Project3 技术报告

姓名：欧穗新 学号：16340173 班级：教务三班

目录

[Project3 技术报告 1](#_Toc483943562)

[一、目的 1](#_Toc483943563)

[二、实验环境 1](#_Toc483943564)

[三、需求分析 2](#_Toc483943565)

[四、实现思路 3](#_Toc483943566)

[A.功能结构图： 3](#_Toc483943567)

[B.数据与函数设计： 4](#_Toc483943568)

[C.设计思路： 7](#_Toc483943569)

[五、输入与输出 8](#_Toc483943570)

[六、心得体会 8](#_Toc483943571)

# 一、目的

实现教务选课系统（包括学生端，教师端以及教务端）

# 二、实验环境

Linux下（g++编译）

# 三、需求分析

根据自己的选课体验，选课系统应该有如下功能：

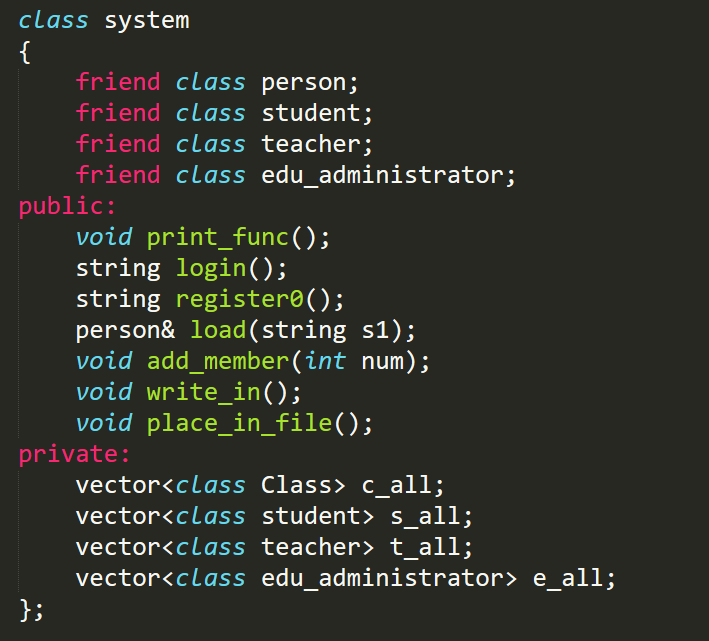
|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 教务端 | 教师端 | 学生端 |
| 用户 | 教务员 | 开课教师 | 在校学生 |
| 需求 | 公共部分 | | |
| 1.所有用户都能够登陆教务系统；  2.如果没有账户可以进行注册；  3.都可以修改自己的密码；  4.都能够查看当前所有课程信息（课程信息包括：课时数，学分，上课时间地点，开课老师，课程最大学生容量，已选人数）；  5.都能够根据 课程名/教师名 称搜索课程； | | |
| 特殊功能 | | |
| 1.能够展示待审核课程（按照老师要求审核的时间顺序排列）；  2.能够上架开课教师所开课程；  3.能够下架所有开课教师所开课程； | 1.能够只展示自己所开的课程（分为上架和未上架）；  2.能够开课（需要经由教务员审核）；  3.能够下架自己开放的课程（可以直接进行操作，下架某个课程）； | 1.可以屏蔽与自己上课时间冲突的课程；  2. 能够屏蔽所有已选完课程；  3.可以选择已经上架的课程；  4.可以退课； |
| 信息记录功能 | 所以用户和课程相关信息都被载入文件，下一次打开程序时信息又被载入到程序当中 | | |

# 四、实现思路

### A.功能结构图：

### B.数据与函数设计：

1. 表示选课系统的system类：

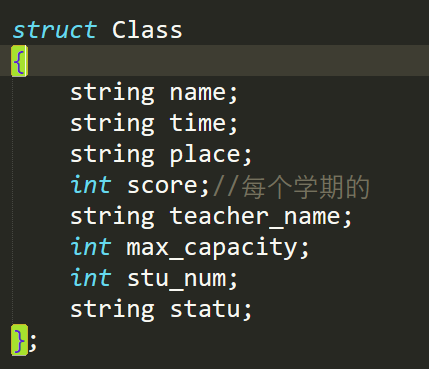


这个类的数据包含：

1. 全部课程信息（Class类的vector）
2. 全部学生的信息（student的vector）
3. 全部老师的信息（teacher类的vector）
4. 全部教务员信息（education administer类的vector）

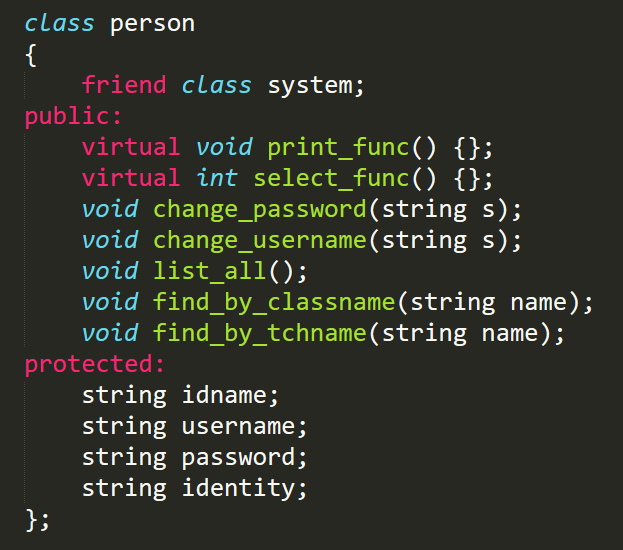
这个类的接口（功能函数）：

1. 打印系统功能（包括登陆、注册）print\_func( )
2. 登陆功能 login（）
3. 注册功能 register（）
4. 添加成员（如果是注册就需要添加成员）add\_member( )
5. 登陆/注册以后载入函数信息 load（）
6. 把文档中成员信息写入程序write\_in（）
7. 推出程序之前把信息保存到文档place\_in\_file（）
8. 课程类class：



数据内容包括课程名字，上课时间、地点，老师名字，每学期学分，最大容量，现有人数，课程状态（已通过批准或者还未通过教务员批准）

1. 成员类person：



作为学生，老师，教务员的父类

数据成员：

用户id，用户名（相当于每个人的名字，可以更改），用户密码，用户身份（分为学生，老师，教务员三种）

函数成员（也即所有成员都可以执行的操作）：

a．虚函数（这个两个函数会根据具体身份调用其子类的函数）

virtual void print\_func(); virtual int select\_func();

b．更改密码或者用户名

void change\_password(string s); void change\_username(string s);

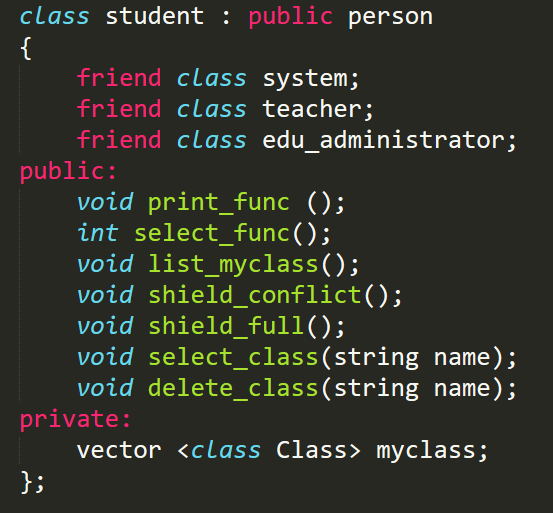
c．列出所有课程

void list\_all();

d．根据名称或者老师查找课程：

void find\_by\_classname(string name); void find\_by\_tchname(string name);

1. Person的子类student、teacher、education administer



学生类，相比于父类，

多出来的数据成员有装载该学生拥有的所有课程的vector ---- myclass

虚函数：void print\_func (); int select\_func();

多出来的操作即学生的操作：

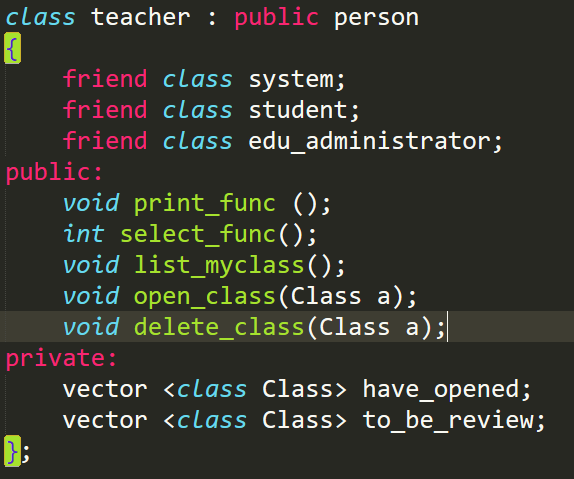
void list\_myclass();列出自己的课程

void shield\_conflict();隐藏冲突课程

void shield\_full();隐藏满员课程

void select\_class(string name);选择课程

void delete\_class(string name);退课



老师类，相比父类，

多出来的数据成员有

装载该老师已开放并通过审核的课程的vector ---- have\_opened

以及正在申请对学生开放的课程的vector ---- to\_be\_review

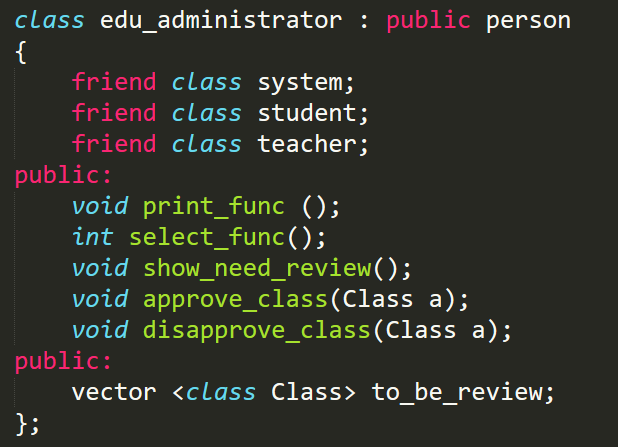
虚函数：void print\_func (); int select\_func();

多出来的函数成员：

void list\_myclass(); 列出课程

void open\_class(Class a); 开课

void delete\_class(Class a); 删除课程



教务员，相比父类

多出来的数据成员有待审核的所以课程to\_be\_review

虚函数：void print\_func (); int select\_func();

多出来的函数：

void show\_need\_review(); 展示所有待审核课程

void approve\_class(Class a); 通过某一课程的审核

void disapprove\_class(Class a); 将某一课程下架（可以让该课程不再对学生开放）

### C.设计思路：

将所有代码及程序文件和数据保存文件分别放在两个不同的文件夹中

*1.当程序打开时，write\_in（）打开数据文件并把文档中成员信息写入程序*

（被写入程序的有：所有课程信息，所有用户信息）

*2.然后进入登陆/注册环节*

登陆：核对用户的id与密码是否相符合

注册：需要用户提供相应的代码来确定用户的身份（是学生还是老师还是教务员）

审核通过以后注册成功

*3.载入用户信息load函数*

*4.进入操作环节*

**步骤三的load函数返回一个person类的引用temp，**

**然后用temp调用虚函数print\_func根据具体载入的用户类型调用具体类型的打印函数；**

**再用temp调用虚函数select\_func根据具体载入的用户类型调用具体类型的选择操作；**

**公共操作：**

更改用户名、密码（这个直接改，很简单）

列出所有课程（这个直接遍历vector<Class> c\_all 打印即可）

查找相应课程（遍历vector，如果该课程name与输入相符，打印即可）

差别操作：

**学生的：**

void list\_myclass();列出自己的课程，遍历vector<Class>myclass 即可

a．隐藏冲突课程---- void shield\_conflict();这个函数通过比较class类的time信息来判断具体课程是否与该学生的课程（myclass）的time冲突，如果冲突，就不打印该课程；

b．隐藏满员课程---- void shield\_full();如果某课程的最大容量等于当前学生人数（即Class类的capacity==stu\_num），就不打印该课程；

c．选择课程void select\_class(string name);即在该学生的vector<Class>myclass中加入这个课程并且所有相应的存有stu\_num的数据都惊醒stu\_num++的操作

d．退课void delete\_class(string name);即在该学生的vector<Class>myclass中删除（erase）这个课程，并且所有相应的存有stu\_num的数据都惊醒stu\_num--的操作

**老师的：**

a．列出课程void list\_myclass(); 遍历vector <Class>have\_opened和 vector <Class>to\_be\_review

即可

b．开课void open\_class(Class a); 定义一个Class对象，要求老师输入课程名称并且完善相应课程信息到该对象，并且判断老师的课程是否存在重名、时间，地点冲突等问题。完成以后，将该对象插入到该老师的vector <Class>to\_be\_review当中，并且插入所有教务员的vector <Class>to\_be\_review中

c．删除课程void delete\_class(Class a);

遍历vector <Class>have\_opened和 vector <Class>to\_be\_review找到名字和对象a的名字相同的成员，删除该成员，

然后遍历所有教务员e\_all的vector <Class>to\_be\_review和所有学生s\_all的vector<Class>myclass，找到同名的对象并删除；

**教务员的：**

a．展示所有待审核课程void show\_need\_review(); 遍历该教务员的vector <Class>to\_be\_review

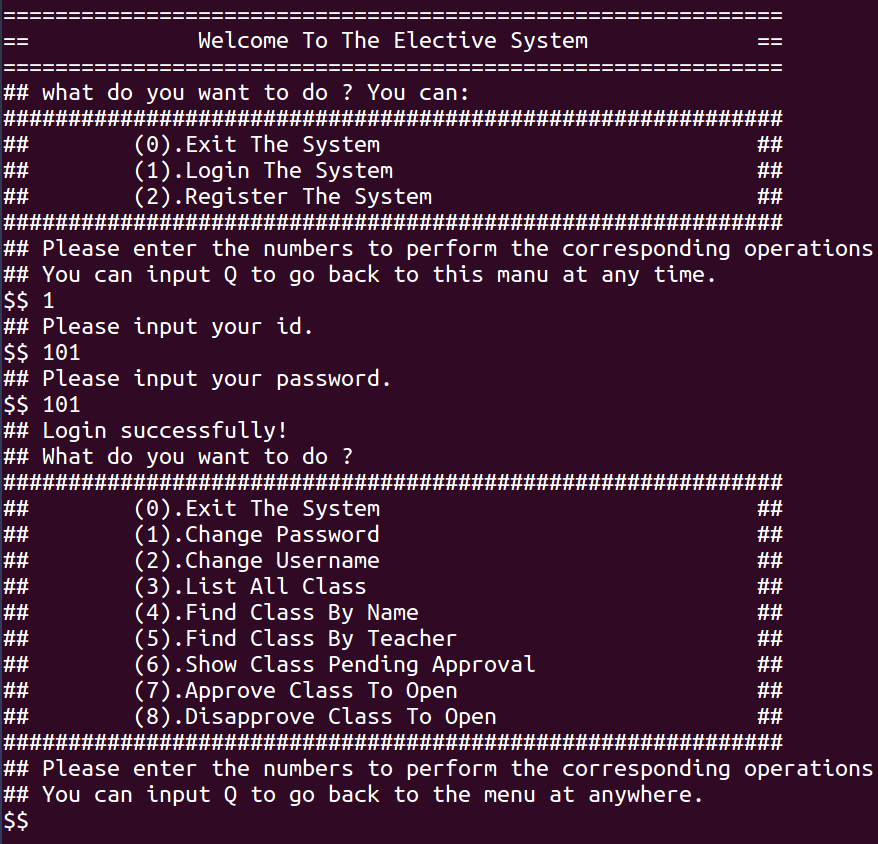
b．通过某一课程的审核void approve\_class(Class a); 遍历该教务员的vector <Class>to\_be\_review，删除名字与对象相同的成员；在全局的c\_all中加入这个对象；遍历所有老师的vector <Class>to\_be\_review，找到相同名字的成员并删除；

c．将某一课程下架（可以让该课程不再对学生开放）void disapprove\_class(Class a); 在所有教务员的vector <Class>to\_be\_review中加入这个对象，stu\_num置为0，表示这个对象需要重新审核；把这个课程对应的老师的vector <Class>have\_opened和 vector <Class>to\_be\_review分别减少和增加一个该对象，stu\_num置为0；遍历所有学生的vector<Class>myclass找到同名课程，删除掉；

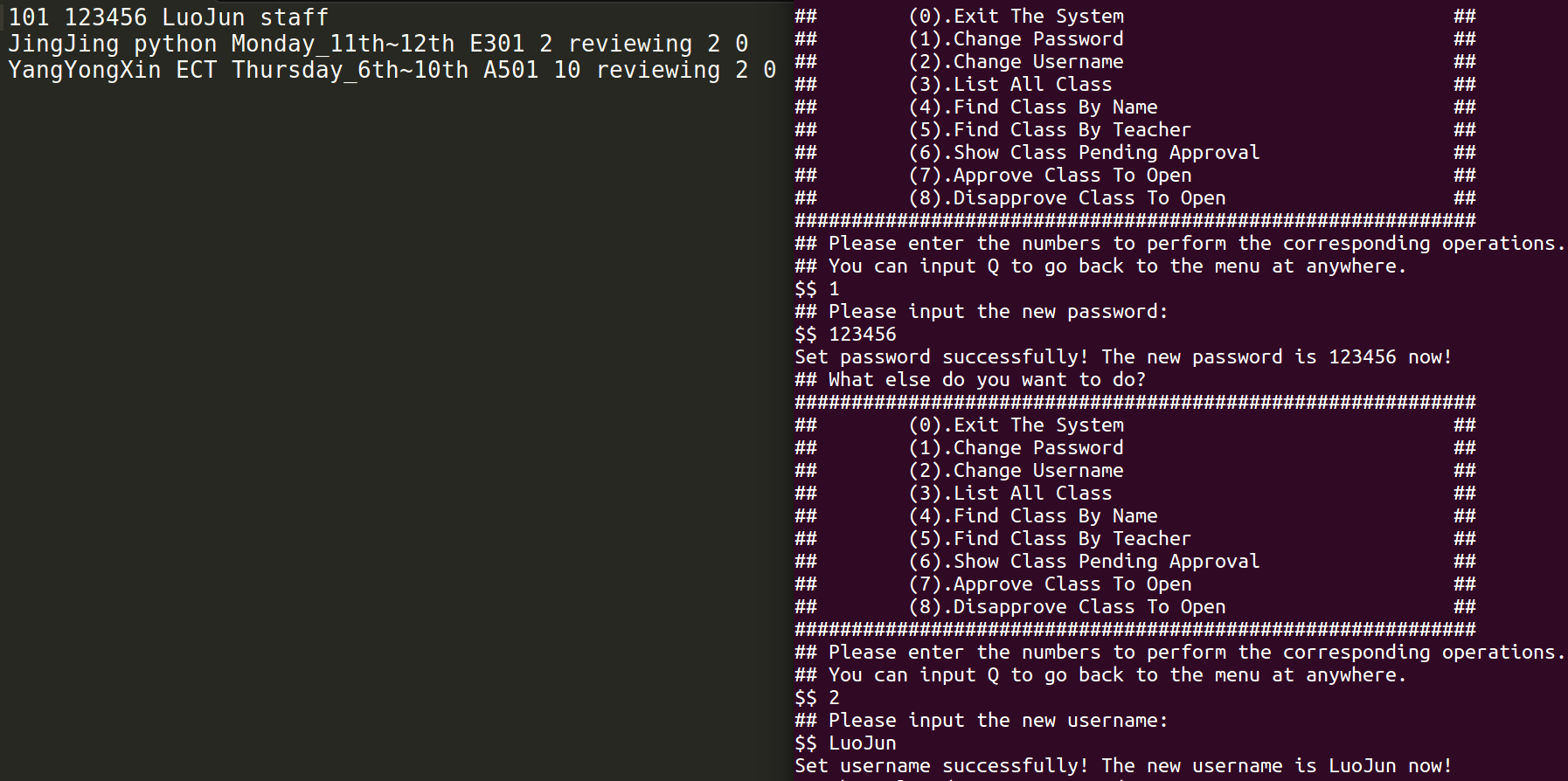
*5.操作完成退出程序*

# 五、输入与输出

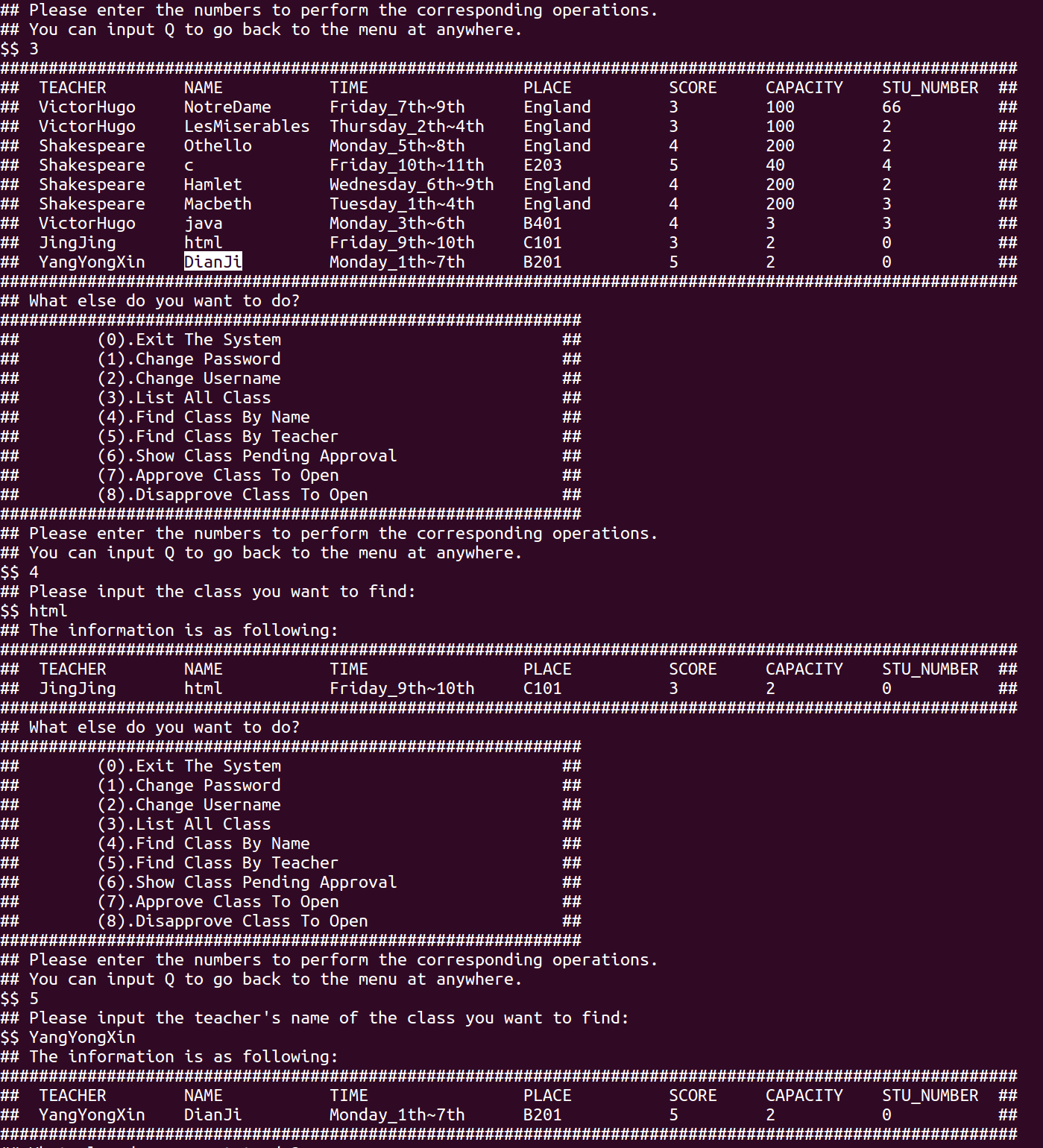
##### 1.进入登陆界面



##### 2.1体验教务员（输入密码和用户名）：

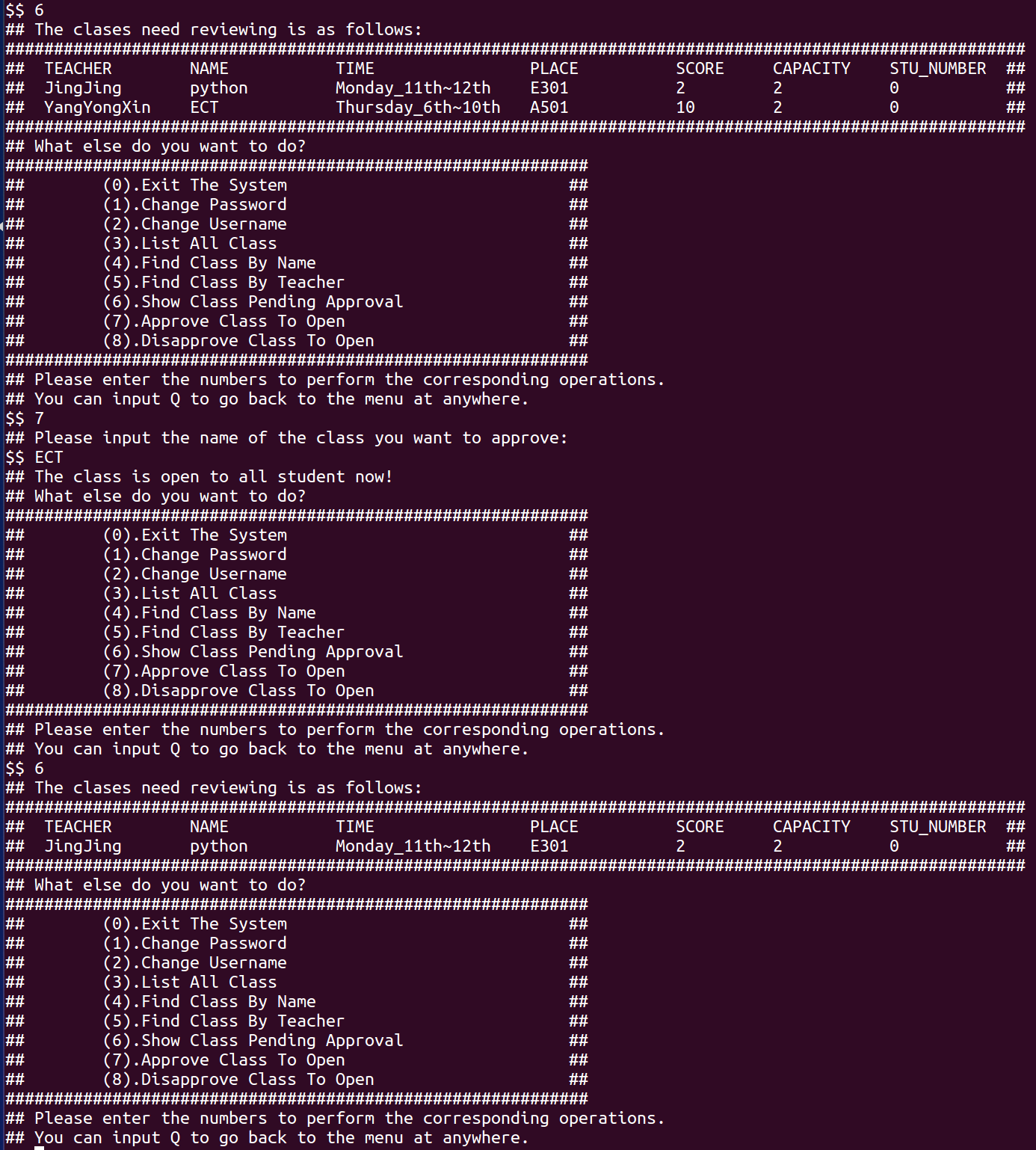


可以看到，更改用户名和密码以后，文件里面保存的信息变化了；



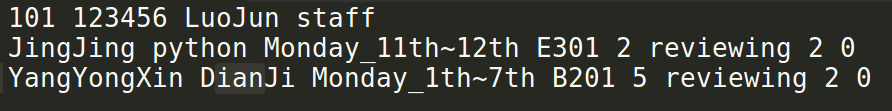
我们尝试了列出所有课程，根据教师，课程名称查找课程的功能；

由于这些功能都由person类实现，所以不同的子类执行会得到相同的结果，所以后续不在测试；

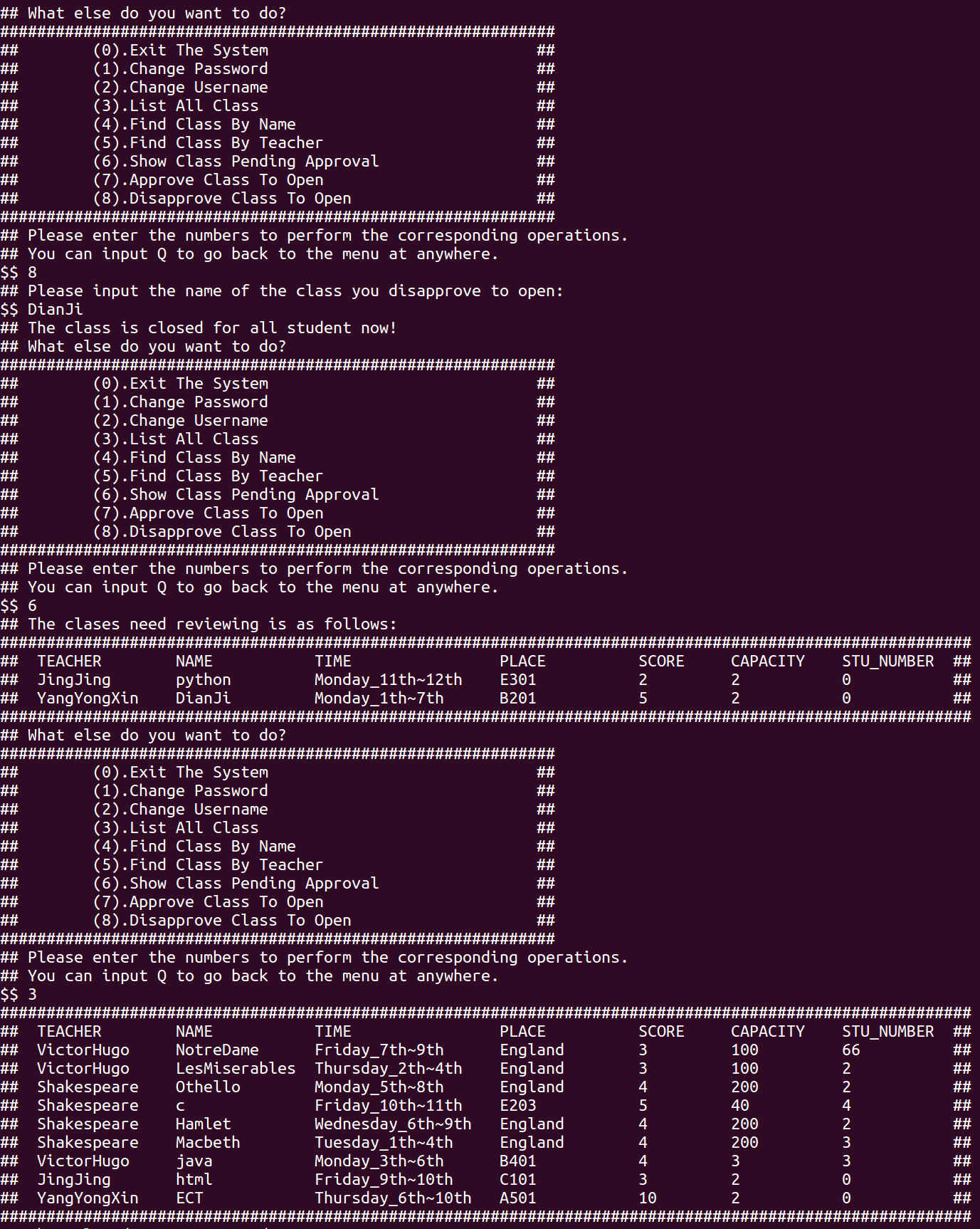


我们通过了ECT课程的审核，可以看到，审核之后，待审核项目少了这一项；

同时，下面给出一张数据文件的截图，可以看出，该课程已经通过了审核；



ECT这一项不见了

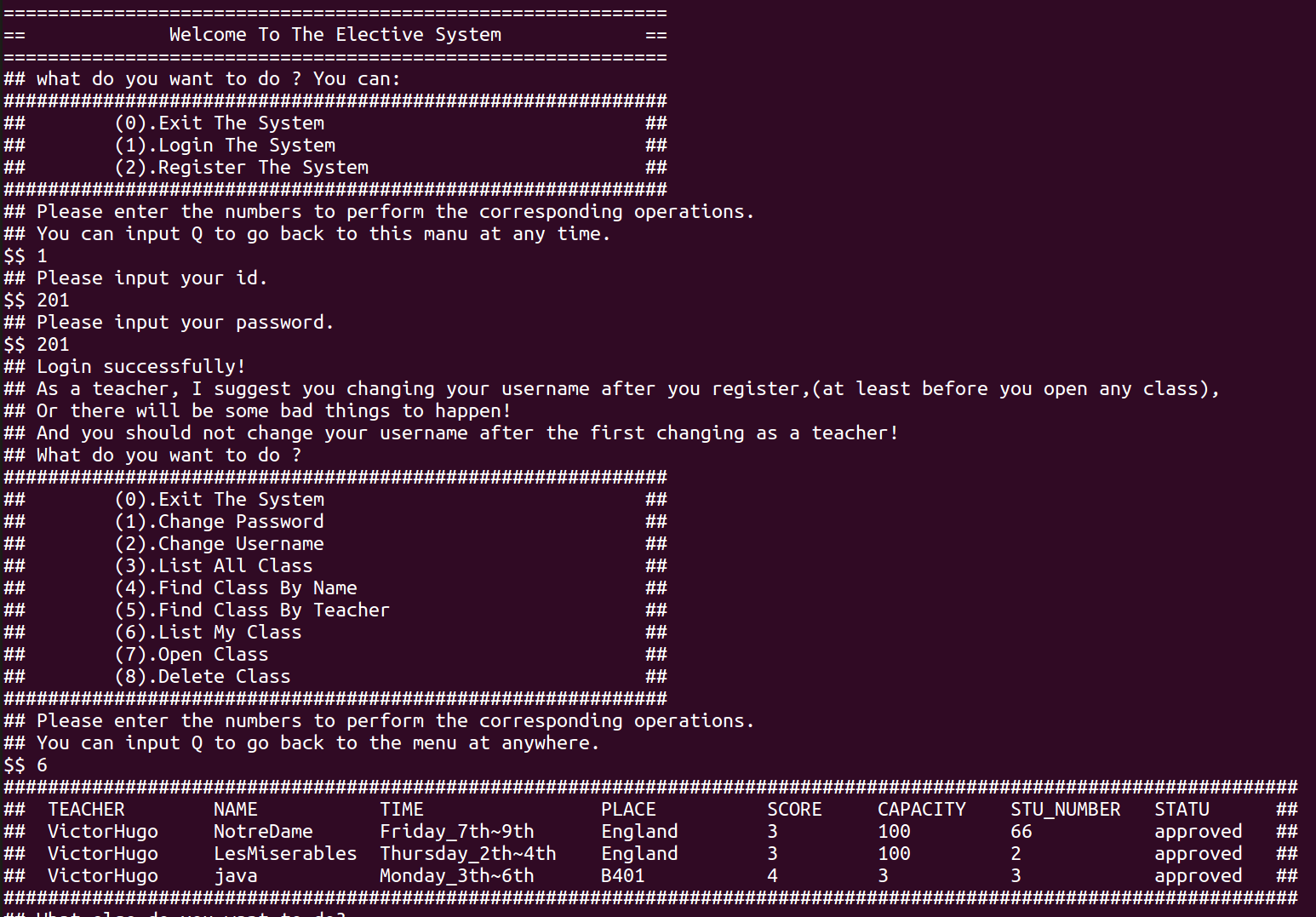


我们又进行了下架课程的操作，发现这个课程已经不见，而且到了待审核项目中；

同时，上一步上架的课程ECT也出现在全部课程中，所以也证明上一部操作真确；

**教务端体验完毕。**

##### 2.2体验教师端：

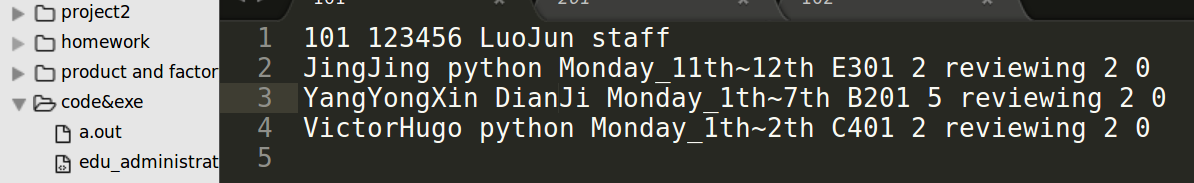


列出老师自己的所有课程；

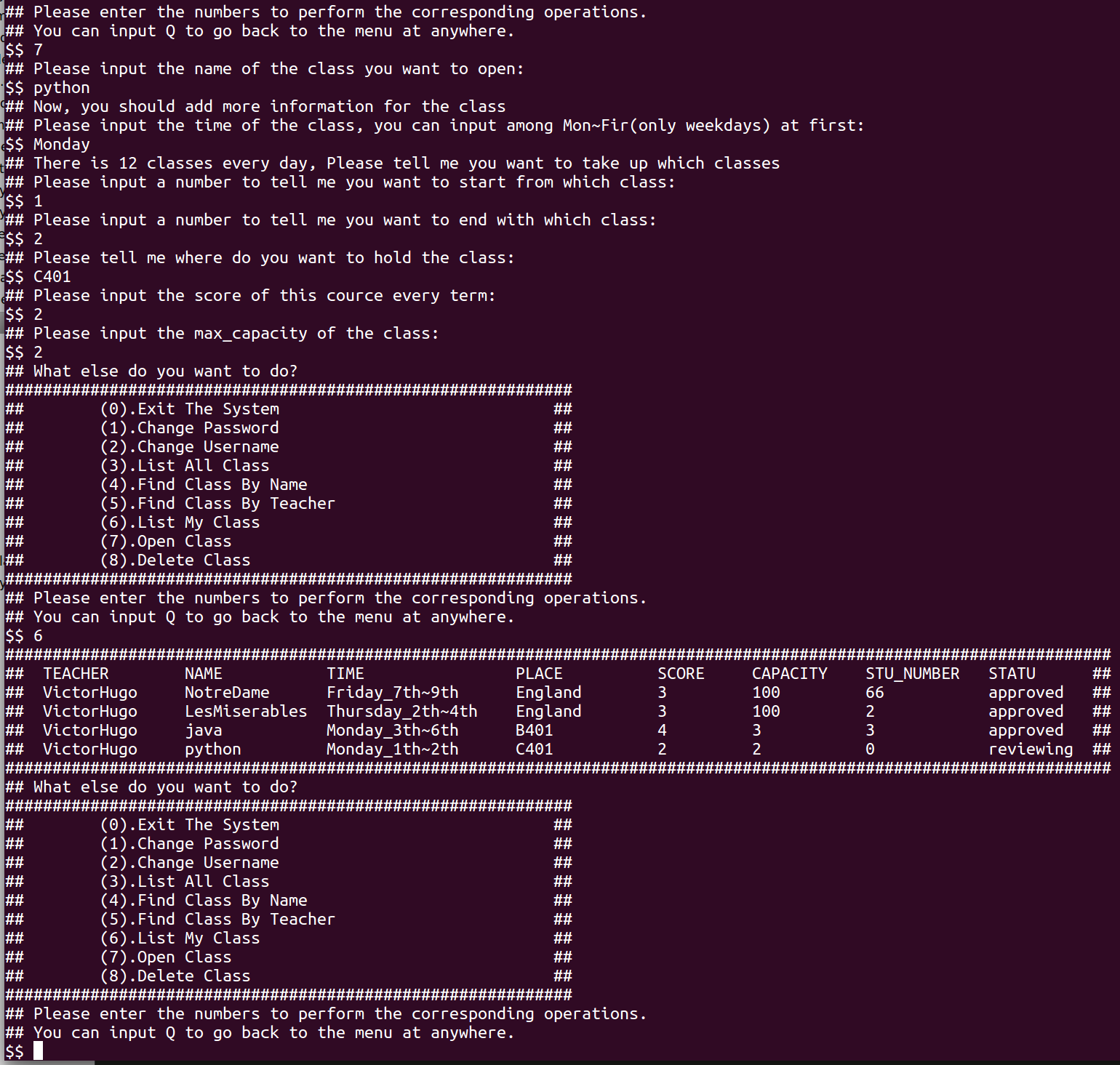
我们可以看到，这里推荐老师在注册以后最好马上更改用户名，否则会导致以前开设的课程需要更改老师的名称，但是这个系统不提供直接更改课程信息的函数，所以需要删除原来开的课，重新再开一次，会很麻烦，所以推荐老师在注册以后最好马上更改用户名，并且不要频繁更改用户名；

在下面两张图片我们可以看到，不可用开设同名的课程，不可以开设冲突的课程，

而且开课以后，在myclass中可以看到，而且状态是reviewing

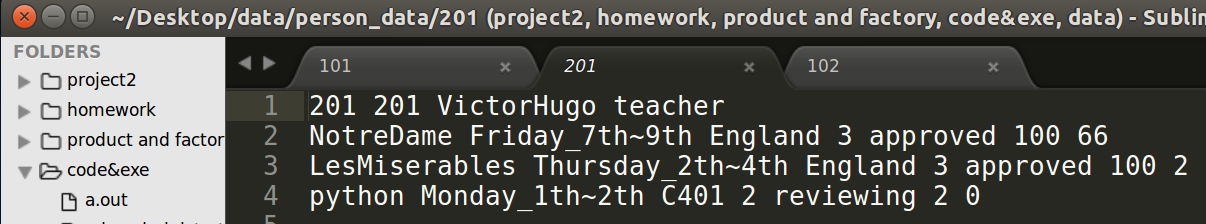


这里看到，在教务员的文件中，出现了这个待审核的文件

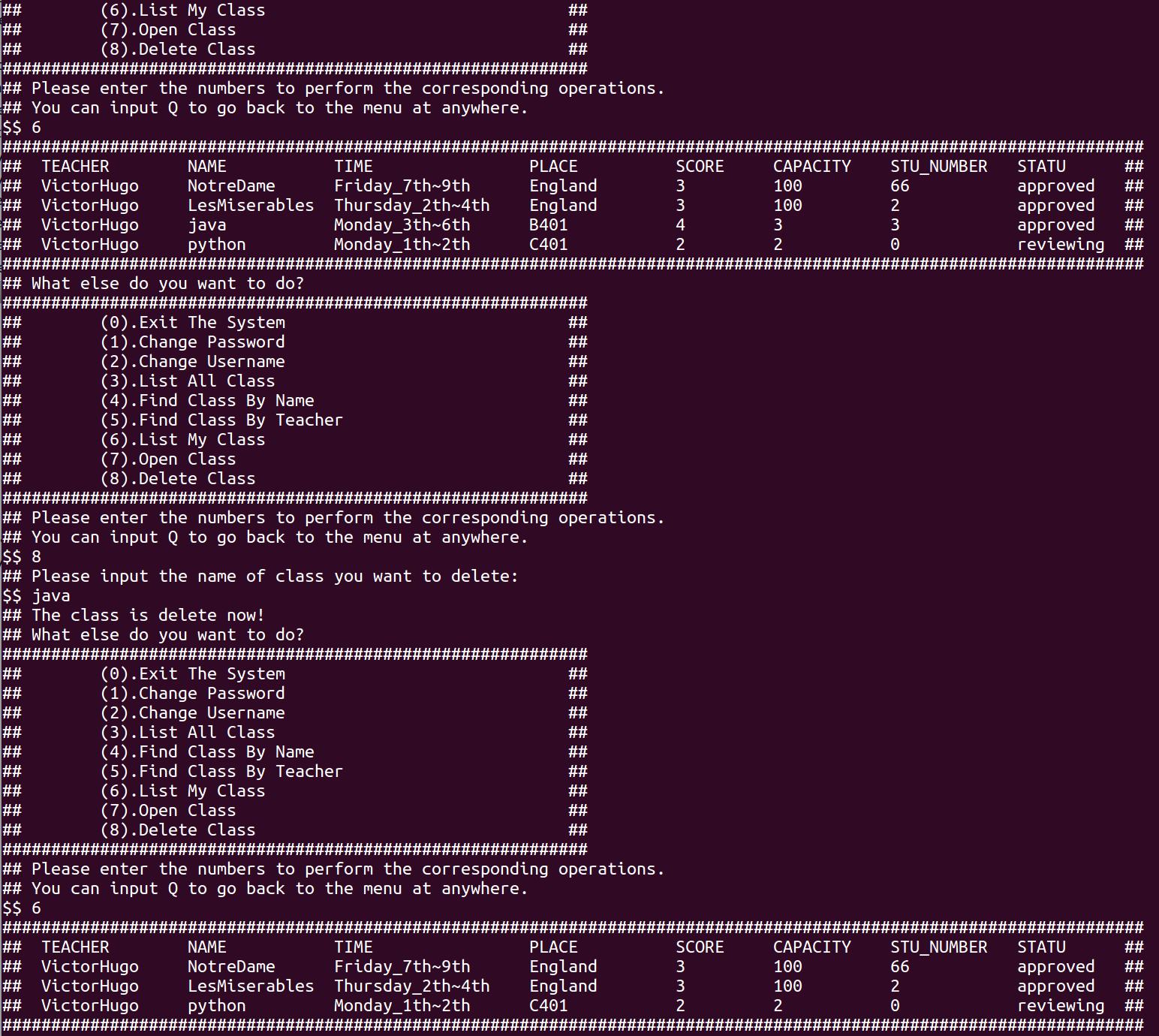


在下面这两张图片，我们可以看到，删除了一个课程，删除以后，在老师的myclass中，没有这个课程了

而且，在相应的文件夹中，Java课程不见了这表示删除成功

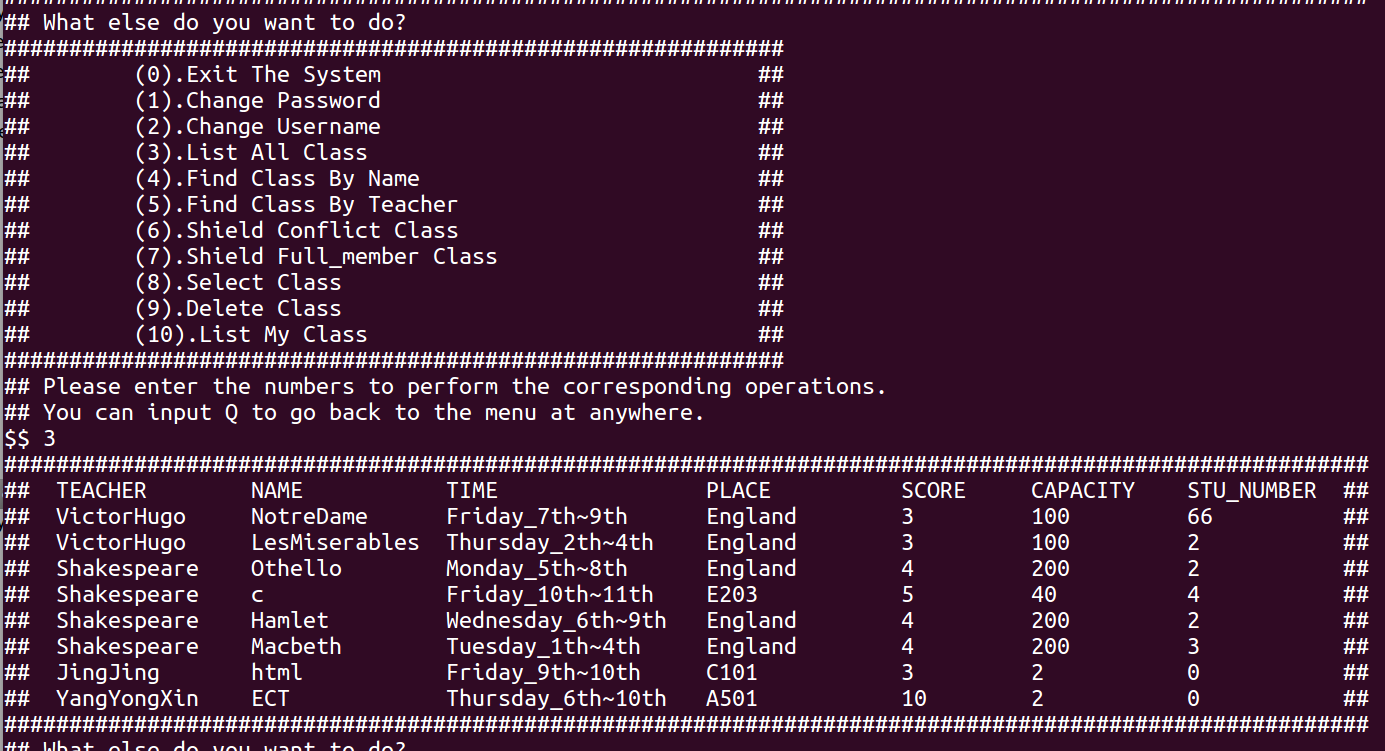


这个文件夹没有Java课程，而且是id为201 的用户的文件夹

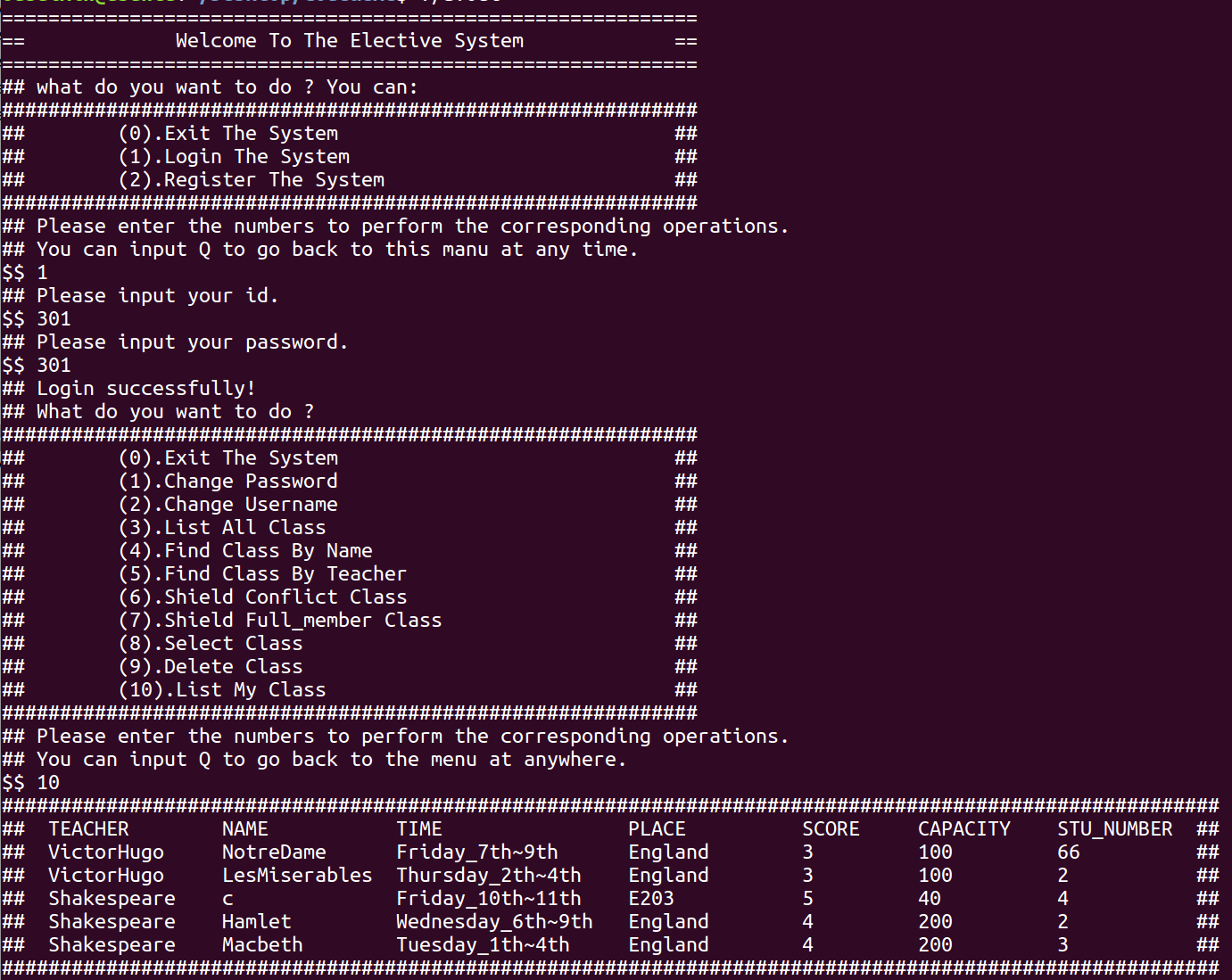


教师端体验完毕；

##### 2.3体验学生端：



首先我们看到一共有哪些课程，这在后续会有用



然后我们进行了打印自己的课程的操作

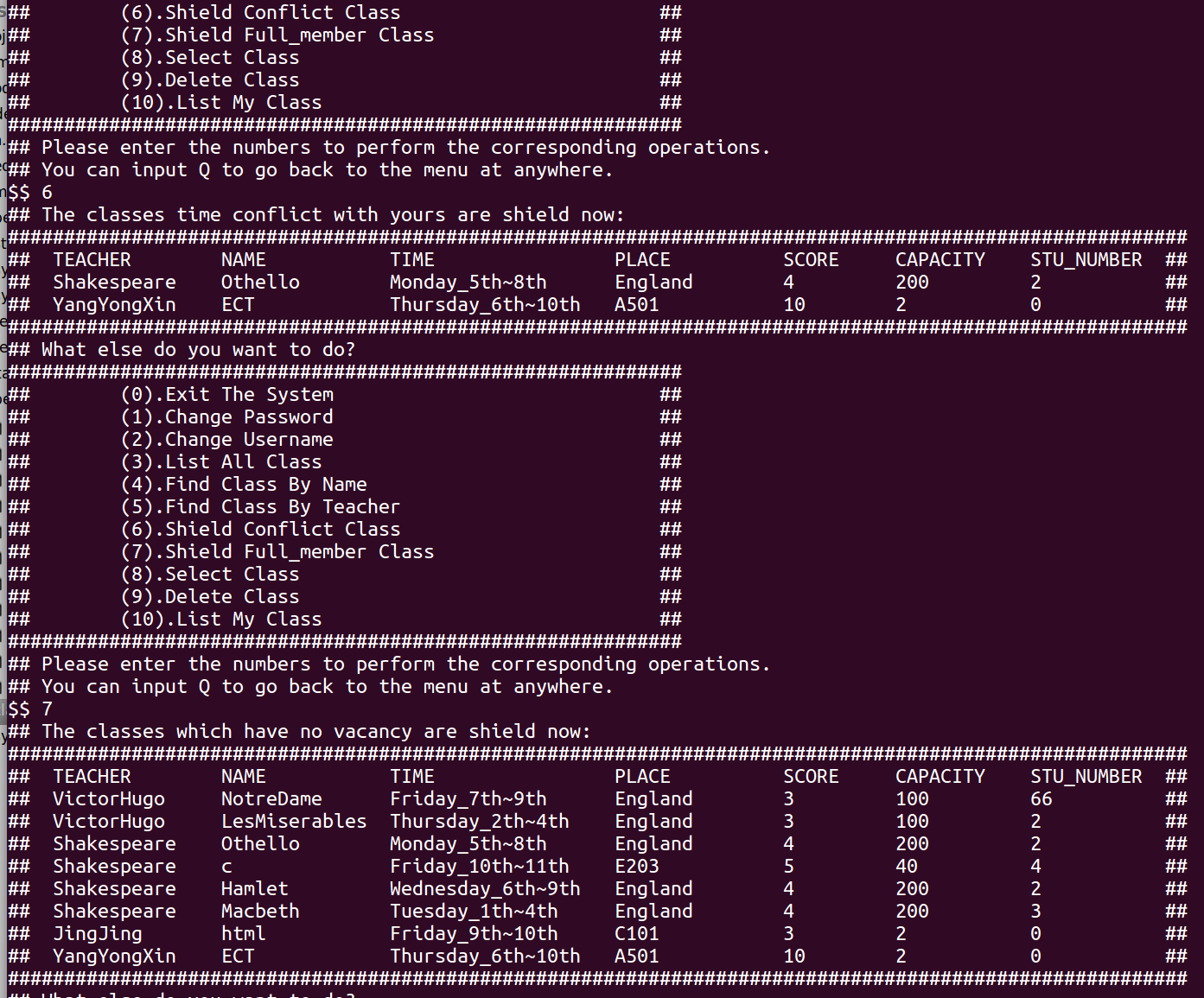
这也是为后续观察做铺垫，同时体验了一下功能10

下面这一张图片则是进行了

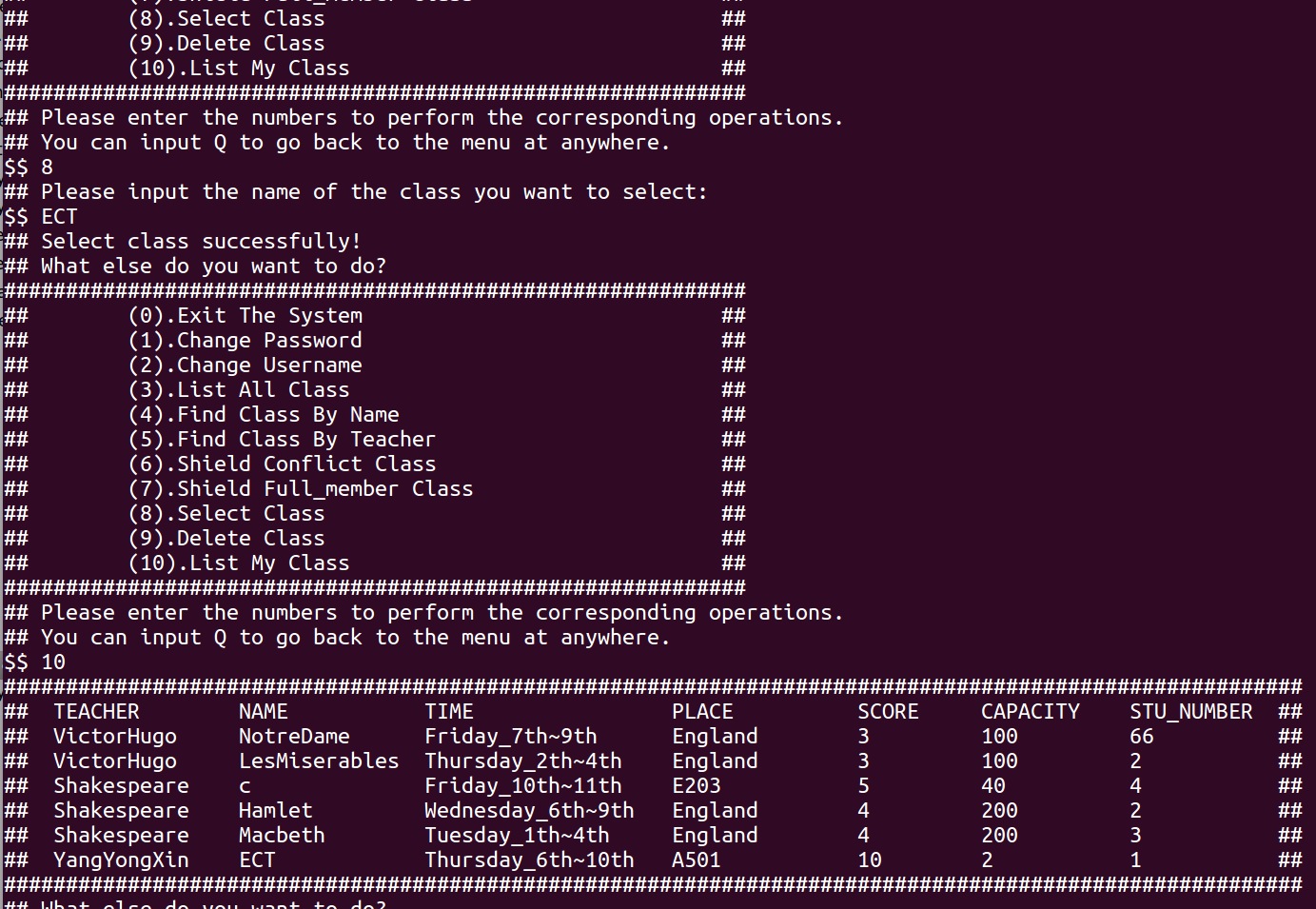
屏蔽冲突课程操作；

屏蔽满员课程操作；

并且结合上面的图片可以看出来，的确是屏蔽了与自己课程冲突的课程

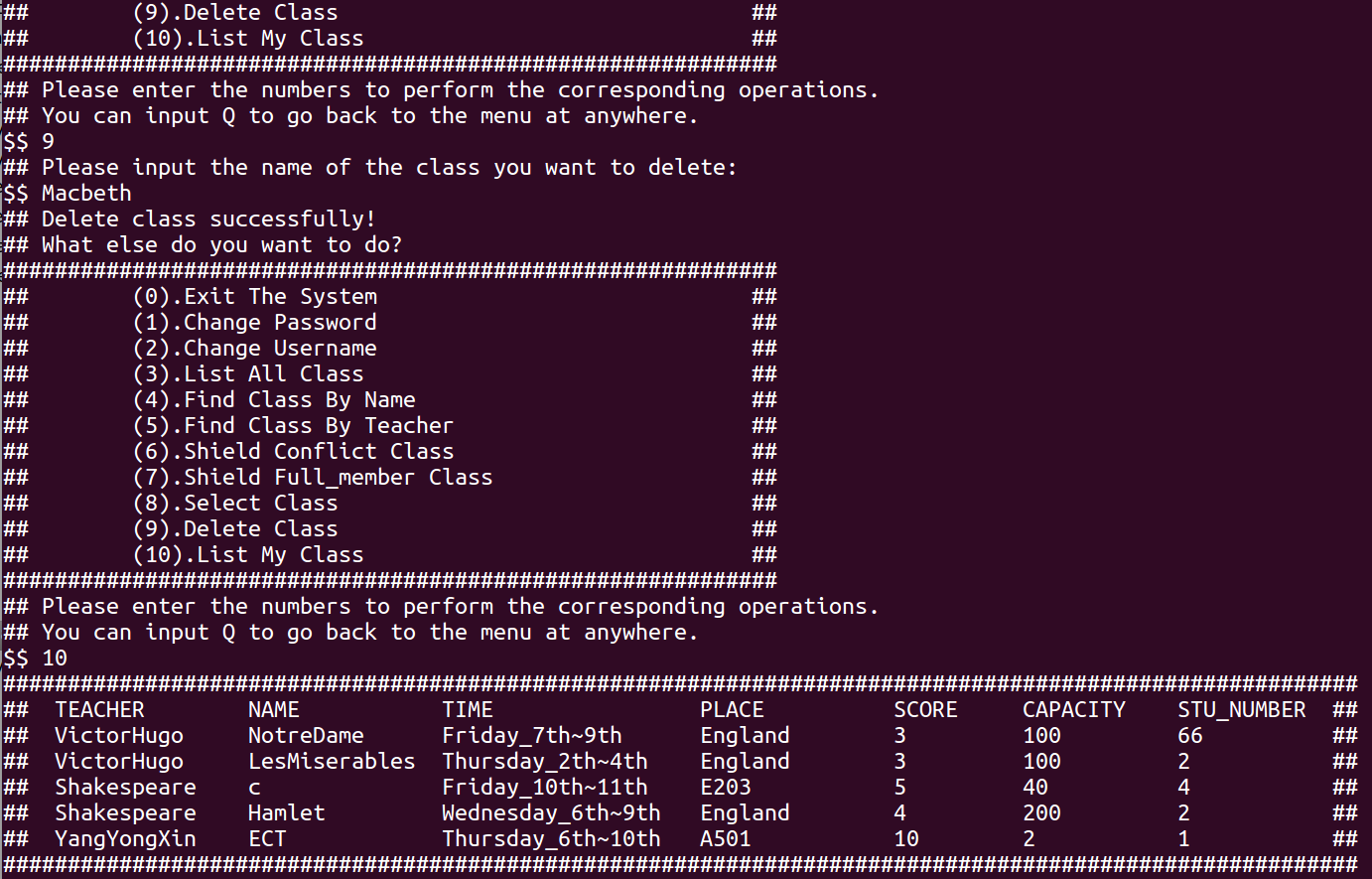


这下面我们体验了选课操作：



选完课以后可以看到，自己的课程里面多了这一项，而且，学生人数有所增加，由原来的0变成了1；

接下来体验删除课程操作：



选完课以后可以看到，自己的课程里面少了这一项，而且，学生人数有所减少，由原来的3变成了2；

# 六、心得体会

我不提供老师修改课程的函数，如果要修改课程，应该先删除原有课程，然后再新建课程，这样也刚刚好可以给教务员进行批准

这一次的project大翻车

其实应该把所有的class都放在一个容器里面，这样既节省了空间，又节省了时间，现在搞成这样简直想死，各种改来改去，我都怕了，作了大死啊，真的是写了好久啊，以后一定要先想好怎么写啊啊啊啊啊啊啊啊，气死了，吃力不讨好。

其实冷静一点就只要在每个函数做一点点的，但是这次真的是气的不行，一开始的设计有问题，导致了事倍功半