**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 honoured the principles of academic integrity and have upheld 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Course  (CE2002 or CZ2002) | Lab Group | | Signature /Date |
| Neera | CZ2002 | SSP1 | Neera/14 Nov 2012 | |
| Chaudhuri Kaustav | CZ2002 | SSP1 | Kaustav/14 Nov 2012 | |
| Vij Akriti | CZ2002 | SSP1 | Akriti/14Nov 2012 | |

**Design considerations:**

**Approach**

The program features an object oriented and modular approach. The various entities in context are represented in the form of classes, and their related methods are encapsulated along with the data fields.

The program is free from Symptoms of Rotting Design. The main aim was to minimise impact of change.

We have designed the application keeping Reuse, Extensibility and Maintainability in mind.

These different classes exhibit a variety of relationships between them.

The following features of Object Oriented Program have been utilized:

1. Inheritance :

The class Sub extends SubComponent.

The idea behind this is that SubComponent is used to store the information about the subcomponents of a course- such as exam, lab assessment, class participation, etc. However, for specific students, we will need to store their marks as well.   
So we derive Sub class from SubComponent- this contains the marks stored by the student for each component in addition to the component details.

1. Method Overriding :

There is over-riding of “equals” method of java.lang.Object

1. Auto-unboxing

Example : int =Integer.ParseInt(input)

Here the wrapper “Integer Object” implicitly gets converted to the primitive type.

The sophisticated collections and generics provided in java as a part of java.util provide collections such as array list that are more convenient to implement in a dynamic scenario.

**Principles:**

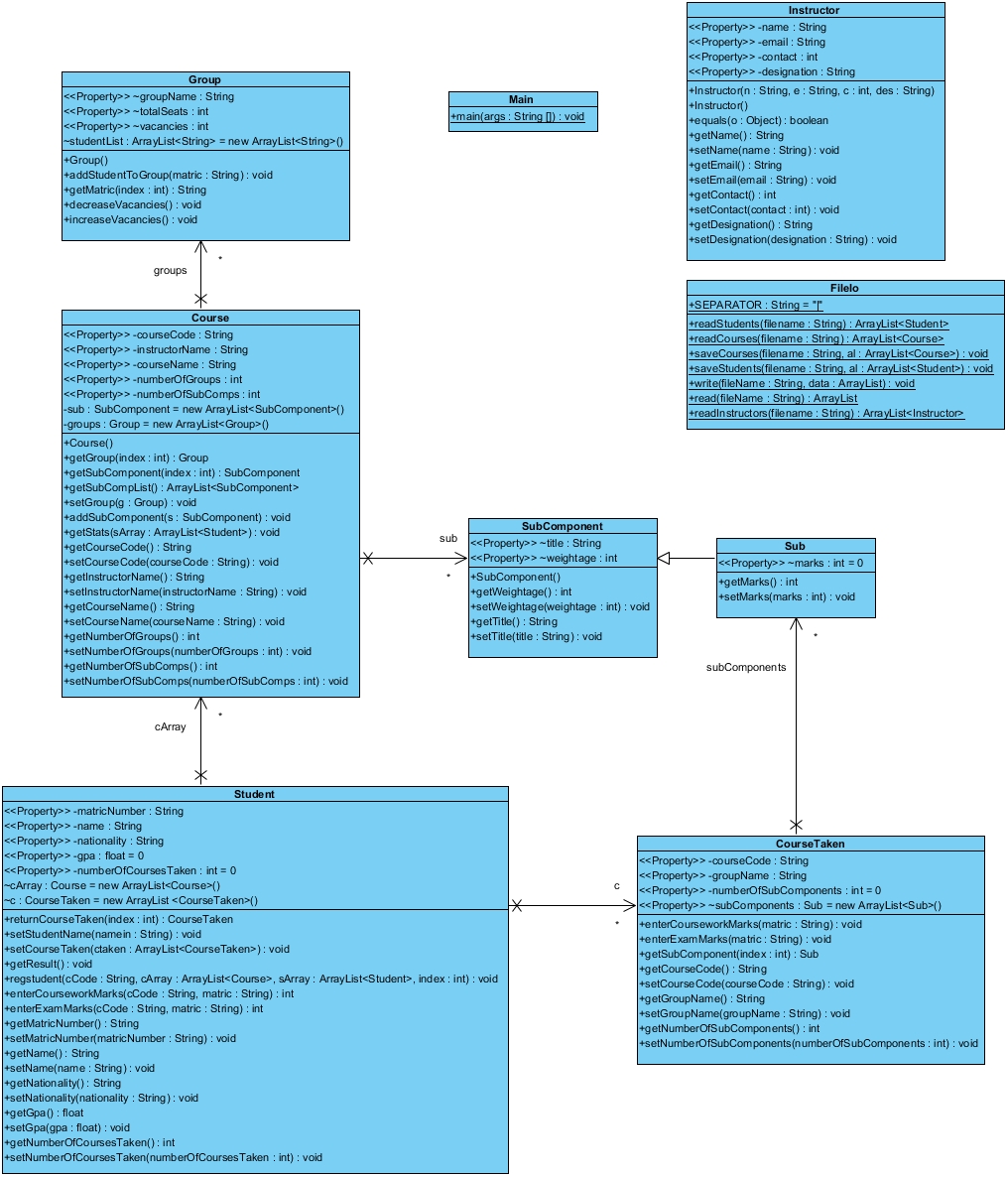
1. Data hiding. Data should be secure- hence private accessibility with public getter-setter functions.
2. Encapsulation- internal representation of an object is generally hidden from view outside of the object's definition.
3. Abstraction: The data and programs are defined with a representation similar in form to its meaning), while hiding away the implementation details. The internal compositions of a class, or the internal working of a method, are hidden from the outer world. E.g., Result class/whatever method will simply be invoked.

**Assumptions made:**

1. Every student can register a maximum of 3 courses
2. Lab and tutorial groups are the same- i.e. if a student is in Group SS1, then it is group SS1 for lab, as well as group SS1 for tutorial.
3. All marks entries are based on 100 marks. Application will scale the marks to its component weightage percentage
4. A course, once registered, cannot be dropped
5. The courses can vary in terms of coursework and assignment sub components, and their respective weightage, and also in terms of lab and tutorial structure. However, all the courses are of 3 Academic credits.
6. The length of matric number should be 10.
7. The length of course code should be 7.

**UML – CLASS DIAGRAM**

Class diagram in UML is a type of static structure diagram that tells us how to define a class itself.

In the class diagram different classes are shown with their attributes, operations and the relationship among the classes. In our Student Course Registration and mark entry application we have created eight classes. 

* ASSOCIATION-

1-Between Course and SubComponent- (Course uses instance of SubComponent)

2-Course and Group - (Course uses instance of Group)

3-Student and Courses-(Student uses instance of class courses)

4-Student and CourseTaken - (Student uses instance of Class CourseTaken)

5-Course and SubComponent - (Course uses instance of Class SubComponent)

* INHERITANCE

1-Sub is inherited from SubComponent [Explained in design and approaches]

* DEPENDENCY

1-Class CourseTaken is dependant on Group

Any changes made in Group class i.e. if the Number of vacancies changes (which

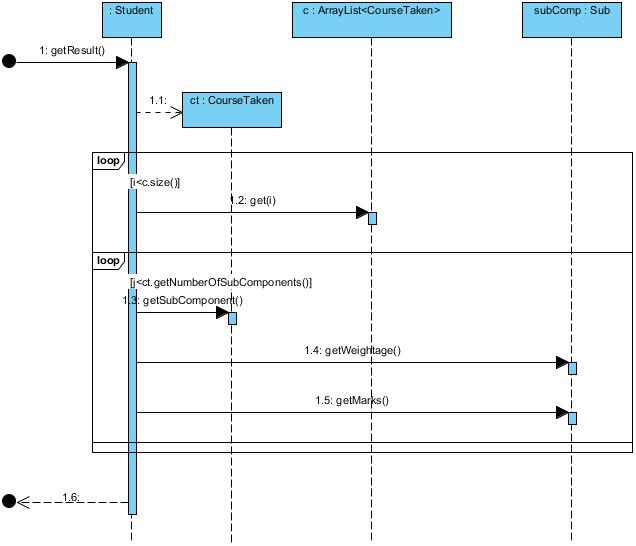
depends on the total seats) then the group name assigned or registered to the

student will also change.

**Sequence diagram**

One disadvantage of class diagram is that it doesn’t tell the user how to code the method itself

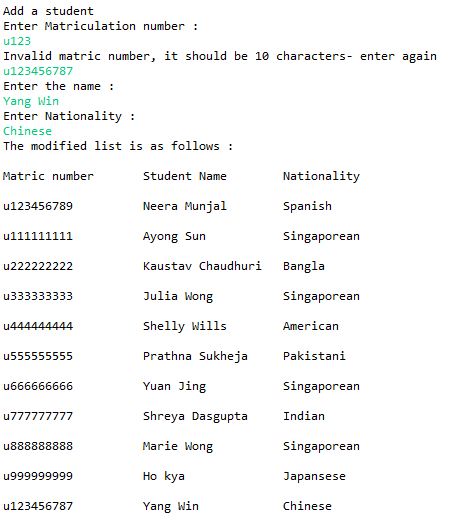
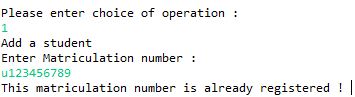
Sequence diagram help us to know the flow of methods or flow of functionality (for example printing class statistics) and how to implement the methods.



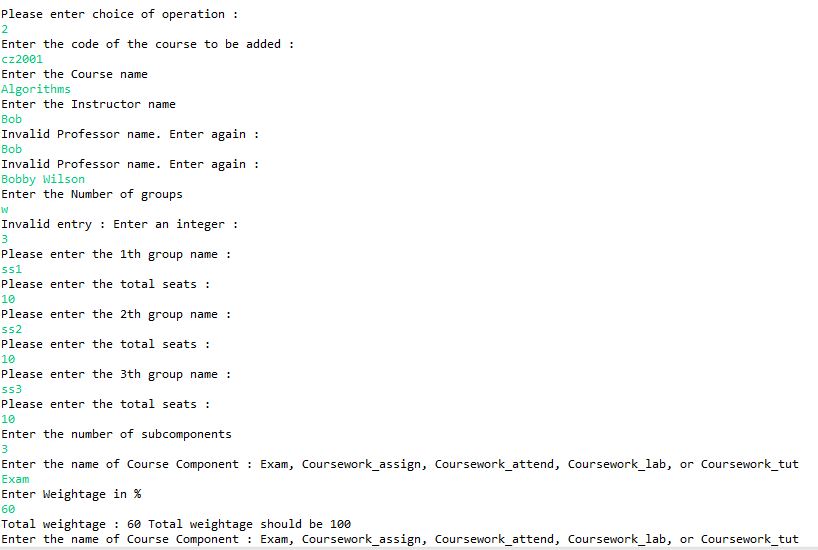
The sequence diagram above represents the logic flow for the getResult( ) function which calculates the GPA and also prints the transcript with detailed information about the student’s result. Since it is a method of Student class itself, there is direct authorized access to the student marks.

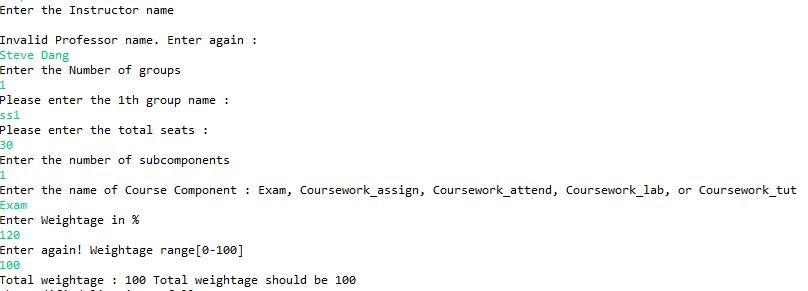
The function has a loop iterating over the courses taken by the student. For each “CourseTaken” of the ArrayList of CourseTaken type, control is passed to a loop iteration going through the “SubComponents”. Within each SubComponent, the individual weightages and components are multiplied and added to relevant fields, to calculate average exam score, average coursework score, total average score, number of students above average and so on.

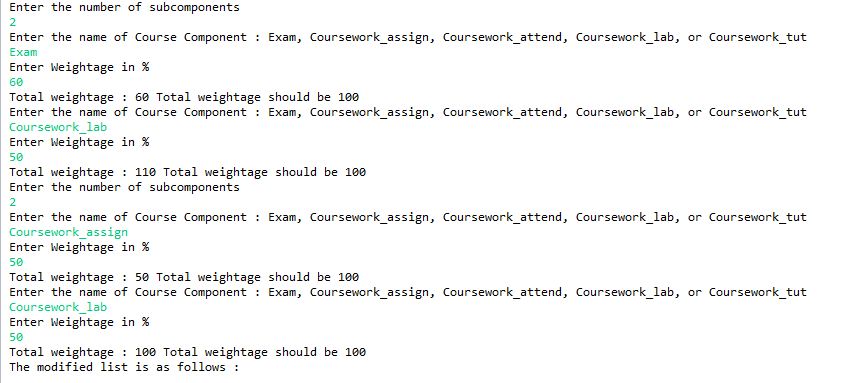
TEST CASES

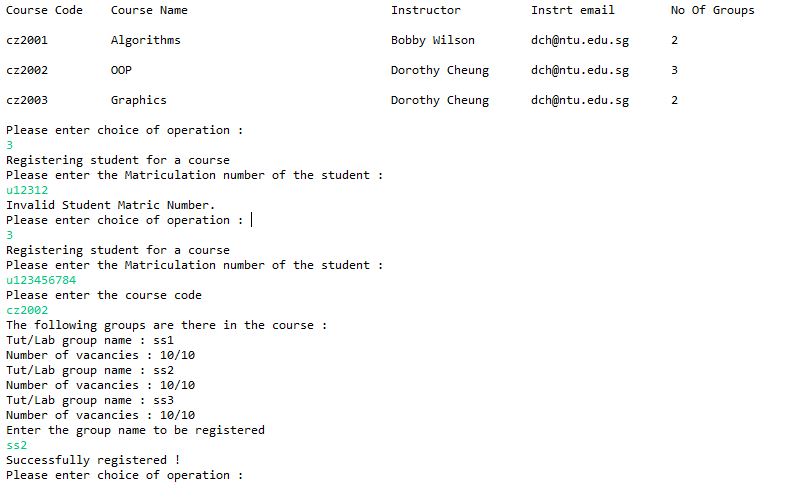
Most test cases have been covered in the video. The following screenshots represent the functioning of the program at various points in the program. Cases of error handling have also been included. Error handling is essential to make the program robust and to avoid inaccurate/inconsistent data.

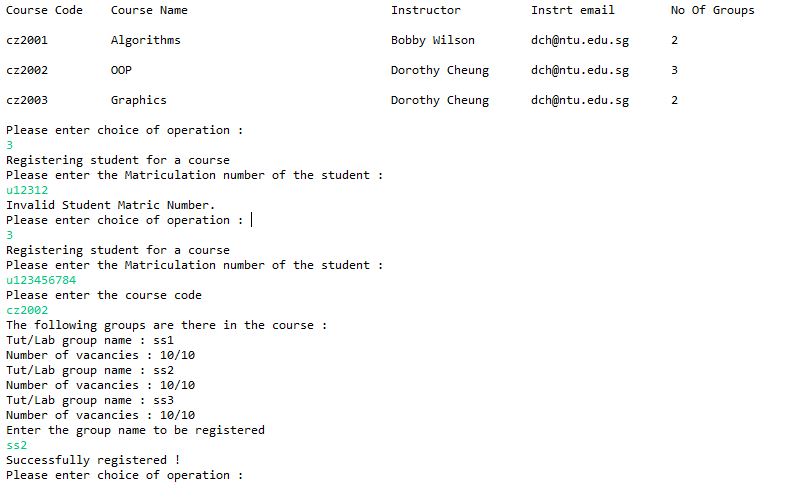
Error Handling while Adding Student

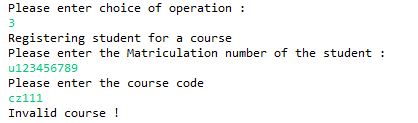
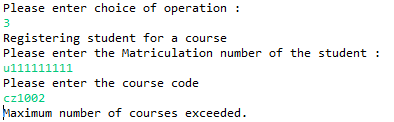
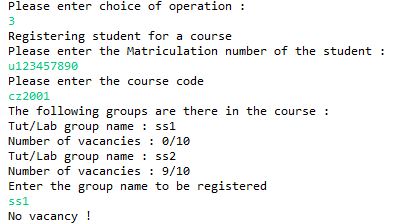
🡨-Standard Student Display List

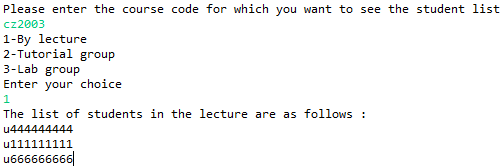


Various error checks need to be added while adding a course. Checking for valid course code, valid professor name(-using data taken from file), valid subcomponent name, (total weightage==100)- all implemented using if/else or while. **If total weightage is not 100, user gets a chance to reset subcomponents**. Check for number of groups id using **InputMismatchException.**

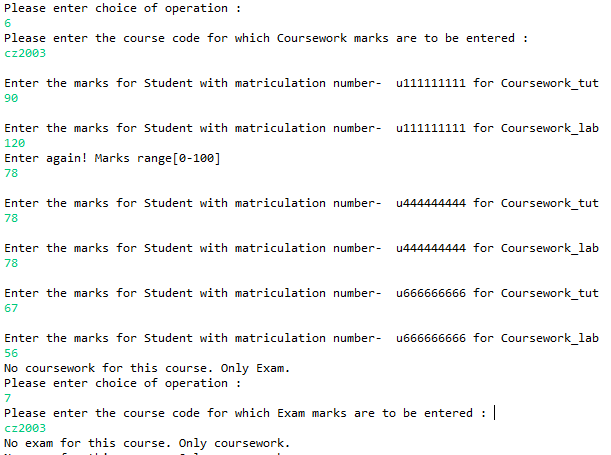


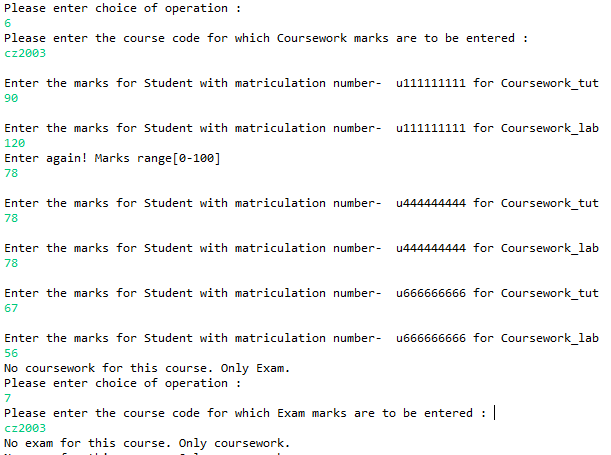
Standard List display for courses- including Instructor information

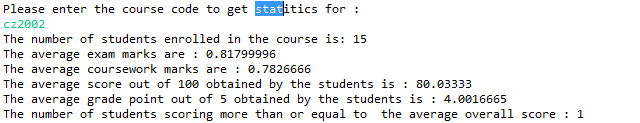
Test cases for Course Registration of Student



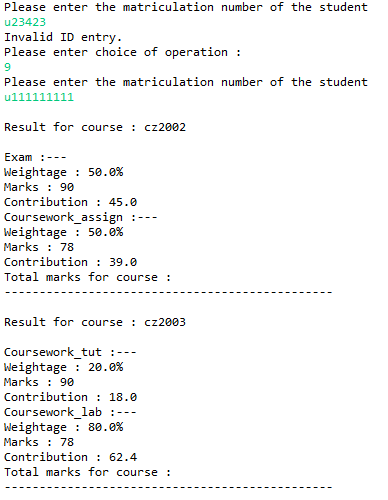
Obtaining the list of student matric numbers by lecture, lab and tutorial groups

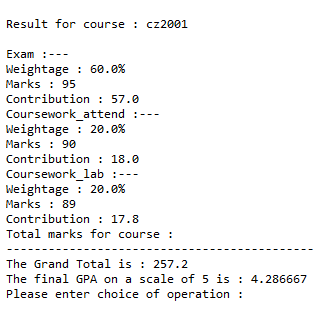




Entering Exam and Coursework marks for ALL the students enrolled in the course with checking for course code and marks.

Print statistics for a course



Transcript with error checking in beginning