинг 5 – Код класса LineSegment

package src.lab6\_2;

public class LineSegment {

Point start;

Point end;

public LineSegment(Point start, Point end){

this.start = start;

this.end = end;

}

public Point getStart() {

return start;

}

public Point getEnd() {

return end;

}

public Point FindIntersection(LineSegment b){

float t = ((this.start.getX() - b.start.getX()) \* (b.start.getY() - b.end.getY()) - (this.start.getY() - b.start.getY()) \* (b.start.getX() - b.end.getX()))

/ ((this.start.getX()- this.end.getX()) \* (b.start.getY() - b.end.getY()) - (this.start.getY() - this.end.getY()) \* (b.start.getX() - b.end.getX()));

float u = ((this.start.getX() - b.start.getX()) \* (this.start.getY() - this.end.getY()) - (this.start.getY() - b.start.getY()) \* (this.start.getX() - this.end.getX()))

/ ((this.start.getX()- this.end.getX()) \* (b.start.getY() - b.end.getY()) - (this.start.getY() - this.end.getY()) \* (b.start.getX() - b.end.getX()));

if ((t >= 0 && t <= 1.0) && (u >= 0 && u <= 1.0)) {

return new Point((this.start.getX() + t \* (this.end.getX() - this.start.getX())), (this.start.getY() + t \* (this.end.getY() - this.start.getY())));

}

return null;

}

}

Листинг 6 – Код основной программы подзадачи 1

package src.lab6\_2;

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.ArrayList;

import java.util.Comparator;

import java.util.HashMap;

import java.util.Map;

public class Program1 {

public static void main(String[] args) throws IOException {

ArrayList<Point> pointsArray = new ArrayList<>();

Path file = Paths.get("src/lab6\_2/data/output.txt");

pointsArray.add(new Point(1, 1));

pointsArray.add(new Point(2, 2));

pointsArray.add(new Point(1, 4));

pointsArray.add(new Point(4, 4));

HashMap<Line, Integer> linePoints = new HashMap<>();

ArrayList<Line> lines = new ArrayList<>();

for (int i = 0; i < pointsArray.size(); i++){

for (int j = i + 1; j < pointsArray.size(); j++){

float slope;

slope = getSlope(pointsArray.get(i), pointsArray.get(j));

Line line = new Line(slope);

line.addPoint(pointsArray.get(i));

line.addPoint(pointsArray.get(j));

for (int z = j + 1; z < pointsArray.size(); z++){

slope = getSlope(pointsArray.get(i), pointsArray.get(z));

if (Float.compare(line.getSlope(), slope) == 0) {

line.addPoint(pointsArray.get(z));

}

}

lines.add(line);

}

}

lines.sort((o1, o2)->o2.points.size()-o1.points.size());

for (int i = 0; i < lines.size(); i++){

for (int j = i + 1; j < lines.size(); j++){

if (lines.get(i).points.containsAll(lines.get(j).points)){

lines.get(j).setDuplicate(true);

}

}

}

for (Line j: lines) {

if (!j.getDuplicate()) {

System.out.println(j);

linePoints.put(j, j.points.size());

}

}

Files.writeString(file,"");

for (Map.Entry<Line, Integer> entry : linePoints.entrySet()) {

Files.writeString(file, entry.getKey().toString() + " number of points: " + entry.getValue() + "\n",

StandardOpenOption.APPEND);

}

}

public static float getSlope(Point a, Point b){

if (b.getX() - a.getX() == 0) {

return 90;

}

else {

return (b.getY() - a.getY()) / (b.getX() - a.getX());

}

}

}

Результаты выполнения подзадачи 1 представлены на рисунке 3.

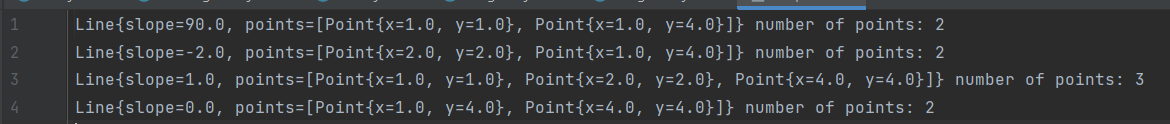


Рисунок 3 – Результат выполнения подзадачи 1

Листинг 7 – Код основной программы подзадачи 2

package src.lab6\_2;

import java.util.TreeMap;

public class Program {

public static void main(String[] args) {

LineSegment a = new LineSegment(

new Point(1, 1), new Point(3, 2)

);

LineSegment b = new LineSegment(

new Point(1,2), new Point(3,1)

);

LineSegment c = new LineSegment(

new Point(4, 1), new Point(5, 3)

);

LineSegment d = new LineSegment(

new Point(4,5), new Point(5,1)

);

TreeMap<Float, Float> tree = new TreeMap<>();

AddToTree(tree, a.FindIntersection(b));

AddToTree(tree, c.FindIntersection(d));

AddToTree(tree, a.FindIntersection(d));

AddToTree(tree, b.FindIntersection(c));

System.out.println(tree);

System.out.println("X: " + tree.firstEntry().getKey() + " Y: " + tree.firstEntry().getValue());

}

public static void AddToTree(TreeMap<Float, Float> tree, Point p){

if (p != null) tree.put(p.getX(), p.getY());

}

}

Результаты выполнения подзадачи 2 представлены на рисунке 4.

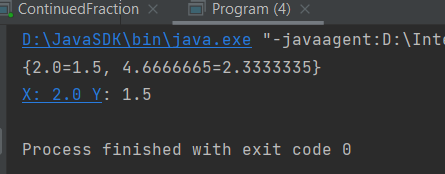


Рисунок 4 – Результаты выполнения подзадачи 2

**Местоположение проектных файлов** –файлы проекта расположены в репозитории GitHub по следующим адресам:

<https://github.com/Fregatty/BigDataLanguages/tree/main/src/lab6_1>

<https://github.com/Fregatty/BigDataLanguages/tree/main/src/lab6_2>

**Вывод** – в результате выполнения лабораторной работы были получены навыки работы с различными коллекциями в языке Java, включая ArrayList, HashMap и TreeMap.