暨南大学本科实验报告专用纸(附页)

较复杂的学生信息管理系统

* 实验项目类型：设计性
*此表由学生按顺序填写

课程名称___面向对象程序设计/JAVA 语言___成绩评定________
实验项目名称__较复杂的学生信息管理系统__指导老师__干晓聪__
实验项目编号_1_实验项目类型__设计性__实验地点_数学系机房_
学生姓名___郭彦培___学号___2022101149___
学院_信息科学技术学院_系_数学系_专业_信息管理与信息系统_
实验时间_2023 年 11 月 1 日上午_~_2023 年 11 月 1 日中午__

一、 实验目的

熟练运用对象设计工具。

熟练掌握以下工具或性质：

1. 访问控制
2. 内部类
3. 接口
4. 继承
5. 多态

了解异常处理

二、 实验环境

计算机：PC X64

操作系统：Windows

编程语言：Java

IDE：Visual Studio Code

在线测试平台：leetcode

三、 程序原理

提供了四个类:

`course`	课程
`score`	成绩
`student`	学生
`SchoolLib`	管理系统

其中 SchoolLib 为主类，提供了对学生以下操作:

1. 添加学生并自动创建学号
2. 添加课程并自动生成课程编号
3. 学生选课
4. 记录考试成绩，补考成绩
5. 与平时分加权，计算总评成绩
6. 计算绩点
7. 以绩点、学号、姓名字典序等对学生排序

保证所有操作线程安全，必要的地方法进行了异常处理与保护性拷贝。

测试用例的控制台规则:

中括号内为需要填入字符串，尖括号为可选参数，默认为列表第一个。

`addStu [student name]`	添加学生
`addCur [Course name] [teacher name] [credit]`	添加课程
`celCur [student UID] [Course name]`	选课
`showStu <sort with:"UID" | "name" | "point">`	显示学生列表
`showCur <sort with:"UID" | "Course name" | " teacher name">`	显示课程列表
`addScore [student UID] [Course name] [Normally score] [Exam score]`	添加成绩
`addResc [student UID] [Course name] [Exam score]`	添加补考成绩
`cacu`	计算总评成绩与绩点

四、 程序代码

文件`sis5\coures.java`实现了`coures`类

```java
package sis5;
import java.util.Comparator;

public class course {//课程
    private String UID;//课程编号
    private String name;//课程名称
    private String teacher;//教师名称
    private int credit;//学分

    @Override public boolean equals(Object b)
    {
        if(!(b instanceof course)) return false;
        course ts = (course)b;
        if(ts.getUID().equals(this.UID)) return true;
        else return false;
    }

    @Override public int hashCode()
    {
        return UID.hashCode();
    }

    @Override public String toString()
    {
        return UID + " " + name + " " + teacher + " " + credit;
    }


    public static class sortWithUID implements Comparator<course>
    {
        @Override public int compare(course o1, course o2)
        {
            return o1.UID.compareTo(o2.UID);
        }
    }

    public static class sortWithName implements Comparator<course>
    {
        @Override public int compare(course o1, course o2)
        {
            return o1.name.compareTo(o2.name);
        }
    }

    public static class sortWithCredit implements Comparator<course>
    {
        @Override public int compare(course o1, course o2)
        {
            if(o1.credit > o2.credit) return 1;
            else if(o1.credit < o2.credit) return -1;
            else return 0;
        }
    }


    course(String _uuid,final String _name,final String _teacher,final int
_credit)
    {
        UID = _uuid;
```

```java
        name = _name;
        teacher = _teacher;
        credit = _credit;
    }

    course(course _course)//深拷贝构造
    {
        this.UID = _course.getUID();
        this.name = _course.getName();
        this.teacher = _course.getTeacher();
        this.credit = _course.getCredit();
    }


    public String getUID()
    {
        return UID;
    }

    public String getName()
    {
        return name;
    }

    public String getTeacher()
    {
        return teacher;
    }

    public int getCredit()
    {
        return credit;
    }
}
```

文件`sis5\Student.java`实现了`Student`类

```java
package sis5;

import java.util.Comparator;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Set;
```

```java
public class Student{
    private String UID;//学号
    private String name;//姓名
    private Set<course> curriculumList;//所选课程列表
    private HashMap<course,score> scoreList;//成绩单
    private double avgGPA = 0;//平均绩点
    private int credit = 0;//已修学分

    private void cacuGPA() //计算平均绩点
    {
        double sum = 0;
        for(course cs : curriculumList)
        {
            sum += scoreList.get(cs).getGPA();
        }
        avgGPA = sum / curriculumList.size();
    }

    @Override public boolean equals(Object b)
    {
        if(!(b instanceof Student)) return false;
        Student ts = (Student)b;
        if(ts.getUID().equals(this.UID)) return true;
        else return false;
    }

    @Override public int hashCode()
    {
        return UID.hashCode();
    }

    @Override public String toString()
    {
        return UID + " " + name + " " + avgGPA + " " + credit;
    }


    public static class sortWithGPA implements Comparator<Student>
    {
        @Override public int compare(Student o1, Student o2)
        {
            if(o1.avgGPA < o2.avgGPA) return 1;
            else if(o1.avgGPA > o2.avgGPA) return -1;
            else return 0;
        }
    }

    public static class sortWithUID implements Comparator<Student>
    {
        @Override public int compare(Student o1, Student o2)
        {
            return o1.UID.compareTo(o2.UID);
        }
    }

    public static class sortWithName implements Comparator<Student>
    {
        @Override public int compare(Student o1, Student o2)
        {
            return o1.name.compareTo(o2.name);
        }
    }


    Student(String _uid,String _name)
    {
        UID = new String(_uid);
        name = new String(_name);
        curriculumList = new HashSet<course>();
        scoreList = new HashMap<course,score>();
    }

    Student(Student _student) //深拷贝构造
    {
        this.UID = _student.getUID();
        this.name = _student.getName();
        this.scoreList = new HashMap<course,score>(_student.scoreList);
        this.avgGPA = _student.avgGPA;
        this.credit = _student.credit;
        curriculumList = new HashSet<course>();
        for(course cs : _student.curriculumList)
        {
            this.curriculumList.add(new course(cs));
        }
    }
    public void addCourse(course _cs)  //选课
    {
        curriculumList.add(_cs);
        credit += _cs.getCredit();
    }

    public void ranking(course _cs,score rank) throws Exception //首次录入成
绩
    {
        if(scoreList.containsKey(_cs)) throw new Exception("重复录入成绩
\n");
        else scoreList.put(_cs,rank);
    }

    public void reRanking(course _cs,score rank) throws Exception //补考
    {
        if(!scoreList.containsKey(_cs)) throw new Exception("未找到对应成绩
\n");
        else scoreList.replace(_cs, rank);
    }


    public String getUID()
    {
        return new String(UID);
    }

    public String getName()
    {
        String rt = new String(name);
        return rt;
    }

    public double getAvgGPA()
    {
        cacuGPA();
        return avgGPA;
    }

    public score getScore(course _cs) throws Exception
    {
        if(!scoreList.containsKey(_cs)) throw new Exception("未找到对应成绩
\n");
        else return new score(scoreList.get(_cs));
    }
}
```

文件`sis5\score.java`实现了`score`类

```
package sis5;

public class score {

    private Integer normallyScore;//平时成绩
    private Integer examScore;//期末成绩
    private Integer finalScore;//总评成绩
    private double GPA;//绩点
    private boolean ifReExam;//是否补考

    Integer NORMALLY_WEIGHT;//平时成绩权重
    Integer EXAM_WEIGHT;//期末成绩权重
    Integer TOTAL_WEIGHT;//总评成绩权重

    @Override public boolean equals(Object b)
    {
        if(!(b instanceof score)) return false;
        score ts = (score)b;
        if(ts.getFinalScore() == this.finalScore) return true;
        else return false;
    }

    @Override public int hashCode()
    {
        return finalScore;
    }


    private void cacuGPA() //根据暨大标准计算GPA
    {
        if(finalScore > 90) GPA = 4+(finalScore-90)/10.0;
        else if(finalScore > 80) GPA = 3+(finalScore-80)/10.0;
        else if(finalScore > 70) GPA = 2+(finalScore-70)/10.0;
        else if(finalScore > 60) GPA = 1+(finalScore-60)/10.0;
        else GPA = 0;

        if(ifReExam) GPA = Math.min(GPA, 1.6);
    }

    private void cacuFinalScore() //计算分配权重后的总评成绩
    {
        finalScore = (int)(normallyScore.intValue() *
((double)NORMALLY_WEIGHT/(double)TOTAL_WEIGHT) + examScore.intValue() *
((double)EXAM_WEIGHT/(double)TOTAL_WEIGHT));
    }

    public score(Integer _normallyScore,Integer _examScore)
    {
        ifReExam = false;
        NORMALLY_WEIGHT = 3;
        EXAM_WEIGHT = 7;
        TOTAL_WEIGHT = NORMALLY_WEIGHT + EXAM_WEIGHT;
        normallyScore = _normallyScore;
        examScore = _examScore;
        cacuFinalScore();
        cacuGPA();
    }

    public score(Integer _normallyScore,Integer _examScore,Integer
_normallyWeight,Integer _examWeight)
    {
        ifReExam = false;
        NORMALLY_WEIGHT = _normallyWeight;
        EXAM_WEIGHT = _examWeight;
        TOTAL_WEIGHT = NORMALLY_WEIGHT + EXAM_WEIGHT;
        normallyScore = _normallyScore;
        examScore = _examScore;
        cacuFinalScore();
        cacuGPA();
    }

    public score(score sc) throws Exception //深拷贝构造
    {
        this.normallyScore = sc.getNormallyScore();
        this.examScore = sc.getExamScore();
        this.finalScore = sc.getFinalScore();
        this.GPA = sc.getGPA();
        this.ifReExam = sc.ifReExam;
        this.NORMALLY_WEIGHT = sc.NORMALLY_WEIGHT;
        this.EXAM_WEIGHT = sc.EXAM_WEIGHT;
        this.TOTAL_WEIGHT = sc.TOTAL_WEIGHT;
    }


    public void resetWeight(Integer _normallyWeight,Integer _examWeight) //
重设分数分配权重
    {
        NORMALLY_WEIGHT = _normallyWeight;
        EXAM_WEIGHT = _examWeight;
        TOTAL_WEIGHT = NORMALLY_WEIGHT + EXAM_WEIGHT;
        cacuFinalScore();
        cacuGPA();
    }

    public void resetNormallyScore(Integer _normallyScore) //重设平时成绩
    {
        normallyScore = _normallyScore;
        cacuFinalScore();
        cacuGPA();
    }

    public void resetExamScore(Integer _examScore) throws Exception //补考
    {
        if(ifReExam) throw new Exception("重复补考\n");
        if(finalScore >= 60) throw new Exception("非法补考\n");
        ifReExam = true;
        examScore = _examScore;
        cacuFinalScore();
        cacuGPA();
    }


    public Integer getNormallyScore()
    {
        return Integer.valueOf(normallyScore);
    }

    public Integer getExamScore()
    {
        return Integer.valueOf(examScore);
    }

    public Integer getFinalScore()
    {
        return Integer.valueOf(finalScore);
    }

    public double getGPA()
    {
        return GPA;
    }

}
```

文件`sis5\SchoolLib.java`实现了`SchoolLib`类

```
package sis5;
import java.util.ArrayList;
import java.util.HashMap;

import sis2.UIDmanager;

public class SchoolLib {
    private HashMap<String,Student> studentList;
    private HashMap<String,course> courseList;

    final UIDmanager uidm = new UIDmanager();

    private enum sortStuWith
    {
        UID,
        name,
        GPA
    }

    private enum sortCurWith
    {
        UID,
        name,
        credit
    }

    public SchoolLib()
    {
        studentList = new HashMap<String,Student>();
        courseList = new HashMap<String,course>();
    }
    public SchoolLib(final SchoolLib _scl) //深拷贝构造
    {
        studentList = new HashMap<String,Student>(_scl.studentList);
        courseList = new HashMap<String,course>(_scl.courseList);
    }
    private void addStudent(final Student student)
    {
        studentList.put(student.getUID(),new Student(student));
    }

    private void addCourse(final course cs)
    {
        courseList.put(cs.getName(),new course(cs));
    }

    private void celCur(final String stuUID,final String csName) throws
Exception
    {
        if(!studentList.containsKey(stuUID)) throw new Exception("未找到对应
学生\n");
        if(!courseList.containsKey(csName)) throw new Exception("未找到对应
课程\n");
        Student stu = studentList.get(stuUID);
        course cs = courseList.get(csName);
        stu.addCourse(cs);
    }

    private ArrayList<Student> sortLocalStuList(final sortStuWith sw)
    {
        ArrayList<Student> tmp = new ArrayList<Student>();
        for(Student stu : studentList.values())
        {
            tmp.add(new Student(stu));
        }
        if(sw == sortStuWith.UID)
        {
            tmp.sort(new Student.sortWithUID());
        }
        else if(sw == sortStuWith.name)
        {
            tmp.sort(new Student.sortWithName());
        }
        else if(sw == sortStuWith.GPA)
        {
            tmp.sort(new Student.sortWithGPA());
        }
        return tmp;
    }

    private ArrayList<course> sortLocalCurList(final sortCurWith sw)
    {
        ArrayList<course> tmp = new ArrayList<course>();
        for(course cs : courseList.values())
        {
            tmp.add(new course(cs));
        }
        if(sw == sortCurWith.UID)
        {
            tmp.sort(new course.sortWithUID());
        }
        else if(sw == sortCurWith.name)
        {
            tmp.sort(new course.sortWithName());
        }
        else if(sw == sortCurWith.credit)
        {
            tmp.sort(new course.sortWithCredit());
        }
        return tmp;
    }

    public String parseCommand(final String cmd) throws Exception//解析命令
    {
        String[] cmdList = cmd.split(" ");
        if(cmdList[0].equals("addStu"))
        {
            if(cmdList.length != 2) throw new Exception("命令格式错误\n");
            String name = cmdList[1];
            String uid = uidm.nextDate();
            Student stu = new Student(uid,name);
            uidm.bindUID(uid, stu);
            addStudent(stu);
            return "添加学生成功, UID 为" + uid + "\n";
        }
        else if(cmdList[0].equals("addCur"))
        {
            if(cmdList.length != 4) throw new Exception("命令格式错误\n");
            String name = cmdList[1];
            String teacher = cmdList[2];
            int credit = Integer.parseInt(cmdList[3]);
            course cs = new course(uidm.nextSeq(),name,teacher,credit);
            uidm.bindUID(cs.getUID(), cs);
            addCourse(cs);
            return "添加课程成功, UID 为" + cs.getUID() + "\n";
        }
        else if(cmdList[0].equals("celCur"))
        {
            if(cmdList.length != 3) throw new Exception("命令格式错误\n");
            String stuUID = cmdList[1];
            String csName = cmdList[2];
            celCur(stuUID,csName);
            return "选课成功\n";
        }
        else if(cmdList[0].equals("showStu"))
        {
            if(cmdList.length != 2) throw new Exception("命令格式错误\n");
            String sortWith = cmdList[1];
            ArrayList<Student> tmp =
sortLocalStuList(sortStuWith.valueOf(sortWith));
            StringBuilder sb = new StringBuilder();
            for(Student stu : tmp)
            {
                sb.append(stu.toString());
                sb.append("\n");
            }
            return sb.toString();
        }
        else if(cmdList[0].equals("showCur"))
        {
            if(cmdList.length != 2) throw new Exception("命令格式错误\n");
            String sortWith = cmdList[1];
            ArrayList<course> tmp =
sortLocalCurList(sortCurWith.valueOf(sortWith));
            StringBuilder sb = new StringBuilder();
            for(course cs : tmp)
            {
                sb.append(cs.toString());
                sb.append("\n");
            }
            return sb.toString();
        }
        else if(cmdList[0].equals("addScore"))
        {
            if(cmdList.length != 5) throw new Exception("命令格式错误\n");
            String stuUID = cmdList[1];
            String csName = cmdList[2];
            int normallyScore = Integer.parseInt(cmdList[3]);
            int examScore = Integer.parseInt(cmdList[4]);
            if(!studentList.containsKey(stuUID)) throw new Exception("未找
到对应学生\n");
            if(!courseList.containsKey(csName)) throw new Exception("未找到
对应课程\n");
            Student stu = studentList.get(stuUID);
            course cs = courseList.get(csName);
            stu.ranking(cs, new score(normallyScore,examScore));
            return "添加成绩成功\n";
        }
        else if(cmdList[0].equals("addResc"))
        {
            if(cmdList.length != 4) throw new Exception("命令格式错误\n");
            String stuUID = cmdList[1];
            String csName = cmdList[2];
            int examScore = Integer.parseInt(cmdList[3]);
            if(!studentList.containsKey(stuUID)) throw new Exception("未找
到对应学生\n");
            if(!courseList.containsKey(csName)) throw new Exception("未找到
对应课程\n");
            Student stu = studentList.get(stuUID);
            course cs = courseList.get(csName);
            score tmp = stu.getScore(cs);
            tmp.resetExamScore(examScore);
            stu.reRanking(cs, tmp);
            return "重设补考成绩成功\n";
        }
        else if(cmdList[0].equals("cacu"))
        {
            for(Student stu : studentList.values())
            {
                stu.getAvgGPA();
            }
            return "计算完成\n";
        }
        else throw new Exception("未知命令\n");
    }
}
```

文件`sis5\Test`实现了主函数入口与输入输出

```java
package sis5;

import java.util.Scanner;

public class Test {
    public static void main(String[] args) {
        try (Scanner sc = new Scanner(System.in)) {
            SchoolLib lib = new SchoolLib();
            for(;;)
            {
                String s = sc.nextLine();
                try {
                    System.out.print(lib.parseCommand(s));
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

五、 出现的问题、原因与解决方法

编码过程中选择命令解析方式时，初始思路为在`Test`类中直接实现，但是会导致抛出异常后程序直接结束，且不易灵活判定指令输入的内容是否正确，因此最终实现时，选择在类中实现一个解析命令的方法`parseCommand`，传入一行指令，返回程序输出，可以使得封装更加完备，类的使用更加方便。同时，在完备的异常处理下，此结构可以保证程序遇到指令错误时仍然可以跳过此条指令并继续执行。

在具体实现排序学生列表功能时，考虑到匿名函数的复用性较差，因此选择在内部类`sortWith...`中实现接口`Comparator<>`实现各种排序方法的灵活选择。调用时只需要`sort(new course.sortWith...())`即可。

同时，在决定排序方式时，避免使用了复杂的分支判断，而选择使用`enum Sort...With`的枚举类型，提高代码复用性、可读性与鲁棒性，且更方便地进行异常处理。

六、 测试数据与运行结果

输入	输出	解释

`addStu GYP`
`addStu XSX`
`addStu CXK`
`addStu ZY`
	添加学生成功，UID 为 2312150001
	添加学生成功，UID 为 2312150002
	添加学生成功，UID 为 2312150003
	添加学生成功，UID 为 2312150004
		添加学生

`addCur java GXC 4`
`addCur cpp FZZ 3`
`addCur py LCC 2`
	添加课程成功，UID 为 0000000001
	添加课程成功，UID 为 0000000002
	添加课程成功，UID 为 0000000003
		添加课程

`celCur 2312150001 java`
`celCur 2312150001 cpp`
`celCur 2312150002 py`
`celCur 2312150002 java`
`celCur 2312150003 cpp`
`celCur 2312150003 py`
`celCur 2312150003 game`
`celCur 2312150004 java`
	选课成功
	选课成功
	选课成功
	选课成功
	选课成功
	选课成功
	未找到对应课程
	选课成功
		学生选课
		由于没有
		game 课程
		因此报错
		不影响
		继续运行

`addScore 2312150001 java 90 90`
`addScore 2312150001 cpp 80 80`
`addScore 2312150001 py 70 70`
`addScore 2312150002 java 60 60`
`addScore 2312150002 py 50 50`
`addScore 2312150003 cpp 93 34`
`addScore 2312150003 py 77 88`
`addScore 2312150003 game 100`
`100`
`addScore 2312150004 java 100`
`100`
	添加成绩成功
	添加成绩成功
	添加成绩成功
	添加成绩成功
	添加成绩成功
	添加成绩成功
	添加成绩成功
	未找到对应课程
	添加成绩成功
		添加学生
		平时成绩与
		期末成绩
		由于没有
		game 课程
		因此报错
		不影响
		继续运行

`cacu`	计算完成	

`shouStu GPA`
`showStu GPA`
	未知命令
	`2312150004 ZY 5.0 4`
	`2312150001 GYP 3.5 7`
	`2312150003 CXK 1.7 5`
	`2312150002 XSX 0.0 6`
		计算并
		排序，显示
		学生的 GPA

		续表

输入	**输出**	**解释**

`addResc 2312150002 py 99`
`addResc 2312150003 cpp 99`
`addResc 2312150003 py 99`
	重设补考成绩成功
	重设补考成绩成功
	非法补考
		总评 60 以下的
		可以补考

`cacu`
`showStu GPA`
	计算完成
	`2312150004 ZY 5.0 4`
	`2312150001 GYP 3.5 7`
	`2312150003 CXK 2.5 5`
	`2312150002 XSX 0.8 6`
		补考后重新计算 GPA

`showStu name`	`2312150003 CXK 2.5 5`
	`2312150001 GYP 3.5 7`
	`2312150002 XSX 0.8 6`
	`2312150004 ZY 5.0 4`
		按姓名排序