#### 姓名： 谢宝玛

#### 学号：1120233506

#### 实验日期： 2025-04-18

## （1）实验内容

使用Lambda表达式比较器Comparator给List对象排序，分别按Name、Age（倒序）、Grade排序。List对象内容如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Age | Grade |
| 1 | ZhangSan | 28 | 98 |
| 2 | LiSi | 21 | 100 |
| 3 | KangKang | 27 | 89 |
| 4 | LiMing | 19 | 92 |
| 5 | WangGang | 22 | 66 |
| 6 | ZhaoXin | 24 | 85 |
| 7 | LiuWei | 20 | 78 |
| 8 | BaiZhanTang | 16 | 99 |

## （2）实验思路

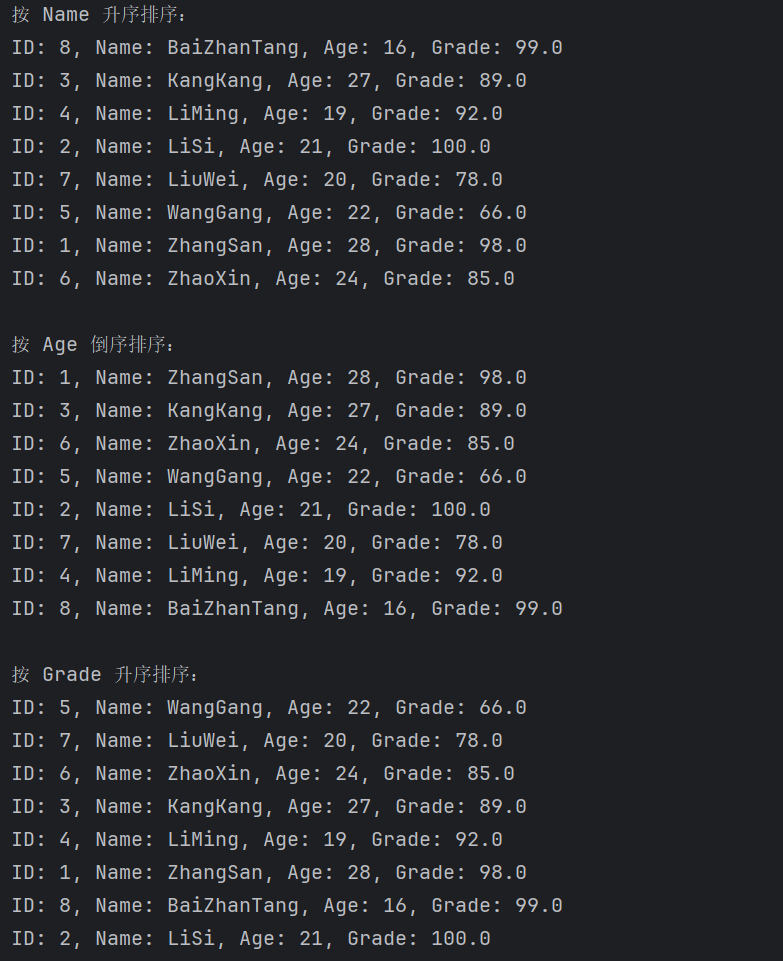
使用 Lambda 表达式和 Comparator 对 List 中的对象按 Name（字典序升序）、Age（倒序）、Grade（升序）进行排序

## 实验源码

import java.util.\*;  
  
class Student {  
 private int id;  
 private String name;  
 private int age;  
 private double grade;  
  
 public Student(int id, String name, int age, double grade) {  
 this.id = id;  
 this.name = name;  
 this.age = age;  
 this.grade = grade;  
 }  
  
 // Getter methods  
 public int getId() { return id; }  
 public String getName() { return name; }  
 public int getAge() { return age; }  
 public double getGrade() { return grade; }  
  
 @Override  
 public String toString() {  
 return "ID: " + id + ", Name: " + name + ", Age: " + age + ", Grade: " + grade;  
 }  
}  
  
public class LambdaSortDemo {  
 public static void main(String[] args) {  
 List<Student> students = new ArrayList<>();  
 students.add(new Student(1, "ZhangSan", 28, 98));  
 students.add(new Student(2, "LiSi", 21, 100));  
 students.add(new Student(3, "KangKang", 27, 89));  
 students.add(new Student(4, "LiMing", 19, 92));  
 students.add(new Student(5, "WangGang", 22, 66));  
 students.add(new Student(6, "ZhaoXin", 24, 85));  
 students.add(new Student(7, "LiuWei", 20, 78));  
 students.add(new Student(8, "BaiZhanTang", 16, 99));  
  
 // 按 Name 升序排序  
 students.sort(Comparator.comparing(Student::getName));  
 System.out.println("按 Name 升序排序：");  
 students.forEach(System.out::println);  
  
 // 按 Age 倒序排序  
 students.sort(Comparator.comparing(Student::getAge).reversed());  
 System.out.println("\n按 Age 倒序排序：");  
 students.forEach(System.out::println);  
  
 // 按 Grade 升序排序  
 students.sort(Comparator.comparing(Student::getGrade));  
 System.out.println("\n按 Grade 升序排序：");  
 students.forEach(System.out::println);  
 }  
}

LibrarySystem.java

## (4)结果

****

#### **(3)**实验心得

如果想要同时进行多重排序（比如先按 Name，再按 Age），也可以链式使用 thenComparing