**Java程序设计实践报告**

序 号： 15

姓 名： 刘积松

学 号： 2022000974

学 院： 地球物理与石油资源学院

专业班级： 大数据22203班

指导老师： 蔡明

2024年 月 日

**目 录**

[一、 题目 1](#_Toc106311929)

[二、 系统需求分析 1](#_Toc106311930)

[三、 系统设计 1](#_Toc106311931)

[四、 系统实现 1](#_Toc106311932)

[五、 系统测试 1](#_Toc106311933)

[六、 使用说明 2](#_Toc106311934)

[七、 总结 2](#_Toc106311935)

[八、 附录：源程序代码 2](#_Toc106311936)

1. 题目

简易测井资料处理分析系统的设计与实现

1. 系统需求分析

简易测井资料处理分析系统，能够利用菜单调用各个子方法，分别实现如下功能：

1. 数据提取与检查。通过屏幕查看自己提取的原始测井数据及参数数据，检查数据提取是否正常。
2. 测井数据处理。利用提取的数据分别计算各深度点的孔隙度、泥质含量、含有饱和度。
3. 处理结果分析。
4. 通过屏幕显示查询各个深度点的处理成果数据条（输入深度点序号查询），含原始数据和处理结果。
5. 统计处理深度段泥质含量、孔隙度和含有饱和度的最小值、最大值和平均值，并显示到屏幕上。
6. 按含油饱和度从大到小的顺序通过屏幕显示查看完整的测井数据处理成果表。
7. 通过屏幕显示查询不同等级储层深度点的处理成果数据，并通过屏幕显示不同等级储层深度点的数目。储层等级分级标准为：所有储层泥质含量小于等于25%；另外，孔隙度大于12%为Ⅰ类储层，孔隙度介于[8%,12%]为Ⅱ类储层，孔隙度介于[5%,8%]为Ⅲ类储层，孔隙度小于5%为Ⅳ类储层。
8. 通过屏幕显示查询好油层深度点的处理成果数据。好油层条件为泥质含量小于等于25%、孔隙度大于等于6%、含油饱和度大于等于60%。
9. 系统设计

用Java写的

1. 系统实现

Java实现

1. 系统测试

这里撰写相关内容

1. 使用说明

不太好用

1. 总结

这

1. 附录：源程序代码

import java.io.\*;

import java.util.\*;

class Main {

    public static void main(String[] args) {

        String filePath = "D:\\parameters.txt";

        String logdata\_Path = "D:\\well\_logging\_data.txt";

        LogDataProcessor processor = new LogDataProcessor(logdata\_Path, filePath);

        while (true) {

            System.out.println("---------菜单----------");

            System.out.println("0. 退出");

            System.out.println("1. 查看原始测井数据和处理参数");

            System.out.println("2. 计算泥质含量、孔隙度、饱和度并输出");

            System.out.println("3. 查看单深度点处理成果数据条");

            System.out.println("4. 统计查看储层参数的极值和平均值");

            System.out.println("5. 按饱和度对处理成果数据表排序");

            System.out.println("6. 统计显示不同等级储层信息");

            System.out.println("7. 统计查看好油层信息");

            System.out.print("请选择一个选项：");

            Scanner scanner = new Scanner(System.in);

            int choice = scanner.nextInt();

            System.out.println("");

            switch (choice) {

                case 0:

                    System.out.println("退出程序");

                    scanner.close();

                    System.exit(0);

                    break;

                case 1:

                    try {

                        List<LogData> logDataList = processor.readLogData();

                        System.out.println("原始测井数据：");

                        processor.printFirstLine();

                        processor.printLogData(logDataList);

                        System.out.println("");

                        System.out.println("参数信息：");

                        processor.parameters.printParameters();

                    } catch (FileNotFoundException e) {

                        System.err.println("找不到数据文件：" + logdata\_Path);

                    }

                    break;

                case 2:

                case 3:

                case 4:

                case 5:

                case 6:

                case 7:

                    try {

                        List<LogData> logDataList = processor.readLogData();

                        processor.processLogData(logDataList);

                        if (choice == 2) {

                            processor.printFirstLine();

                            processor.printLogData(logDataList);

                        }else if (choice == 3) {

                            System.out.print("请输入要查看的深度点(0~15)：");

                            int depth\_point = scanner.nextInt();

                            processor.printFirstLine();

                            if (depth\_point >= 0 && depth\_point < 16) {

                                logDataList.get(depth\_point).print();

                                System.out.println("");

                            }else{

                                System.out.println("深度点不合法");

                            }

                        }else if (choice == 4) {

                            processor.printStatistics(logDataList);

                            break;

                        }else if (choice == 5) {

                            processor.sortBySO(logDataList);

                        }else if (choice == 6) {

                            processor.classByPOR(logDataList);

                        }else if (choice == 7) {

                            processor.showFullOfOil(logDataList);

                        }

                    }catch (FileNotFoundException e) {

                        System.err.println("找不到数据文件：" + logdata\_Path);

                    }

                    break;

                default:

                    System.out.println("无效的选项，请重新选择");

                    break;

            }

        }

    }

}

class LogDataProcessor {

    private String logdata\_Path;

    private double VSHmax = 0;

    private double VSHmin = 0;

    private double VSHall = 0;

    private double PORmax = 0;

    private double PORmin = 0;

    private double PORall = 0;

    private double SOmax = 0;

    private double SOmin = 0;

    private double SOall = 0;

    private double processTime= 0;

    // private String filePath;

    String[] types = {};

    // 读取参数

    public LogParameters parameters;

    public LogDataProcessor(String logdata\_Path, String filePath) {

        // this.filePath = filePath;

        this.logdata\_Path = logdata\_Path;

        this.parameters = new LogParameters();

        try {

            this.parameters.loadParametersFromFile(filePath);

        } catch (IOException e) {

            System.err.println("无法加载参数文件：" + e.getMessage());

        }

    }

    // 从文件读取测井数据，读取第15到30行有depth的数据

    public List<LogData> readLogData() throws FileNotFoundException {

        List<LogData> logDataList = new ArrayList<>();

        Scanner fileScanner = new Scanner(new File(logdata\_Path));

        int lineNumber = 0; // 用于跟踪文件中的行号

        while (fileScanner.hasNextLine()) {

            String line = fileScanner.nextLine().trim();

            lineNumber++;

            if (line.matches("^#.\*$")){

                types = line.split("\\s+");

            }

            if (line.matches("^\\d.\*$")) { // 检查行是否以数字开头

                if (lineNumber >= 15 && lineNumber <= 30) {

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

                    float depth = Float.parseFloat(tokens[0]);

                    float[] values = new float[tokens.length - 1];

                    for (int i = 1; i < tokens.length; i++) {

                        values[i - 1] = Float.parseFloat(tokens[i]);

                    }

                    logDataList.add(new LogData(depth, values));

                }

            }

        }

        fileScanner.close();

        return logDataList;

    }

    // 计算泥质含量、孔隙度、饱和度

    public void processLogData(List<LogData> logDataList) {

        float GRmin = parameters.getParameter("GRmin");

        float GRmax = parameters.getParameter("GRmax");

        float GCUR = parameters.getParameter("GCUR");

        float DTma = parameters.getParameter("DTma");

        float DTf = parameters.getParameter("DTf");

        float a = parameters.getParameter("a");

        float b = parameters.getParameter("b");

        float Rw = parameters.getParameter("Rw");

        float m = parameters.getParameter("m");

        float n = parameters.getParameter("n");

        for (LogData logData : logDataList) {

            float[] values = logData.getValues();

            // 计算泥质含量VSH

            float SH = (values[2]-GRmin) / (GRmax-GRmin);

            double VSH = (Math.pow(2, GCUR\*SH) - 1) / (Math.pow(2, GCUR) - 1);

            if (VSH > 1) {

                VSH = 1;

            }else if (VSH < 0) {

                VSH = 0;

            }

            // 计算孔隙度

            double POR = (values[1] - DTma) / (DTf - DTma);

            if (POR > 0.4) {

                POR = 0.4;

            }else if (POR < 0) {

                POR = 0.005;

            }

            // 计算饱和度

            double Sw = Math.pow(a\*b\*Rw / (Math.pow(POR,m)\*values[3]) , 1/n);

            double SO = 1 - Sw;

            if (SO < 0) {

                SO = 0;

            }else if (SO > 1) {

                SO = 1;

            }

            // 创建一个新的float数组，并设置正确的值

            float[] newValues = new float[values.length + 3];

            System.arraycopy(values, 0, newValues, 0, values.length);

            newValues[values.length] = (float) VSH;

            newValues[values.length + 1] = (float) POR;

            newValues[values.length + 2] = (float) SO;

            logData.setValues(newValues);

            //types更新

            String[] types\_old = Arrays.copyOf(types, types.length);

            types = new String[13];

            System.arraycopy(types\_old, 0, types, 0, Math.min(types\_old.length, types.length));

            // 在新数组的末尾添加三个特定的字符串

            types[types.length - 3] = "VSH";

            types[types.length - 2] = "POR";

            types[types.length - 1] = "SO";

            VSHall += VSH;

            PORall += POR;

            SOall += SO;

            if (VSH >= VSHmax) {

                VSHmax = VSH;

            }else {

                VSHmin = VSH;

            }

            if (POR >= PORmax) {

                PORmax = POR;

            }else {

                PORmin = POR;

            }

            if (SO >= SOmax) {

                SOmax = SO;

            }else {

                SOmin = SO;

            }

        }processTime += 1;

       //保存数据到文件

       try {

            FileWriter file = new FileWriter("D:\\大数据22203\_Results\_15.txt");

            for (String type : types) {

                file.write(String.format("%12s", type));

            }file.write("\n");

            for (LogData logData : logDataList) {

                file.write(String.format("%-10.6s", String.valueOf(logData.getDepth()))+ " ");

                for (float value : logData.getValues()) {

                    file.write(String.format("%-10.6s", String.valueOf(value))+ " ");

                }file.write("\n");

            }

            file.close();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    // 打印第一行

    public void printFirstLine() {

        for (String type : types) {

            System.out.printf("%-10s",type);

        }System.out.println("");

    }

    // 打印测井数据

    public void printLogData(List<LogData> logDataList) {

        for (LogData logData : logDataList) {

            logData.print();

        }

    }

    // 打印极值、平均值

    public void printStatistics(List<LogData> logDataList){

        System.out.println("VSHmax = " + VSHmax);

        System.out.println("VSHmin = " + VSHmin);

        System.out.println("VSHavg = " + VSHall / logDataList.size() / processTime);

        System.out.println("");

        System.out.println("PORmax = " + PORmax);

        System.out.println("PORmin = " + PORmin);

        System.out.println("PORavg = " + PORall / logDataList.size() / processTime);

        System.out.println("");

        System.out.println("SOmax = " + SOmax);

        System.out.println("SOmin = " + SOmin);

        System.out.println("SOavg = " + SOall / logDataList.size() / processTime);

        System.out.println("");

    }

    // 按饱和度排序

    public void sortBySO(List<LogData> logDatalList){

        int[] array= new int[]{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};

        for (int i = 0 ; i < logDatalList.size() - 1 ; i++) {

            double max = logDatalList.get(array[i]).getValues()[11];

            int maxnum = i;

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

                if (logDatalList.get(array[j]).getValues()[11] > max){

                    max = logDatalList.get(array[j]).getValues()[11];

                    maxnum = j;

                }

            }int tmp = array[i];

            array[i] = array[maxnum];

            array[maxnum] = tmp;

        }System.out.println("按照饱和度由大到小排序：");

        printFirstLine();

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

            logDatalList.get(array[i]).print();

        }

    }

    // 按照储层分级

    public void classByPOR(List<LogData> logDatalList){

        int[] time = {0,0,0,0};

        printFirstLine();

        System.out.println("Ⅰ 类储层:");

        for (LogData logData : logDatalList){

            if (logData.getValues()[10] > 0.12 && logData.getValues()[9] <= 0.25){

                logData.print();

                time[0]++;

            }

        }System.out.println("共有" + time[0] + "个深度点");

        System.out.println("");

        System.out.println("Ⅱ 类储层:");

        for (LogData logData : logDatalList){

            if (logData.getValues()[10] > 0.08 && logData.getValues()[10] <= 0.12 && logData.getValues()[9] <= 0.25){

                logData.print();

                time[1]++;

            }

        }System.out.println("共有" + time[1] + "个深度点");

        System.out.println("");

        System.out.println("Ⅲ 类储层:");

        for (LogData logData : logDatalList){

            if (logData.getValues()[10] > 0.05 && logData.getValues()[10] <= 0.08 && logData.getValues()[9] <= 0.25){

                logData.print();

                time[2]++;

            }

        }System.out.println("共有" + time[2] + "个深度点");

        System.out.println("");

        System.out.println("Ⅳ 类储层:");

        for (LogData logData : logDatalList){

            if (logData.getValues()[10] <= 0.05 && logData.getValues()[9] <= 0.25){

                logData.print();

                time[3]++;

            }

        }System.out.println("共有" + time[3] + "个深度点");

    }

    // 显示查询好油层深度点处理成果数据

    public void showFullOfOil(List<LogData> logDatalList){

        System.out.println("好油层深度点数据：");

        for (LogData logData : logDatalList){

            if (logData.getValues()[9] <= 0.25 && logData.getValues()[10] > 0.06 && logData.getValues()[11] >= 0.6){

                logData.print();

            }

        }

    }

}

class LogData {

    private float depth;

    private float[] values;

    public LogData(float depth, float[] values) {

        this.depth = depth;

        this.values = values;

    }

    // 获取深度值

    public float getDepth() {

        return depth;

    }

    // 获取值数组

    public float[] getValues() {

        return values;

    }

    public void setValues(float[] values) {

        this.values = values;

    }

    public void print() {

        System.out.printf("%-10.4f", depth);

        for (float value : values) {

            System.out.printf("%-10.4f", value);

        }

        System.out.println();

    }

}

class LogParameters {

    private Map<String, Float> parameters;

    public LogParameters() {

        parameters = new HashMap<>();

    }

    public void loadParametersFromFile(String filePath) throws IOException {

        BufferedReader reader = new BufferedReader(new FileReader(filePath));

        String line;

        // 跳过文件头部

        reader.readLine();

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

            String[] parts = line.trim().split("\\s+");

            if (parts.length == 3) {

                String name = parts[1];

                float value = Float.parseFloat(parts[2]);

                parameters.put(name, value);

            }

        }

        reader.close();

    }

    public Float getParameter(String name) {

        return parameters.get(name);

    }

    public void printParameters() {

        for (Map.Entry<String, Float> entry : parameters.entrySet()){

            System.out.println(entry.getKey() + " = " + entry.getValue());

        }

    }

}

格式字数要求(最终报告确定本部分内容)：前七部分字数不少于3000字；宋体小四字号，1.25倍行距，A4双面打印。