|  |  |
| --- | --- |
| 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 № 12**

**Название:** коллекции

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | Н.М. Кулинич |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (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 класса Main:

//3. 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.

import java.util.HashSet;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner console = new Scanner(System.in);

HashSet<Integer> array = new HashSet<Integer>();

System.out.println("Сколько будет чисел: ");

int col = console.nextInt();

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

array.add(console.nextInt());

}

int kol = -1;

while (kol != 1){

int i = 0;

int[] arr = new int[array.size()];

for (int ele:array){

arr[i++] = ele;

}

int kol\_array = array.size();

array.clear();

for (int j = 0; j <= kol\_array - 1; j += 2) {

if (j+1 == kol\_array){

array.add(arr[j]);

}else {

array.add(arr[j] + arr[j+1]);

}

}

System.out.println(array);

kol = array.size();

}

}

}

Работа программы показана на рисунке 1.

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 1 – Работа программы

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

Код класса Main:

// Сложить два многочлена заданной степени, если коэффициенты многочленов хранятся в объекте HashMap.

import java.util.HashMap;

import java.util.Map;

public class Main {

public static void main(String[] args) {

// Создание двух многочленов в виде HashMap

HashMap<Integer, Integer> polynomial1 = new HashMap<>();

polynomial1.put(0, 2); // 2

polynomial1.put(1, 3); // 3x

polynomial1.put(2, 5); // 5x^2

HashMap<Integer, Integer> polynomial2 = new HashMap<>();

polynomial2.put(0, 1); // 1

polynomial2.put(1, -3); // -3x

polynomial2.put(3, 4); // 4x^3

// Выводим исходные многочлены

System.out.println("Polynomial 1: " + polynomialToString(polynomial1));

System.out.println("Polynomial 2: " + polynomialToString(polynomial2));

// Складываем два многочлена

HashMap<Integer, Integer> result = addPolynomials(polynomial1, polynomial2);

// Выводим результирующий многочлен

System.out.println("Resulting Polynomial: " + polynomialToString(result));

}

// Метод для сложения двух многочленов

public static HashMap<Integer, Integer> addPolynomials(HashMap<Integer, Integer> poly1, HashMap<Integer, Integer> poly2) {

HashMap<Integer, Integer> result = new HashMap<>(poly1);

for (Map.Entry<Integer, Integer> entry : poly2.entrySet()) {

result.merge(entry.getKey(), entry.getValue(), Integer::sum);

}

return result;

}

// Метод для представления многочлена в виде строки

public static String polynomialToString(HashMap<Integer, Integer> polynomial) {

StringBuilder result = new StringBuilder();

boolean firstTerm = true;

for (int i = polynomial.size() - 1; i >= 0; i--) {

if (polynomial.containsKey(i)) {

int coefficient = polynomial.get(i);

if (coefficient != 0) {

if (!firstTerm && coefficient > 0) {

result.append("+");

}

if (i == 0) {

result.append(coefficient);

} else if (i == 1) {

result.append(coefficient).append("x");

} else {

result.append(coefficient).append("x^").append(i);

}

firstTerm = false;

}

}

}

return result.toString();

}

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 2 – Работа программы

Работа программы показана на рисунке 2.

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

Код класса Main:

import java.io.FileReader;

import java.io.IOException;

import java.util.HashMap;

public class Main {

public static void main(String[] args) {

HashMap<Integer, Integer> CA = new HashMap<>();

HashMap<Integer, Integer> CB = new HashMap<>();

char[] matrix = new char[54];

int[] matrix1 = new int[9];

int[] matrix2 = new int[9];

int kol = 0;

try (FileReader reader = new FileReader("C:/Users/User/IdeaProjects/jawa\_lab/Matrix.txt")) {

int c;

while ((c = reader.read()) != -1) {

matrix[kol] = (char) c;

kol++;

}

} catch (IOException e) {

e.printStackTrace();

}

// Обработка строки для удаления пробелов и новых строк

String data = new String(matrix).trim();

String[] tokens = data.split("\\s+");

// Проверка, что количество токенов соответствует ожиданиям

if (tokens.length != 18) {

System.out.println("Ошибка: Некорректный формат входного файла.");

return;

}

// Заполнение matrix1 и matrix2 из токенов

for (int i = 0; i < 9; i++) {

matrix1[i] = Integer.parseInt(tokens[i]);

matrix2[i] = Integer.parseInt(tokens[i + 9]);

}

// Построение циклически связанных списков

createCyclicList(CA, matrix1);

createCyclicList(CB, matrix2);

System.out.println("CA: " + CA);

System.out.println("CB: " + CB);

// Вычисление суммы и произведения матриц

int[] matrix\_S = new int[9];

int[] matrix\_P = new int[9];

System.out.println("Сумма матриц: ");

for (int i = 0; i < 9; i++) {

matrix\_S[i] = matrix1[i] + matrix2[i];

System.out.print(matrix\_S[i] + " ");

if ((i + 1) % 3 == 0) {

System.out.println();

}

}

System.out.println("Умножение матриц: ");

for (int i = 0; i < 9; i++) {

matrix\_P[i] = matrix1[i] \* matrix2[i];

System.out.print(matrix\_P[i] + " ");

if ((i + 1) % 3 == 0) {

System.out.println();

}

}

}

// Метод для построения циклически связанного списка

private static void createCyclicList(HashMap<Integer, Integer> map, int[] matrix) {

int first = 0;

boolean first\_check = false;

int t = 0;

for (int i = 0; i < matrix.length; i++) {

if (matrix[i] > 0) {

map.put(matrix[i], t);

t = matrix[i];

if (!first\_check) {

first = matrix[i];

first\_check = true;

}

}

}

map.replace(first, t);

}

}

Работа программы показана на рисунке 3.

Изображение выглядит как текст, Мультимедийное программное обеспечение, снимок экрана

Автоматически созданное описание

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 класса Main:

import java.io.FileReader;

import java.io.IOException;

import java.util.\*;

public class Main {

public static void main(String[] args) {

String[] objects = new String[9];

HashMap<Integer, String> C1 = new HashMap<>();

int kol = 0;

try(FileReader reader = new FileReader("C:/Users/User/IdeaProjects/jawa\_lab/Objects.txt"))

{

int c;

while((c=reader.read())!=-1){

objects[kol] = Character.toString((char)c);

kol++;

}

} catch (IOException e) {

}

int j = 3;

for (int i = 0; i < 9; i += 3) {

C1.put(j, objects[i]);

j--;

}

TreeMap<Integer, String> sorted\_C1 = new TreeMap<>();

sorted\_C1.putAll(C1);

System.out.println(C1);

Iterator<Integer> keyIter = C1.keySet().iterator();

HashMap<String, Integer> C2 = new HashMap<>();

while (keyIter.hasNext()) {

Integer key = keyIter.next();

String value = C1.get(key);

C2.put(value, key);

}

Iterator<Integer> keyIter2 = C2.values().iterator();

HashMap<Integer, String> C3 = new HashMap<>();

while (keyIter2.hasNext()) {

Integer key = keyIter2.next();

String value = C1.get(key);

C3.put(key, value);

}

System.out.println(C3);

}

}

Работа программы показана на рисунке 4.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 4 – Работа программы

**Вывод:** были изучена работа с исключениями и файлами в java.