**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](#_Toc167972003)

[二、 系统需求分析 1](#_Toc167972004)

[三、 系统设计 1](#_Toc167972005)

[四、 系统实现 2](#_Toc167972006)

[五、 系统测试 4](#_Toc167972007)

[六、 使用说明 11](#_Toc167972008)

[七、 总结 12](#_Toc167972009)

[八、 附录：源程序代码 13](#_Toc167972010)

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

创建三个类：LogParameters、LogData和LogDataProcessor。

1. LogParameters类实现参数的读取，能够读取参数文件并将参数加载到系统中；
2. LogData类用于存储每个深度点的测井数据，并提供打印数据的功能；
3. LogDataProcessor类实现主要的处理逻辑，包括读取测井数据、计算泥质含量和其他要求数据、打印测井数据、对数据饱和度排序输出打印、统计储层分级和好油层等功能；
4. 主类包含了main方法，这是程序的入口点，main方法实现了一个简单的菜单，通过用户输入选择调用不同的功能。
5. 系统实现

LogParameters:

1. loadParametersFromFile(String filePath):从文件filePath中加载参数到系统。

2. getParameter(String name):获取参数name的值。

3. printParameters():打印所加载的所有参数。

class LogParameters {

    private Map<String, Float> parameters;

    public LogParameters() {…

    public void loadParametersFromFile(String filePath) throws IOException {…

    public Float getParameter(String name) {…

    public void printParameters() {…

}

LogData:

1. getDepth():获取深度值。
2. getValues():获取测井数据。
3. setValues(float[] values):更新测井数据。
4. print():打印单个深度点的数据。

class LogData {

    private float depth;

    private float[] values;

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

    public float getDepth() {…

    public float[] getValues() {…

    public void setValues(float[] values) {…

    public void print() {…

}

LogDataProcessor:

1. readLogData():从文件读取测井数据，读取第15到30行有depth的数据。
2. processLogData():计算读取数据的泥质含量、孔隙度、饱和度及其均值和极值。
3. printFirstLine()、printLogData():打印计算完成的测井数据。
4. printStatistics():打印泥质含量、孔隙度、饱和度的极值和平均值。
5. sortBySO():按饱和度从大到小打印深度点测井数据。
6. classByPOR():对各深度点进行储层分级。
7. showFullOfOil():显示好油层深度点测井数据。

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;

    String[] types = {};

    public LogParameters parameters;

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

    public List<LogData> readLogData() throws FileNotFoundException {…

    public void processLogData(List<LogData> logDataList) {…

    public void printFirstLine() {…

    public void printLogData(List<LogData> logDataList) {…

    public void printStatistics(List<LogData> logDataList){…

    public void sortBySO(List<LogData> logDatalList){…

    public void classByPOR(List<LogData> logDatalList){…

    public void showFullOfOil(List<LogData> logDatalList){…

}

1. 系统测试

运行结果如下：

---------菜单----------

0. 退出

1. 查看原始测井数据和处理参数

2. 计算泥质含量、孔隙度、饱和度并输出

3. 查看单深度点处理成果数据条

4. 统计查看储层参数的极值和平均值

5. 按饱和度对处理成果数据表排序

6. 统计显示不同等级储层信息

7. 统计查看好油层信息

请选择一个选项：1

原始测井数据：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #DEPTH | CAL | DT | GR | LLD | LLS | MSFL | NPHI | RHOB | SP |
| 193.8750 | 15.5330 | 103.3140 | 69.0110 | 9.3950 | 3.6870 | 1.7050 | 0.2580 | 2.3470 | -6.3330 |
| 194.0000 | 15.5360 | 103.4530 | 69.9920 | 9.0110 | 3.7710 | 1.6630 | 0.2620 | 2.3440 | -6.7670 |
| 194.1250 | 15.5360 | 103.6110 | 69.6520 | 10.3410 | 3.7960 | 1.6640 | 0.2620 | 2.3420 | -6.0510 |
| 194.2500 | 15.5330 | 103.8490 | 69.4150 | 11.0940 | 3.7990 | 1.5460 | 0.2640 | 2.3420 | -4.7170 |
| 194.3750 | 15.5390 | 104.1140 | 72.2370 | 8.1610 | 3.8870 | 1.7640 | 0.2760 | 2.3430 | -4.5980 |
| 194.5000 | 15.5390 | 104.0120 | 73.3130 | 9.1460 | 3.8520 | 1.8780 | 0.2680 | 2.3500 | -4.3420 |
| 194.6250 | 15.5560 | 103.7280 | 75.2520 | 9.2660 | 3.7390 | 2.3280 | 0.2670 | 2.3560 | -3.8140 |
| 194.7500 | 15.6010 | 103.4890 | 74.7430 | 8.7900 | 3.7000 | 2.0660 | 0.2570 | 2.3590 | -3.5740 |
| 194.8750 | 15.6240 | 103.4210 | 73.9560 | 9.5680 | 3.7260 | 1.8740 | 0.2530 | 2.3580 | -3.2910 |
| 195.0000 | 15.6120 | 103.6400 | 73.9280 | 10.9020 | 3.7520 | 1.8290 | 0.2560 | 2.3520 | -2.7700 |
| 195.1250 | 15.6250 | 104.2510 | 72.2540 | 7.8440 | 3.6800 | 1.9350 | 0.2490 | 2.3440 | -2.3100 |
| 195.2500 | 15.5990 | 104.3530 | 72.7670 | 9.0500 | 3.6300 | 1.8290 | 0.2560 | 2.3360 | -2.0280 |
| 195.3750 | 15.6140 | 104.6980 | 71.1770 | 7.8560 | 3.7130 | 1.8020 | 0.2550 | 2.3340 | -2.0700 |
| 195.5000 | 15.6290 | 104.1740 | 70.9800 | 7.7380 | 3.7550 | 1.7310 | 0.2560 | 2.3320 | -1.9350 |
| 195.6250 | 15.6330 | 103.0160 | 71.7950 | 8.9240 | 3.6840 | 1.8440 | 0.2550 | 2.3330 | -1.7010 |
| 195.7500 | 15.6300 | 102.1120 | 71.9790 | 9.5540 | 3.5380 | 2.1690 | 0.2410 | 2.3400 | -1.6800 |

参数信息：

a = 0.66

b = 1.0

DTma = 88.0

Rw = 0.04

GRmax = 156.001

DTf = 189.0

GCUR = 3.7

GRmin = 52.212

RHOma = 2.6

RHOf = 1.0

m = 1.6496

n = 2.0

---------菜单----------

0. 退出

1. 查看原始测井数据和处理参数

2. 计算泥质含量、孔隙度、饱和度并输出

3. 查看单深度点处理成果数据条

4. 统计查看储层参数的极值和平均值

5. 按饱和度对处理成果数据表排序

6. 统计显示不同等级储层信息

7. 统计查看好油层信息

请选择一个选项：2

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#DEPTH** | **CAL** | **DT** | **GR** | **LLD** | **LLS** | **MSFL** | **NPHI** | **RHOB** | **SP** | **VSH** | **POR** | **SO** |
| **193.8750** | **15.5330** | **103.3140** | **69.0110** | **9.3950** | **3.6870** | **1.7050** | **0.2580** | **2.3470** | **-6.3330** | **0.0429** | **0.1516** | **0.7488** |
| **194.0000** | **15.5360** | **103.4530** | **69.9920** | **9.0110** | **3.7710** | **1.6630** | **0.2620** | **2.3440** | **-6.7670** | **0.0460** | **0.1530** | **0.7454** |
| **194.1250** | **15.5360** | **103.6110** | **69.6520** | **10.3410** | **3.7960** | **1.6640** | **0.2620** | **2.3420** | **-6.0510** | **0.0449** | **0.1546** | **0.7643** |
| **194.2500** | **15.5330** | **103.8490** | **69.4150** | **11.0940** | **3.7990** | **1.5460** | **0.2640** | **2.3420** | **-4.7170** | **0.0442** | **0.1569** | **0.7753** |
| **194.3750** | **15.5390** | **104.1140** | **72.2370** | **8.1610** | **3.8870** | **1.7640** | **0.2760** | **2.3430** | **-4.5980** | **0.0534** | **0.1595** | **0.7415** |
| **194.5000** | **15.5390** | **104.0120** | **73.3130** | **9.1460** | **3.8520** | **1.8780** | **0.2680** | **2.3500** | **-4.3420** | **0.0571** | **0.1585** | **0.7546** |
| **194.6250** | **15.5560** | **103.7280** | **75.2520** | **9.2660** | **3.7390** | **2.3280** | **0.2670** | **2.3560** | **-3.8140** | **0.0639** | **0.1557** | **0.7525** |
| **194.7500** | **15.6010** | **103.4890** | **74.7430** | **8.7900** | **3.7000** | **2.0660** | **0.2570** | **2.3590** | **-3.5740** | **0.0621** | **0.1534** | **0.7427** |
| **194.8750** | **15.6240** | **103.4210** | **73.9560** | **9.5680** | **3.7260** | **1.8740** | **0.2530** | **2.3580** | **-3.2910** | **0.0593** | **0.1527** | **0.7525** |
| **195.0000** | **15.6120** | **103.6400** | **73.9280** | **10.9020** | **3.7520** | **1.8290** | **0.2560** | **2.3520** | **-2.7700** | **0.0592** | **0.1549** | **0.7708** |
| **195.1250** | **15.6250** | **104.2510** | **72.2540** | **7.8440** | **3.6800** | **1.9350** | **0.2490** | **2.3440** | **-2.3100** | **0.0534** | **0.1609** | **0.7382** |
| **195.2500** | **15.5990** | **104.3530** | **72.7670** | **9.0500** | **3.6300** | **1.8290** | **0.2560** | **2.3360** | **-2.0280** | **0.0552** | **0.1619** | **0.7575** |
| **195.3750** | **15.6140** | **104.6980** | **71.1770** | **7.8560** | **3.7130** | **1.8020** | **0.2550** | **2.3340** | **-2.0700** | **0.0498** | **0.1653** | **0.7442** |
| **195.5000** | **15.6290** | **104.1740** | **70.9800** | **7.7380** | **3.7550** | **1.7310** | **0.2560** | **2.3320** | **-1.9350** | **0.0492** | **0.1601** | **0.7354** |
| **195.6250** | **15.6330** | **103.0160** | **71.7950** | **8.9240** | **3.6840** | **1.8440** | **0.2550** | **2.3330** | **-1.7010** | **0.0519** | **0.1487** | **0.7380** |
| **195.7500** | **15.6300** | **102.1120** | **71.9790** | **9.5540** | **3.5380** | **2.1690** | **0.2410** | **2.3400** | **-1.6800** | **0.0525** | **0.1397** | **0.7335** |

---------菜单----------

0. 退出

1. 查看原始测井数据和处理参数

2. 计算泥质含量、孔隙度、饱和度并输出

3. 查看单深度点处理成果数据条

4. 统计查看储层参数的极值和平均值

5. 按饱和度对处理成果数据表排序

6. 统计显示不同等级储层信息

7. 统计查看好油层信息

请选择一个选项：3

请输入要查看的深度点(0~15)：4

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#DEPTH** | **CAL** | **DT** | **GR** | **LLD** | **LLS** | **MSFL** | **NPHI** | **RHOB** | **SP** | **VSH** | **POR** | **SO** |
| **194.3750** | **15.5390** | **104.1140** | **72.2370** | **8.1610** | **3.8870** | **1.7640** | **0.2760** | **2.3430** | **-4.5980** | **0.0534** | **0.1595** | **0.7415** |

---------菜单----------

0. 退出

1. 查看原始测井数据和处理参数

2. 计算泥质含量、孔隙度、饱和度并输出

3. 查看单深度点处理成果数据条

4. 统计查看储层参数的极值和平均值

5. 按饱和度对处理成果数据表排序

6. 统计显示不同等级储层信息

7. 统计查看好油层信息

请选择一个选项：4

VSHmax = 0.06394354971443271

VSHmin = 0.052499119484068726

VSHavg = 0.0528071448936232

PORmax = 0.16532671451568604

PORmin = 0.1397227644920349

PORavg = 0.15546719636768103

SOmax = 0.7752702008888386

SOmin = 0.7335025874513397

SOavg = 0.7497009261249894

---------菜单----------

0. 退出

1. 查看原始测井数据和处理参数

2. 计算泥质含量、孔隙度、饱和度并输出

3. 查看单深度点处理成果数据条

4. 统计查看储层参数的极值和平均值

5. 按饱和度对处理成果数据表排序

6. 统计显示不同等级储层信息

7. 统计查看好油层信息

请选择一个选项：5

按照饱和度由大到小排序：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#DEPTH** | **CAL** | **DT** | **GR** | **LLD** | **LLS** | **MSFL** | **NPHI** | **RHOB** | **SP** | **VSH** | **POR** | **SO** |
| **194.2500** | **15.5330** | **103.8490** | **69.4150** | **11.0940** | **3.7990** | **1.5460** | **0.2640** | **2.3420** | **-4.7170** | **0.0442** | **0.1569** | **0.7753** |
| **195.0000** | **15.6120** | **103.6400** | **73.9280** | **10.9020** | **3.7520** | **1.8290** | **0.2560** | **2.3520** | **-2.7700** | **0.0592** | **0.1549** | **0.7708** |
| **194.1250** | **15.5360** | **103.6110** | **69.6520** | **10.3410** | **3.7960** | **1.6640** | **0.2620** | **2.3420** | **-6.0510** | **0.0449** | **0.1546** | **0.7643** |
| **195.2500** | **15.5990** | **104.3530** | **72.7670** | **9.0500** | **3.6300** | **1.8290** | **0.2560** | **2.3360** | **-2.0280** | **0.0552** | **0.1619** | **0.7575** |
| **194.5000** | **15.5390** | **104.0120** | **73.3130** | **9.1460** | **3.8520** | **1.8780** | **0.2680** | **2.3500** | **-4.3420** | **0.0571** | **0.1585** | **0.7546** |
| **194.6250** | **15.5560** | **103.7280** | **75.2520** | **9.2660** | **3.7390** | **2.3280** | **0.2670** | **2.3560** | **-3.8140** | **0.0639** | **0.1557** | **0.7525** |
| **194.8750** | **15.6240** | **103.4210** | **73.9560** | **9.5680** | **3.7260** | **1.8740** | **0.2530** | **2.3580** | **-3.2910** | **0.0593** | **0.1527** | **0.7525** |
| **193.8750** | **15.5330** | **103.3140** | **69.0110** | **9.3950** | **3.6870** | **1.7050** | **0.2580** | **2.3470** | **-6.3330** | **0.0429** | **0.1516** | **0.7488** |
| **194.0000** | **15.5360** | **103.4530** | **69.9920** | **9.0110** | **3.7710** | **1.6630** | **0.2620** | **2.3440** | **-6.7670** | **0.0460** | **0.1530** | **0.7454** |
| **195.3750** | **15.6140** | **104.6980** | **71.1770** | **7.8560** | **3.7130** | **1.8020** | **0.2550** | **2.3340** | **-2.0700** | **0.0498** | **0.1653** | **0.7442** |
| **194.7500** | **15.6010** | **103.4890** | **74.7430** | **8.7900** | **3.7000** | **2.0660** | **0.2570** | **2.3590** | **-3.5740** | **0.0621** | **0.1534** | **0.7427** |
| **194.3750** | **15.5390** | **104.1140** | **72.2370** | **8.1610** | **3.8870** | **1.7640** | **0.2760** | **2.3430** | **-4.5980** | **0.0534** | **0.1595** | **0.7415** |
| **195.1250** | **15.6250** | **104.2510** | **72.2540** | **7.8440** | **3.6800** | **1.9350** | **0.2490** | **2.3440** | **-2.3100** | **0.0534** | **0.1609** | **0.7382** |
| **195.6250** | **15.6330** | **103.0160** | **71.7950** | **8.9240** | **3.6840** | **1.8440** | **0.2550** | **2.3330** | **-1.7010** | **0.0519** | **0.1487** | **0.7380** |
| **195.5000** | **15.6290** | **104.1740** | **70.9800** | **7.7380** | **3.7550** | **1.7310** | **0.2560** | **2.3320** | **-1.9350** | **0.0492** | **0.1601** | **0.7354** |
| **195.7500** | **15.6300** | **102.1120** | **71.9790** | **9.5540** | **3.5380** | **2.1690** | **0.2410** | **2.3400** | **-1.6800** | **0.0525** | **0.1397** | **0.7335** |

---------菜单----------

0. 退出

1. 查看原始测井数据和处理参数

2. 计算泥质含量、孔隙度、饱和度并输出

3. 查看单深度点处理成果数据条

4. 统计查看储层参数的极值和平均值

5. 按饱和度对处理成果数据表排序

6. 统计显示不同等级储层信息

7. 统计查看好油层信息

请选择一个选项：6

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#DEPTH** | **CAL** | **DT** | **GR** | **LLD** | **LLS** | **MSFL** | **NPHI** | **RHOB** | **SP** | **VSH** | **POR** | **SO** |

Ⅰ 类储层:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **193.8750** | **15.5330** | **103.3140** | **69.0110** | **9.3950** | **3.6870** | **1.7050** | **0.2580** | **2.3470** | **-6.3330** | **0.0429** | **0.1516** | **0.7488** |
| **194.0000** | **15.5360** | **103.4530** | **69.9920** | **9.0110** | **3.7710** | **1.6630** | **0.2620** | **2.3440** | **-6.7670** | **0.0460** | **0.1530** | **0.7454** |
| **194.1250** | **15.5360** | **103.6110** | **69.6520** | **10.3410** | **3.7960** | **1.6640** | **0.2620** | **2.3420** | **-6.0510** | **0.0449** | **0.1546** | **0.7643** |
| **194.2500** | **15.5330** | **103.8490** | **69.4150** | **11.0940** | **3.7990** | **1.5460** | **0.2640** | **2.3420** | **-4.7170** | **0.0442** | **0.1569** | **0.7753** |
| **194.3750** | **15.5390** | **104.1140** | **72.2370** | **8.1610** | **3.8870** | **1.7640** | **0.2760** | **2.3430** | **-4.5980** | **0.0534** | **0.1595** | **0.7415** |
| **194.5000** | **15.5390** | **104.0120** | **73.3130** | **9.1460** | **3.8520** | **1.8780** | **0.2680** | **2.3500** | **-4.3420** | **0.0571** | **0.1585** | **0.7546** |
| **194.6250** | **15.5560** | **103.7280** | **75.2520** | **9.2660** | **3.7390** | **2.3280** | **0.2670** | **2.3560** | **-3.8140** | **0.0639** | **0.1557** | **0.7525** |
| **194.7500** | **15.6010** | **103.4890** | **74.7430** | **8.7900** | **3.7000** | **2.0660** | **0.2570** | **2.3590** | **-3.5740** | **0.0621** | **0.1534** | **0.7427** |
| **194.8750** | **15.6240** | **103.4210** | **73.9560** | **9.5680** | **3.7260** | **1.8740** | **0.2530** | **2.3580** | **-3.2910** | **0.0593** | **0.1527** | **0.7525** |
| **195.0000** | **15.6120** | **103.6400** | **73.9280** | **10.9020** | **3.7520** | **1.8290** | **0.2560** | **2.3520** | **-2.7700** | **0.0592** | **0.1549** | **0.7708** |
| **195.1250** | **15.6250** | **104.2510** | **72.2540** | **7.8440** | **3.6800** | **1.9350** | **0.2490** | **2.3440** | **-2.3100** | **0.0534** | **0.1609** | **0.7382** |
| **195.2500** | **15.5990** | **104.3530** | **72.7670** | **9.0500** | **3.6300** | **1.8290** | **0.2560** | **2.3360** | **-2.0280** | **0.0552** | **0.1619** | **0.7575** |
| **195.3750** | **15.6140** | **104.6980** | **71.1770** | **7.8560** | **3.7130** | **1.8020** | **0.2550** | **2.3340** | **-2.0700** | **0.0498** | **0.1653** | **0.7442** |
| **195.5000** | **15.6290** | **104.1740** | **70.9800** | **7.7380** | **3.7550** | **1.7310** | **0.2560** | **2.3320** | **-1.9350** | **0.0492** | **0.1601** | **0.7354** |
| **195.6250** | **15.6330** | **103.0160** | **71.7950** | **8.9240** | **3.6840** | **1.8440** | **0.2550** | **2.3330** | **-1.7010** | **0.0519** | **0.1487** | **0.7380** |
| **195.7500** | **15.6300** | **102.1120** | **71.9790** | **9.5540** | **3.5380** | **2.1690** | **0.2410** | **2.3400** | **-1.6800** | **0.0525** | **0.1397** | **0.7335** |

共有16个深度点

Ⅱ 类储层:

共有0个深度点

Ⅲ 类储层:

共有0个深度点

Ⅳ 类储层:

共有0个深度点

---------菜单----------

0. 退出

1. 查看原始测井数据和处理参数

2. 计算泥质含量、孔隙度、饱和度并输出

3. 查看单深度点处理成果数据条

4. 统计查看储层参数的极值和平均值

5. 按饱和度对处理成果数据表排序

6. 统计显示不同等级储层信息

7. 统计查看好油层信息

请选择一个选项：7

好油层深度点数据：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **193.8750** | **15.5330** | **103.3140** | **69.0110** | **9.3950** | **3.6870** | **1.7050** | **0.2580** | **2.3470** | **-6.3330** | **0.0429** | **0.1516** | **0.7488** |
| **194.0000** | **15.5360** | **103.4530** | **69.9920** | **9.0110** | **3.7710** | **1.6630** | **0.2620** | **2.3440** | **-6.7670** | **0.0460** | **0.1530** | **0.7454** |
| **194.1250** | **15.5360** | **103.6110** | **69.6520** | **10.3410** | **3.7960** | **1.6640** | **0.2620** | **2.3420** | **-6.0510** | **0.0449** | **0.1546** | **0.7643** |
| **194.2500** | **15.5330** | **103.8490** | **69.4150** | **11.0940** | **3.7990** | **1.5460** | **0.2640** | **2.3420** | **-4.7170** | **0.0442** | **0.1569** | **0.7753** |
| **194.3750** | **15.5390** | **104.1140** | **72.2370** | **8.1610** | **3.8870** | **1.7640** | **0.2760** | **2.3430** | **-4.5980** | **0.0534** | **0.1595** | **0.7415** |
| **194.5000** | **15.5390** | **104.0120** | **73.3130** | **9.1460** | **3.8520** | **1.8780** | **0.2680** | **2.3500** | **-4.3420** | **0.0571** | **0.1585** | **0.7546** |
| **194.6250** | **15.5560** | **103.7280** | **75.2520** | **9.2660** | **3.7390** | **2.3280** | **0.2670** | **2.3560** | **-3.8140** | **0.0639** | **0.1557** | **0.7525** |
| **194.7500** | **15.6010** | **103.4890** | **74.7430** | **8.7900** | **3.7000** | **2.0660** | **0.2570** | **2.3590** | **-3.5740** | **0.0621** | **0.1534** | **0.7427** |
| **194.8750** | **15.6240** | **103.4210** | **73.9560** | **9.5680** | **3.7260** | **1.8740** | **0.2530** | **2.3580** | **-3.2910** | **0.0593** | **0.1527** | **0.7525** |
| **195.0000** | **15.6120** | **103.6400** | **73.9280** | **10.9020** | **3.7520** | **1.8290** | **0.2560** | **2.3520** | **-2.7700** | **0.0592** | **0.1549** | **0.7708** |
| **195.1250** | **15.6250** | **104.2510** | **72.2540** | **7.8440** | **3.6800** | **1.9350** | **0.2490** | **2.3440** | **-2.3100** | **0.0534** | **0.1609** | **0.7382** |
| **195.2500** | **15.5990** | **104.3530** | **72.7670** | **9.0500** | **3.6300** | **1.8290** | **0.2560** | **2.3360** | **-2.0280** | **0.0552** | **0.1619** | **0.7575** |
| **195.3750** | **15.6140** | **104.6980** | **71.1770** | **7.8560** | **3.7130** | **1.8020** | **0.2550** | **2.3340** | **-2.0700** | **0.0498** | **0.1653** | **0.7442** |
| **195.5000** | **15.6290** | **104.1740** | **70.9800** | **7.7380** | **3.7550** | **1.7310** | **0.2560** | **2.3320** | **-1.9350** | **0.0492** | **0.1601** | **0.7354** |
| **195.6250** | **15.6330** | **103.0160** | **71.7950** | **8.9240** | **3.6840** | **1.8440** | **0.2550** | **2.3330** | **-1.7010** | **0.0519** | **0.1487** | **0.7380** |
| **195.7500** | **15.6300** | **102.1120** | **71.9790** | **9.5540** | **3.5380** | **2.1690** | **0.2410** | **2.3400** | **-1.6800** | **0.0525** | **0.1397** | **0.7335** |

---------菜单----------

0. 退出

1. 查看原始测井数据和处理参数

2. 计算泥质含量、孔隙度、饱和度并输出

3. 查看单深度点处理成果数据条

4. 统计查看储层参数的极值和平均值

5. 按饱和度对处理成果数据表排序

6. 统计显示不同等级储层信息

7. 统计查看好油层信息

请选择一个选项：0

退出程序

1. 使用说明

确保在D盘目录下有参数文件parameters.txt和测井数据文件well\_logging\_data.txt。运行Java文件能够启动此测井资料处理分析系统，菜单将显示0~7的选项能够分别实现不同的功能，功能如下：

0. 退出

1. 查看原始测井数据和处理参数

2. 计算泥质含量、孔隙度、饱和度并输出

3. 查看单深度点处理成果数据条

4. 统计查看储层参数的极值和平均值

5. 按饱和度对处理成果数据表排序

6. 统计显示不同等级储层信息

7. 统计查看好油层信息

键盘输入数字(只能输入整数数字，不然程序会报错)，系统将实现相应的功能。当系统实现选项3时，系统将会提示选择一个深度点：

请输入要查看的深度点(0~15)：

从键盘输入0~15中的一个深度点，系统将会打印相应深度点的测井数据。在系统完成一次选项后，会再次运行，提示选择选项，直到选择0退出程序。

1. 总结

该系统是一个简易的测井资料处理分析系统，通过菜单调用各个子方法，分别实现数据提取与检查、测井数据处理及处理结果分析等功能。主要功能模块包括查看原始测井数据和处理参数、计算孔隙度、泥质含量、含油饱和度以及对处理结果进行分析和统计。

此简易测井资料处理分析系统通过自动化的数据提取、处理和分析功能，不仅提高了工作效率和数据处理的准确性，还为油气田的勘探和开发提供了重要的决策支持工具。通过科学合理的参数计算和储层评价，系统可以显著提升油气资源开发的经济效益，具有广泛的现实应用价值。

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。