



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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И
СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.04.01 Интеллектуальные системы анализа,
обработки и интерпретации больших данных

ОТЧЕТ

по лабораторной работе № 6

Вариант 13

Название: Коллекции

Дисциплина: Языки программирования для работы с большими данными

Студент

ИУ6-22М

(Группа)

(Подпись, дата)

В.А.Ловцов

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2024

Цель: изучить работу с коллекциями в java.

Задание 1: с использованием множества выполнить попарное суммирование произвольного конечного ряда чисел по следующим правилам: на первом этапе суммируются попарно рядом стоящие числа, на втором этапе суммируются результаты первого этапа и т.д. до тех пор, пока не останется одно число.

Код:

```
import java.util.HashSet;
import java.util.Set;

public class Third_1 {
    public static void main(String[] args) {
        Set<Integer> numbers = new HashSet<>();
        // Добавьте произвольные числа в множество
        numbers.add(1);
        numbers.add(2);
        numbers.add(3);
        numbers.add(4);
        numbers.add(5);

        while (numbers.size() > 1) {
            Set<Integer> newNumbers = new HashSet<>();
            Integer prev = null;
            for (Integer num : numbers) {
                if (prev == null) {
                    prev = num;
                } else {
                    newNumbers.add(prev + num);
                    prev = null;
                }
            }
            if (prev != null) {
                newNumbers.add(prev);
            }
            numbers = newNumbers;
        }

        System.out.println("Результат попарного суммирования:
" + numbers.iterator().next());
    }
}
```

Работа программы показана на рисунке 1.

```
/Library/Java/JavaVirtualMachines/jdk-2
Результат попарного суммирования: 15

Process finished with exit code 0
```

Рисунок 1 – Работа программы

Задание 2: сложить два многочлена заданной степени, если коэффициенты многочленов хранятся в объекте HashMap.

Код:

```
import java.math.BigInteger;
import java.util.*;

class Polynomial {
    HashMap<Integer, Integer> coefficients;

    public Polynomial(HashMap<Integer, Integer> coefficients)
    {
        this.coefficients = coefficients;
    }

    public Polynomial add(Polynomial other) {
        int maxDegree = Math.max(getMaxDegree(this),
getMaxDegree(other));
        HashMap<Integer, Integer> result = new HashMap<>();
//        Polynomial result = new Polynomial();
        for (int degree = 0; degree <= maxDegree; ++degree) {
            Integer sum = coefficients.get(degree) +
other.coefficients.get(degree);
            result.put(degree, sum);
        }
        return new Polynomial(result);
    }

    private static int getMaxDegree(Polynomial polynomial) {
        int maxDegree = 0;
        for (HashMap.Entry<Integer, Integer> entry :
polynomial.coefficients.entrySet()) {
            maxDegree = Math.max(maxDegree, entry.getKey());
        }
        return maxDegree;
    }

    @Override
    public String toString() {
```

```

        StringBuilder sb = new StringBuilder("(");
        boolean firstTermAdded = false;

        for (HashMap.Entry<Integer, Integer> term :
coefficients.entrySet()) {
            if (term.getValue() != 0 || firstTermAdded) {
                if (firstTermAdded) {
                    sb.append("+ ");
                }
                firstTermAdded = true;

                if (term.getValue() != 0) {
sb.append(term.getValue()).append("x^").append(term.getKey()).
append(" ");
                } else {
                    sb.append((Object) 0).append("x ");
                }
            }
        }

        return sb.toString();
    }
}

public class Fourth_1 {
    public static void main(String[] args) {
        HashMap<Integer, Integer> a = new HashMap<>();
        a.put(0, 0);
        a.put(1, -10);
        a.put(2, 10);
        a.put(3, 15);
        Polynomial first = new Polynomial(a);

        HashMap<Integer, Integer> b = new HashMap<>();
        b.put(0, 1);
        b.put(1, 2);
        b.put(2, -3);
        b.put(3, 4);
        Polynomial second = new Polynomial(b);

        Polynomial sum = first.add(second);

        System.out.println(first);
        System.out.println(second);
        System.out.println(sum);
    }
}

```

Работа программы показана на рисунке 2.

```
(-10x^1) + (10x^2) + (15x^3)
(1x^0) + (2x^1) + (-3x^2) + (4x^3)
(1x^0) + (-8x^1) + (7x^2) + (19x^3)

Process finished with exit code 0
```

Рисунок 2 – Работа программы

Задание 3: Во входном файле хранятся две разреженные матрицы A и B. Построить циклически связанные списки SA и SB, содержащие ненулевые элементы соответственно матриц A и B. Просматривая списки, вычислить: а) сумму $S = A + B$; б) произведение $P = A * B$.

Код:

```
import java.util.*;

//package com.journaldev.readfileslinebyline;

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

class Node {
    int row;
    int col;
    int value;

    public Node(int row, int col, int value) {
        this.row = row;
        this.col = col;
        this.value = value;
    }

    @Override
    public String toString() {
        return "{" + value +
            "[" + row +
            ", " + col +
            "]}";
    }
}
```

```

class SparseMatrix{
    LinkedList<Node> list;
    int lenrows;
    int lencols;

    public SparseMatrix(int[][] matrix){

        assert matrix.length > 0;
        this.lencols = matrix.length;
        this.lenrows = matrix[0].length;
        list = new LinkedList<>();
        for (int i = 0; i < matrix.length; i++)
            for (int j = 0; j < matrix[0].length; j++){
                if (matrix[i][j] != 0){
                    Node tmp = new Node(i, j, matrix[i][j]);
                    list.add(tmp);
                }
            }
    }

    public SparseMatrix(LinkedList<Node> list) {
        this.list = list;
    }

    public SparseMatrix add(SparseMatrix other){
        LinkedList<Node> result = new LinkedList<>();

        ListIterator<Node> it1 = list.listIterator();
        ListIterator<Node> it2 = other.list.listIterator();

        while (it1.hasNext() || it2.hasNext()) {
            if (!it1.hasNext()) {
                Node cur = it2.next();
                // Node tmp = new Node(it2.g().row,
it2.next().col, it2.previous().value);
                result.add(cur);
            } else if (!it2.hasNext()) {
                Node cur = it1.next();
                // Node tmp = new Node(it1.next().row,
it1.next().col, it1.previous().value);
                result.add(cur);
            } else {
                Node node1 = it1.next();
                Node node2 = it2.next();
                if (node1.col == node2.col && node1.row ==
node2.row) {
                    Node sum = new Node(node1.row, node1.col,
node1.value + node2.value);
                    result.add(sum);
                } else if (node1.row < node2.row | (node1.row
== node2.row & node1.col < node2.col)) {

```

```

        result.add(node1);
        it2.previous();
    } else {
        result.add(node2);
        it1.previous();
    }
    }
}
return new SparseMatrix(result);
}

public SparseMatrix mul(SparseMatrix other){

    assert (lenrows == other.lencols);
    LinkedList<Node> result = new LinkedList<>();

    for (int i = 0; i < lenrows; i++) {
        for (int j = 0; j < other.lencols; j++) {
//            System.out.println("(" + i + "; " + j +
//");");
            int val = 0;
            for (Node cur : list) {
//                System.out.println("\tit1: " + cur);
                if (cur.row == i) {
                    for (Node cur_2 : other.list) {
//                        System.out.println("\tit2: " +
//cur_2);
                        if (cur_2.col == j && cur_2.row ==
cur.col) {
                            val += cur.value *
cur_2.value;
//                            System.out.println(" [" + i
+ " " + j + "] = " + cur.value * cur_2.value);
                            break;
                        }
                    }
                }
            }
            if (val > 0) {
                result.add(new Node(i, j, val));
            }
        }
    }

//    ListIterator<Node> it2 = other.list.listIterator();
//    result.add(node2);
    return new SparseMatrix(result);
}

```

```

        public void print(){
            ListIterator<Node> it1 = list.listIterator();
            while (it1.hasNext()) {
                Node cur = it1.next();
                System.out.print(cur + " ");
            }
        }
    }

class ReaderFile{
    public int[][] readFile(String filename){

        try {
            Scanner scanner = new Scanner(new File(filename));
            String sizes = "";
            if (scanner.hasNextLine()) {
                sizes = scanner.nextLine();
            }
            int nrows = Integer.parseInt(sizes.split(" ")[0]);
            int ncols = Integer.parseInt(sizes.split(" ")[1]);
            int[][] arr = new int[nrows][ncols];
            int i = 0;
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                for (int j = 0; j < ncols; j++) {
                    arr[i][j] = Integer.parseInt(line.split("
")[j]);
                }
                i++;
            }

            System.out.println(Arrays.deepToString(arr));
            return arr;
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        return null;
    }
}

public class Third_2 {
    public static void main(String[] args) {
        // Scanner scanner = new Scanner(System.in);
        // System.out.print("Введите длину массива n: ");
        // int length = scanner.nextInt();
        //
        // int[][] a = {{1, 0, 0}, {0, 0, 0}, {0, 1, 0}};
        // int[][] b = {{0, 1, 0}, {0, 0, 0}, {1, 0, 0}};
        ReaderFile reader = new ReaderFile();
    }
}

```



```

        SparseMatrix a = new
SparseMatrix(reader.readFile("matrix.txt"));
        SparseMatrix b = new
SparseMatrix(reader.readFile("matrix_2.txt"));
        System.out.println("list a: " + a.list);
        System.out.println("list b: " + b.list);

        SparseMatrix sum = a.add(b);
        System.out.println("sum: " + sum.list);

        SparseMatrix mul = a.mul(b);
        System.out.println("mul: " + mul.list);
    }
}

```

Работа программы показана на рисунке 3.

```

[[1, 1, 0], [0, 0, 0], [0, 1, 0]]
[[1, 0, 0], [1, 0, 0], [1, 0, 0]]
list a: [{1[0, 0]}, {1[0, 1]}, {1[2, 1]}]
list b: [{1[0, 0]}, {1[1, 0]}, {1[2, 0]}]
sum: [{2[0, 0]}, {1[0, 1]}, {1[1, 0]}, {1[2, 0]}, {1[2, 1]}]
mul: [{2[0, 0]}, {1[2, 0]}]

Process finished with exit code 0

```

Рисунок 3 – Работа программы

Задание 4: Во входном файле хранятся наименования некоторых объектов. Построить список C1, элементы которого содержат наименования и шифры данных объектов, причем элементы списка должны быть упорядочены по возрастанию шифров. Затем “сжать” список C1, удаляя дублирующие наименования объектов.

Код:

```

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.Scanner;

class HashName{
    String name;

```

```

int code;

HashMap(String name){
    this.name = name;
    this.code = this.name.hashCode();
}

@Override
public String toString() {
    return "{" + name + ", " + code + '}';
}
}

class ListNames{

    LinkedList<HashMap> names;

    public void append(String name){
        System.out.println(name);
        HashMap new_name = new HashMap(name);
        int i = 0;
        if (names.isEmpty()){
            this.names.add(new_name);
            System.out.println("\tadded");
        }
        else {
            boolean bigger_previous = true;
            for (HashMap cur : names) {
                if (new_name.code < cur.code) {
                    this.names.add(i, new_name);
                    System.out.println("\tadded");
                    return;
                }
                i++;
            }
            this.names.add(new_name);
            System.out.println("\tadded");
        }
    }

    ListNames(){
        this.names = new LinkedList<>();
    }

    ListNames(String filename){
        this.names = new LinkedList<>();
        try {
            Scanner scanner = new Scanner(new File(filename));
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                System.out.println(line);
                this.append(line);
            }
        }
    }
}

```

```

        }

//          System.out.println(names);
//          scanner.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}

public void compress(){
    ListNames compressed = new ListNames();
    HashName previous = names.getFirst();
    for (HashName cur : names) {
        if (cur.code != previous.code) {
            compressed.append(previous.name);
            previous = cur;
        }

    }
    compressed.append(names.getLast().name);
    this.names = compressed.names;
}

}

public class Fourth_2 {
    public static void main(String[] args) {
//        ListNames reader = new ListNames();
//        LinkedList<String> a = new
LinkedList<>(reader.readNames("names.txt"));
        ListNames entity = new ListNames("names.txt");
        System.out.println(entity.names);
        entity.compress();
        System.out.println(entity.names);
    }
}

```

Работа программы показана на рисунке 4.

```
moscow
added
moscow
added
saintp
added
kazan
added
moscow
added
kazan
added
paris
added
paris
added
[{"moscow", -1068354342}, {"moscow", -1068354342}, {"moscow", -1068354342}, {"saintp", -909795921}, {"kazan", 101826833}, {"kazan", 101826833}, {"paris", 106437003}, {"paris", 106437003}]
moscow
added
saintp
added
kazan
added
paris
added
[{"moscow", -1068354342}, {"saintp", -909795921}, {"kazan", 101826833}, {"paris", 106437003}]
Process finished with exit code 0
```

Рисунок 4 – Работа программы

Вывод: была изучена работа с коллекциями в java.