



《Java 与面向对象设计》

项目报告

Java 语言实现的课程管理系统

学 院 名 称 : 数据科学与计算机学院

专业 (班级) : 17 软件工程 2 班

王明业 17343107

姓 名 学 号 : 温卓沛 17343115

幸赞 17343128

时 间 : 2018 年 1 月 9 日

目录

一、	系统功能介绍.....	3
二、	系统类图	3
三、	关键模块说明.....	4
1.	COURSE 类	4
2.	TEACHER 类.....	5
3.	STUDENT 类.....	6
4.	TEST 类	7
5.	UI.....	9
四、	知识点应用与创新点、技术难点说明.....	10
1.	除 UI 外的类.....	10
2.	UI 类.....	12
五、	未解决的问题与难点讨论.....	13
六、	分工说明.....	14

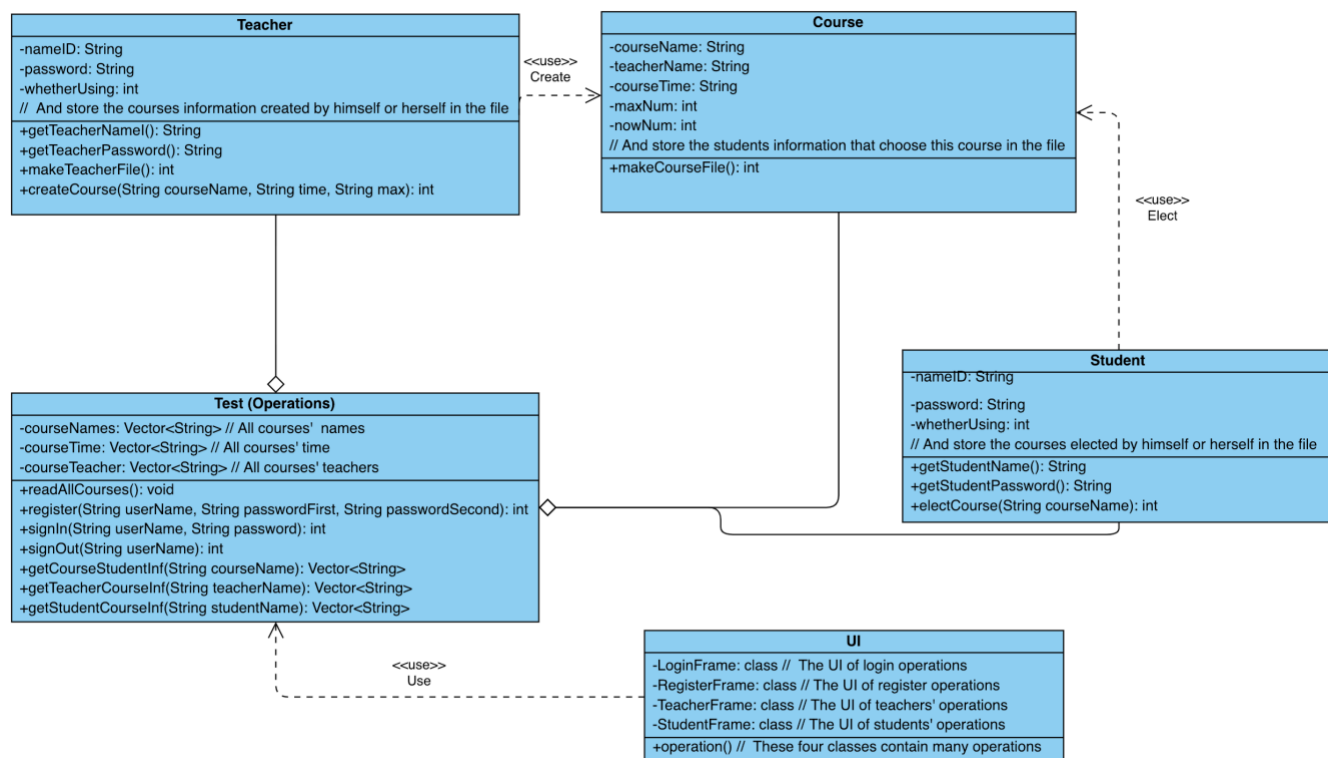
此目录带有超链接，点击目录中的条目可跳转至相应内容

一、系统功能介绍

一个基于 Java 语言的课程管理系统，实现了：教师用户与学生用户的注册和登录，教师发布新课程，教师查看自身已发布的课程，学生选择课程，学生查看自身已选课程，学生查看现阶段可选课程等功能。并将所有教师、学生、课程的信息保存在本地.txt 文件中。

二、系统类图

即 Teacher 类使用 Course 类（即创建课程操作），Student 类使用 Course 类（即选课操作）。Teacher 类、Student 类、Course 类均集成在 Test 类中被使用，Test 类中有各种与具体操作有关的函数。比如从文件中读取信息，验证登录登出等等。



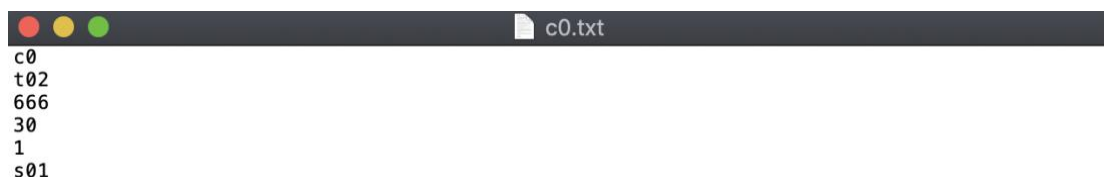
三、 关键模块说明

1. Course 类

Course
<pre> -courseName: String -teacherName: String -courseTime: String -maxNum: int -nowNum: int // And store the students information that choose this course in the file +makeCourseFile(): int </pre>

类成员变量有课程名称、任课教师名称、上课时间、最大人数、当前人数。每个课程的信息都会被储存在一个以课程名为文件名的.txt 文件中，放置于 Course 文件夹。每个文件中都会包含课程名称、教师名称、上课时间、最大人数、当前人数、已选该课程的学生这些信息。

Course 储存文件内容如下图所示：



```

c0
t02
666
30
1
s01

```

其中第一行为课程名称，第二行为授课教师名称，第三行为上课时间，第四行为最大人数，第五行为当前已选人数，第六行及后续是选择该课程学生的名称。

关键函数：

```

/*
 * -1 表示课程已存在 无法再次创建
 * 1 表示课程创建成功
 */
public int makeCourseFile()

```


该函数在 Course 文件夹中创建该 Course 类对应的.txt 文件，若 Course 文件夹不存在，则先创建 Course 文件夹再创建.txt 文件。

2. Teacher 类

Teacher
-nameID: String -password: String -whetherUsing: int // And store the courses information created by himself or herself in the file
+getTeacherNameID(): String +getTeacherPassword(): String +makeTeacherFile(): int +createCourse(String courseName, String time, String max): int

类成员变量有名称 ID，密码，该账户是否正被使用（即是否已被登录，值为 0 表示该账户目前未被登录；值为 1 表示该账户已被登录，正在使用中）。每个教师的信息都会被储存在一个以教师名为文件名的.txt 文件中，放置于 Teacher 文件夹。每个文件中都会包含账户是否正在被使用、教师名称、该账户密码、该教师创建的所有课程这些信息。

Teacher 储存文件内容如下图所示：



```
0
t02
123456
c0 666 30
```

其中第一行表示账户是否正被使用，第二行为教师名称，第三行为密码，第四行及后续为该教师创建课程的信息，格式为“课程名 上课时间 最大人数”。

关键函数：

```
/*
 * -1 表示该教师用户文件已存在
 * 1 表示教师用户文件创建成功
 *
 * */
public int makeTeacherFile()
```

该函数在 Teacher 文件夹中创建该 Teacher 类对应的.txt 文件，若 Teacher 文件夹不存在，则先创建 Teacher 文件夹再创建.txt 文件。

```

/*
-1 表示该课程已存在，无法再创建同名课程
1 表示创建课程成功
*/
public int creatCourse(String courseName, String time, int max)

```

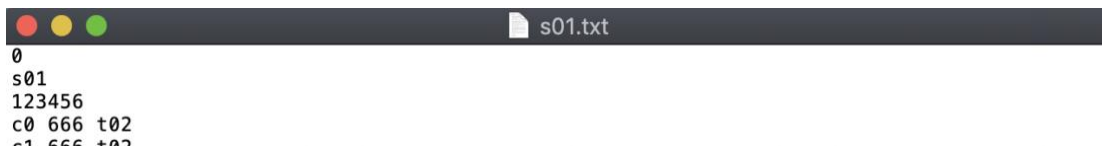
该函数创建新课程，若成功创建则会生成新创建课程的对应文件，并将新课程信息添加到本教师的存储文件中。

3. Student 类

Student
-nameID: String -password: String -whetherUsing: int // And store the courses elected by himself or herself in the file
+getStudentName(): String +getStudentPassword(): String +electCourse(String courseName): int

类成员变量有名称 ID，密码，该账户是否正被使用（0 表示该账户目前未被登录；1 表示该账户已被登录，正在使用中）。每个学生的信息都会被储存在一个以学生名为文件名的.txt 文件中，放置于 Student 文件夹。每个文件中都会包含账户是否正在被使用、教师名称、该账户密码、该学生已选的所有课程这些信息。

Student 储存文件内容如下图所示：



```

0
s01
123456
c0 666 t02
~1 666 +n?

```

其中第一行表示账户是否正被使用，第二行为学生名称，第三行为密码，第四行及后续为该学生已选课程的信息，格式为“课程名 上课时间 授课老师名称”。

关键函数：

```

/*
 *   -1 表示该学生用户文件已存在
 *    1 表示学生用户文件创建成功
 *
 * */
public int makeStudentFile()

```

该函数在 Student 文件夹中创建该 Student 类对应的.txt 文件，若 Student 文件夹不存在，则先创建 Student 文件夹再创建.txt 文件。

```

/*
 *   -2 表示该课程不存在
 *   -1 表示课程人数已满
 *    0 表示该课程已选过
 *    1 表示选课成功
 *
 * */
public int electCourse(String courseName)

```

该函数选择某课程，若选择成功则在本学生的存储文件中添加该课程信息.并在该课程的存储文件中把当前选课人数增加 1，并在文件中添加该学生名称。

4. Test 类

Test (Operations)
-courseNames: Vector<String> // All courses' names -courseTime: Vector<String> // All courses' time -courseTeacher: Vector<String> // All courses' teachers
+readAllCourses(): void +register(String userName, String passwordFirst, String passwordSecond): int +signIn(String userName, String password): int +signOut(String userName): int +getCourseStudentInf(String courseName): Vector<String> +getTeacherCourseInf(String teacherName): Vector<String> +getStudentCourseInf(String studentName): Vector<String>

类成员变量为三个向量，分别是所有课程的名字、时间与开课老师。成员函数为顶层的功能函数，直接被 UI 部分使用。

关键函数：

```
/*
 * 遍历course文件夹内所有文件，
 * 将课程名字放入courseNames向量中，
 * 对应的课程时间放入对应下标的courseTime向量中，
 * 对应的课程老师ID放入对应下标的courseTeacher向量中，
 * !!!已满人的课程不会被放入这些向量
 */
public static void readAllCourses()
```

```
/*
 * -3 表示用户名格式错误
 * -2 表示两次输入的密码不相同
 * -1 该用户名已存在
 *
 * 1 表示以教师身份注册成功
 * 2 表示以学生身份注册成功
 */
public static int register(String userName, String passwordFirst, String passwordSecond)
```

```
/*
 * -3 表示用户名格式错误
 * -2 此用户不存在
 * -1 表示密码错误
 * 0 表示该账户正在被使用
 * 1 表示以教师身份登陆成功
 * 2 表示以学生身份登陆成功
 */
public static int signin(String userName, String password)
```

若登录成功，该函数会将用户对应储存文件中第一行的值由 0 改为 1，表示该账户正在被使用。

```
/*
 * 1 表示该用户退出成功
 */
public static int signOut(String userName)
```

若退出成功，该函数会将用户对应储存文件中第一行的值有 1 改为 0，表示该账户目前未被使用。

```
/*
 * 把这个课程的所有学生的学生ID放入向量中返回，向量中每个元素表示一个课程信息，格式为“学生ID”
 */
public static Vector<String> getCourseStudentInf(String courseName)
```



```

/*
 * 把这个老师创建的所有课程信息存在向量中返回，向量中每个元素表示一个课程信息，格式为“课程名 课程时间 最大人数”
 */
public static Vector<String> getTeacherCourseInf(String teacherName)

/*
 * 把这个学生选的所有课程的信息存在向量中返回，向量中每个元素代表一个课程信息，格式为“课程名 课程时间 老师名”
 */
public static Vector<String> getStudentCourseInf(String studentName)

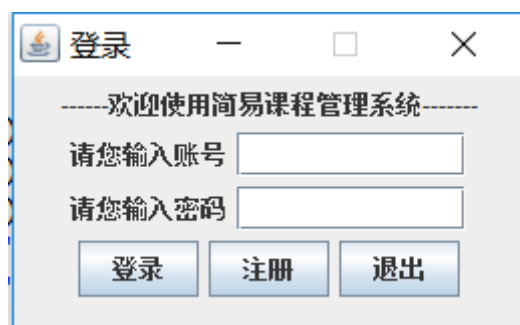
```

5. UI

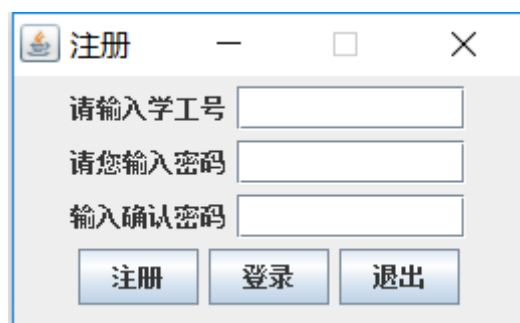
UI
-LoginFrame: class // The UI of login operations -RegisterFrame: class // The UI of register operations -TeacherFrame: class // The UI of teachers' operations -StudentFrame: class // The UI of students' operations +operation() // These four classes contain many operations

整个系统的 UI 分为四个大类——分别为登录界面、注册界面、教师操作界面、学生操作界面。

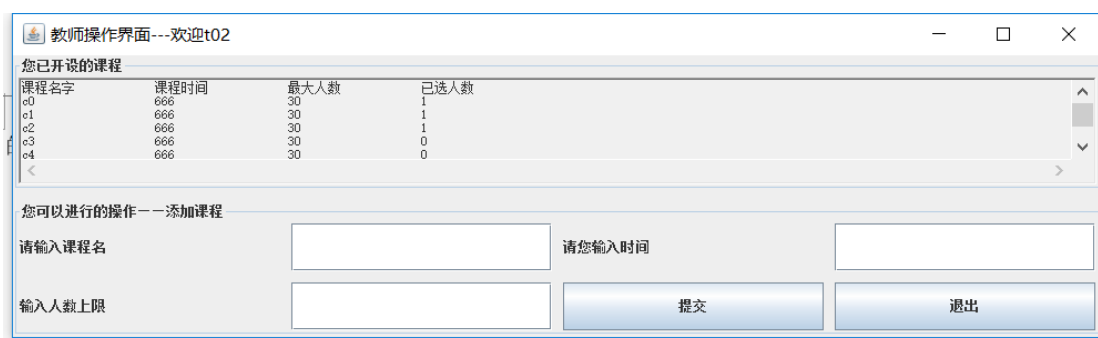
登录界面如下：



注册界面如下：



教师操作界面如下：



教师操作界面---欢迎t02

您已开设的课程

课程名字	课程时间	最大人数	已选人数
c0	666	30	1
c1	666	30	1
c2	666	30	1
c3	666	30	0
c4	666	30	0

您可以进行的操作——添加课程

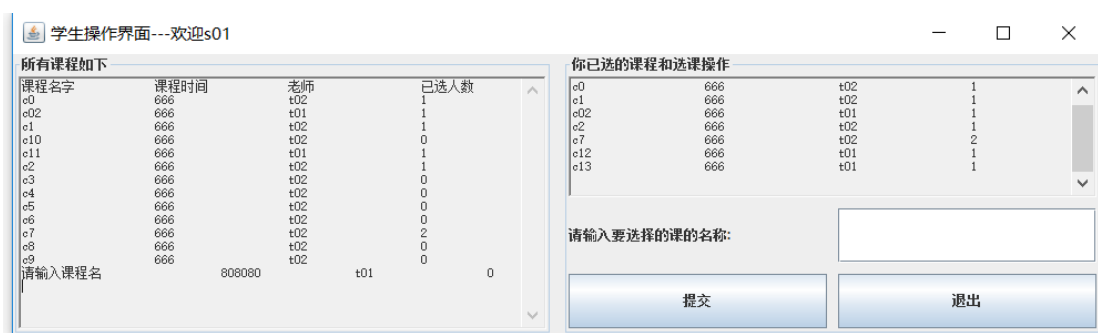
请输入课程名:

请输入课程时间:

输入人数上限:

提交 退出

学生操作界面如下：



学生操作界面---欢迎s01

所有课程如下

课程名字	课程时间	老师	已选人数
c0	666	t02	1
c02	666	t01	1
c1	666	t02	1
c10	666	t02	0
c11	666	t01	1
c2	666	t02	1
c3	666	t02	0
c4	666	t02	0
c5	666	t02	0
c6	666	t02	0
c7	666	t02	2
c8	666	t02	0
c9	666	t02	0

请输入课程名: 808080

你已选的课程和选课操作

课程名字	课程时间	老师	已选人数
c0	666	t02	1
c1	666	t02	1
c02	666	t01	1
c2	666	t02	1
c7	666	t02	2
c12	666	t01	1
c13	666	t01	1

请输入要选择的课的名称:

提交 退出

四、 知识点应用与创新点、技术难点说明

1. 除 UI 外的类

运用了类和对象、异常处理、文件存储读写等基础面向对象编程的知识点

此外，实现这些类的成员函数时，由于要对文件进行频繁的读写操作，而且许多功能的实现需要对文件进行精确到特定行的读取。因此我额外添加了一个 FileOperate 类，其中的成员函数封装了对文件的一些操作，使其他类可以更方便地执行程序对文件的特定操作。

关键函数：

```
// 读取文件指定行。
public static String readAppointedLineNumber(File sourceFile, int lineNumber)
    throws IOException
{
    FileReader in = new FileReader(sourceFile);
    LineNumberReader reader = new LineNumberReader(in);
    String s = "";

    int lines = 0;

    while (s != null)
    {
        lines++;
        s = reader.readLine();
        if((lines - lineNumber) == 0)
        {
            reader.close();
            in.close();
            return s;
        }
    }
    reader.close();
    in.close();
    return s;
}
```

实现了对某文件特定行的读取。（行数从 1 开始）

```
// 文件内共有多少行
static int getTotalLines(File file) throws IOException
{
    FileReader in = new FileReader(file);
    LineNumberReader reader = new LineNumberReader(in);
    String s = reader.readLine();
    int lines = 0;
    while (s != null)
    {
        lines++;
        s = reader.readLine();
    }

    reader.close();
    in.close();
    return lines;
}
```

此函数返回文件总行数，用于一些判断语句。

对于文件特定行内容的修改，我使用的方法是先逐行读出文件内容到一个向量中，然后在此向量中修改值，而后将向量逐行写入原文件（覆盖性写入）。例如：

```

for(int i = 0; i < FileOperate.getTotalLines(courseFile); i++)
{
    everyLineOfFile.add(FileOperate.readAppointedLineNumber(courseFile, i + 1));
}
int newNum = Integer.parseInt(everyLineOfFile.get(4)) + 1;
if(newNum > Integer.parseInt(everyLineOfFile.get(3)))
    return -1;
everyLineOfFile.set(4, Integer.toString(newNum));
BufferedWriter updatePeopleNum = new BufferedWriter(new FileWriter(courseFile));
for(int i = 0; i < everyLineOfFile.size(); i++)
    updatePeopleNum.write(everyLineOfFile.get(i) + newLineChar);
updatePeopleNum.flush();
updatePeopleNum.close();

```

这段代码实现了：学生用户选课后，对应课程文件内“当前选课人数”值的更新。第一个 for 循环将文件每行内容读入到向量中，而后进行当前选课人数与最大人数的比较。人数关系符合逻辑后，更新向量中下表 4 的元素的值（对应文件中“当前选课人数”那一行），然后把向量内容逐行覆盖写入园文件。注意 FileWriter 的构造中没有第二个参量“true”，表示覆盖性的写操作。

2. UI 类

在注册界面与登录界面中使用 Swing 组件中的 JFrame 窗体及流布局 (FlowLayout) 管理器，其中添加相关 JLabel 标签组件及 JTextField 文本组件以及 JButton 按钮组件，最后使用添加事件监听器的方法调用 Test 类的方法实现登录检验等功能。

与登录界面不同的是，教师操作界面使用了 JPanel 面板(一种继承自 java.awt.Container 类)的容器，在 JPanel 面板可以聚集一些组件来进行布局。而上图的教师操作界面正是基于此原理进行编排的。

而提交操作是在添加事件监听下调用了 Test 类中的课程创建的相关静态方法进行验证与创建，在类中还有一私有(private)成员 Teacher 用来保存当前教师的信息和调用相关函数，同时在每一次进行操作之前进行信息校对：

```

@Override
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    if(e.getSource()==btLogout) {
        Test.signOut(teacher.getTeacherNameID());
        JOptionPane.showMessageDialog(this,"退出成功");
        this.dispose();
        new LoginFrame();
    }
    else if(e.getSource()==btPost) {
        updates(teacher.getTeacherNameID());
        String LbName=tfName.getText();
        String Times=pfTime.getText();
        String Num=pfNumber.getText();
        int flag=teacher.creatCourse(LbName, Times, Integer.parseInt(Num));
        if(flag==1) {
            JOptionPane.showMessageDialog(this,"创建课程成功");
            updates(teacher.getTeacherNameID());
        }
        else if(flag==-1) {
            JOptionPane.showMessageDialog(this,"课程已存在，无法创建同名课程");
        }
    }
}

```

以使得多登入用户操作时数据能实时更新。

与教师操作界面类似，学生操作界面仅流布局(FlowLayout)的排布有些许不同。

监听事件管理器也调用了 Test 类中相关静态方法，在类中还有一私有(private)成员 Student 用来保存当前学生的信息和调用相关函数，与教师操作相同，学生的每次操作之前都会对数据进行同步化。

五、 未解决的问题与难点讨论

文件特定行的修改操作并不是很好，每次读出全部内容再写回的方法虽然便于实现，但是运行效率不高。并且这种方法难以用于删除操作。总的来说不是一个很好的解决方案。需要改善文件特定行的修改、删除方法，增进程序的效率，并可额外实现课程删除、退课等功能。

六、 分工说明

王明业 17343107: 除 UI 外所有类和函数的实现, 底层功能的测试, 实验报告的编辑 (除 UI 说明部分)。

温卓沛 17343115: UI 的实现, 其他类和函数与 UI 的对接, 程序封装完成后的测试, 实验报告的编辑 (UI 说明部分)。

幸赞 17343128: 参与程序系统大纲的讨论, 代码的整理, 各项资源的整合与统筹。