





Declaration of Original Work for CE/CZ2002 Assignment

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below.

We have honored the principles of academic integrity and have upheld the Student Code of Academic Conduct in the completion of this work.

We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work. In addition, disciplinary actions may be taken.

| Name | Course | Lab group | Signature/Date |
|----------------------------|--------|-----------|--|
| Angel Han Feng Yi | CZ2002 | SS1 |  24/11/2020 |
| Aileen Laksmono Lie | CZ2002 | SS1 |  24/11/2020 |
| Bachhas Nikita | CZ2002 | SS1 |  24/11/2020 |
| Han Xiao | CZ2002 | SS1 |  24/11/2020 |



NANYANG
TECHNOLOGICAL
UNIVERSITY

CZ2002 Object-Oriented Design & Programming

Semester 1 AY 2020/21

Project Report

SS1

Group 4

Angel Han Feng Yi (U1922554E)

Aileen Laksmono Lie (U1920118E)

Bachhas Nikita (U1921630F)

Han Xiao (U1920108G)

Submission Date: 25 November 2020

Design considerations, principles and OO concepts

Approach taken:

Our design approach is to separate the classes to have separate responsibilities and prevent having a god class.

There are 5 main classes identified by our group, they are **login**, **index**, **student**, **admin** and **course**. The student and admin class is designed to facilitate actions conducted by student and admin through accepting input from the boundary class and using the methods in their respective entity classes to fulfil the selected command.

We stored our data in CSV files. BufferedReader is used to read and write the CSV files.

| CSV Filename | Details |
|-----------------------|--|
| student login details | Contains Student Username and Hashed Password |
| admin login details | Contains Admin Username and Hashed Password |
| student | Contains Student name, matric number, gender and nationality |
| courses | Contains Course's School, CourseCode, IndexNumber and AU |
| StudentCourseList | Contains list of registered courses |

The login class consists of sub-menu methods so as to modify code easily and also have a neat structure. It will be easily readable and understandable.

User Interface:

- Users have to enter between 1 to ? to select the option in the menu on display
- Each student has a schedule containing the list of courses registered.
- When a user is entering the password, the password would be masked to prevent third-party users from viewing it. The password key would be converted to a string of characters for example. This string of characters will then be compared to the 2nd column of login details CSV file. This way, passwords stored in the CSV file are not in clear text to increase the level of security.

- Exception handling is done to check for invalid input. Eg. entering alphabets when numbers should be entered.
- Logic check is done on user's input. A few examples are shown below:

| Situation | Checks |
|--|---|
| Student types in Course Code of course to add course | Check if the student is already registered for the course. If student is already registered for the course, student cannot add the course again. |
| Student types index she wants | Check if index entered is for the course code entered |
| Student wants to swap index with friend | Check if both friend and user are registered for the entered course code. Check if both friend and user are registered for the index entered. Check for clash if the user and friend has the new index. |

Principles:

Single responsibility Principle

For each class in our project, they have their own responsibility. For example, the lecture class only contains the methods related to the lecture venue and date and time, and anything unrelated will not be inside.

Open-Closed Principle

Abstraction has been implemented for lecture, tutorial, seminar and lab. They all use different databases, however they use similar methods to get the venue, date and time. This allows the different classes to access different databases without changing the source code of the module.

Assumptions:

1. AU limit of 21 for students.
2. Each course only belongs to one school.
3. The administrator will enter data that makes sense. Eg. when adding course, admin will enter an existing/valid course code.
4. The email entered by the administrator is valid when adding a student in the database.
5. The pre-loaded data in the CSV databases are all correct and accurate.
6. All CSV databases are pre-loaded with existing data.

OO Concepts:

Inheritance

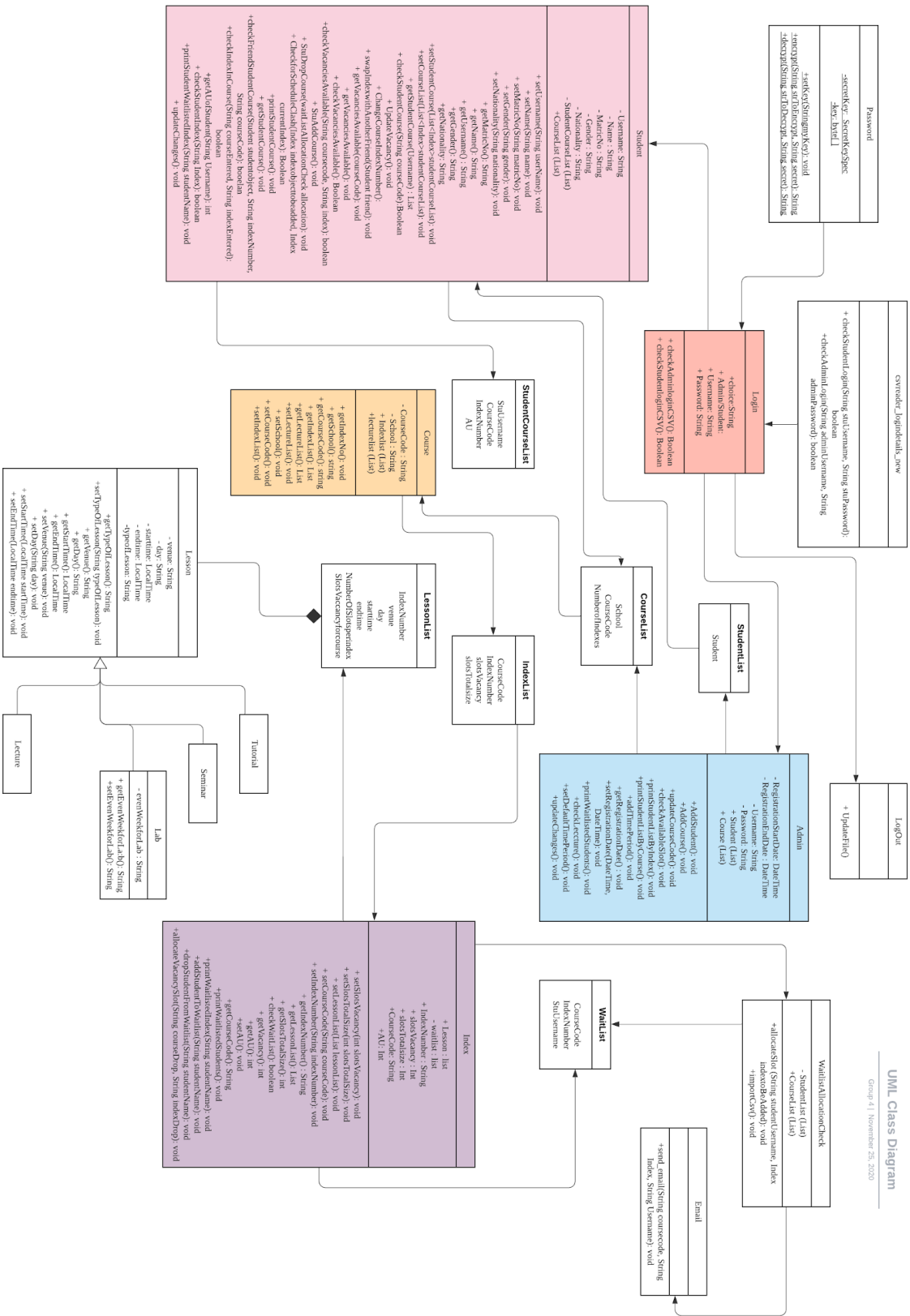
- Lesson is the parent class, lecture, tutorial and seminar class inherit the attributes and behaviour from the parent class. This enables code reuse and can greatly reduce programming effort. This is an “is-a” relationship.

Object composition

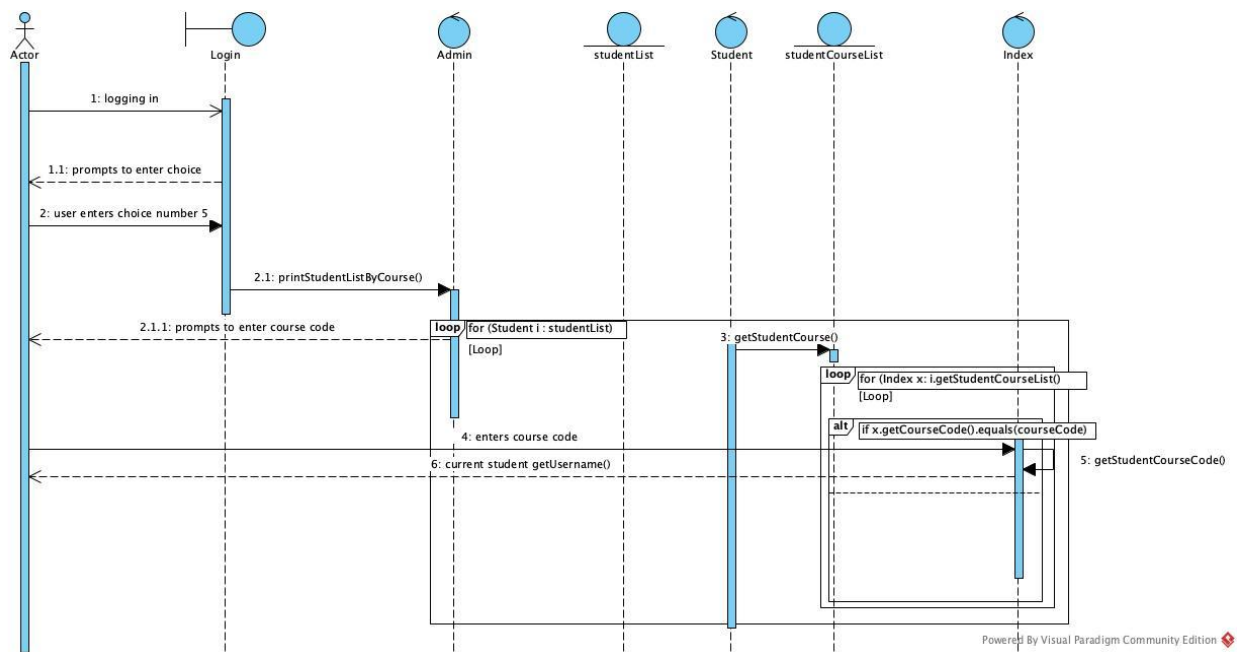
Whole-part relationship

The Index object is contained in the Course object. The contained object, Course, cannot exist without the existence of the container object. In this case, Course is the whole and the index is part of the course object.

UML Class Diagram



UML Sequence Diagram : “Print student list by course” function.



Testing

Student login

| | Test Case | Expected Outcome |
|---|-------------------------------------|---------------------------|
| a | Login before allowed period (dates) | Invalid registration date |
| b | Login after allowed period (dates) | Invalid registration date |
| c | Wrong password and username | Login unsuccessful |

Add Course and student

| | Test Case | Expected Outcome |
|---|-----------------------------|------------------------------|
| a | Add a course with new index | course added in CSV |
| b | Add a new student | student details added in CSV |

Print student list by course and index

| | Test Case | Expected Outcome |
|---|-----------------|------------------|
| a | Print by course | List appear |
| b | Print by index | List appear |

Check slots in a class

| | Test Case | Expected Outcome |
|---|---------------------------------|-----------------------------|
| a | Enter the index and course code | Number of vacancy displayed |

Day/time clash with other course

| | Test Case | Expected Outcome |
|---|-------------------------------|------------------|
| a | Add student to a course index | Show clash |

Swap index

| | Test Case | Expected Outcome |
|---|-------------------|----------------------|
| a | Successful swap | Successful message |
| b | Unsuccessful swap | Unsuccessful message |

Waitlist notification

| | Test Case | Expected Outcome |
|---|---|---|
| a | Add student to course with 0 vacancy | Student added to waitlist, email sent |
| b | Student B drop course and student A is added in from waitlist | Student A added into the course, email sent |

Link to project demonstration:

<https://youtu.be/uH1fUIElwYA>