西南民族大学

**实验报告**

2021------2022学年第1学期

课程名称：软件工程

学院：计算机科学与工程学院

专业：计算机科学与技术

年级：2019级 班级： 计科1903

学号：201931304029 姓名：雷文堃

同组人：无

|  |
| --- |
| 西南民族大学学生实验报告  教学单位：计算机科学与工程学院 实验室名称：BS223 实验时间：2022年3月17日  姓名：雷文堃 专业：计算机科学与技术 班级：计科1903 学号: 201931304029 |
| 实验项目名称: SE实践2 实验成绩： 教师签名：周绪川 |
| 实验报告内容（1、实验背景（目的、意义及原理等）；2、材料与方法；3、实验主要过程与结果；  4、分析讨论及实验心得；5、教师评阅）。  **一、实验目的与背景**  **在实验二的基础上继续：**  **1、输入文件名在命令行输入（即输入文件可以指定）；输出文件名在命令行输出（即输出文件可以指定）；**  **2、可以输出指定省的信息**  **具体示例：**  **>yq yq\_in\_03.txt yq\_out\_03.txt**  **//表示输入文件名为yq\_in\_03.txt，输出文件名为yq\_out\_03.txt，包括所有省份信息，格式和实验二一致**  **>yq yq\_in\_03.txt yq\_out\_03.txt 浙江省**  **//输出文件中仅包括浙江省的信息**  **在实验三的基础上继续：**  **输入文件为yq\_in\_04.txt，输出文件yq\_out\_04.txt,但有如下要求：**   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；**   **二、材料与方法**  **PC机一台、GitHub、JAVA、IDEA**  **三、实验主要过程与结果**  **基本思想：**  **实验三：**  **在上次实验的基础上，将test的main函数的参数改为args字符串数组索引得到，便可在命令行通过空格分割参数，实现外部输入参数的目的。关于指定省份输出数据，最简单的方式便是遍历往文件写入数据时，增加判定，只写入满足指定省份名字的数据。**  **实验四：**  **主旨是数据存储容器的选择以及Comparator排序接口的重写。分别创建拥有省份名、感染总人数、下属城市数据ArrayList三个属性的省份类和拥有城市名、感染人数两个属性的城市类，用于实例化之后存储各条数据，再把各条数据存储进ArrayList中。然后实现Comparator接口，重写比较方法，改为比较类的人数属性的大小，如果相同则通过Collator的getInstance(Locale.CHINESE)比较汉字的拼音。**  **代码展示：**  **DealFile.java**  */\*\**  *\** ***@ClassName*** *DealFile*  *\** ***@Description*** *TODO*  *\** ***@Author*** *Wenkun Lei*  *\** ***@Date*** *2022/3/19 14:26*  *\** ***@Version*** *3.0*  *\*\*/*  import java.io.\*;  import java.text.Collator;  import java.util.ArrayList;  import java.util.Comparator;  import java.util.Locale;  // 实现Comparator接口，自定义比较方法，比较CityData类中的人数  class CityComparator implements Comparator {  public int compare(Object o1, Object o2) {  CityData e1 = (CityData) o1;  CityData e2 = (CityData) o2;  if (e1.getPeopleNum() < e2.getPeopleNum()) {  return 1;  } else if (e1.getPeopleNum() > e2.getPeopleNum()) {  return -1;  } else {  // 如果人数相同则比较汉字拼音  return Collator.getInstance(Locale.CHINESE).compare(e1.getCityName(), e2.getCityName());  }  }  }  // 实现Comparator接口，自定义比较方法，比较ProvinceData类中的人数  class ProvinceComparator implements Comparator {  public int compare(Object o1, Object o2) {  ProvinceData e1 = (ProvinceData) o1;  ProvinceData e2 = (ProvinceData) o2;  if (e1.getPeopleNum() < e2.getPeopleNum())  return 1;  else if (e1.getPeopleNum() > e2.getPeopleNum()) {  return -1;  } else {  // 如果人数相同则比较汉字拼音  return Collator.getInstance(Locale.CHINESE).compare(e1.getProvinceName(), e2.getProvinceName());  }  }  }  public class DealFile {  public String inputFile;  public String outputFile;  public String outputFile2;  public DealFile() {  }  public DealFile(String inputFile, String outputFile) {  this.inputFile = inputFile;  this.outputFile = outputFile;  }  public DealFile(String inputFile, String outputFile, String outputFile2) {  this.inputFile = inputFile;  this.outputFile = outputFile;  this.outputFile2 = outputFile2;  }  // 处理文件格式，输入参数依次为文件的编码格式、待处理的输入文件的路径、处理完成的输出文件的路径  public void dealTxtFile(String designatedProvince) throws IOException {  // 所有省数据类放置与ArrayList容器  ArrayList provinceDataList = new ArrayList<>();  try {  File file = new File(this.inputFile);  // 判断文件是否存在  if (file.isFile() && file.exists()) {  String encoding = getFileFormat(this.inputFile);  // 考虑到编码格式，使用对应编码格式将文件输入流加载为输入读取流  InputStreamReader read = new InputStreamReader(new FileInputStream(file), encoding);  // 创建文件读取缓冲区  BufferedReader br = new BufferedReader(read);  // 创建文件输出流  FileWriter fw = new FileWriter(this.outputFile);  // 创建文件写入缓冲区  BufferedWriter bw = new BufferedWriter(fw);  // 创建读取缓冲区的每行的字符串变量  String lineTxt = null;  // 创建每行数据的省份字符串变量  String currentProvince = "0";  // 创建每个城市感染人数变量  int cityPeopleNum;  // 创建每个省份感染总人数变量  int totalPeopleNum = 0;  int countFlag = 0;  // 每个省所有城市数据类放置与ArrayList容器  ArrayList singleDataList = new ArrayList<>();  // 缓冲区有数据就一直一行一行地读取  while ((lineTxt = br.readLine()) != null) {  // 使用空格作为分割符，将每行的字符串分割为字符串数组  String[] lineArray = lineTxt.split("\\s+");  // 根据指定的省份进行操作，只写入和统计指定省份的数据，如果是空串，就执行所有省份  if (!lineArray[0].equals(designatedProvince) && !designatedProvince.equals("")) {  continue;  }  // 略去“待明确地区”  if (lineArray[1].equals("待明确地区")) {  continue;  }  // 将当前城市的省份进行比较，如果省份不一样了，那么将对当前省份进行替换  if (!currentProvince.equals(lineArray[0])) {  if (countFlag == 0) {  countFlag = 1;  } else {  bw.write("总计: " + totalPeopleNum);  // System.out.println(currentProvince);  // 省份名，总感染人数，下辖城市疫情情况实例化省份对象  ProvinceData provinceData = new ProvinceData(currentProvince, totalPeopleNum, singleDataList);  // 将该省份数据add进容器  provinceDataList.add(provinceData);  singleDataList = new ArrayList<>();  totalPeopleNum = 0;  bw.newLine();  }  if (!currentProvince.equals("0")) {  // 写入一个换行符  bw.newLine();  }  currentProvince = lineArray[0];  bw.write(lineArray[0]);  // 写入一个换行符  bw.newLine();  }  // 向缓冲区写入除了省份的其余数据  bw.write(lineArray[1] + "\t" + lineArray[2]);  cityPeopleNum = Integer.valueOf(lineArray[2]);  totalPeopleNum += cityPeopleNum;  // 城市名，感染人数，实例化城市对象  CityData cityData = new CityData(lineArray[1], cityPeopleNum);  // 将该城市数据add进容器  singleDataList.add(cityData);  // 写入一个换行符  bw.newLine();  bw.flush();  }  bw.write("总计: " + totalPeopleNum);  // System.out.println(currentProvince);  ProvinceData provinceData = new ProvinceData(currentProvince, totalPeopleNum, singleDataList);  provinceDataList.add(provinceData);  bw.flush();  // 关闭io流  bw.close();  read.close();  System.out.println("执行完毕");  } else {  System.out.println("找不到指定的文件");  }  } catch (Exception e) {  // 抛出异常  System.out.println("读取文件内容出错");  e.printStackTrace();  }  sortCity(provinceDataList);  }  public void sortCity(ArrayList provinceDataList) throws IOException {  // 创建文件输出流  FileWriter fw = new FileWriter(this.outputFile2);  // 创建文件写入缓冲区  BufferedWriter bw = new BufferedWriter(fw);  // 必须是Comparator中的compare方法和Collections.sort方法配合使用才管用  ProvinceComparator pct = new ProvinceComparator();  CityComparator cct = new CityComparator();  // 排序省感染总人数  provinceDataList.sort(pct);  // 遍历实例化省份数据  for (Object o : provinceDataList) {  ProvinceData provinceData = (ProvinceData) o;  ArrayList cityList = provinceData.getCityDataList();  // 排序一个省内城市感染人数  cityList.sort(cct);  // 写入排序后的省份名和感染总人数  bw.write(provinceData.getProvinceName() + "\t" + provinceData.getPeopleNum());  // 写入一个换行符  bw.newLine();  // 遍历实例化城市数据  for (Object value : cityList) {  CityData cityData = (CityData) value;  // 写入排序后的城市名和感染人数  bw.write(cityData.getCityName() + "\t" + cityData.getPeopleNum());  // 写入一个换行符  bw.newLine();  }  // 写入一个换行符  bw.newLine();  bw.flush();  }  // 关闭io流  bw.close();  }  // 对文件的编码格式进行判断  public String getFileFormat(String filePath) throws Exception {  File file = new File(filePath);  InputStream inputStream = new FileInputStream(file);  byte[] head = new byte[3];  String code = "GBK";  if (head[0] == -1 && head[1] == -2) {  code = "UTF-16";  }  if (head[0] == -2 && head[1] == -1) {  code = "Unicode";  }  if (head[0] == -17 && head[1] == -69 && head[2] == -65) {  code = "UTF-8";  }  System.out.println("The format of this file is " + code);  inputStream.close();  return code;  }  public static void main(String[] args) throws IOException {  String inputFile = args[0];  String outputFile = args[1];  String outputFile2 = args[2];  // 简易可变长参数  String designatedProvince = "";  if (args.length == 4) {  designatedProvince = args[3];  }  // String inputFile = "src/yq\_in\_03.txt";  // String outputFile = "src/my\_yq\_out\_03.txt";  // String outputFile2 = "src/my\_yq\_out\_04.txt";  // String designatedProvince = "";  DealFile rf = new DealFile(inputFile, outputFile, outputFile2);  rf.dealTxtFile(designatedProvince);  }  }  **CityData.java**  */\*\**  *\** ***@ClassName*** *CityData*  *\** ***@Description*** *TODO*  *\** ***@Author*** *Wenkun Lei*  *\** ***@Date*** *2022/3/19 15:24*  *\** ***@Version*** *1.0*  *\*\*/*  public class CityData {  private String cityName;  private int peopleNum;  public String getCityName() {  return cityName;  }  public void setCityName(String empNo) {  this.cityName = cityName;  }  public int getPeopleNum() {  return peopleNum;  }  public void setPeopleNum(int peopleNum) {  this.peopleNum = peopleNum;  }  public CityData() {  }  public CityData(String cityName, int peopleNum) {  this.cityName = cityName;  this.peopleNum = peopleNum;  }  }  **ProvinceData.java**  import java.util.ArrayList;  */\*\**  *\** ***@ClassName*** *ProvinceData*  *\** ***@Description*** *TODO*  *\** ***@Author*** *Wenkun Lei*  *\** ***@Date*** *2022/3/19 15:40*  *\** ***@Version*** *1.0*  *\*\*/*  public class ProvinceData {  private String provinceName;  private int peopleNum;  private ArrayList cityDataList;  public String getProvinceName() {  return provinceName;  }  public void setProvinceName(String empNo) {  this.provinceName = provinceName;  }  public ArrayList getCityDataList() {  return cityDataList;  }  public void setCityDataList(ArrayList cityDataList) {  this.cityDataList = cityDataList;  }  public int getPeopleNum() {  return peopleNum;  }  public void setPeopleNum(int peopleNum) {  this.peopleNum = peopleNum;  }  public ProvinceData() {  }  public ProvinceData(String provinceName, int peopleNum, ArrayList cityDataList) {  this.provinceName = provinceName;  this.peopleNum = peopleNum;  this.cityDataList = cityDataList;  }  }  **结果展示**  **实验三：**        **指定省份参数可以作为可变长参数进行缺省**    **排序后的结果**    **个人Git链接：** [**https://github.com/Wenkun2001/SE-homeworks.git**](https://github.com/Wenkun2001/SE-homeworks.git)  **四、实验心得与分析:**  **之前设计问题，回头看，写的稀烂，于是重构了部分代码。数据存储结构换成ArrayList了，Map也行，尝试过，不过感觉这种情况既然繁琐于存储，不如直接简单粗暴List，还方便排序。实现Comparator接口，重写排序方法的操作算是传统艺能了，但是汉字拼音排序之前没搞过，不知道这里能不能有好效果，但是逻辑上基本上就是那样了，在将原本的相等return 0换成了拼音字母比较。**   1. **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。