实验内容

第一个版本：排序时树映射的第一个泛型是封装类，测试程序时，会发现这个版本在学生成绩都不相同时是没有问题的，如果有成绩相同的情况，请观察程序的显示结果，有什么问题？

第二个版本：对于第一个版本出现的问题，编写第二个版本，使得有成绩相同的情况，也都可以正确显示。

（1）假设源程序StuList.java中的类实现了接口MyList，创建源程序MyList.java，其中的内容是接口MyList的声明。

（2）在StuList.java中根据要求补充代码。其中，学生信息不再存放在数组中，而是存放在双链表泛型类的对象中；排序时不再使用冒泡法或者选择法排序，而是使用树映射泛型类的对象进行排序输出。

（3）StuList.java，MyList.java，Undergraduate.java，Postgraduate.java和Student.java在同一个包中。Main.java在无名包中。

上述源程序可以完成如下任务：

创建学生链表，使用者根据提示信息，选择从键盘上输入本科生或者研究生的学生信息，将其加入学生链表，添加学生信息，删除学生信息，显示学生信息，按照学生成绩排序等。

实现方法

1. 学生列表用LinkedList建立

LinkedList<Student> 用来存储学生对象。利用LinkedList 我们可以创建一个学生对象的列表。列表被限制为只能包含 Student 类或其子类的对象，通过将不同类型的学生对象（如本科生和研究生）添加到相同的列表中，不需要为每个学生类型创建单独的列表。

1. 学生排序用TreeMap实现

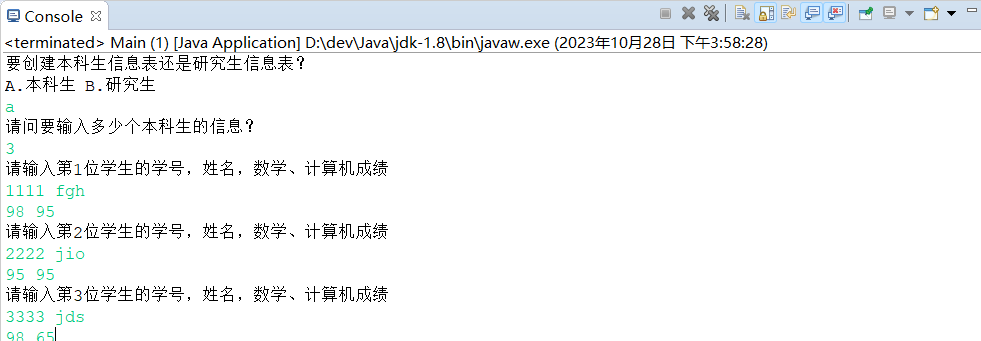
在sort 方法中，TreeMap 被用于对学生进行排序。TreeMap<Double, Student>中 Double 表示键的类型，而 Student 表示值的类型。TreeMap 允许根据标准（此处是选择数学或计算机科学成绩）对学生进行排序。

由于用Double封装类在创建TreeMap对象时会丢失键相同的Student，比如数学成绩相同的同学只能保留一个，所以对代码进行更新。在更新后的代码中，引入了自定义的Key类来替代原先使用Double作为键类型的TreeMap，自定义Key类实现了Comparable接口，允许对学生信息进行排序，其中比较键值相同时返回正值，确保相同比较值的学生信息不会丢失，而是按照添加的顺序保留在TreeMap中。

实验结果

1. 第一个版本：排序时树映射的第一个泛型是封装类

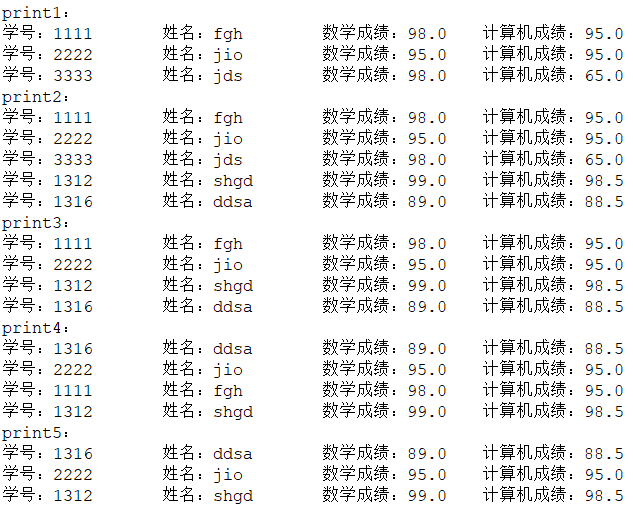
输入：



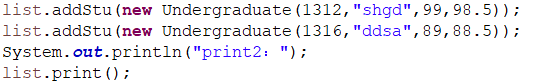
输出分别为：

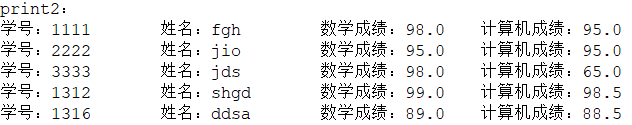
1. 直接对输入的学生信息进行输出。



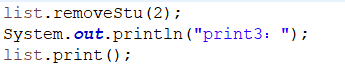


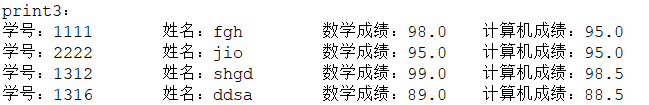
1. 添加两名新学生信息后进行输出。



、

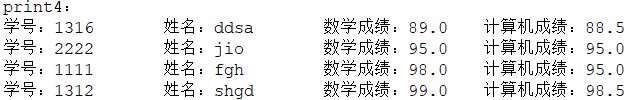
（3）删除下标为2的同学的信息，即为第三个同学的信息后进行全部学生信息的输出。





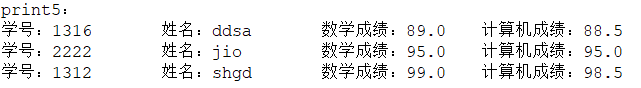
（4）按照数学成绩排序的输出。





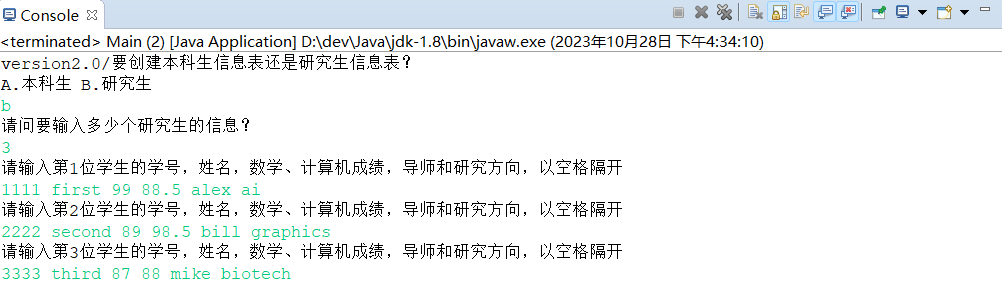
（5）按照计算机成绩排序的输出。这里可以看到和2222号有相同计算机成绩的1111号同学信息没有被输出。





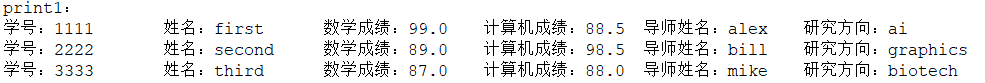
1. 树映射的第一个泛型不是封装类

输入：

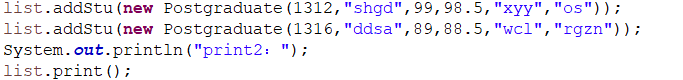


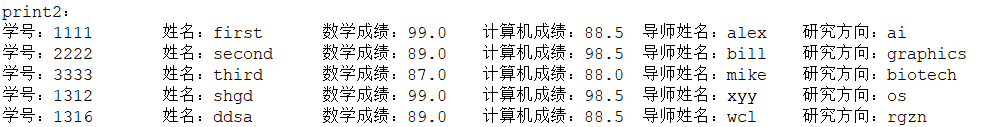
1. 直接对输入的学生信息进行输出。



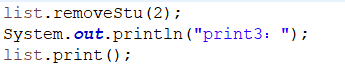


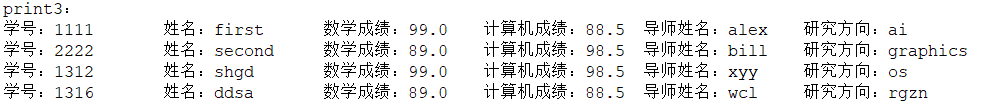
（2）添加两名新学生信息后进行输出。





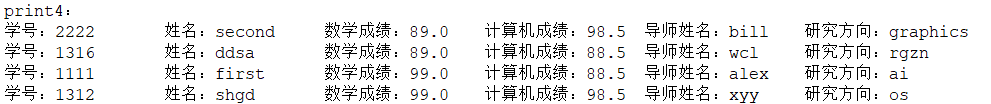
（3）删除下标为2的同学的信息，即为第三个同学的信息后进行全部学生信息的输出。





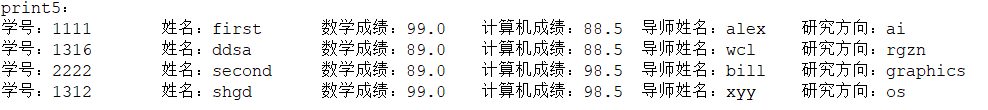
（4）按照数学成绩排序的输出。这里可以看到相同数学成绩学生信息的都被输出了。





（5）按照计算机成绩排序的输出。这里可以看到相同计算机成绩学生信息的都被输出了。





结论分析

1. 对Comparable接口的实现总结：

Key 类实现了Comparable接口，要求重写compareTo方法来确定如何比较两个Key对象。在此处Key对象是通过成绩进行比较的。

compareTo 方法返回一个整数值，表示两个Key对象之间的比较关系。如果指定的数与参数相等返回 0。如果指定的数小于参数返回 -1。如果指定的数大于参数返回 1。

通过这些更改，代码不再依赖于默认的比较方式，而是使用自定义的Key对象来实现对学生信息的排序。这种方式提供了更多灵活性，允许以不同的方式排序学生信息，同时确保相同比较值的学生信息不会丢失。

1. 泛型的总结。

使用泛型数据结构，例如 LinkedList<Student>，允许不同类型的学生（例如本科生和研究生）被添加到同一个列表中，不需要为每种类型的学生创建单独的数据结构。

通过引入自定义的Key类，泛型使代码能够使用自定义的比较方式对学生信息进行排序，有助于处理相同比较值的情况来确保相同成绩的学生信息不会丢失。

泛型还能实现以不同的标准对学生信息进行排序，通过修改TreeMap的键类型来实现不同的排序方式。