1. **使用IO包中的类读取D盘上exam.txt文本文件的内容，每次读取一行内容，将每行作为一个输入放入ArrayList的泛型集合中并将集合中的内容使用加强for进行输出显示。**

**public** **class** Test {

**public** **static** **void** main(String[] args) **throws** IOException {

String path="D:\\exam.txt";

*outputMethod*(path);

}

**public** **static** **void** outputMethod(String path) **throws** IOException {

List<String> list = **new** ArrayList<String>(); // 创建集合对象

// 创建缓冲区对象

BufferedReader br = **new** BufferedReader(**new** FileReader(path));

String line = br.readLine(); // 读取数据每次读一行

**while** (line != **null**) {

list.add(line);

line = br.readLine();

}

br.close(); //关闭

**for**(String s:list){

System.*out*.println(s);

}

}

}

1. **由控制台按照固定格式输入学生信息，包括学号，姓名，年龄信息，当输入的内容为exit退出；将输入的学生信息分别封装到一个Student对象中，再将每个Student对象加入到一个集合中，要求集合中的元素按照年龄大小正序排序；最后遍历集合，将集合中学生信息写入到记事本，每个学生数据占单独一行。**

推荐步骤：

1. 创建Student类，并指定按照年龄正序排列
2. 通过控制台输入多个不同Student信息。格式规定为：编号#姓名#年龄
3. 取出字符串中相应信息放入Student对象，并将Student加入到集合中
4. 遍历集合的过程中将学生的信息输入到记事本

难点：

1. 如何指定学生按照年龄正序排列
2. 如果从字符串“编号#姓名#年龄”中提取学生信息
3. 放入哪种集合后可以保证学生按照年龄大小正序排列
4. 如何将集合中学生信息写入记事本，每个学生数据占单独一行

**public** **class** Student **implements** Comparable<Student>{

**private** Integer num;

**private** String name;

**private** Integer age;

**//省略getter和setter方法**

**//省略构造方法**

**public** **int** compareTo(Student stu) {

**return** **this**.age-stu.age;

}

**public** String toString() {

**return** "Student [age=" + age + ", name=" + name +

", num=" + num + "]";

}

}

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Set<Student> stuSet = *saveStudentInfo*();

*outputInfo*(stuSet);

}

**private** **static** Set<Student> saveStudentInfo() {

Scanner input = **new** Scanner(System.*in*);

// 保存学生信息的TreeSet集合对象

Set<Student> stuSet = **new** TreeSet<Student>();

**while** (**true**) {

// 输入提示

System.*out*.println("请输入学生信息:(学号#姓名#年龄)");

String inputData = input.nextLine();

// 判断是否退出 inputData.equals("exit")

**if** ("exit".equals(inputData)) {

**break**;

}

// 将用户输入的学生信息分割为String[]

String[] info = inputData.split("#");

// 将输入信息封装到Student对象中

Student stu =

**new** Student(Integer.*parseInt*(info[0]), info[1],

Integer.*parseInt*(info[2]));

// 将学生对象加入集合

stuSet.add(stu);

}

**return** stuSet;

}

**private** **static** **void** outputInfo(Set<Student> stuSet) {

File file = **new** File("e:/student.txt");

// 创建文件输出流对象

FileWriter fw = **null**;

**try** {

fw = **new** FileWriter(file);

Iterator<Student> it = stuSet.iterator();

**while** (it.hasNext()) {

String info = it.next().toString();

// 将info字符串，写入记事本

fw.write(info);

// 完成换行功能

fw.write("\r\n");

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

**try** {

fw.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

}

评分标准：

正确编写学生类， 正确完成学生信息输入并放入集合，正确将集合内容写入文件。具体得分点如下：

1. 创建Student类，提供三个属性和setter/getter方法，提供构造方法
2. Student类实现Comparable，重写compareTo( )，按照年龄正序排列
3. 通过控制台输入多个Student信息。格式规定为：编号#姓名#年龄
4. 当输入的内容为exit退出。
5. 将用户输入的学生信息使用split( )分割为String[]。
6. 将输入信息封装到Student对象中。
7. 创建TreeSet对象，并将每个Student对象放入TreeSet。
8. 创建和关闭输出流对象。
9. 正确的遍历TreeSet集合。
10. 将每个学生信息写入文件。

此题目的难点主要有：

实现Comparable接口指定按照年龄正序排列；放入TreeSet会按照规则自动排序

使用split( )分割字符串；创建输出流写学生信息到文件并换行。

如果没有以上思路实现，只要能完成功能即可，按照相应功能步骤给分。

1. **利用递归实现将某个目录下所有内容copy到另一个目录中。**

**public** **class** CopyDir2 {

**public** **static** **void** main(String[] args) {

**try** {

*copyDirectiory*("d:/306","d:/306-2");

} **catch** (IOException e) {

e.printStackTrace();

}

}

/\*\*

\* 复制单个文件

\* **@param** sourceFile 源文件

\* **@param** targetFile 目标文件

\*/

**private** **static** **void** copyFile(File sourceFile, File targetFile) **throws** IOException {

BufferedInputStream inBuff = **null**;

BufferedOutputStream outBuff = **null**;

**try** {

// 新建文件输入流

inBuff = **new** BufferedInputStream(**new** FileInputStream(sourceFile));

// 新建文件输出流

outBuff = **new** BufferedOutputStream(**new** FileOutputStream(targetFile));

// 缓冲数组

**byte**[] b = **new** **byte**[1024 \* 5];

**int** len;

**while** ((len = inBuff.read(b)) != -1) {

outBuff.write(b, 0, len);

}

// 刷新此缓冲的输出流

outBuff.flush();

} **finally** {

// 关闭流

**if** (inBuff != **null**)

inBuff.close();

**if** (outBuff != **null**)

outBuff.close();

}

}

/\*\*

\* 复制目录

\* **@param** sourceDir 源目录

\* **@param** targetDir 目标目录

\* **@throws** IOException

\*/

**private** **static** **void** copyDirectiory(String sourceDir, String targetDir) **throws** IOException {

// 检查源目录

File fSourceDir = **new** File(sourceDir);

**if**(!fSourceDir.exists() || !fSourceDir.isDirectory()){

**return**;

}

//检查目标目录，如不存在则创建

File fTargetDir = **new** File(targetDir);

**if**(!fTargetDir.exists()){

fTargetDir.mkdirs();

}

// 遍历源目录下的文件或目录

File[] file = fSourceDir.listFiles();

**for** (**int** i = 0; i < file.length; i++) {

**if** (file[i].isFile()) {

// 源文件

File sourceFile = file[i];

// 目标文件

File targetFile = **new** File(fTargetDir, file[i].getName());

*copyFile*(sourceFile, targetFile);

}

//递归复制子目录

**if** (file[i].isDirectory()) {

// 准备复制的源文件夹

String subSourceDir = sourceDir + File.***separator*** + file[i].getName();

// 准备复制的目标文件夹

String subTargetDir = **targetDir** + File.***separator*** + file[i].getName();

// 复制子目录

*copyDirectiory*(subSourceDir, subTargetDir);

}

}

}

}