Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе № 5

тема «Файлы в Java»

по дисциплине «Информатика»

Выполнил: студент группы ИСТ-22-1б Михайлов Александр

Проверил: ассистент каф. ВММБ Нетбай Г.В.

Пермь, 2023

Оглавление

Оглавление

[Вариант 14 3](#_Toc136719547)

[Задание 1 3](#_Toc136719548)

[Постановка задачи 3](#_Toc136719549)

[Решение задачи 3](#_Toc136719550)

[Проверка результата 5](#_Toc136719551)

[Задание 2 6](#_Toc136719552)

[Постановка задачи 6](#_Toc136719553)

[Решение задачи 6](#_Toc136719554)

[Проверка результата 6](#_Toc136719555)

[Задание 3 7](#_Toc136719556)

[Постановка задачи 7](#_Toc136719557)

[Решение задачи 7](#_Toc136719558)

[Проверка результата 7](#_Toc136719559)

[Задание 4 8](#_Toc136719560)

[Постановка задачи 8](#_Toc136719561)

[Решение задачи 8](#_Toc136719562)

[Проверка результата 9](#_Toc136719563)

[Задание 5 10](#_Toc136719564)

[Постановка задачи 10](#_Toc136719565)

[Решение задачи 10](#_Toc136719566)

[Проверка результата 11](#_Toc136719567)

[Задание 6 12](#_Toc136719568)

[Постановка задачи 12](#_Toc136719569)

[Решение задачи 12](#_Toc136719570)

[Проверка результата 13](#_Toc136719571)

[Задание 7 14](#_Toc136719572)

[Постановка задачи 14](#_Toc136719573)

[Решение задачи 14](#_Toc136719574)

[Проверка результата 15](#_Toc136719575)

[Задание 8 16](#_Toc136719576)

[Постановка задачи 16](#_Toc136719577)

[Решение задачи 16](#_Toc136719578)

[Проверка результата 20](#_Toc136719579)

[Задание 9 21](#_Toc136719580)

[Постановка задачи 21](#_Toc136719581)

[Решение задачи 21](#_Toc136719582)

[Проверка результата 24](#_Toc136719583)

[Задание 10 25](#_Toc136719584)

[Постановка задачи 25](#_Toc136719585)

[Решение задачи 25](#_Toc136719586)

[Проверка результата 26](#_Toc136719587)

[Задание 11 27](#_Toc136719588)

[Постановка задачи 27](#_Toc136719589)

[Решение задачи 27](#_Toc136719590)

[Проверка результата 28](#_Toc136719591)

[Задание 12 29](#_Toc136719592)

[Постановка задачи 29](#_Toc136719593)

[Решение задачи 29](#_Toc136719594)

[Проверка результата 30](#_Toc136719595)

[Задание 13 31](#_Toc136719596)

[Постановка задачи 31](#_Toc136719597)

[Решение задачи 31](#_Toc136719598)

[Проверка результата 33](#_Toc136719599)

[Задание 14 34](#_Toc136719600)

[Постановка задачи 34](#_Toc136719601)

[Решение задачи 34](#_Toc136719602)

[Проверка результата 36](#_Toc136719603)

[Задание 15 37](#_Toc136719604)

[Постановка задачи 37](#_Toc136719605)

[Решение задачи 37](#_Toc136719606)

[Проверка результата 41](#_Toc136719607)

# Вариант 14

# Задание 1

## Постановка задачи

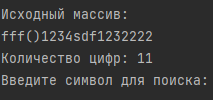
Переделать программу задания 3 лабораторной работы 9. Массив данных записан в файл и считывается в программе для обработки. Результаты обработки массива согласно вопросам задания 3 лабораторной работы 9 записываются в конец файла построчно с заголовками расшифровывающими, что за данные представлены ниже.

## Решение задачи

Код программы:

import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.Scanner;  
import java.io.FileWriter;  
  
public class Main {  
 private static boolean containsChar(char[] array, char ch) {  
 for (char element : array) {  
 if (element == ch) {  
 return true;  
 }  
 }  
 return false;  
 }  
 public static void print\_massive(char[] massive){  
 String stroka = "";  
 for (int i = 0;i<massive.length;i++){  
 stroka = stroka + Character.*toString*(massive[i]);  
 }  
 System.*out*.println(stroka);  
 }  
 public static void main(String[] args){  
 String line;  
 File file = new File("C:\\Users\\mikha\\Downloads\\Telegram Desktop\\task\_1");  
  
 try (BufferedReader br = new BufferedReader(new FileReader(file)))  
 {  
 line = br.readLine();  
 } catch (IOException e) {  
 e.printStackTrace();  
 line = "";  
 }  
 int len = line.split(",").length;  
 char[] characters = new char[len];  
 characters = line.toCharArray();  
 boolean increasing\_sequence = false;  
 boolean decreasing\_sequence = false;  
 System.*out*.println("Исходный массив:");  
 *print\_massive*(characters);  
  
 int digitCount = 0;  
 for (char ch : characters) {  
 if (Character.*isDigit*(ch)) {  
 digitCount++;  
 }  
 }  
 int[] massive\_number = new int[digitCount];  
 System.*out*.println("Количество цифр: " + digitCount);  
 int g = 0;  
 for (char ch: characters) {  
 if (Character.*isDigit*(ch)){  
 massive\_number[g] = ch - 48;  
 g=g+1;  
 }  
 }  
 for (int i=0;i<massive\_number.length-3;i++){  
 for (int k = i+1;k<massive\_number.length -2 ; k++){  
 for (int j=k+1;j<massive\_number.length-1;j++){  
 if (massive\_number[i] < massive\_number[i+1]){  
 increasing\_sequence = true;  
 }  
 if (massive\_number[j] > massive\_number[j+1]) {  
 decreasing\_sequence = true;  
 }  
  
 }  
 }  
 }  
  
  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Введите символ для поиска: ");  
 char searchChar = scanner.nextLine().charAt(0);  
 int searchCount = 0;  
 for (char ch : characters) {  
 if (ch == searchChar) {  
 searchCount++;  
 }  
 }  
 System.*out*.println("Количество символов поиска: " + searchCount);  
  
  
 System.*out*.print("Введите номер для поиска: ");  
 String searchNumber = scanner.nextLine();  
 boolean allDigitsFound = true;  
 for (char digit : searchNumber.toCharArray()) {  
 if (!*containsChar*(characters, digit)) {  
 allDigitsFound = false;  
 break;  
 }  
 }  
 System.*out*.println("Все найденные цифры: " + allDigitsFound);  
 System.*out*.println("Возрастающая последовательность:" + increasing\_sequence);  
 System.*out*.println("Убывающая последовательность:" + decreasing\_sequence);  
  
  
 String brackets = "(){}[]";  
 boolean closedPairsFound = false;  
 for (int i = 0; i < characters.length - 1; i++) {  
 String pair = "" + characters[i] + characters[i+1];  
 if (brackets.contains(pair)) {  
 if (pair.equals("()") || pair.equals("{}") || pair.equals("[]")) {  
 closedPairsFound = true;  
 }  
 }  
 }  
 System.*out*.println("Найдены замкнутые пары различных комбинаций скобок: " + closedPairsFound);  
  
  
 String punctuation = ".,:;!?";  
 boolean adjacentIdenticalFound = false;  
 for (int i = 0; i < characters.length - 1; i++) {  
 if (punctuation.contains("" + characters[i]) && punctuation.contains("" + characters[i+1])) {  
 if (characters[i] == characters[i+1]) {  
 adjacentIdenticalFound = true;  
 }  
 }  
 }  
 System.*out*.println("Соседние идентичные символы, относящиеся к найденным знакам препинания: " + adjacentIdenticalFound);  
 try {  
 FileWriter writer = new FileWriter("C:\\Users\\mikha\\Downloads\\Telegram Desktop\\task\_1",true);  
 writer.write("\n" +"Все найденные цифры: " + allDigitsFound + "\n");  
 writer.write("Возрастающая последовательность:" + increasing\_sequence+ "\n");  
 writer.write("Убывающая последовательность:" + decreasing\_sequence+ "\n");  
 writer.write("Найдены замкнутые пары различных комбинаций скобок: " + closedPairsFound+ "\n");  
 writer.write("Соседние идентичные символы, относящиеся к найденным знакам препинания: " + adjacentIdenticalFound+ "\n");  
 writer.close();  
 } catch (IOException e) {  
 System.*out*.println("An error occurred.");  
 e.printStackTrace();  
 }  
 }  
}

## Проверка результата



# Задание 2

## Постановка задачи

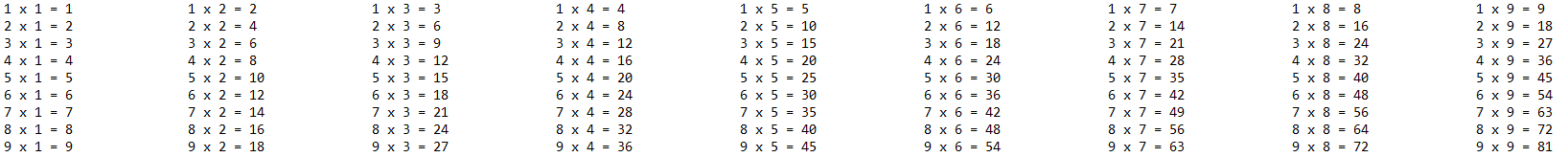
Записать в файл таблицу из задания 7 лабораторной работы 6. Если в таблице присутствуют дробные ответы, то вывести данные в формате 4 знака после запятой.

## Решение задачи

Код программы:

import java.io.FileWriter;  
import java.io.IOException;  
  
public class Main {  
 public static void main(String[] args) {  
 try {  
 FileWriter file = new FileWriter("C:\\\\Users\\\\mikha\\\\Downloads\\\\Telegram Desktop\\\\task\_2",true);  
 int[] pif = {1, 2, 3, 4, 5, 6, 7, 8, 9};  
 for (int q : pif) {  
 for (int w : pif) {  
 file.write(q + " x " + w + " = " + q \* w + " \t" + "\t");  
 }  
 file.write("\n");  
 }  
 file.close();  
 }  
 catch (IOException e){  
 System.*out*.println("opa-na.");  
 e.printStackTrace();  
 }  
 }  
}

## Проверка результата



# Задание 3

## Постановка задачи

Напечатать все слова из текстового файла, содержащее сочетание двух соседних букв латинского алфавита (например сочетания «ab», «bc» и т.д.)

## Решение задачи

Код программы:

import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
  
public class Main {  
 public static void main(String[] args){  
 String word = " ";  
 try (FileReader file = new FileReader("C:\\Users\\mikha\\Downloads\\Telegram Desktop\\task\_3")){  
 BufferedReader in = new BufferedReader(file);  
 word = in.readLine();  
  
 for (String line:word.split(" ")) {  
 char[] Syblols = line.toCharArray();  
 for (int i = 0; i < Syblols.length - 1; i++) {  
 if (Math.*abs*(Character.*getNumericValue*(Syblols[i]) - Character.*getNumericValue*(Syblols[i + 1])) == 1) {  
  
 }  
 }  
 }  
 }  
 catch (IOException e){  
 System.*out*.println("opana, chto-to ne tak ((");  
 e.printStackTrace();  
 }  
 }  
}

## Проверка результата

# Задание 4

## Постановка задачи

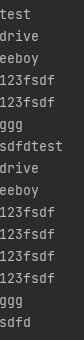
Выделить в текстовом файле все слова, разделенные символами-разделителями «\_.,;:\n\t!?», и все слова, которые состоят после цифр.

## Решение задачи

Код программы:

import java.io.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 String fileName = "C:\\\\Users\\\\mikha\\\\Downloads\\\\Telegram Desktop\\\\task\_4";  
 String[] separators = {" ", "\_", ".", ",", ";", ":", "\n", "\t", "!", "?"};  
  
 try (BufferedReader lines = new BufferedReader(new FileReader(fileName))) {  
 BufferedWriter file = new BufferedWriter(new FileWriter(fileName,true));  
 String line;  
 while ((line = lines.readLine()) != null) {  
 String[] words = line.split("[\\s\_.,;:\\n\\t!?]+");  
 for (String word : words) {  
 System.*out*.println(word);  
 file.write(word + "\n");  
 if (word.matches("\\d+\\p{Alpha}+")) {  
 System.*out*.println(word);  
 file.write(word + "\n");  
 }  
 }  
 }  
 file.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

## Проверка результата



# Задание 5

## Постановка задачи

Дано два файла с предложениями. Дописать в конец второго файла предложения из первого файла, которые включают все буквы русского алфавита.

## Решение задачи

Код программы:

import java.io.\*;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class Main {  
 public static void main(String[] args) {  
 String filename1 = "C:\\\\\\\\Users\\\\\\\\mikha\\\\\\\\Downloads\\\\\\\\Telegram Desktop\\\\\\task\_5\_1";  
 String filename2 = "C:\\\\\\\\Users\\\\\\\\mikha\\\\\\\\Downloads\\\\\\\\Telegram Desktop\\\\\\task\_5\_2";  
  
 try {  
 BufferedReader reader1 = new BufferedReader(new FileReader(filename1));  
 BufferedReader reader2 = new BufferedReader(new FileReader(filename2));  
 BufferedWriter writer2 = new BufferedWriter(new FileWriter(filename2, true));  
 String line;  
 String regex\_rus = "[а-яёА-ЯЁ]+";  
 String regex\_eng = "[a-zA-Z]+\\.?";  
 Pattern pattern\_rus = Pattern.*compile*(regex\_rus);  
 Pattern pattern\_eng = Pattern.*compile*(regex\_eng);  
  
 // Флаг для отслеживания, включены ли все буквы алфавита в предложении  
 boolean hasAllLetters = true;  
 boolean hasAllLetterRus = true;  
  
 // Алфавит  
 String alphabet = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя,?";  
  
 while ((line = reader1.readLine()) != null) {  
 Matcher m\_rus = pattern\_rus.matcher(line);  
 Matcher m\_eng = pattern\_eng.matcher(line);  
 hasAllLetterRus = false;  
 if (m\_eng.find()){  
 hasAllLetters = false;  
 }  
 else {  
 if (m\_rus.find()){  
 hasAllLetterRus = true;  
 }  
 }  
  
 // Если предложение содержит все буквы алфавита, добавляем его во второй файл  
 if (hasAllLetters&&hasAllLetterRus) {  
 writer2.newLine();  
 writer2.write(line);  
 }  
  
 // Сбрасываем флаг  
 hasAllLetters = true;  
 }  
  
 // Закрываем все потоки  
 reader1.close();  
 reader2.close();  
 writer2.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

## Проверка результата



# Задание 6

## Постановка задачи

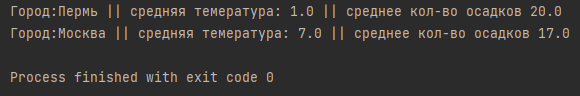
Входной файл содержит сведения с метеорологическими данными городов: название города, месяц, средняя температура в месяце, среднее количество осадков в месяце. Подсчитать для каждого города среднюю температуру и количество осадков в декаду (каждые 3 месяца: январь, февраль, март), дописать ответ в конец файла построчно.

## Решение задачи

Код программы:

import java.io.\*;  
import java.util.\*;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
import java.io.BufferedReader;  
import java.io.FileReader;  
  
public class Main {  
 private static boolean check(String[] arr, String toCheckValue)  
 {  
 for (String element : arr) {  
 if (element.equals(toCheckValue)) {  
 return true;  
 }  
 }  
 return false;  
  
 }  
 public static void main(String[] args){  
 String line;  
 Map<String, ArrayList<City>> information = new HashMap<>();  
 String[] massive\_month = {"январь","февраль","март"};  
 try {  
 String filename1 = "C:\\\\\\\\\\\\\\\\Users\\\\\\\\\\\\\\\\mikha\\\\\\\\\\\\\\\\Downloads\\\\\\\\\\\\\\\\Telegram Desktop\\\\\\\\\\\\task\_6";  
 BufferedReader reader1 = new BufferedReader(new FileReader(filename1));  
 while ((line = reader1.readLine()) != null){  
 String[] info = line.split(" ");  
 String name\_city = info[0];  
 String month = info[1];  
 int temperature = Integer.*parseInt*(info[2]);  
 int rainfall = Integer.*parseInt*(info[3]);  
 ArrayList<City> New\_city = information.get(name\_city); // null  
  
 City city = new City(month,temperature,rainfall);  
 try{  
 New\_city = information.get(info[0]);  
 New\_city.add(city);  
 information.put(info[0],New\_city);  
 } catch (Exception exception){  
 New\_city = new ArrayList<City>();  
 New\_city.add(city);  
 information.put(info[0],New\_city);  
 }  
  
 }  
 for (String name\_city:information.keySet()){ //Пробегаемся по городам  
// System.out.println("Город:" + name\_city + " Значение:" + information.get(name\_city));  
 ArrayList<City> list = information.get(name\_city);  
 int count = 0;  
 int temperature = 0;  
 int rainfall = 0;  
 for (int i = 0;i<list.size();i++){ //Пробегаемся по инофрмации о городах  
 if (*check*(massive\_month,list.get(i).toString().split(" ")[0])){ //проверка на нужные месяца  
 temperature = temperature + Integer.*parseInt*(list.get(i).toString().split(" ")[1]);  
 rainfall = rainfall + Integer.*parseInt*(list.get(i).toString().split(" ")[2]);  
 count = count + 1;  
 }  
  
 }  
 double mid\_temp = temperature / count;  
 double mid\_rainfall = rainfall/count;  
 System.*out*.println("Город:" + name\_city + " || средняя темература: " + mid\_temp + " || среднее кол-во осадков " + mid\_rainfall );  
 }  
 }  
 catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
}  
class City{  
 public String toString() {  
 return name + " " + temperature + " " + rainfall;  
 }  
 public City(String Name, int temp, int rain) {  
 this.name = Name;  
 this.temperature = temp;  
 this.rainfall = rain;  
 }  
 String name;  
 int temperature;  
 int rainfall;  
  
 public void setName(String name) {  
 this.name = name;  
 }  
}

## Проверка результата



# Задание 7

## Постановка задачи

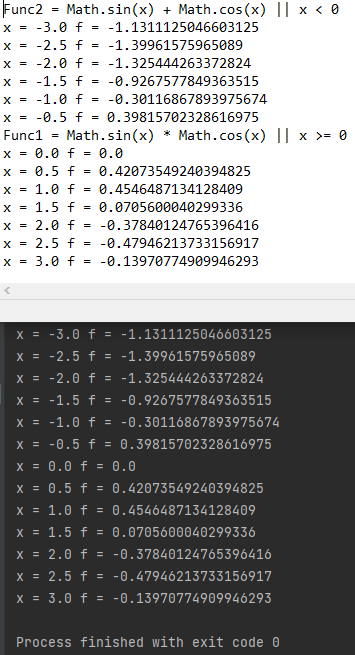
Записать в файл значения функции f(x) из задания 3 лабораторной работы 5 для всего диапазона переменной в виде: значение x; значение f(x). В начале файла должна быть указана постановка задачи, т.е. диапазон аргумента и вид функции (запись на языке java). Перед выводом цифровых значений должен быть сделан заголовок о том, в каком порядке идет вывод.

## Решение задачи

Код программы:

import java.io.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 String line = "x";  
 boolean name\_func\_1 = true;  
 boolean name\_func\_2 = true;  
 try (BufferedWriter file = new BufferedWriter(new FileWriter("C:\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\Users\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\mikha\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\Downloads\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\Telegram Desktop\\\\\\\\\\\\\\\\\\\\\\\\task\_7",true))) {  
 float x = -3;  
 while (x != 3.5) {  
 if (x >= 0) {  
 if (name\_func\_1){  
 file.write("Func1 = Math.sin(x) \* Math.cos(x) || x >= 0" + "\n");  
 name\_func\_1 = false;  
 }  
 double f = Math.*sin*(x) \* Math.*cos*(x);  
 System.*out*.println("x = " + x + " f = " + f);  
 line = "x = " + x + " f = " + f;  
 file.write(line + "\n");  
 } else {  
 if (name\_func\_2){  
 file.write("Func2 = Math.sin(x) + Math.cos(x) || x < 0" + "\n");  
 name\_func\_2 = false;  
 }  
 double f1 = Math.*sin*(x) + Math.*cos*(x);  
 System.*out*.println("x = " + x + " f = " + f1);  
 line = "x = " + x + " f = " + f1;  
 file.write(line + "\n");  
 }  
 x = x + 0.5f;  
 }  
 }  
 catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
}

## Проверка результата



# Задание 8

## Постановка задачи

Составить файл постановки и решения задачи для попадания точки в область лабораторная работа 5 задание 5.

Структура файла: постановка задачи; массив точек для проверки; стилизованный ответ.

Постановка задачи (для каждой фигуры, с заголовком о номере фигуры): уравнение кривой с порядковым номером, отрезок на котором она определена.

Массив точек для проверки: номер точки, координата x; координата y.

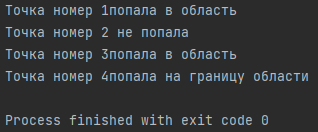
Стилизованный ответ: номер точки, ответ о попадании точки в область, на границу или о том, что точка не попала.

## Решение задачи

Код программы:

import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.FileReader;  
import java.io.FileWriter;  
  
public class Main {  
 public static String *path* = "C:\\\\Users\\\\mikha\\\\Downloads\\\\Telegram Desktop\\\\task\_8";  
  
 public static double k(double x1, double y1, double x2, double y2) {  
 return (y1 - y2) / (x1 - x2);  
 }  
  
 public static double b(double x1, double y1, double x2, double y2) {  
 return y1 - x1 \* *k*(x1, y1, x2, y2);  
 }  
  
 public static class FigureOne {  
 public static int figureOne(double x, double y) {  
 if (y > *k*(-4, -6, -6, -4) \* x + *b*(-4, -6, -6, -4) && y < *k*(-6, -4, -4, -2) \* x + *b*(-6, -4, -4, -2) && x <= -4) {  
 return 1;  
 }  
 if ((y == *k*(-4, -6, -6, -4) \* x + *b*(-4, -6, -6, -4) || y == *k*(-6, -4, -4, -2) \* x + *b*(-6, -4, -4, -2)) && x <= -4 && x >= -6) {  
 return 0;  
 }  
 return -1;  
 }  
  
 public static int figureTwo(double x, double y) {  
 if (x >= -4 && y <= -2 && y > *k*(-4, -6, -2, -2) \* x + *b*(-4, -6, -2, -2)) {  
 return 1;  
 }  
 if (x >= -4 && y <= -2 && y == *k*(-4, -6, -2, -2) \* x + *b*(-4, -6, -2, -2)) {  
 return 0;  
 }  
 return -1;  
 }  
  
 public static int figureThree(double x, double y) {  
 if (y < *k*(-4, -2, -2, 0) \* x + *b*(-4, -2, -2, 0) && x <= -2 && y >= -2) {  
 return 1;  
 }  
 if (y == *k*(-4, -2, -2, 0) \* x + *b*(-4, -2, -2, 0) && x <= -2 && x >= -4) {  
 return 0;  
 }  
 return -1;  
 }  
  
 public static int figureFour(double x, double y) {  
 if (y > *k*(0, 1, -2, -2) \* x + *b*(0, 1, -2, -2) && x >= -2 && y < *k*(-2, 2, 0, 1) \* x + *b*(-2, 2, 0, 1)) {  
 return 1;  
 }  
 if ((y == *k*(0, 1, -2, -2) \* x + *b*(0, 1, -2, -2) || y == *k*(-2, 2, 0, 1) \* x + *b*(-2, 2, 0, 1)) && x >= -2 && x <= 0) {  
 return 0;  
 }  
 return -1;  
 }  
  
 public static int figureFive(double x, double y) {  
 if (y > *k*(-2, 0, -6, 4) \* x + *b*(-2, 0, -6, 4) && x <= -2 && y < *k*(-6, 4, -2, 2) \* x + *b*(-6, 4, -2, 2)) {  
 return 1;  
 }  
 if (y == *k*(-2, 0, -6, 4) \* x + *b*(-2, 0, -6, 4) && x >= -6 && x <= -2) {  
 return 0;  
 }  
 return -1;  
 }  
  
 public static int figureSix(double x, double y) {  
 if (y > *k*(-2, 2, -6, 4) \* x + *b*(-2, 2, -6, 4) && y <= 4 && y < *k*(-3, 4, -2, 2) \* x + *b*(-3, 4, -2, 2)) {  
 return 1;  
 }  
 if (y <= *k*(-3, 4, -2, 2) \* x + *b*(-3, 4, -2, 2) && x < -2 && x >= -3) {  
 return 0;  
  
 }  
 return -1;  
 }  
  
 public static int figureSeven(double x, double y) {  
 if (y > *k*(-6, 4, -7, 5) \* x + *b*(-6, 4, -7, 5) && y >= 4 && y < *k*(-7, 5, -3, 4) \* x + *b*(-7, 5, -3, 4)) {  
 return 1;  
 }  
 if ((y == *k*(-6, 4, -7, 5) \* x + *b*(-6, 4, -7, 5) || y == *k*(-7, 5, -3, 4) \* x + *b*(-7, 5, -3, 4) && x <= -3 && x >= -7)) {  
 return 0;  
 }  
 return -1;  
 }  
 }  
 public static class FigureTwo {  
 public static int figureOne(double x, double y) {  
 if (y > *k*(0, -6, -1, -2) \* x + *b*(0, -6, -1, -2) && y <= -2 && y > *k*(0, -6, 1, -2) \* x + *b*(0, -6, 1, -2)) {  
 return 1;  
 }  
 if (y == *k*(0, -6, -1, -2) \* x + *b*(0, -6, -1, -2) && x >= -1 && x<=0){  
 return 0;  
 }  
 return -1 ;  
 }  
  
 public static int figureTwo(double x, double y) {  
 if (y <= *k*(0, -6, 1, -2) \* x + *b*(0, -6, 1, -2) && x <= 1 && y >= *k*(1, -7, 0, -6) \* x + *b*(1, -7, 0, -6)) {  
 return 1;  
 }  
 if (x>=0 && x <= 1 && y == *k*(1, -7, 0, -6) \* x + *b*(1, -7, 0, -6)){  
 return 0;  
 }  
 return -1;  
 }  
  
 public static int figureThree(double x, double y) {  
 if (y <= *k*(1, -2, 5, 2) \* x + *b*(1, -2, 5, 2) && x >= 1 && y >= *k*(1, -7, 5, 2) \* x + *b*(1, -7, 5, 2)) {  
 return 1;  
 }  
 return -1;  
 }  
  
 public static int figureFour(double x, double y) {  
 if (y <= *k*(1, -7, 5, 2) \* x + *b*(1, -7, 5, 2) && y >= *k*(5, 2, 4, -6) \* x + *b*(5, 2, 4, -6) && y > *k*(4, -6, 1, -7) \* x + *b*(4, -6, 1, -7)) {  
 return 1;  
 }  
 if (y == *k*(4, -6, 1, -7) \* x + *b*(4, -6, 1, -7) && x>=1 && x<=4){  
 return 0;  
 }  
 return -1;  
 }  
  
 public static int figureFive(double x, double y) {  
 if (y <= *k*(4, -6, 5, 2) \* x + *b*(4, -6, 5, 2) && y <= *k*(5, 2, 6, -3) \* x + *b*(5, 2, 6, -3) && y > *k*(4, -6, 6, -3) \* x + *b*(4, -6, 6, -3)) {  
 return 1;  
 }  
 if (y == *k*(4, -6, 6, -3) \* x + *b*(4, -6, 6, -3) && x>=4 && x<=6){  
 return 0;  
 }  
 return -1;  
 }  
  
 public static int figureSix(double x, double y) {  
 if (y < *k*(5, 2, 7, 4) \* x + *b*(5, 2, 7, 4) && y > *k*(7, 4, 6, -3) \* x + *b*(7,4,6,-3) && y >= *k*(6,-3,5,2) \* x + *b*(6,-3,5,2)) {  
 return 1;  
 }  
 if ((y == *k*(5, 2, 7, 4) \* x + *b*(5, 2, 7, 4) && y == *k*(7, 4, 6, -3) \* x + *b*(7,4,6,-3)) && x>=5 && x<=7) {  
 return 0;  
 }  
 return -1;  
 }  
  
 public static int figureSeven(double x, double y) {  
 if (y >= *k*(1, -2, 5, 2) \* x + *b*(1, -2, 5, 2) && y < *k*(1, -2, 4, 4) \* x + *b*(1, -2, 4, 4) && y < *k*(4, 4, 5, 2) \* x + *b*(4, 4, 5, 2)) {  
 return 1;  
 }  
 if ((y == *k*(1, -2, 4, 4) \* x + *b*(1, -2, 4, 4) || y == *k*(4, 4, 5, 2) \* x + *b*(4, 4, 5, 2)) && x>=1 && x<=4){  
 return 0;  
 }  
 return -1;  
 }  
  
 }  
 public static void main(String[] args){  
 try {  
 BufferedReader file\_r = new BufferedReader(new FileReader(*path*));  
 BufferedWriter file\_w = new BufferedWriter(new FileWriter(*path*,true));  
 String line;  
 file\_w.write('\n' + "Стилиризованный ответ:" + '\n');  
 while ((line = file\_r.readLine()) != null) {  
 if (line.startsWith("Массив точек:")) {  
 // Извлекаем строку с массивом точек  
 String pointsLine = line.substring(line.indexOf(":") + 1).trim();  
  
 // Разбиваем строку на отдельные точки  
 String[] points = pointsLine.split(" ");  
  
 // Выводим каждую точку в требуемом формате  
 for (String point : points) {  
 String[] coordinates = point.split(",");  
 int pointNumber = Integer.*parseInt*(coordinates[0].trim());  
 double x = Double.*parseDouble*(coordinates[1].trim());  
 double y = Double.*parseDouble*(coordinates[2].trim());  
 if (FigureOne.*figureOne*(x,y) == 1 || FigureOne.*figureTwo*(x,y) == 1 || FigureOne.*figureThree*(x,y) == 1 ||  
 FigureOne.*figureFour*(x,y) == 1 || FigureOne.*figureFive*(x,y) == 1 || FigureOne.*figureSix*(x,y) == 1 || FigureOne.*figureSeven*(x,y) == 1){  
 System.*out*.println("Точка номер " + pointNumber + "попала в область");  
 file\_w.write("Точка номер " + pointNumber + " попала в 1 область" + '\n');  
 continue;  
 }  
 else if (FigureOne.*figureOne*(x,y) == 0 || FigureOne.*figureTwo*(x,y) == 0 || FigureOne.*figureThree*(x,y) == 0 ||  
 FigureOne.*figureFour*(x,y) == 0 || FigureOne.*figureFive*(x,y) == 0 || FigureOne.*figureSix*(x,y) == 0 || FigureOne.*figureSeven*(x,y) == 0) {  
 System.*out*.println("Точка номер " + pointNumber + "попала на границу области");  
 file\_w.write("Точка номер " + pointNumber + " попала на границу в 1 область" + '\n');  
 continue;  
 }  
 if (FigureTwo.*figureOne*(x,y) == 1 || FigureTwo.*figureTwo*(x,y) == 1 || FigureTwo.*figureThree*(x,y) == 1 || FigureTwo.*figureFour*(x,y) == 1 ||  
 FigureTwo.*figureFive*(x,y) == 1 || FigureTwo.*figureSix*(x,y) == 1 || FigureTwo.*figureSeven*(x,y) == 1){  
 System.*out*.println("Точка номер " + pointNumber + "попала в область");  
 file\_w.write("Точка номер " + pointNumber + " попала в 2 область" + '\n');  
 continue;  
 } else if (FigureTwo.*figureOne*(x,y) == 0 || FigureTwo.*figureTwo*(x,y) == 0 || FigureTwo.*figureThree*(x,y) == 0 || FigureTwo.*figureFour*(x,y) == 0 ||  
 FigureTwo.*figureFive*(x,y) == 0 || FigureTwo.*figureSix*(x,y) == 0 || FigureTwo.*figureSeven*(x,y) == 0) {  
 System.*out*.println("Точка номер " + pointNumber + " попала на границу области");  
 file\_w.write("Точка номер " + pointNumber + " попала на границу в 2 область" + '\n');  
 continue;  
 }  
 else{  
 System.*out*.println("Точка номер " + pointNumber + " не попала");  
 file\_w.write("Точка номер " + pointNumber + " не попала в область" + '\n');  
 }  
 }  
 }  
 }  
 file\_w.close();  
 }  
  
 catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
}

## Проверка результата



# Задание 9

## Постановка задачи

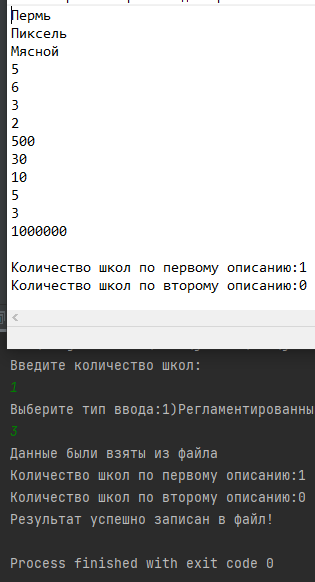
Создать файл с данными по структурированному вводу данных согласно постановке задачи 6 лабораторной работы 9. Изменить программу задания 6 лабораторной работы 9 с анализом данных по тем же критериям из файла. Ответы построчно дописать в исходный файл.

## Решение задачи

Код программы:

import java.io.\*;  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class Main {  
 public static final int *criteria* = 13;  
 public static final String *path* = "C:\\\\Users\\\\mikha\\\\Downloads\\\\Telegram Desktop\\\\Task\_9";  
 public static void print\_option(int i){  
 if (i == 0){  
 System.*out*.println("район города:");  
 }  
 if (i == 1){  
 System.*out*.println("Номер/Название школы:");  
 }  
 if (i == 2){  
 System.*out*.println("тип школы:");  
 }  
 if (i == 3){  
 System.*out*.println("число работников:");  
 }  
 if (i == 4){  
 System.*out*.println("количетсво кандидатов наук:");  
 }  
 if (i == 5){  
 System.*out*.println("количество докторов наук:");  
 }  
 if (i == 6){  
 System.*out*.println("количество учителей со званием 'заслужанный учитель':");  
 }  
 if (i == 7){  
 System.*out*.println("число обучающихся:");  
 }  
 if (i == 8){  
 System.*out*.println("количество классов:");  
 }  
 if (i == 9){  
 System.*out*.println("колиество аудиторий:");  
 }  
 if (i == 10){  
 System.*out*.println("количество компьютерных классов:");  
 }  
 if (i == 11){  
 System.*out*.println("количество спортивных залов:");  
 }  
 if (i == 12){  
 System.*out*.println("общую стоимость оборудование школы:");  
 }  
  
 }  
 public static void enter\_reglam(int count\_school){  
 Scanner scanner = new Scanner(System.*in*);  
 String inform;  
 System.*out*.println("Регламентированный ввод");  
 String[][] massive = new String[*criteria*][count\_school];  
 for (int i = 0;i<count\_school;i++){  
 System.*out*.println("Вы вводите информацию о "+ (i+1) + " школе!");  
 for (int g=0;g<*criteria*;g++){  
 System.*out*.print("Введите ");  
 *print\_option*(g);  
 inform = scanner.nextLine();  
 massive[g][i] = inform;  
 }  
 }  
 *information\_about\_the\_shool*(massive);  
 }  
  
 public static void enter\_file(int count\_school){  
 Scanner scanner = new Scanner(System.*in*);  
 String inform;  
 System.*out*.println("Данные были взяты из файла");  
 String[][] massive = new String[*criteria*][count\_school];  
 try {  
 BufferedReader file\_read = new BufferedReader(new FileReader(*path*));  
 BufferedWriter file\_write = new BufferedWriter(new FileWriter(*path*,true));  
 for (int i = 0; i < count\_school; i++) {  
 for (int g = 0; g < *criteria*; g++) {  
 inform = file\_read.readLine();  
 massive[g][i] = inform;  
 }  
 }  
 System.*out*.println(*information\_about\_the\_shool*(massive));  
 file\_write.write("\n" + *information\_about\_the\_shool*(massive));  
 file\_write.close();  
 System.*out*.println("Результат успешно записан в файл!");  
 }  
 catch (IOException e ){  
 e.printStackTrace();  
 }  
 }  
 public static void enter\_not\_reglam(int count\_school){  
 System.*out*.println("Нерегламентированный ввод");  
 Scanner scanner = new Scanner(System.*in*);  
 String inform;  
 String[][] massive = new String[*criteria*][count\_school];  
 ArrayList<Integer> massive\_exception = new ArrayList<Integer>();  
 int crit;  
 int count = count\_school \* *criteria*;  
 System.*out*.println("Вам нужно ввести номер критерия, который хотите ввести!");  
 for (int b = 0;b<*criteria*;b++){  
 System.*out*.print(b+1 + ")");  
 *print\_option*(b);  
 }  
 for (int i = 0;i<count\_school;i++){  
 System.*out*.println("Вы вводите информацию о "+ (i+1) + " школе!");  
 while (count != 0){  
 System.*out*.println("Введите номер критерия:");  
 crit = scanner.nextInt()-1;  
 if (!massive\_exception.contains(crit)) {  
 massive\_exception.add(crit);  
 System.*out*.print("Введите ");  
 *print\_option*(crit);  
 inform = scanner.next();  
 massive[crit][i] = inform;  
 count = count - 1;  
 }  
 else{  
 System.*out*.println("Вы ввелите критерий который ранее заполнили!");  
 }  
 }  
 }  
 *information\_about\_the\_shool*(massive);  
 }  
 public static String information\_about\_the\_shool(String[][] massive){  
 int count\_school\_1 = 0;  
 int count\_school\_2 = 0;  
 String text1 = "";  
 for (int i = 0;i<massive[0].length;i++){  
 int number\_personal = Integer.*parseInt*(massive[3][i]);  
 int number\_students = Integer.*parseInt*(massive[7][i]);  
 long summ\_money = Integer.*parseInt*(massive[12][i]);  
 int candidate\_sciences = Integer.*parseInt*(massive[4][i]);  
 int doctor\_sciences = Integer.*parseInt*(massive[5][i]);  
 int number\_computer\_class = Integer.*parseInt*(massive[10][i]);  
 int number\_sport\_class = Integer.*parseInt*(massive[11][i]);  
 double kof = (double)number\_students/number\_personal;  
 if (kof >= 0.005 && summ\_money <= 5000000){  
 count\_school\_1 = count\_school\_1+1;  
 }  
 if (candidate\_sciences >= 1 && doctor\_sciences >= 1 && number\_computer\_class == 2 && number\_sport\_class == 0){  
 count\_school\_2=count\_school\_2+1;  
 }  
  
 }  
 text1 = "Количество школ по первому описанию:" + count\_school\_1 + "\n" + "Количество школ по второму описанию:" + count\_school\_2;  
 return text1;  
  
 }  
 public static void main(String[] args){  
 Scanner scanner = new Scanner(System.*in*);  
 int count\_school;  
 int option;  
 System.*out*.println("Введите количество школ:");  
 count\_school = scanner.nextInt();  
 System.*out*.println("Выберите тип ввода:1)Регламентированный 2)Нерагламентированный 3)Данные из файла");  
 option = scanner.nextInt();  
 switch (option) {  
 case (1) :  
 *enter\_reglam*(count\_school);  
 scanner.close();  
 break;  
 case (2):  
 *enter\_not\_reglam*(count\_school);  
 break;  
 case (3):  
 *enter\_file*(count\_school);  
 break;  
  
 default:  
 System.*out*.println("У вас всего два выбора!");  
  
 }  
  
 }  
}

## Проверка результата



# Задание 10

## Постановка задачи

На основе программы задания 5 лабораторной работы 6 организовать запись в файл данных о результатах нахождения функции по ряду Маклорена.

Примерный вид файла ответов:

Функция f(x) = … для x = … равняется …

Результаты определения значений функции f(x) = … с помощью ряда Маклорена

Погрешность итерационной процедуры …

Значение функции по Маклорену Погрешность, % Число итераций

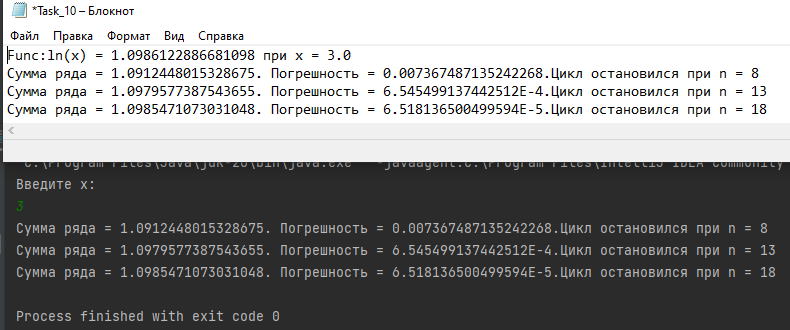
10 4 1

## Решение задачи

Код программы:

import java.io.FileWriter;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class Main {  
 public static String *path* = "C:\\\\\\\\Users\\\\\\\\mikha\\\\\\\\Downloads\\\\\\\\Telegram Desktop\\\\\\\\Task\_10";  
 public static void Write\_file(double summ, double rate , int i){  
 try{  
 FileWriter file = new FileWriter(*path*,true);  
 file.write("Сумма ряда = "+ summ + ". Погрешность = "+ rate + ".Цикл остановился при n = "+ i);  
 file.write("\n");  
 file.close();  
 }  
 catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
 public static void Write\_file\_func(double x ,double value){  
 try{  
 FileWriter file = new FileWriter(*path*,true);  
 file.write("Func:ln(x) = " + value + " при x = " + x);  
 file.write("\n");  
 file.close();  
 }  
 catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
 public static void main(String[] args) {  
 String path = "C:\\Users\\Пётр\\IdeaProjects\\Function\\src\\laba\_8\\Task\_10";  
 Scanner scanner = new Scanner(System.*in*);  
 double x;  
 System.*out*.println("Введите x:");  
 x = scanner.nextDouble();  
 *Write\_file\_func*(x , Math.*log*(x));  
 if (x > 0.5) {  
 double logarifmik = Math.*log*(x);  
 int n = 1000;  
 double summ = 0;  
 int error\_st = -2;  
 double error\_rate = Math.*pow*(10,error\_st);  
 for (int i=1;i<n;i++){  
 summ = summ + Math.*pow*((x - 1),i)/(Math.*pow*(x,i) \* i);  
 if (Math.*abs*(logarifmik - summ) < error\_rate){  
 System.*out*.println("Сумма ряда = "+ summ + ". Погрешность = "+ Math.*abs*(logarifmik - summ)+ ".Цикл остановился при n = "+ i);  
 *Write\_file*(summ,Math.*abs*(logarifmik - summ), i);  
 if (error\_st == -4){  
 break;  
 }  
 error\_st = error\_st - 1;  
 error\_rate = Math.*pow*(10,error\_st);  
 }  
 }  
 }  
 else{  
 System.*out*.println("x Должен быть > 0.5 !");  
 }  
 }  
  
}

## Проверка результата



# Задание 11

## Постановка задачи

Организовать запись данных о поверхности из лабораторной работы по диаграмма из УИР в файл. Файл содержит шапку о содержимом: формула поверхности, интервалы определения по координатам x и y.

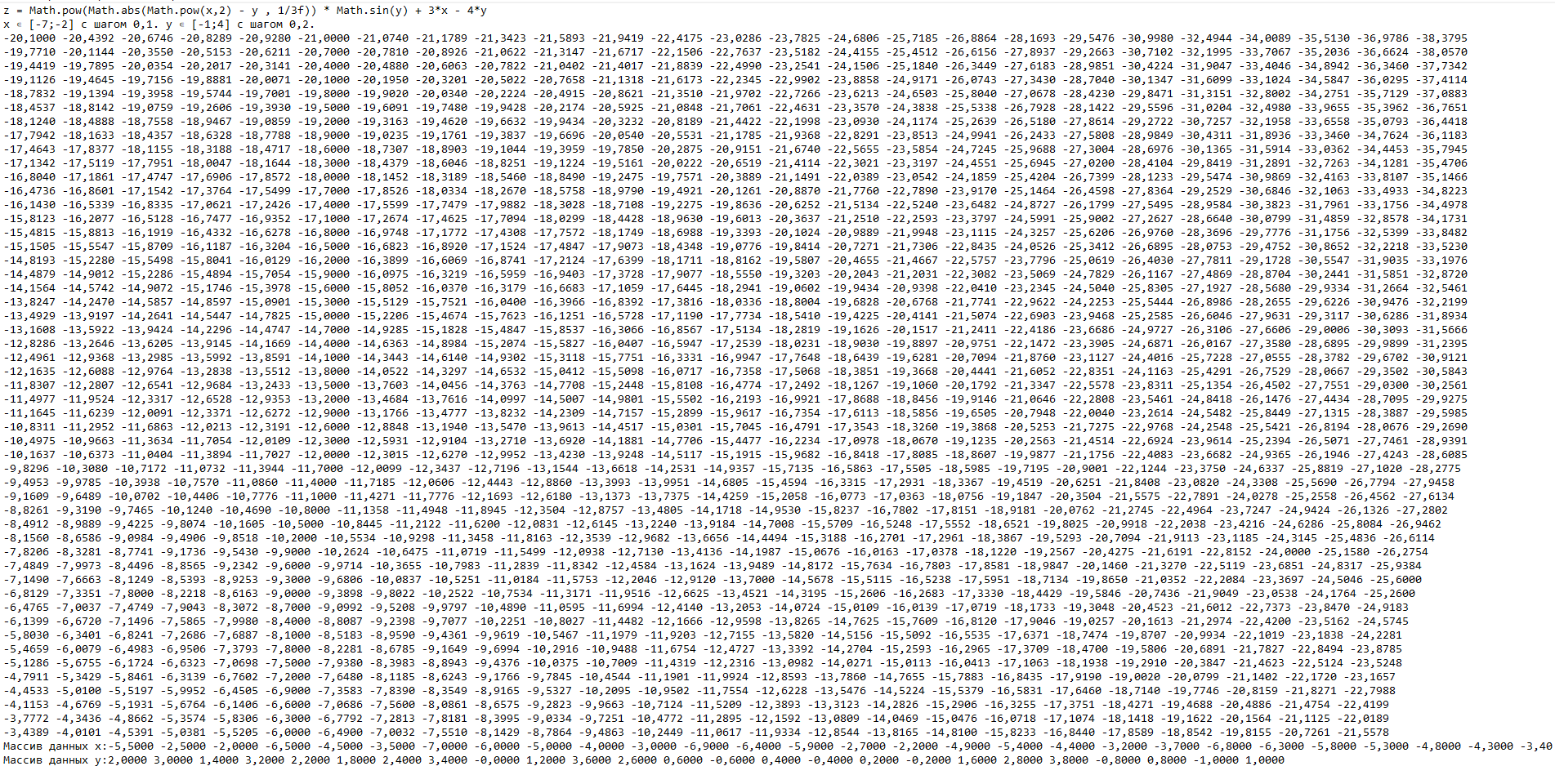
Запись выполнить в определенном формате с понятными заголовками: массив координат x, массив координат y, матрица значений поверхности (z).

## Решение задачи

Код программы:

import java.io.BufferedWriter;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.\*;  
  
public class Main {  
  
 public static String *path* = "C:\\\\\\\\\\\\\\\\Users\\\\\\\\\\\\\\\\mikha\\\\\\\\\\\\\\\\Downloads\\\\\\\\\\\\\\\\Telegram Desktop\\\\\\\\\\\\\\\\Task\_12";  
 public static void write\_massive\_data(ArrayList<Double> list){  
 try{  
 BufferedWriter file = new BufferedWriter(new FileWriter(*path*,true));  
 for (double value:list){  
// file.write(String.format("%." + 4 + "f",value) + " ");  
 file.write(String.*valueOf*(value));  
 }  
 file.close();  
 }  
 catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
  
 public static void write\_function() {  
 try {  
 BufferedWriter file = new BufferedWriter(new FileWriter(*path*,true));  
 String func = "z = Math.pow(Math.abs(Math.pow(x,2) - y , 1/3f)) \* Math.sin(y) + 3\*x - 4\*y";  
 String comment\_func = "x ∈ [-7;-2] с шагом 0,1. y ∈ [-1;4] с шагом 0,2.";  
 file.write(func + "\n");  
 file.write(comment\_func + "\n");  
 file.close();  
 }  
 catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 public static void main(String[] args){  
 *write\_function*();  
 try {  
 BufferedWriter file\_1 = new BufferedWriter(new FileWriter(*path*,true));  
 double z;  
 ArrayList<Double> list\_x = new ArrayList<>();  
 ArrayList<Double> list\_y = new ArrayList<>();  
 for (double x = -7.0; x <= -2.0; x = x + 0.1){  
 for (double y = -1.0; y <= 4.0; y = y + 0.2){  
 z = Math.*pow*(Math.*abs*(Math.*pow*(x,2) - y) , 1/3f) \* Math.*sin*(y) + 3\*x - 4\*y;  
 file\_1.write(String.*format*("%." + 4 + "f",z) + " ");  
 list\_y.add(y);  
 list\_x.add(x);  
 }  
 file\_1.write("\n");  
 }  
 Set<Double> unique\_x = new HashSet<>(list\_x);  
 ArrayList<Double> list\_unique\_x = new ArrayList<>(unique\_x);  
 Set<Double> unique\_y = new HashSet<>(list\_y);  
 ArrayList<Double> list\_unique\_y = new ArrayList<>(unique\_y);  
 file\_1.write("Массив данных x:");  
 for (Double value: list\_unique\_x){  
 file\_1.write(String.*format*("%." + 4 + "f",value) + " ");  
 }  
 file\_1.newLine();  
 file\_1.write("Массив данных y:");  
 for (Double value: list\_unique\_y){  
 file\_1.write(String.*format*("%." + 4 + "f",value) + " ");  
 }  
 file\_1.close();  
  
  
 }  
 catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
}

## Проверка результата



# Задание 12

## Постановка задачи

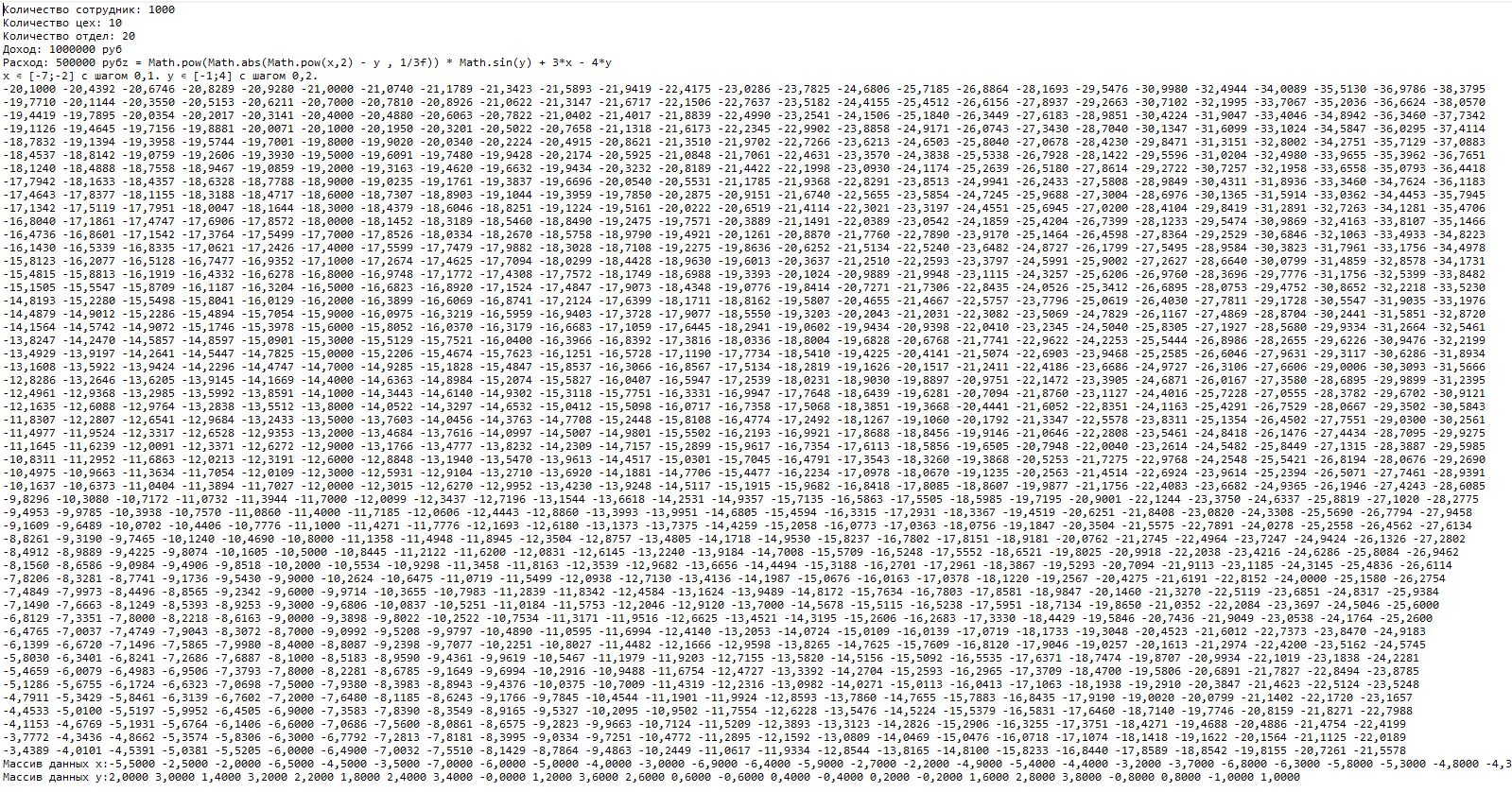
Переделать программу задания 12 лабораторной работы 9 по работе с регулярными выражениями. Дан файл с текстом, применить анализ и редактирование текст файла с использованием ранее написанного алгоритма форматирования текста. Записать отредактированный текст в файл после исходного текста через пустую строку и заголовок.

## Решение задачи

Код программы:

import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class Main {  
 public static String *path* = "C:\\\\\\\\Users\\\\\\\\mikha\\\\\\\\Downloads\\\\\\\\Telegram Desktop\\\\\\\\Task\_12";  
 public static void main(String[] args) {  
 int count = 0;  
 try {  
 BufferedReader file\_1 = new BufferedReader(new FileReader(*path*));  
 String line;  
 while ((line = file\_1.readLine()) != null){  
 count++;  
 }  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 }  
 String[] companyInfo = new String[count];  
 int i = 0;  
 try {  
 BufferedReader file\_2 = new BufferedReader(new FileReader(*path*));  
 String lines;  
 while ((lines = file\_2.readLine()) != null){  
 companyInfo[i++] = lines;  
 }  
 }  
 catch (IOException e){  
 e.printStackTrace();  
 }  
 // Создаем регулярное выражение для поиска слов, требующих склонения  
 Pattern pattern = Pattern.*compile*("\\b(сотрудник|цех|отдел)\\b");  
  
 // Проходим по массиву строк и заменяем слова, требующие склонения  
 try {  
 FileWriter file = new FileWriter(*path*);  
 for (String info : companyInfo) {  
 Matcher matcher = pattern.matcher(info);  
 if (matcher.find()) {  
 String word = matcher.group();  
 String newWord = *getDeclension*(word);  
 String newInfo = matcher.replaceAll(newWord);  
 file.write(newInfo+ "\n");  
 }  
 else {  
 file.write(info + "\n");  
 }  
 }  
 file.close();  
 }  
 catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
 public static String getDeclension(String word) {  
 return word + "ов";  
 }  
}

## Проверка результата



# Задание 13

## Постановка задачи

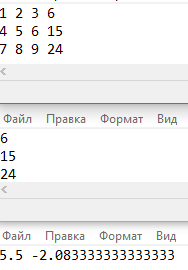
В первом файле хранится k матриц из n строк и n+1 столбцов каждая (последний столбец – столбец свободных членов). Во втором файле хранится k столбцов решения СЛАУ, записанных в файле 1. Проверить соответствие результатов (т.е. при умножении матрица коэффициентов на столбец решения, мы должны получить столбец свободных членов). Если решение не соответствует СЛАУ, тогда заменить его на правильное, иначе оставить без изменения (правильные данные записываются в третий файл).

## Решение задачи

Код программы:

import java.io.File;  
import java.io.IOException;  
import java.io.PrintWriter;  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 // Чтение данных из файлов  
 File inputMatrixFile = new File("C:\\\\\\\\\\\\\\\\Users\\\\\\\\\\\\\\\\mikha\\\\\\\\\\\\\\\\Downloads\\\\\\\\\\\\\\\\Telegram Desktop\\\\\\\\\\\\\\\\Task\_13\_1"); // содержит матрицу коэффициентов СЛАУ;  
 File inputSolutionFile = new File("C:\\\\\\\\\\\\\\\\Users\\\\\\\\\\\\\\\\mikha\\\\\\\\\\\\\\\\Downloads\\\\\\\\\\\\\\\\Telegram Desktop\\\\\\\\\\\\\\\\Task\_13\_2"); // содержит столбец решений СЛАУ, записанный в файле  
  
 Scanner matrixScanner = null;  
 Scanner solutionScanner = null;  
  
 try {  
 matrixScanner = new Scanner(inputMatrixFile);  
 solutionScanner = new Scanner(inputSolutionFile);  
 } catch (IOException e) {  
 System.*out*.println("Ошибка при открытии файлов: " + e.getMessage());  
 }  
  
 int k = matrixScanner.nextInt(); // Количество матриц  
 int n = matrixScanner.nextInt(); // Размерность матрицы  
 double[][][] matrices = new double[k][n][n + 1]; // Массив матриц  
 double[][] solutions = new double[k][n]; // Массив решений  
  
 // Чтение матриц и решений из файлов  
 for (int i = 0; i < k; i++) {  
 for (int j = 0; j < n; j++) {  
 for (int l = 0; l < n + 1; l++) {  
 matrices[i][j][l] = matrixScanner.nextDouble();  
 }  
 }  
 for (int j = 0; j < n; j++) {  
 solutions[i][j] = solutionScanner.nextDouble();  
 }  
 }  
  
 // Проверка соответствия решения СЛАУ  
 for (int i = 0; i < k; i++) {  
 double[] solution = matrices[i][0].clone(); // Копия первой строки матрицы коэффициентов  
 for (int j = 0; j < n; j++) {  
 double sum = 0;  
 for (int l = 0; l < n; l++) {  
 sum += matrices[i][j][l] \* solution[l];  
 }  
 if (sum != matrices[i][j][n]) { // Если решение не соответствует СЛАУ  
 solution = *solve*(matrices[i]); // Находим правильное решение  
 break;  
 }  
 }  
 solutions[i] = solution; // Сохраняем правильное решение  
 }  
  
 // Запись правильных решений в третий файл  
 File outputSolutionFile = new File("C:\\\\\\\\\\\\\\\\Users\\\\\\\\\\\\\\\\mikha\\\\\\\\\\\\\\\\Downloads\\\\\\\\\\\\\\\\Telegram Desktop\\\\\\\\\\\\\\\\Task\_13\_3");  
 PrintWriter outputSolutionWriter = null;  
  
 try {  
 outputSolutionWriter = new PrintWriter(outputSolutionFile);  
 } catch (IOException e) {  
 System.*out*.println("Ошибка при открытии файла: " + e.getMessage());  
 System.*exit*(1);  
 }  
  
 for (int i = 0; i < k; i++) {  
 for (int j = 0; j < n; j++) {  
 outputSolutionWriter.print(solutions[i][j] + " ");  
 }  
 outputSolutionWriter.println();  
 }  
  
 outputSolutionWriter.close();  
 }  
  
 // Метод Гаусса  
 public static double[] solve(double[][] matrix) {  
 int n = matrix.length;  
  
 for (int i = 0; i < n; i++) {  
 int maxRow = i;  
 for (int j = i + 1; j < n; j++) {  
 if (Math.*abs*(matrix[j][i]) > Math.*abs*(matrix[maxRow][i])) {  
 maxRow = j;  
 }  
 }  
  
 double[] temp = matrix[i];  
 matrix[i] = matrix[maxRow];  
 matrix[maxRow] = temp;  
  
 for (int j = i + 1; j < n; j++) {  
 double factor = matrix[j][i] / matrix[i][i];  
 for (int k = i; k < n + 1; k++) {  
 matrix[j][k] -= factor \* matrix[i][k];  
 }  
 }  
 }  
  
 // Обратный ход метода Гаусса  
 double[] solution = new double[n];  
 for (int i = n - 1; i >= 0; i--) {  
 double sum = 0;  
 for (int j = i + 1; j < n; j++) {  
 sum += matrix[i][j] \* solution[j];  
 }  
 solution[i] = (matrix[i][n] - sum) / matrix[i][i];  
 }  
  
 return solution;  
 }  
}

## Проверка результата



# Задание 14

## Постановка задачи

Дан файл результатов исследования (листинг натурного эксперимента, результаты численного решения задачи и т.п.). Выполнить анализ данных: 1. Найти максимальное, минимальное и среднее значение параметров по столбцу; 2. Отклонение каждого значения параметра от среднего.

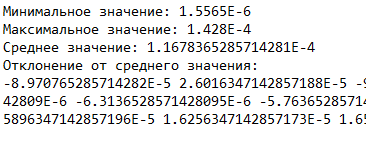
Записать данные в конец файла построчно. Сначала ответ на первый вопрос, потом ответ на 2 вопрос.

## Решение задачи

Код программы:

import java.io.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
public class Main {  
  
 private static List<Double> getColumnValues(String filename, int columnIndex) {  
 List<Double> columnValues = new ArrayList<>();  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {  
 String line;  
 boolean foundColumn = false;  
  
 while ((line = reader.readLine()) != null) { //скипаем лишние  
 if (line.startsWith(" NODE")) {  
 foundColumn = true;  
 continue;  
 }  
  
 if (foundColumn && line.trim().isEmpty()) {  
 break;  
 }  
  
 if (foundColumn) {  
 String[] values = line.trim().split("\\s+");  
 if (values.length > columnIndex) {  
 String sanitizedValue = values[columnIndex].replaceAll("[^\\d.E-]", "");  
 double value = Double.*parseDouble*(sanitizedValue);  
 columnValues.add(value);  
 }  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 return columnValues;  
 }  
  
 private static double findMinValue(List<Double> values) {  
 double min = Double.*MAX\_VALUE*;  
 for (double value : values) {  
 if (value < min) {  
 min = value;  
 }  
 }  
 return min;  
 }  
  
 private static double findMaxValue(List<Double> values) {  
 double max = Double.*MIN\_VALUE*;  
 for (double value : values) {  
 if (value > max) {  
 max = value;  
 }  
 }  
 return max;  
 }  
  
 private static double findAverage(List<Double> values) {  
 double sum = 0.0;  
 for (double value : values) {  
 sum += value;  
 }  
 return sum / values.size();  
 }  
  
 public static void main(String[] args) {  
 String filename = "C:\\Users\\mikha\\Downloads\\Telegram Desktop\\Variant\_4,14.txt"; // Имя файла с результатами исследования  
 List<Double> columnValues = *getColumnValues*(filename, 2); // Передаем индекс столбца, с которым хотим работать (например, 2 для EPTOX)  
 try{  
 BufferedWriter file = new BufferedWriter(new FileWriter(filename,true));  
 if (!columnValues.isEmpty()) {  
 double min = *findMinValue*(columnValues);  
 double max = *findMaxValue*(columnValues);  
 double average = *findAverage*(columnValues);  
  
 System.*out*.println("Минимальное значение: " + min);  
 System.*out*.println("Максимальное значение: " + max);  
 System.*out*.println("Среднее значение: " + average);  
 file.newLine();  
 file.write("Минимальное значение: " + min + "\n");  
 file.write("Максимальное значение: " + max + "\n");  
 file.write("Среднее значение: " + average + "\n");  
  
 System.*out*.println("Отклонение от среднего значения:");  
 file.write("Отклонение от среднего значения:" + "\n");  
 for (double value : columnValues) {  
 double deviation = value - average;  
 System.*out*.println(deviation);  
 file.write(String.*valueOf*(deviation) + " ");  
 }  
 file.newLine();  
 }  
 file.close();  
 }  
 catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
}

## Проверка результата



# Задание 15

## Постановка задачи

Создать файл с координатами 10 участков произвольной четырехугольной формы. Создать файл со структурированными данными о наличии на территории, где расположены участки, объектов инфраструктуры: вышки сотовой связи, опоры ЛЭП, озера, ключи с питьевой водой, жилые постройки. Данные заполнены с координатами объектов, озера и ключи представить в виде координат центра и радиуса.

Для каждого участка определить количество лежащих в его пределах объектов инфраструктуры, сформировать файл ответа по типу:

————————

N участка Площадь Кол-во % площади участка, % площади участка,

участка объектов которую занимает которую занимает

водоем ключ

1 4 3 50 10

...

————————

## Решение задачи

Код программы:

import java.awt.\*;  
import java.io.\*;  
  
public class Main {  
 public static String *path\_file\_1* = "C:\\\\Users\\\\mikha\\\\Downloads\\\\Telegram Desktop\\\\Task\_15\_1";  
 public static String *path\_file\_2* = "C:\\\\Users\\\\mikha\\\\Downloads\\\\Telegram Desktop\\\\Task\_15\_2";  
 public static String *path\_file\_3* = "C:\\\\Users\\\\mikha\\\\Downloads\\\\Telegram Desktop\\\\Task\_15\_3";  
  
 public static double getPercent(double B, double S) {  
 return S / B \* 100;  
 }  
  
 //формула Герона  
 public static double getQuadrilateralArea(Point[] points) {  
 double a = points[0].distance(points[1]);  
 double b = points[1].distance(points[2]);  
 double c = points[2].distance(points[3]);  
 double d = points[3].distance(points[0]);  
 double p1 = (a + b + c) / 2;  
 double p2 = (c + d + a) / 2;  
 double s1 = Math.*sqrt*(p1 \* (p1 - a) \* (p1 - b) \* (p1 - c));  
 double s2 = Math.*sqrt*(p2 \* (p2 - c) \* (p2 - d) \* (p2 - a));  
 double area = s1 + s2;  
 return area;  
 }  
  
 public static boolean isSmallRectangleInsideBigRectangle(Point[] bigRect, Point[] smallRect) {  
 // Проверяем, что все вершины маленького четырехугольника находятся внутри большого четырехугольника  
 for (Point p : smallRect) {  
 if (!*isPointInsideRectangle*(p, bigRect)) {  
 return false;  
 }  
 }  
 // Если все вершины маленького четырехугольника находятся внутри большого, значит, он лежит внутри  
 return true;  
 }  
  
 public static boolean isPointInsideRectangle(Point p, Point[] rect) {  
 // Проверяем, что точка лежит внутри четырехугольника, используя теорему о принадлежности точки многоугольнику  
 int n = rect.length;  
 boolean inside = false;  
 for (int i = 0, j = n - 1; i < n; j = i++) {  
 if (((rect[i].y > p.y) != (rect[j].y > p.y)) &&  
 (p.x < (rect[j].x - rect[i].x) \* (p.y - rect[i].y) / (rect[j].y - rect[i].y) + rect[i].x)) {  
 inside = !inside;  
 }  
 }  
 return inside;  
 }  
  
 public static boolean isCircleInsideQuadrilateral(Point[] quadrilateral, Point circleCenter, double radius) {  
 double centerX = (quadrilateral[0].getX() + quadrilateral[1].getX() + quadrilateral[2].getX() + quadrilateral[3].getX()) / 4;  
 double centerY = (quadrilateral[0].getY() + quadrilateral[1].getY() + quadrilateral[2].getY() + quadrilateral[3].getY()) / 4;  
 boolean isCenterInside = *rayCasting*(centerX, centerY, quadrilateral);  
 if (!isCenterInside) {  
 return false;  
 }  
 for (int i = 0; i < 360; i++) {  
 double x = circleCenter.getX() + radius \* Math.*cos*(Math.*toRadians*(i));  
 double y = circleCenter.getY() + radius \* Math.*sin*(Math.*toRadians*(i));  
 boolean isPointInside = *rayCasting*(x, y, quadrilateral);  
 if (!isPointInside) {  
 return false;  
 }  
 }  
 return true;  
 }  
  
 private static boolean rayCasting(double x, double y, Point[] vertices) {  
 boolean inside = false;  
 for (int i = 0, j = vertices.length - 1; i < vertices.length; j = i++) {  
 double xi = vertices[i].getX();  
 double yi = vertices[i].getY();  
 double xj = vertices[j].getX();  
 double yj = vertices[j].getY();  
 boolean intersect = ((yi > y) != (yj > y))  
 && (x < (xj - xi) \* (y - yi) / (yj - yi) + xi);  
 if (intersect) {  
 inside = !inside;  
 }  
 }  
 return inside;  
 }  
  
 public static double getCircleArea(double radius) {  
 double pi = Math.*PI*;  
 double area = pi \* Math.*pow*(radius, 2);  
 return area;  
 }  
  
  
 public static void main(String[] args) {  
 String line = "";  
 try {  
 BufferedReader file\_1 = new BufferedReader(new FileReader(*path\_file\_1*));  
 BufferedReader file\_2 = new BufferedReader(new FileReader(*path\_file\_2*));  
 int count\_object;  
 int count\_water;  
 double sq\_water;  
 int count\_key;  
 double sq\_key;  
 int number\_region = 0;  
 Point[] bigRect = new Point[4];  
 int count = 0;  
 while ((line = file\_1.readLine()) != null) {  
 count\_object = 0;  
 number\_region++;  
 count\_water = 0;  
 sq\_key = 0;  
 count\_key = 0;  
 sq\_water = 0;  
 for (String l : line.split(" ")) {  
 int x = Integer.*parseInt*(l.split(";")[0]);  
 int y = Integer.*parseInt*(l.split(";")[1]);  
 bigRect[count] = new Point(x, y);  
 count = count + 1;  
 }  
  
 int left\_count;  
 Point[] smallRect = new Point[4];  
 count = 0;  
 while ((line = file\_2.readLine()) != null) { //1-Участок, 2-ЛЭП, 3-Скважина, 4-Дом, 5-Баня 6 - водоём 7 - ключ  
 for (String l : line.split(" ")) {  
 count = 0;  
 if (String.*valueOf*(l.charAt(0)).equals("6") || String.*valueOf*(l.charAt(0)).equals("7")) {// Если круглые объекты  
 if (String.*valueOf*(l.charAt(0)).equals("6")) {  
 String[] coords = l.substring(2).split(";");  
 Point point = new Point(Integer.*parseInt*(coords[0]), Integer.*parseInt*(coords[1]));  
 double Radius = Double.*valueOf*(coords[2]);  
 if (*isCircleInsideQuadrilateral*(bigRect, point, Radius)) {  
 count\_water++;  
 sq\_water = sq\_water + *getCircleArea*(Radius);  
 }  
 }  
 if (String.*valueOf*(l.charAt(0)).equals("7")) {  
 String[] coords = l.substring(2).split(";");  
 Point point = new Point(Integer.*parseInt*(coords[0]), Integer.*parseInt*(coords[1]));  
 double Radius = Double.*valueOf*(coords[2]);  
 if (*isCircleInsideQuadrilateral*(bigRect, point, Radius)) {  
 count\_key++;  
 sq\_key = sq\_key + *getCircleArea*(Radius);  
 }  
 }  
  
 } else {  
 String[] parts = l.substring(2).split(";");  
 for (int i = 0, j = 0; i < parts.length; i += 2, j++) {  
 int x = Integer.*parseInt*(parts[i]);  
 int y = Integer.*parseInt*(parts[i + 1]);  
 smallRect[count] = new Point(x, y);  
 count++;  
 }  
 if (*isSmallRectangleInsideBigRectangle*(bigRect, smallRect)) {  
 count\_object++;  
 }  
 }  
 }  
 count = 0;  
  
 }  
 Region r = new Region(number\_region, *getQuadrilateralArea*(bigRect), (count\_object + count\_water + count\_key), *getPercent*(*getQuadrilateralArea*(bigRect), sq\_water), *getPercent*(*getQuadrilateralArea*(bigRect), sq\_key));  
 try {  
 BufferedWriter file\_3 = new BufferedWriter(new FileWriter(*path\_file\_3*, true));  
 file\_3.write("Ответ:" + '\n');  
 file\_3.write("Номер региона:" + r.getNumber()+ '\n');  
 file\_3.write("Площадь участка" + r.getSquare\_region()+ '\n');  
 file\_3.write("Кол-во объектов в участке:" + r.getCount\_object()+ '\n');  
 file\_3.write("Процент площади участка, которую занимает водоём:" + r.getSquare\_water()+ '\n');  
 file\_3.write("Площадь участка, который занимает ключ:" + r.getSquare\_key()+ '\n');  
 file\_3.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
  
 class Region {  
 public int number;  
 public double square\_region;  
 public int count\_object;  
 public double square\_water;  
 public double square\_key;  
  
 public Region(int number, double square\_region, int count\_object, double square\_water, double square\_key) {  
 this.number = number;  
 this.square\_region = square\_region;  
 this.count\_object = count\_object;  
 this.square\_water = square\_water;  
 this.square\_key = square\_key;  
 }  
  
 public void getFull() {  
 System.*out*.print(number + "\n");  
 System.*out*.print(square\_region + "\n");  
 System.*out*.print(count\_object + "\n");  
 System.*out*.print(square\_water + "\n");  
 System.*out*.print(square\_key + "\n");  
 }  
  
 public int getNumber() {  
 return number;  
 }  
  
 public double getSquare\_region() {  
 return square\_region;  
 }  
  
 public int getCount\_object() {  
 return count\_object;  
 }  
  
 public double getSquare\_water() {  
 return square\_water;  
 }  
  
 public double getSquare\_key() {  
 return square\_key;  
 }  
  
 public void setNumber(int number) {  
 this.number = number;  
 }  
  
 public void setSquare\_region(double square\_region) {  
 this.square\_region = square\_region;  
 }  
  
 public void setCount\_object(int count\_object) {  
 this.count\_object = count\_object;  
 }  
  
 public void setSquare\_water(double square\_water) {  
 this.square\_water = square\_water;  
 }  
  
 public void setSquare\_key(double square\_key) {  
 this.square\_key = square\_key;  
 }  
 }  
 }  
}

## Проверка результата

