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

**Название:** Коллекции

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

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

Москва, 2023

**Задание 1:** умножить два многочлена заданной степени, если коэффициенты многочленов хранятся в различных списках.

Решение задания представлено в листинге 1.

Листинг 1 – Программный код модуля Main.java для задания 1

import java.util.ArrayList;

import java.util.List;

public class Main {

public static void main(String[] args) {

List<Integer> numbers = new ArrayList<>();

numbers.add(2);

numbers.add(-5);

numbers.add(10);

numbers.add(-3);

numbers.add(8);

int size = numbers.size();

int positiveIndex = 0;

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

if (numbers.get(i) < 0) {

int negativeValue = numbers.remove(i);

numbers.add(negativeValue);

i--; // Чтобы корректно обработать новый элемент, замещающий удаленный элемент

size--; // Размер списка уменьшается после удаления элемента

} else {

int positiveValue = numbers.remove(i);

numbers.add(positiveIndex, positiveValue);

positiveIndex++;

}

}

// Выводим результат

for (int number : numbers) {

System.out.print(number + " ");

}

}

}

**Задание 2:** не используя вспомогательных объектов, переставить отрицательные элементы данного списка в конец, а положительные – в начало этого списка.

Решение задания представлено в листинге 2.

Листинг 2 – Программный код модуля Main.java для задания 2

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class Main {

public static void main(String[] args) {

List<String> lines = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader("D:\\JavaLabs\\JavaLabs\\lab6.2\\src\\file.txt"))) {

String line;

while ((line = reader.readLine()) != null) {

lines.add(line);

}

} catch (IOException e) {

System.err.println("Ошибка при чтении файла: " + e.getMessage());

return;

}

Collections.sort(lines);

// Выводим отсортированные строки

for (String line : lines) {

System.out.println(line);

}

}

}

**Задание 3:** во входном файле расположены два набора положительных чисел; между наборами стоит отрицательное число. Построить два списка C1 и С2, элементы которых содержат соответственно числа 1-го и 2-го набора таким образом, чтобы внутри одного списка числа были упорядочены по возрастанию. Затем объединить списки C1 и С2 в один упорядоченный список, изменяя только значения полей ссылочного типа.

Решение задания представлено в листинге 3.

Листинг 3 – Программный код модуля Main.java для задания 3

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class Main {

public static void main(String[] args) {

List<Integer> c1 = new ArrayList<>();

List<Integer> c2 = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader("D:\\JavaLabs\\JavaLabs\\lab6.3\\src\\file.txt"))) {

String line;

boolean isFirstSet = true;

while ((line = reader.readLine()) != null) {

int number = Integer.parseInt(line);

if (number < 0) {

isFirstSet = false;

} else if (isFirstSet) {

c1.add(number);

} else {

c2.add(number);

}

}

} catch (IOException e) {

System.err.println("Ошибка при чтении файла: " + e.getMessage());

return;

}

Collections.sort(c1);

Collections.sort(c2);

List<Integer> combinedList = new ArrayList<>(c1);

combinedList.addAll(c2);

Collections.sort(combinedList);

// Выводим объединенный и упорядоченный список

for (int number : combinedList) {

System.out.println(number);

}

}

}

**Задание 4:** 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.

Решение задания представлено в листингах 4-6.

Листинг 4 – Программный код модуля Main.java для задания 4

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class Main {

public static void main(String[] args) {

List<Point> points = new ArrayList<>();

points.add(new Point(1, 2));

points.add(new Point(3, 4));

points.add(new Point(5, 6));

points.add(new Point(1, 6));

points.add(new Point(3, 8));

points.add(new Point(5, 10));

Map<String, Line> lineMap = new HashMap<>();

// Перебираем все комбинации пар точек

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

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

Point point1 = points.get(i);

Point point2 = points.get(j);

// Рассчитываем угловой коэффициент и свободный член прямой

double slope = (double) (point2.getY() - point1.getY()) / (point2.getX() - point1.getX());

double intercept = point1.getY() - slope \* point1.getX();

// Создаем уникальный идентификатор прямой

String lineId = slope + "\_" + intercept;

// Если прямая уже существует в HashMap, увеличиваем счетчик точек

if (lineMap.containsKey(lineId)) {

Line line = lineMap.get(lineId);

line.incrementPointCount();

} else {

// Создаем новую прямую и добавляем в HashMap

Line line = new Line(slope, intercept);

lineMap.put(lineId, line);

}

}

}

// Записываем описания прямых в файл

try (FileWriter writer = new FileWriter("D:\\JavaLabs\\JavaLabs\\lab6.4\\src\\lines.txt")) {

for (Line line : lineMap.values()) {

writer.write(line.toString() + System.lineSeparator());

}

} catch (IOException e) {

System.err.println("Ошибка при записи в файл: " + e.getMessage());

}

}

}

Листинг 5 – Программный код модуля Point.java для задания 4

class Point {

private int x;

private int y;

public Point(int x, int y) {

this.x = x;

this.y = y;

}

public int getX() {

return x;

}

public int getY() {

return y;

}

}

Листинг 6 – Программный код модуля Line.java для задания 4

class Line {

private double slope;

private double intercept;

private int pointCount;

public Line(double slope, double intercept) {

this.slope = slope;

this.intercept = intercept;

this.pointCount = 2; // Прямая проходит через две точки: начальную и конечную

}

public void incrementPointCount() {

pointCount++;

}

public int getPointCount() {

return pointCount;

}

@Override

public String toString() {

return "Прямая с угловым коэффициентом " + slope + ", свободным членом " + intercept +

" и проходит через " + pointCount + " точек.";

}

}

**Вывод:** в ходе выполнения данной лабораторной работы была изучена и применена на практике библиотека для работы с коллекциями на языке Java.