Instruction:

After unzipping the project, please open the course\_registration\_system folder and click on Course.java and click run. You will not have to change the path of the MyUniversityCourses.csv since it is already in the folder.

Program Design:

|  |
| --- |
| *User* |
| -username: String  -password: String  -fname: String  -lname: String |
| #User()  +getUserN(): String  +getPassW(): String  +setUserN(u: String): void  +setPassW(p: String): void  +getFirst(): String  +getLast(): String  +setFirst(f: String): void  +setLast(l: String): void  +reportall(courses: ArrayList<Course>): void  +reportrgcourse(n: String, courses: ArrayList<Course>): void |

⇩ ⇩

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Admin | | -username: String  -password: String  -fname: String  -lname: String | | #Admin()  +getUserN(): String  +getPassW(): String  +setUserN(u: String): void  +setPassW(p: String): void  +getFirst(): String  +getLast(): String  +setFirst(f: String): void  +setLast(l: String): void  +veriLoginfo(u: String, p: String): boolean  +search(id: String, sn: String, courses: ArrayList<Course>): Course  +create(cn: String, id: String, m: **int**, csn: **int**, sn: String[], i: String, snum: String, l: String, courses: ArrayList<Course>): void  +delete(id: String, sn: String, courses: ArrayList<Course>): void  +edit(id: String, sn: String, s: String, i: String, courses: ArrayList<Course>): void  +display(n: String, s: String, courses: ArrayList<Course>): void  +register(f: String, l: String): Student  +reportall(courses: ArrayList<Course>): void  +reportfull(courses: ArrayList<Course>): void  +writefull(courses: ArrayList<Course>): void  +reportnames(id: String, csn: String, courses: ArrayList<Course>): void  +reportrgcourse(n: String, courses: ArrayList<Course>): void | | |  | | --- | | Student | | -username: String  -password: String  -fname: String  -lname: String | | #Student()  #Student(String f, String l)  +getUserN(): String  +getPassW(): String  +setUserN(u: String): void  +setPassW(p: String): void  +getFirst(): String  +getLast(): String  +setFirst(f: String): void  +setLast(l: String): void  +veriName(f: String, l: String): boolean  +veriLoginfo(u: String, p: String): boolean  +search(f: String, l: String, students: ArrayList<Student>): Student  +serialization(students: ArrayList<Student>): void  +deserialize(): ArrayList<Student>  +reportall(courses: ArrayList<Course>): void  +reportNofull(courses: ArrayList<Course>): void  +register(id: String, csn: String, n: String, courses: ArrayList<Course>): void  +withdraw(id: String, csn: String, n: String, courses: ArrayList<Course>): void  +reportrgcourse(n: String, courses: ArrayList<Course>): void | |

⇧ ⇧

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | AdminItf | | +username: final  +password: final  +fname: final  +lname: final | | +getUserN(): String  +getPassW(): String  +setUserN(u: String): void  +setPassW(p: String): void  +getFirst(): String  +getLast(): String  +setFirst(f: String): void  +setLast(l: String): void  +veriLoginfo(u: String, p: String): boolean  +search(id: String, sn: String, courses: ArrayList<Course>): Course  +create(cn: String, id: String, m: **int**, csn: **int**, sn: String[], i: String, snum: String, l: String, courses: ArrayList<Course>): void  +delete(id: String, sn: String, courses: ArrayList<Course>): void  +edit(id: String, sn: String, s: String, i: String, courses: ArrayList<Course>): void  +display(n: String, s: String, courses: ArrayList<Course>): void  +register(f: String, l: String): Student  +reportall(courses: ArrayList<Course>): void  +reportfull(courses: ArrayList<Course>): void  +writefull(courses: ArrayList<Course>): void  +reportnames(id: String, csn: String, courses: ArrayList<Course>): void  +reportrgcourse(n: String, courses: ArrayList<Course>): void | | |  | | --- | | StudentItf | | +username: final  +password: final  +fname: final  +lname: final | | +getUserN(): String  +getPassW(): String  +setUserN(u: String): void  +setPassW(p: String): void  +getFirst(): String  +getLast(): String  +setFirst(f: String): void  +setLast(l: String): void  +veriName(f: String, l: String): boolean  +veriLoginfo(u: String, p: String): boolean  +search(f: String, l: String, students: ArrayList<Student>): Student  +serialization(students: ArrayList<Student>): void  +deserialize(): ArrayList<Student>  +reportall(courses: ArrayList<Course>): void  +reportNofull(courses: ArrayList<Course>): void  +register(id: String, csn: String, n: String, courses: ArrayList<Course>): void  +withdraw(id: String, csn: String, n: String, courses: ArrayList<Course>): void  +reportrgcourse(n: String, courses: ArrayList<Course>): void | |

|  |
| --- |
| Course |
| -courseName: String  -courseID: String  -maxStudN: int  -currentStudN: int  -studNames: String[]  -instructor: String  -secNum: String  -location: String |
| #Course()  #Course(cn: String, id: String, m: int, csn: int, sn: String[], i: String, snum: String, l: String)  +setCName(cn: String): void  +setID(id: String): void  +setmaxN(m: int): void  +setCurStudN(csn: int): void  +setStudName(sn: String[]): void  +setInstr(i: String): void  +setSecN(snum: String): void  +setLoc(l: String): void  +getCName(): String  +getID(): String  +getmaxN(): int  +getCurStudN(): int  +getStudName(): String[]  +getInstr(): String  +getSecN(): String  +getLoc(): String  +csvreader(): ArrayList<Course>  +serialization(ArrayList<Course> courses): void  +deserialize(): ArrayList<Course>  +main(String[] args): void |

|  |
| --- |
| StudNumSorter |
| N/A |
| compare(o1: Course, o2: Course): int |

Stage 1: Login:

1. FIRST LOGIN PAGE: Ask if is an admin or student (A for admin and S for student)
2. Ask for username and password to log in (if is an admin, the username and password should be Admin and Admin001; if is a student, then U and P are self-defined) Note: the students can only be registered by the admin; suppose there is only one admin in this system
   1. Student login page: #SECOND LOGIN PAGE# ask for full name first, #THIRD LOGIN PAGE# and then U and P
      1. Check the student name:
         1. For unregistered students (name not in system): display message: You must be registered by the Admin first!
         2. Registered students (name in system):
            1. First-time: ask them to create their own U and P (check if the U has been created by someone else; if yes, throw an error/a message)
            2. If returning:

Ask for U and P

* 1. SECOND LOGIN PAGE: Admin login page:
     1. Only U and P are required (but where should the admin enter full name?)

Stage 2: Display menus respectively for the admin and the students

1. Admin main menu: (C or R or E)
   1. If C: Course Management (into submenu) (enter 1~6)
      1. Create a course (ask for every piece of information) (what if already exists)
      2. Delete a course (by ID + section number) (what if does not exist)
      3. Edit a course (everything except for course ID and name, the admin is also allowed to edit the N of students in a course (which is strange) but let it crash if any problem rises)
      4. Display information of a course (provide (name)/ID + section number)
      5. Register a student (First and last names are needed) (what if already registered)
      6. Exit (to the main menu)
   2. If R: Report (into submenu) (enter 1~7)
      1. All courses
      2. Full courses
      3. Copy full courses to a txt file?
      4. Students’ names in a course
      5. A given student’s registered course(s)
      6. Sort courses based on num of students
      7. Exit (to the main menu)
   3. Exit account (to the first login page)
2. Student main menu (enter 1~6):
   1. View courses
   2. View NOT full courses
   3. Register a course (provide full name, course name, section) (what is the student is already registered)
   4. Withdraw from a course, same req as registering (what is the student is not registered in the first place)
   5. View registered course(s)
   6. Exit account (to the first login page)

I overridden the abstract methods in the User abstract class in both of the Admin and Student class, including the getters and setters of username, passwords, first names, and last names, the login information verification method, the method for reporting all courses and the method for reporting courses that a given student registered on. The Admin and Student classes are inherited from the User abstract class, the AdminItf interface and the StudentItf interface. The Course class is inherited from the ArrayList<Course> class. I also encapsulated all my data fields as private for not letting the fields be confounding with each other.