西南民族大学

**实验报告**

**2019------2020**学年第**2**学期

课程名称：软件工程课程设计

学院：计算机科学与技术 专业：计算机科学与技术年级：2017级 班级：1702

学号：201731102242 姓名：谢晓辉

同组人：无

|  |
| --- |
| 西南民族大学学生实验报告  教学单位：计科学院 实验室名称：BS-226 实验时间：2020 年 3 月 19 日  姓名：谢晓辉 专业：计算机科学与技术 班级：1702 学号:201731102242 |
| 实验项目名称：SE实践4 实验成绩： 教师签名：周绪川 |
| **一、实验目的**  1.在上次实验的基础上通过自主查找资料并学习完成此次个人编程。  2.通过个人编程的完成过程来进一步学习编程的思想。  **二、材料与方法**  PC机一台、GitHub、Eclipse、exe4j等 。  **三、实验主要过程与结果**  1.实验环境  （1）语言要求  JAVA  （2）开发环境  内存8G及以上。  （3）工具  Eclipse、exe4j。  2.安装exe4j工具并进行自主学习  （1）在网站上寻到下载地址，并点击下载。  （2）下载完成后根据提示进行安装。  3.编程  （1）内容  在实验三的基础上继续，输入文件为yq\_in.txt，输出文件yq\_out.txt。  （要求）①每个省后面有一个总数；  ②输出省按总数从大到小排序；如果两个省总数一样，按拼音（字母）排序；  ③每个省内各市从大到小排序；如果两个市总数一样，按拼音（字母）排序。  （2）解决思路  因为本次实验要求在cmd中输入输入文件名和输出文件名，因此不需要在Eclipse源代码中确定文件名及路径。  根据本次实验要求来看，需完成全省的汇总、省份的排序以及省份内的城市排序。  排序的优先原则为数量优先级最高，其次是省份或城市拼音的字母优先级。  数据的灵活处理需借助辅助标记。  （3）流程  ①创建数组rank对省份的城市进行分割；创建数组pro进行省份的排序；使用HashMap表示省份:总数；设置中文拼音进排序的语句。  ②创建一个列表list，用来装按行读取到的内容。  ③按行读取文件中的内容。   1. 使用InputStreamReader类（从字节流到字符流的桥接器）。（为了获得最高效率，选择在BufferedReader中包装InputStreamReader） 2. 设置tempString字符串，用reader.readLine()读出一行，并将成功读出的内容传给tempString字符串。 3. 关闭读的接口。   ④进行省份的排序。  ⑤按照省份对省份内的城市进行排序。  ⑥需得到[省份，起始，结束]，设定第一个起始位置为0。  ⑦主函数   1. 按行读取文档的内容到list（list规模设置为128）。 2. 设置总人数，进行省份和总数的统计。 3. 进行结果的输出。   （4）代码  **package** city;  **import** java.text.Collator;  **import** java.util.\*;  **import** java.io.BufferedReader;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.FileOutputStream;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **import** java.io.OutputStreamWriter;  **public** **class** Test\_report {  //对省份的城市进行分割  **static** String[][] *rank*=**new** String[9][3];  //进行省份的排序  **static** String[][] *pro* =**new** String[9][2];  **static** Map<Object,Object> *dic* =**new** HashMap<Object,Object>();  **static** Comparator<Object> *com*=Collator.*getInstance*(java.util.Locale.***CHINA***);  //创建一个列表,用来装按行读取到的内容  **static** List<String> *list*=**new** ArrayList<>();    **public** **static** **void** readFileByLines(String filename) **throws** IOException {  File file = **new** File(filename);  BufferedReader reader=**null**;  InputStreamReader inputStreamReader = **new** InputStreamReader(**new** FileInputStream(file), "GBK");  reader = **new** BufferedReader(inputStreamReader);  String tempString;  **while**((tempString = reader.readLine())!= **null**)  {  *list*.add(tempString);  }  inputStreamReader.close();  reader.close();  }    **public** **static** **void** change(String[][] str) {  String[][] exc = **new** String[str.length][];  **for**(**int** j=0;j<str.length;j++)  {  **for**(**int** k=j+1;k<str.length;k++)  {  **if**(Integer.*valueOf*(str[j][1])>Integer.*valueOf*(str[k][1]))  {  **continue**;  }  **else** **if**(Integer.*valueOf*(str[j][1])<Integer.*valueOf*(str[k][1]))  {  exc[j]=str[k];  str[k]=str[j];  str[j]=exc[j];  }  **else**  {  **if**(*com*.compare(str[k][0],str[j][0])>=0)  {  **continue**;  }  **else**  {  exc[j]=str[k];  str[k]=str[j];  str[j]=exc[j];  }  }  }  }  }    //按照省份对省份内的城市进行排序  //a为一个省份的起始位置，b为一个省份的结束位置  **public** **static** **void** exchange(String[][] str,**int** a,**int** b) {  String[][] exc = **new** String[str.length][];  **for**(**int** j=a;j<b-1;j++)  {  **for**(**int** k=j+1;k<b-1;k++)  {  **if**(Integer.*valueOf*(str[j][2])>Integer.*valueOf*(str[k][2]))  {  **continue**;  }  **else** **if**(Integer.*valueOf*(str[j][2])<Integer.*valueOf*(str[k][2]))  {  exc[j]=str[k];  str[k]=str[j];  str[j]=exc[j];  }  **else**  {  **if**(*com*.compare(str[k][1],str[j][1])>=0)  {  **continue**;  }  **else**  {  exc[j]=str[k];  str[k]=str[j];  str[j]=exc[j];  }  }  }  }  }    **public** **static** **void** rank(String[][] str) {  String s= str[0][0];  **int** count=0;  //设定第一个起始位置为0  *rank*[count][1]=String.*valueOf*(0);  **for**(**int** b=0;b<str.length;b++)  {  **if**(str[b][0].equals(s))  {  **continue**;  }  **else**  {  *rank*[count][0]=str[b-1][0];  *rank*[count][2]=String.*valueOf*(b-1);  count=count+1;  *rank*[count][1]=String.*valueOf*(b);  s=str[b][0];  }  }  *rank*[count][0]=str[str.length-1][0];  *rank*[count][2]=String.*valueOf*(str.length-1);  }  **public** **static** **void** main(String[] args) **throws** IOException {  *readFileByLines*(args[0]);  System.***out***.println(*list*);  String[][] string = **new** String[*list*.size()][];  **int** sum=0;  **for**(**int** i = 0;i<*list*.size();i++)  {  string[i] = *list*.get(i).split("\\s+");  }  String loc=string[0][0];  **for**(**int** r=0;r<string.length;r++)  {  **if**(string[r][0].equals(loc))  {  sum+=Integer.*valueOf*(string[r][2]);  }  **else**  {  *dic*.put(string[r-1][0], sum);  sum=Integer.*valueOf*(string[r][2]);  loc=string[r][0];  }  }  **int** count=0;  *dic*.put(string[string.length-1][0], sum);  **for**(Object key :*dic*.keySet())  {  *pro*[count][0]=key.toString();  *pro*[count][1]=*dic*.get(key).toString();  count++;  }  *change*(*pro*);  System.***out***.println("按总人数从多到少的排序方法进行省份的排序：");  **for**(String[] s :*pro*) {  **for**(String r :s) {  System.***out***.println(r+" ");  }  }  *rank*(string);  System.***out***.println("各省份及其所包含城市的范围：");  **for**(String[] s :*rank*)  {  **for**(String r :s)  {  System.***out***.print(r+" ");  }  }  String location=*pro*[0][0];  **for**(**int** j =0 ;j < *pro*.length;j++)  {  **for**(**int** i=0;i<*rank*.length;i++)  {  **if**(*rank*[i][0].equals(*pro*[j][0]))  {  *exchange*(string,Integer.*valueOf*(*rank*[i][1]),Integer.*valueOf*(*rank*[i][2]));  }  **else**  {  **continue**;  }  }  }  File file = **new** File(args[1]);  FileOutputStream out = **new** FileOutputStream(file);  OutputStreamWriter wr =**new** OutputStreamWriter(out);  **if**(args.length<2||args.length>3)  {  System.***out***.println("参数出错");  }  **else** **if**(args.length==2)  {  //先写入人数最多的省份并加换行到文档  wr.write(location+" "+*pro*[0][1]+"\r\n\r\n");  **for**(**int** j =0 ;j < *pro*.length;j++)  {  **for**(**int** i=0;i<string.length;i++)  {  **if**(string[i][0].equals(*pro*[j][0]))  {  wr.write(string[i][1]+" "+string[i][2]+"\r\n");  }  **else**  {  **continue**;  }  }  **if**(j!=*pro*.length-1)  {  location=*pro*[j+1][0];  wr.write(location+" "+*pro*[j+1][1]+"\r\n\r\n");  }  **else**  **break**;  }  }  **else**  {  location = args[2];  wr.write(location+" "+*pro*[0][1]+"\r\n\r\n");  **for**(**int** j =0 ;j < string.length;j++)  {  **if**(string[j][0].equals(location))  {  wr.write(string[j][1]+" "+string[j][2]+"\r\n");  }  **else**  {  **continue**;  }  }  }  wr.close();  }  }   1. 主要过程   （1）新建一个java项目Epidemic situation,在该项目下新建一个包city，在该包下新建一个类Test\_report。  12  图1.1 新建java项目Epidemic situation  34  图1.2 新建包city 图1.3 新建类Test\_report  （2）在新建的类页面编写代码。  1  图1.4 编写代码  （3）保存并把该项目打成jar包。  ①选择项目名称右击选择【Export...】，出现如图1.5所示的页面，展开【Java】选项，选择【JAR file】点击Next。  5  图1.5  ②点击Browse按钮，选择生成的JAR文件路径，其它选项可以暂时使用默认的，然后点击Next。  47  图1.6 jar文件规格 图1.7 jar包选项  ③直接点击Next按钮，在该页面的其它选项采用默认即可，点击Browse按钮选择项目中主类(带有main方法的类)，如图1.8所示。  8  图1.8 选择主类  ④直接点击【OK】后如图1.9所示,再点击【Finish】按钮，即可完成。  8.2  图1.9 jar清单规格  （4）将java程序转换为exe。  ①找到exe4j下面的bin目录，双击exe4j.exe运行。  10  图2.1 运行exe4j  ②直接点击Next按钮，选择"JAR in EXE" mode，如图2.2所示。  11  图2.2 选择项目类型  ③点击Next按钮，Short name of your application中可随便定义，我选择定义与类相同的名字Test\_report；Output directory中是用来指定生成exe文件的目录，如图2.3所示。7  图2.3 配置应用  ④点击Next按钮出现如图2.4所示的页面，点击Advanced Options后点击32bit or 64bit一栏后出现如图2.5所示的页面，勾选Generate 64-bit executable。  b  图2.4 配置可执行文件 图2.5 选择生成的可执行文件的主要结构  ⑤点击Next按钮出现如图2.6所示的页面后再次点击Next，在Executable name中输入要生成exe文件的名称，如图2.7所示。    图2.6 可执行清单的配置选项 图2.7 编辑生成的exe文件名  ⑥在点击Next按钮后出现的页面上点击右侧绿色的“+”出现如图2.8所示的页面。在Archive中选择我们第一步生成的jar文件，然后直接点击OK按钮，如图2.9所示；点击Main class 后面的按钮，选择程序的入口即程序含有main方法的类，点击OK按钮即可，如图2.10所示。    图2.8 java调用配置  815  图2.9 定义类路径入口 图2.10 选择主类  ⑦点击Next，在最低版本要求处填写了1.4，再次点击Next依次出现图2.12、2.13所示的页面，再次点击Next出现图2.14所示页面即可，即成功完成转换。    图2.11 jre配置    图2.12 配置闪屏 图2.13 配置可执行的消息  c  图2.14 转换完成  a  图2.15 成功生成Test\_report.exe  （5）注意  在导出jar包前需在Eclipse中设置传递main参数。  **点击工具栏Run按钮，在弹出来的选项当中点击Run Configurations。接下来在弹出的界面当中点击Arguments，并在Program arguments框里填写所传参数（输入文件与输出文件的绝对路径），多参数以空格隔开，如图2.15所示（两个参数）。**  **2**  图2.15  **填好以后点击Apply，再点击Run即可。**  Eclipse中运行结果如图2.16所示。  3  图2.16  此时在目标文件夹下也成功生成了yq\_out.txt,如图2.17所示。  5.1  5.2  图2.17  如果传递的参数为3个（图2.18），则目标文件夹下也会成功生成相应的yq\_out.txt,如图2.20所示。  6.1  图2.18  6.26.3  图2.19  5.结果测试  （1）在cmd中通过cd命令进入实验目录。  9  图3.1 进入实验目录  （2）使用jar -tvf Test\_report.jar命令查看该jar包的内容。  10  图3.2 查看jar包内容  （3）在前边已成功生成了Test\_report.exe，因此可在cmd中通过输入：Test\_report yq\_in.txt yq\_out.txt 来运行，如图3.3、3.4所示。（两个参数）  11.1  图3.3  11.2  11.311.4  图3.4  （5）在cmd中通过输入：Test\_report yq\_in.txt yq\_out.txt 浙江省 来运行，如图3.5、3.6所示。（三个参数）  12.1  图3.5  12.212.3  图3.6  （6）在cmd中通过输入：Test\_report yq\_in.txt yq\_out.txt 湖南省 来运行，如图3.7、3.8所示。（三个参数）  13.1  图3.7  13.213.3  图3.8  **四、分析讨论**  1.在上次实验的基础上完成了此次个人编程，自主查找了资料且在完成过程中遇到错误并进行思考解决，因此学到了很多。  2.HashMap是基于[哈希表](https://baike.so.com/doc/6174673-6387914.html" \t "https://baike.so.com/doc/_blank)的Map接口的实现，对于数据的灵活处理选择使用新的数组储存优先级顺序信息和分割省份之间的城市的信息作为辅助标记信息。  3.当BufferedReader在读取文本文件时，会先尽量从文件中读入字符数据并置入缓冲区，  而之后若使用read()方法，会先从缓冲区中进行读取。如果缓冲区数据不足，才会再从文件中读取。  4.使用BufferedWriter时，写入的数据并不会先输出到目的地，而是先存储至缓冲区中。如果缓冲区中的数据满了，才会一次对目的地进行写出。  5.在编写程序时注意添加注释，有利于理清自己的思维以及可在逐步调试或报错时发挥不可或缺的作用。  **五、教师评阅** |

注：实验报告的内容及格式可由学院根据学科专业特点确定；全校各专业必须使用学校统一封面。