



CZ2002 LAB GROUP SS6

ASSIGNMENT GROUP 2

LAB REPORT

STUDENT COURSE REGISTRATION AND MARK ENTRY Application (SCRAME)

Prepared By:

Chan Yuqian

Lee Ming Hin

Nguyen Ngoc Khanh

Thong Hoi Wei

Trinh Tuan Dung

**Contents**

[**1. Design Considerations**](#_wmojbzpr3t6) **3**

[1.1 Approach Taken](#_se7za3412m7w) 3

[1.2 Principles Used](#_463bqj3zgfvy) 4

[1.3 Assumptions Made](#_gvalyqc8sbwd) 4

[1.4 Current Found Errors:](#_bhdox6ydpkmz) 5

[**2. UML Class Diagram**](#_nmrem74araay) **5**

[**3. UML Sequence Diagram**](#_qwyfzvjfj15d) **6**

[**4. Test Case and results**](#_39r2kzd62j9f) **7**

[4.1 Add a student](#_7wmocayasdek) 7

[4.1.1 Add a new student](#_tvax0wialnwp) 7

[4.1.2 Add an existing student](#_ikavnn4ioxq1) 7

4.1.3 [Invalid data entries](#_4vsir9mzvye6) 8

[4.2 Add a course](#_jmplks1ku4j0) 8

[4.2.1 Add a new course](#_n9xgfdg2wkf7) 8

[4.2.1.1 A course with 1 lecture group only](#_6itqo9g4czms) 8

[4.2.1.2 A course with 1 lecture group and 2 tutorial groups](#_2gw5rk4yimki) 9

[4.2.1.3 A course with 1 lecture group , 2 tutorial groups and 2 lab groups](#_i906tagcxorj) 10

[4.2.2 Add an existing course](#_fj2v5kgj7o0q) 11

[4.2.3 Invalid data entries](#_11e1lv6gkoxo) 11

[4.3](#_deah5eb7h2h0) Register for a course 11

[4.3.1](#_icb1noepltht) Add a student to a course with available vacancies in Tut / Lab 11

[4.3.2](#_lm0zoju7n6bs) Add a student to a course with 0 vacancies in Tut / Lab 12

[4.3.3](#_6m1xulbr4e7q) Register the same course again 12

[4.4](#_rs9gfhbfpl89) Check available slot in a class 12

[4.4.1](#_o0t378f4933b) Check for Tutorial / Lab class 12

[4.4.2](#_wdi0c0v1b9kl) Invalid data entries 12

[4.5 Print student list by lecture, tutorial or laboratory session for a course](#_deah5eb7h2h0) 12

[4.5.1 Print list by Lecture](#_icb1noepltht) 12

[4.5.2 Print list by Tutorial / Lab](#_lm0zoju7n6bs) 13

[4.5.3 Invalid data entries](#_6m1xulbr4e7q) 13

[4.6](#_gsrdoeavtsns) Enter course assessment components weightage 13

[4.6.1](#_181qfjp5fi15) Enter course assessment with only exam + 1 coursework component 13  
4.6.2 Enter course assessment with exam + 2 coursework component 13

[4.6.2 Invalid data entries](#_e22cgwirekrk) 13

[4.7](#_deah5eb7h2h0) Enter coursework mark - inclusive of its components 14

4.7.1 Enter valid coursework mark for course with only 1 main component 14

4.7.2 Enter valid coursework marks for course with 2 sub-components 14

4.7.3 Invalid Data Entries 14

[4.8](#_deah5eb7h2h0) Enter exam mark 15

[4](#_icb1noepltht).8.1 Enter valid exam mark for the valid course 15

[4](#_lm0zoju7n6bs).8.2 Invalid Data Entries 15

[4.9](#_deah5eb7h2h0) Print course statistics 15

[4](#_icb1noepltht).9.1 Enter valid course code 15

[4](#_lm0zoju7n6bs).9.2 Invalid Data Entries 15

[4.10](#_deah5eb7h2h0) Print student transcript 15

[4](#_icb1noepltht).10.1 Enter valid student ID 15

[4](#_lm0zoju7n6bs).10.2 Invalid Data Entries 15

# 1. Design Considerations

## 1.1 Approach Taken

**IOInterface.java**: Boundary class for menu, outputs and user input.

**CourseController.java**: Controller class for adding course and editing content inside a course

**DatabaseController.java**: Controller class for loading and saving data into a file

**EnrollmentController.java**: Controller class for enrolling student to a course and printing transcript

**StudentController.java**: Controller class for adding and editing student particulars

**Assessment.java**: Entity class that extends Field class which is used for modifying assessment in course

**Component.java**: Entity class that is used for getting and setting value for a component

**Course.java**: Entity class that is used for creation of course object

**Enrollment.java**: Entity class that is used for creating enrollment object that saves student with the course registered in it

**Field.java**: Entity class for implementing assessment and grade classes

**Grade.java**: Entity class that extends Field class which is used for editing the grade of a student

**Lecture.java**: Entity class that creates lecture in a course

**Student.java**: Entity class that is used for creation of student object

**TimeSlot.java**: Entity class for creating a timeslot for lectures, tutorials and labs.

**Tutorial.java**: Entity class for tutorial and lab creation, assume one tutorial only have one lab.

## 1.2 Principles Used

Our design implements three from five of the SOLID principles:

1. Single Responsibility Principle(SRP)

The principle that one single class only serves a single purpose. Each class in our model only has one single responsibility. For example, Student entity class is used to store and retrieve student’s particulars, Enrollment entity class is used to store and retrieve enrollment information of all students, CourseController controller class is used to create/edit/remove courses and EnrollmentController controller class is used to create/edit/remove enrollment info.

1. Liskov Substitution Principle (LSP)

The principle that a base class object can be substituted with any of derived class object without altering any of the property. For 2 derive classes “Assessment” and “Grade” those have similar properties, we created an abstract base class “Field” which has every property of both “Assessment” and “Grade”. Since “Assessment” and “Grade” have the same type, they can be treated as the same class in higher layer classes.

1. Interface segregation principle (ISP)

The principle states that no client should be forced to depend on methods it does not use, that is what we have done on “Field”, “Assessment” and “Grade”. In order to use only “Assessment”, client just need to import “Field” and “Assessment” without the need of “Grade”.

1. Composition over Inheritance

The principle that classes should achieve polymorphic behavior by their composition. In our application, there are many “types” of Courses, Courses without Tutorial, Courses with Tutorials. We made Tutorials to be aggregations of a Lecture and Lectures to be aggregations of a Course. Since Tutorial class has “getStudentList” that returns a list of Students in that Tutorial, Lecture and Course also have “getStudentList” that returns a list of Student in that Lecture/Course.

## 1.3 Assumptions Made

There are a few assumptions made by us in our code implementation:

1. Only courses that are finished constructed are able to be registered by students.
2. Each course is able to have many lectures and lecture has many tutorials. Each tutorial section has only one corresponding lab section to it (only one tutorial and lab section shared the same index).
3. There are no course with the courseID equal to -1 and student with the studentID that equals to -1. As in our code, -1 will terminate the searching of course and student inside certain functions.

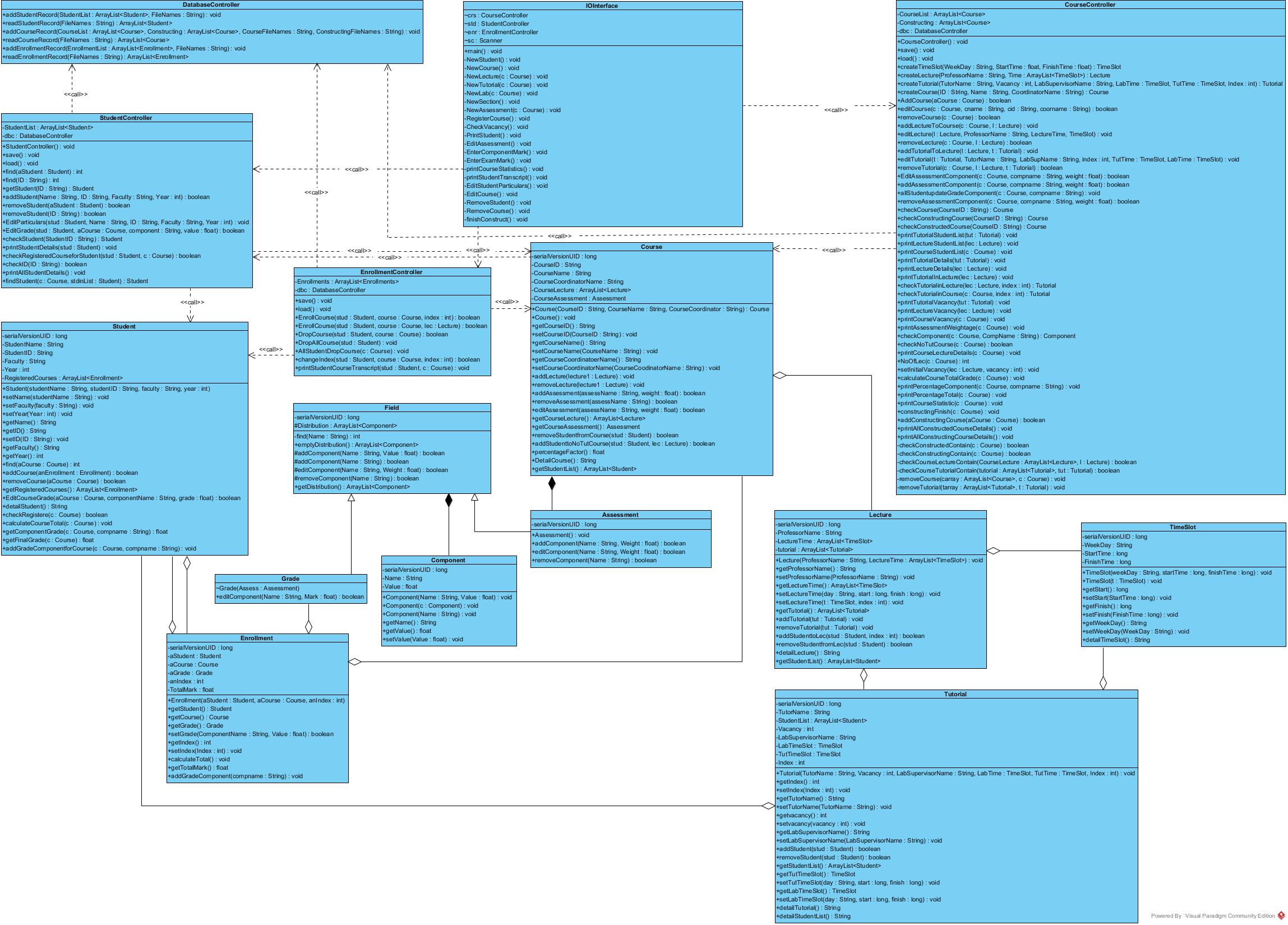
## 1.4 Current Found Errors:

Everytime we restart the program and load the previous data inserted, we can still register course for the same student that already registered certain courses, and it will result in duplicate records.

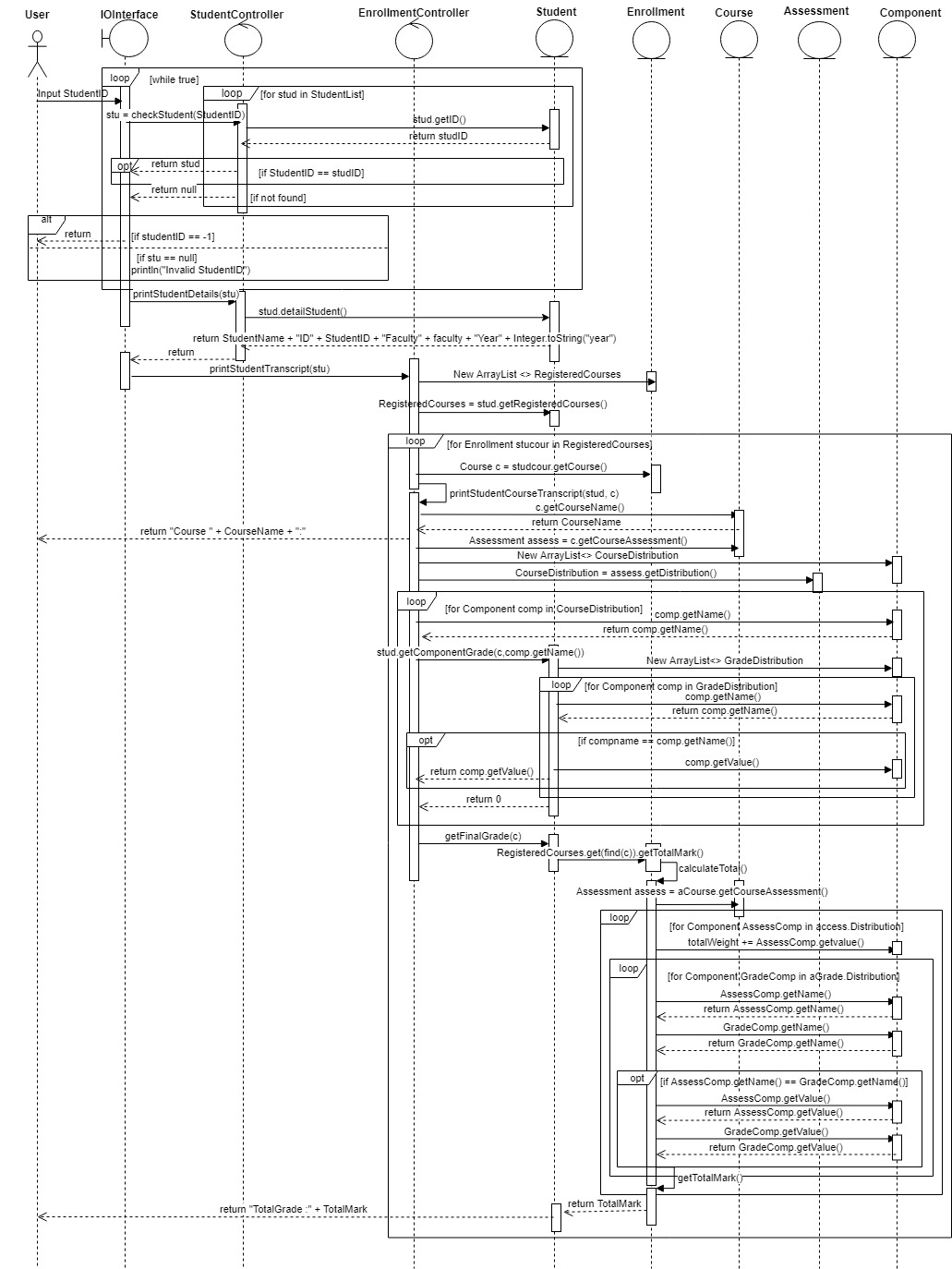
Everytime we restart the program and load the previous data inserted, if we want to remove a student, after removing, the student will still be in the registered courses.

*“There is a list of Enrollment inside Student entity, whenever the App quits, it will save Student objects leads to save the Enrollment then Course that student has taken. It is the same in Course, whenever a Course object is saved, the StudentList also be saved. It has no doubt if in restoring data, all addresses are restored correctly. Unfortunately, it does not.”*

# 2. UML Class Diagram



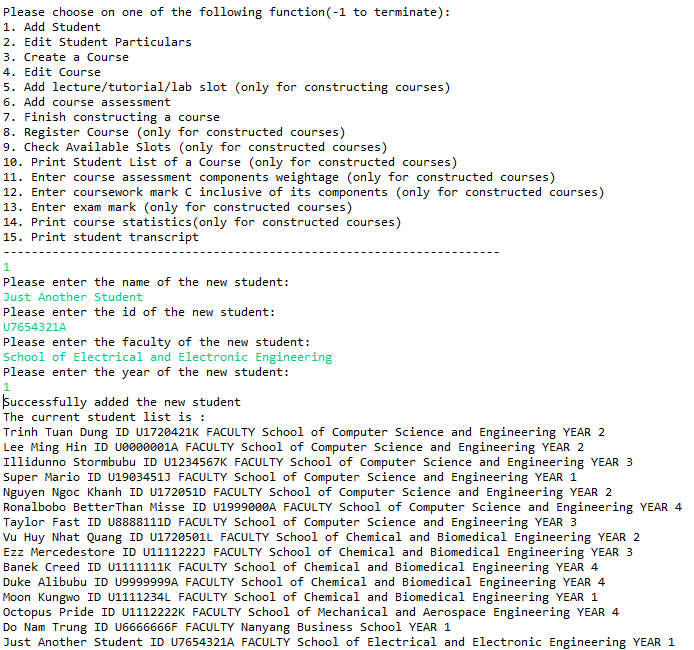
# 3. UML Sequence Diagram



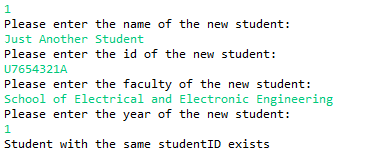
# 4. Test Case and results

## 4.1 Add a student

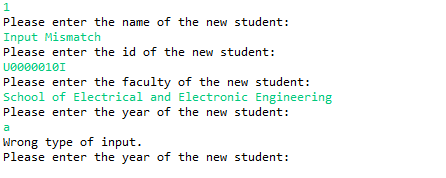
### 4.1.1 Add a new student



### 4.1.2 Add an existing student



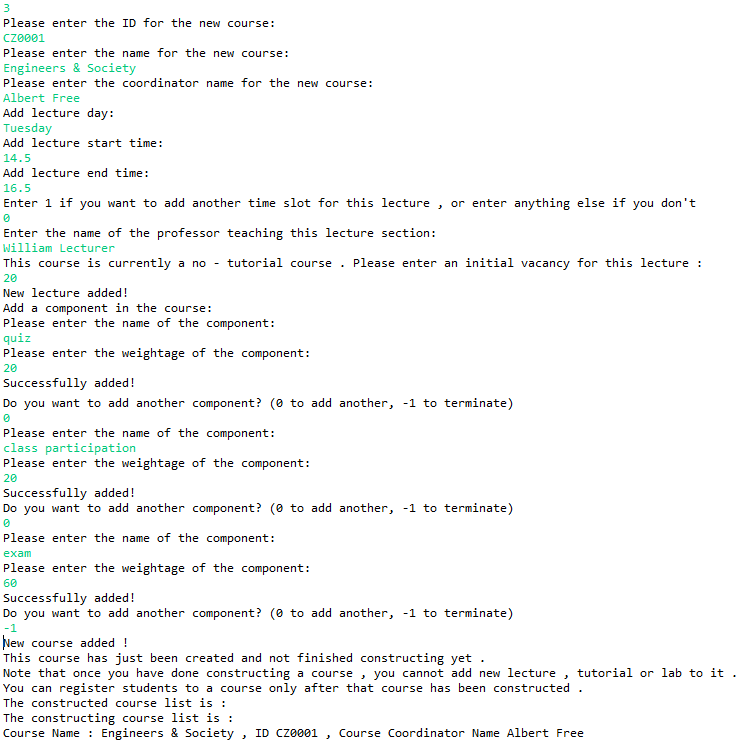
### 4.1.3 Invalid data entries



## 4.2 Add a course

### 4.2.1 Add a new course

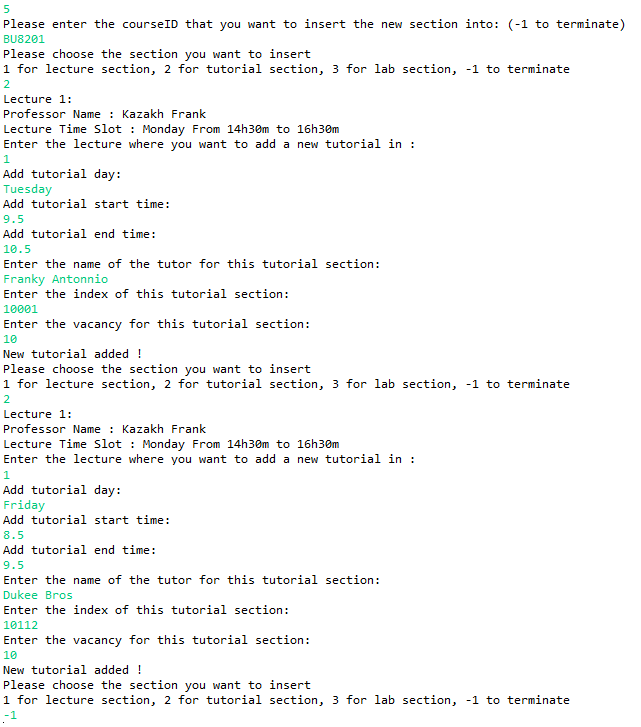
#### 4.2.1.1 A course with 1 lecture group only



#### 4.2.1.2 A course with 1 lecture group and 2 tutorial groups

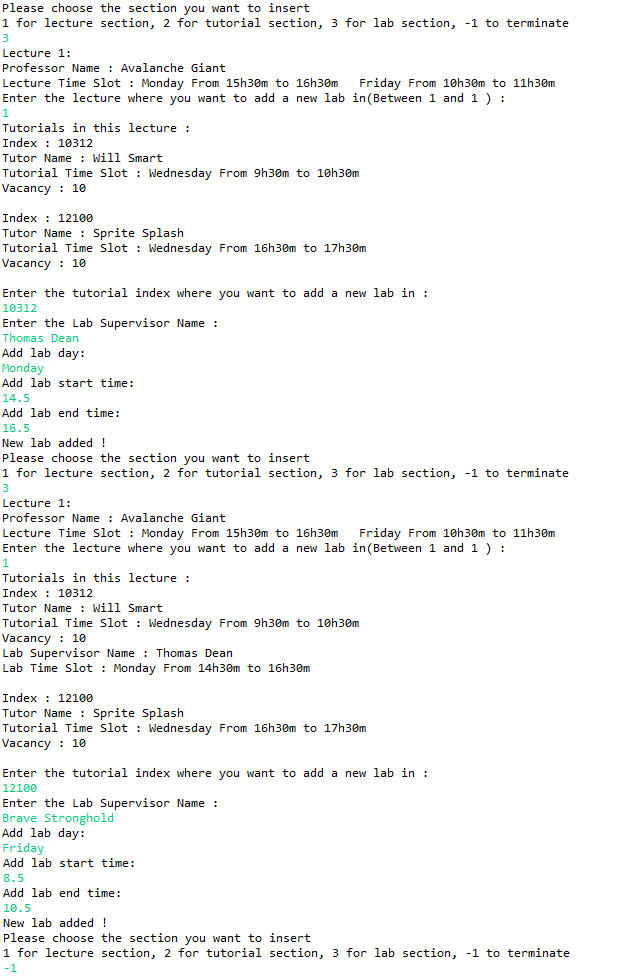
The initial creation of the course is similar to part 4.2.1.1.

Addition of 2 tutorial groups :

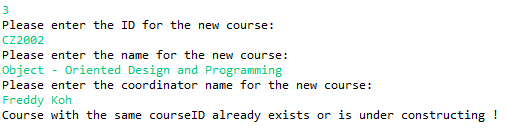


#### 4.2.1.3 A course with 1 lecture group , 2 tutorial groups and 2 lab groups

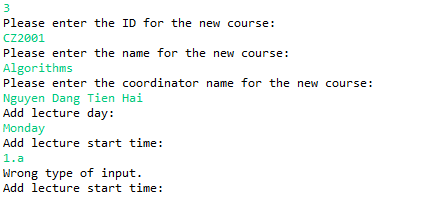
The initial creation and addition of 2 tutorial groups is similar to part 4.2.1.2.



### 4.2.2 Add an existing course



### 4.2.3 Invalid data entries



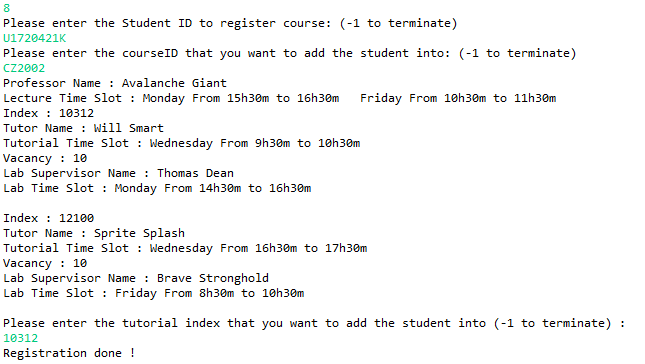
## 4.3 Register for a course

### 4.3.1 Add a student to a course with available vacancies in Tut / Lab

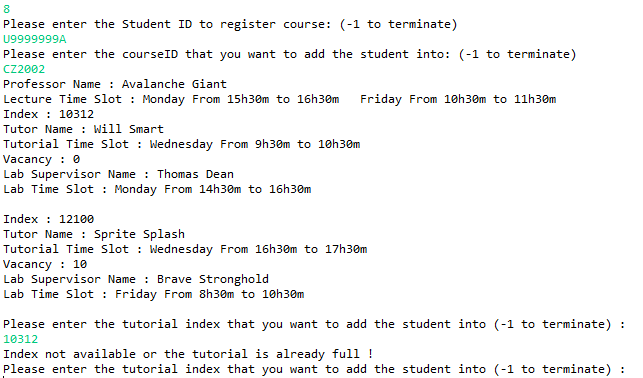
In order to register student for a course , that course needs to be constructed first :



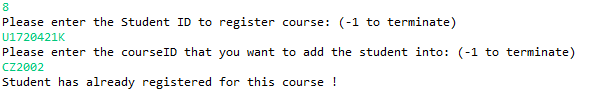
After a course has been constructed , we can start to register students for that course :



### 4.3.2 Add a student to a course with 0 vacancies in Tut / Lab

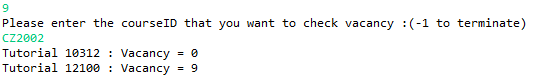


### 4.3.3 Register the same course again

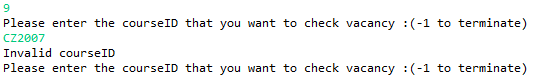


## 4.4 Check available slot in a class

### 4.4.1 Check for Tutorial / Lab class

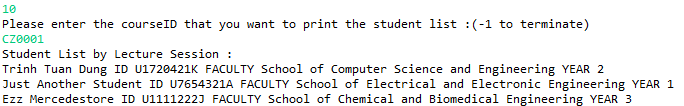


### 4.4.2 Invalid data entries

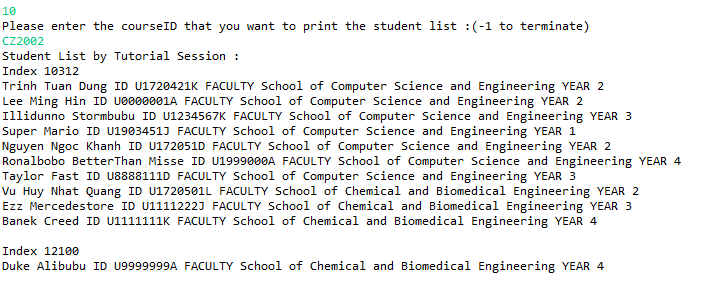


## 4.5 Print student list by lecture, tutorial or laboratory session for a course

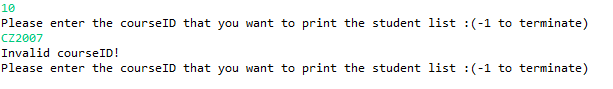
### 4.5.1 Print list by Lecture



### 4.5.2 Print list by Tutorial / Lab

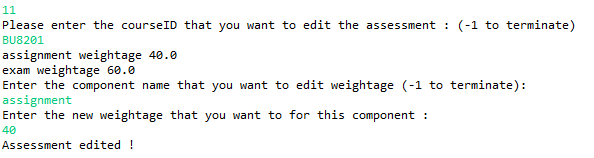


### 4.5.3 Invalid data entries

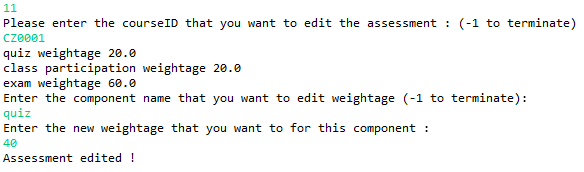


## 4.6 Enter course assessment components weightage

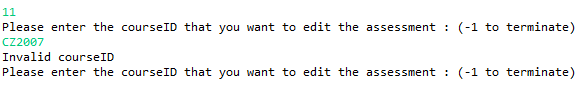
### 4.6.1 Enter course assessment with only exam + 1 coursework component



### 4.6.2 Enter course assessment with exam + 2 coursework component

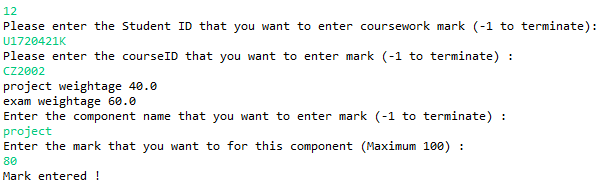


### 4.6.3 Invalid Data Entries

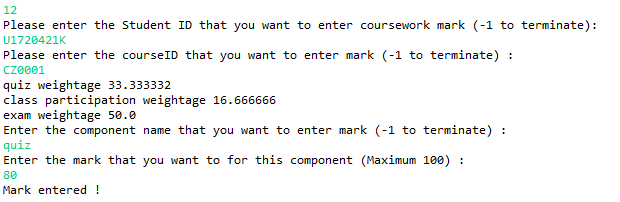


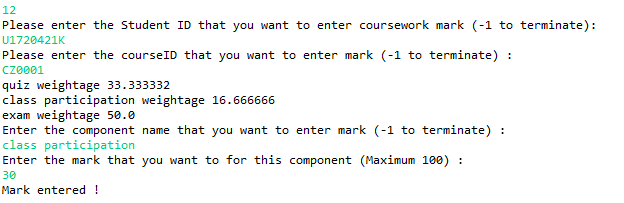
## 4.7 Enter coursework mark – inclusive of its components

### 4.7.1 Enter valid coursework mark for course with only 1 main component

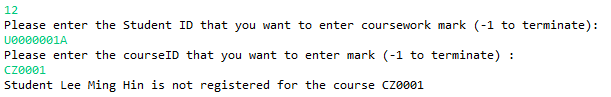


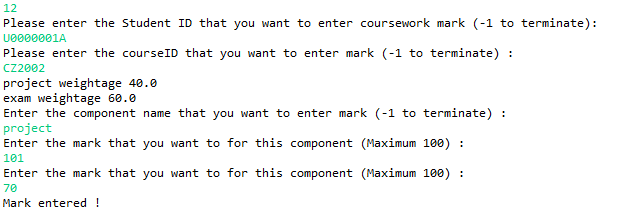
### 4.7.2 Enter valid coursework marks for course with 2 sub-components





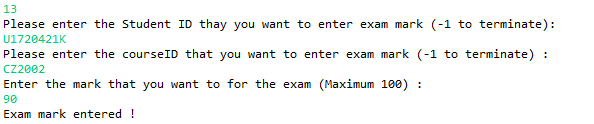
### 4.7.3 Invalid Data Entries



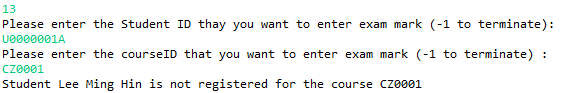


## 4.8 Enter exam mark

### 4.8.1 Enter valid exam mark for the valid course

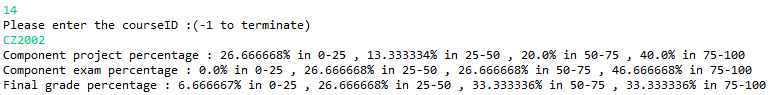


### 4.8.2 Invalid Data Entries

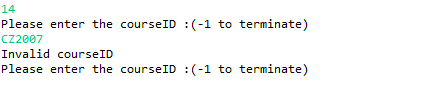


## 4.9 Print course statistics

### 4.9.1 Enter valid course code

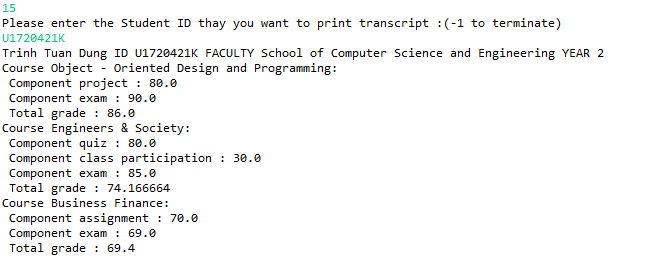


### 4.9.2 Invalid data entries



## 4.10 Print student transcript

### 4.10.1 Enter valid student ID



### 4.10.2 Invalid Data Entries

### 