
# ASSIGNMENT

## CE/CZ2002: Object-Oriented Design & Programming

*Building an OO Application*

**2018/2019 SEMESTER 1**

**SCHOOL OF COMPUTER SCIENCE & ENGINEERING**
**NANYANG TECHNOLOGICAL UNIVERSITY**

## 1. OBJECTIVE

The main objective of this assignment is to:

- apply the Object-Oriented (OO) concepts you have learnt in the course,
- model, design and develop a quality OO application fulfilling the project requirement,
- gain familiarity with using Java as an object oriented programming language,
- work collaboratively as a group to achieve a common goal.

## 2. LABORATORY

Software Lab II (Location: N4-B1c-06).

## 3. EQUIPMENT

Hardware: PC (or Laptop)
Software: Your prefered Java IDE or simply notepad and Java Development ToolKits (JDK)

## 4. THE ASSIGNMENT

The assignment for your group will be to design and develop a **Console-based application (non-Graphical UI)**:

**STUDENT COURSE REGISTRATION AND MARK ENTRY Application (SCRAME)**

The following are information about the application:

a) It is a university application meant for undergraduates students.
b) The application allows the creation of courses and adding of student records. Faculty details are extracted from external source.
c) At the start of each semester, students will be required to register for their courses. Each course (subject) will have information of the professor (or faculty member) who coordinates the course (course coordinator) and available vacancy.
d) Courses may have lectures only, lectures and tutorial only or lectures, tutorial and laboratory sessions.
**e)** For each course, its exam and coursework weightage can be entered, e.g., exam as 60%, coursework as 40%. ****Furthermore, the coursework component can be further breakdown into sub-components, e.g. assignment as 70% and class participation as 30%.**
f) All marks (exam and coursework) relating to the course can be entered using the application.
g) At the end of the semester, system allows the viewing of the course performance by presenting the overall grades percentage, its components grade (eg exam and coursework) percentage.
h) The application also allows the viewing of students' transcript for the current semester, showing the overall and component grades of all courses taken in the semester.

***Functional Requirements (to be demonstrated):***

1. Add a student
2. Add a course
3. Register student for a course (this include registering for Tutorial/Lab classes)
4. Check available slot in a class (vacancy in a class)
5. Print student list by lecture, tutorial or laboratory session for a course.
6. Enter course assessment components weightage
7. Enter coursework mark – inclusive of its components.
8. Enter exam mark

9. Print course statistics
10. Print student transcript.

*(Note : you may re-order or re-phrase the above functionalities when displaying your application menu)*

The application is to be developed as a **Console-based application (non-Graphical UI).** Data should be stored in flat file format, either in text or binary. **Refer to your eLearning Topics in NTULearn on FileI/O. Samples are provided for the reading/writing of text or binary (Serializable) file. [ Learn from the fundamentals].**
***No database application (eg MySQL, MS Access, etc) is to be used. No JSON or XML is to be used.***

You can base on the knowledge and experience of the courses you have taken to consider what are the attributes of the classes you have derived.

You will create your own test cases and data to test your application thoroughly. However, you should also create test cases to test for cases* of over-registered course and class (tutorial or laboratory session).

**Assumptions :**
(1) Need not consider the time slot for courses.
(2) External source implies preexisting records and can be loaded from the file/s.
(3) Course and Students are to be stored in files. The format of storage is up to individual group's considerations.
(4) The application is only accessible by the administrator of the application.

## 5. THE REPORT

Your report will include the following :

a) A detailed UML **Class** Diagram for the application (exported as an image)
- show clearly the class relationship, notation
- notes to explain, if necessary

b) A detailed UML **Sequence** Diagram (exported as an image)
- Show only the flow of the **"Print Student transcript"** function.
- *The diagram should show clearly all participating objects involved with sufficient detailed flow and relevant interaction fragments.*

c) Screen captures of the testings done (those essential test cases not covered in your demo).

b) A **write-up** on your **design considerations**, principles and the use of OO concepts.

d) A duly signed **Declaration of Original Work** form (Appendix B).

e) [**Optional**] Member's work contribution and distribution breakdown.
*If your group feels that marks should be given based on contribution, your group can fill up the WBS.xls(in the same folder as assignment doc) and include it in this report. All members MUST consent to the WBS contents.* ***You must also email the WBS.xls to the course-coordinator with ALL members in the loop.***

## 6. DEMONSTRATION

Your group is to produce a **video and audio recording** to demonstrate the working of the application – presenting ALL the required functionalities of the application and the suggested test cases in Appendix A. It is advised that you planned your demonstration in a story-

boarding flow to facilitate understanding of your application. *Include a group photo of your group members and introduce your members and group number at the start of video.*

In the production, you may include :

a) Explaining essential and relevant information about the application
b) Run-through and elaborate on essential part/s of your implementation/coding

- ***The video duration must not exceed 15 minutes in total.***
- ***The font size used must be large enough to be readable and viewable.***
- ***The video quality must be clear.***
- ***The demo of the application is to done in real-time and NOT pre-run display.***

## 7. THE DELIVERABLE

Your group submission should be in the form of **CD or DVD** and should include the following :

a. The report.
b. Video and audio recording of the demonstration.
c. All implementation codes and java documentation (javadoc).

## 8. ASSESSMENT WEIGHTAGE

UML Class Diagram [**25 Marks**]

- Show mainly the Entity classes, the essential Control and Boundary classes, and enumeration type (if there is).
- Clarity, Correctness and Completeness of details and relationship.

UML Sequence Diagram [**20 Marks**]

- Show only the sequence Diagram mentioned in 5(b)
- Details Clarity, Correctness and Completeness of flow and object interactions.

Design Consideration [**15 Marks**]

- Usage of OO concepts and principle - correctness and appropriateness

Implementation Code [**20 Marks**]

- Diagram to Code correctness, readability, Javanaming convention, exception handling, completeness of Java Doc and overall quality.
- A Java API HTML documentation of **ALL** your defined classes using Javadoc must be submitted. The use of javadoc feature is documented in Appendix D.

Demonstration [**20 Marks**]

- Coverage of application essentials and functionalities, user friendliness, demo flow, innovation.
- ***Based on stated video duration above.***

## 9. DEADLINE

This is a **group assignment**, and one CD/DVD is to be submitted from each group. Report format guidelines is provided in the Appendix below.

1. The CD/DVD needs to be submitted to the **Software Projects Lab @N4-B1b-11** by **16th November 2018, 4.30pm**. Indicate your group members and lab class on the CD/DVD. *Drop your CD/DVD into the pigeon hole indicating **your lab class** and the **course code CE/CZ2002**.*

2. Soft copy of the **report to be uploaded** to **CE/CZ2002 Main Course site**. The link is provided on the left panel "Assignment Report Submission".
   - (Optional) Upload a copy of the video recording as backup in case of issue with the CD/DVD version.
   - Name your file as <labGroup>-<group#>-<video/report>, eg, FEP1-Grp1-report.
   - One copy per group and any group member can submit.

***Important:***
Note that **THREE (3)** marks will be deducted for the delay submission of each **calendar day**.
**The lab is closed on weekends and no submission can be made but late submission penalty still applies.**
**Lateness is based on the date the CD/DVD is submitted, NOT the report upload.**

10. **REFERENCES & TOOLS**

- UML Diagrams tool - Visual Paradigm http://www.visual-paradigm.com/
- http://www.visual-paradigm.com/support/documents/vpuserguide/94/2576/7190_drawingclass.html
- NTULearn Cx2002 main course site content
- NTULearn Cx2002 course site content on "File Input/Output"
- Object Serialization tutorial http://www.javabeginner.com/uncategorized/java-serialization
- Windows Media Encoder ( a suggestion) http://www.microsoft.com/expression/products/EncoderPro_Overview.aspx

**APPENDIX A:**

**Suggested Test Cases for STUDENT COURSE REGISTRATION AND MARK ENTRY Application**

The list of test cases are suggestions for your testing. Depending on your design and user-friendliness of your data entries process, some test cases may be not relevant.

Suggestions :

i. To create about 15 students, 3 courses, and 4 professors (pre-populated prior to running test cases).
**ii. To create courses with only 1 Lecture group, 1 Lecture group + 2 Tutorial group and 1 Lecture group + 2 Tutorial groups + 2 Lab groups.**
iii. To have each tutorial/lab groups to have 10 vacancies each (slightly more than original lecture size) – for testing of zero vacancy in a group and registration for another group.
iv. Need not have follow functionalities sequentially – follow a scenario.

(TBA : to be added by your group)

1. Add a student

| | **Test Case** | **Expected Outcome** |
|---|---|---|
| a | Add a new student | the listing of **all** students should be displayed after the addition<br>[ group to decide the amount of details to display ] |
| b | Add an existing student | Appropriate error message display |
| c | Invalid data entries | Appropriate error message display |
| d | TBA | |

2. Add a course

| | **Test Case** | **Expected Outcome** |
|---|---|---|
| a | Add a new course<br>(with combination of (ii) from above) | listing of **all** courses should be displayed after the addition TOGETHER with Professor in charge (coordinator).<br>[ group to decide the amount of details to display ] |
| b | Add an existing course | Appropriate error message display |
| c | Invalid data entries | Appropriate error message display |
| d | TBA | |

3. Register student for a course (this include registering for Tutorial/Lab classes)

| | **Test Case** | **Expected Outcome** |
|---|---|---|
| a | Add a student to a course with available vacancies in Tut / Lab. | Appropriate message display |
| b | (i) Add a student to a course with 0 vacancies in Tut / Lab.<br>(ii) Add a student to a course with available vacancies in Tut / Lab. | Appropriate message display |
| c | Register the same course again | Appropriate error message display |
| d | Invalid data entries<br>(eg wrong student ID / course code, etc) | Appropriate error message display |
| e | TBA | |

4. Check available slot in a class (vacancy in a class)

| | **Test Case** | **Expected Outcome** |
|---|---|---|
| a | Check for (i) Tutorial class | Appropriate message display, eg 3/10 |

| | | |
|---|---|---|
| | (ii) Lab class | [vacancy/total size] |
| b | Invalid data entries<br>(eg course code, class code etc) | Appropriate error message display |
| c | TBA | |

5. Print student list by lecture, tutorial or laboratory session for a course.

| | **Test Case** | **Expected Outcome** |
|---|---|---|
| a | Print list by<br>(i) Lecture<br>(ii) Tutorial group/s<br>(iii) Lab group/s | Appropriate display.<br>[ group to decide the amount of details to display ] |
| b | Invalid data entries<br>(eg course code, class code etc) | Appropriate error message display |
| c | TBA | |

6. Enter course assessment components weightage

| | **Test Case** | **Expected Outcome** |
|---|---|---|
| a | Enter course assessment with only exam + 1 main coursework component without sub-components<br>(eg, use the case stated in 4(e) in main section) | Appropriate display.<br>[ group to decide the amount of details to display ] |
| b | Enter course assessment with exam + a coursework with 2 sub-components<br>(eg, use the case stated in 4(e) in main section) | Appropriate display.<br>[ group to decide the amount of details to display ] |
| c | Invalid data entries<br>(eg course code, weightage percentage does not tally etc) | Appropriate error message display |
| d | TBA | |

7. Enter coursework mark – inclusive of its components.
***(Note : All mark entries should be based on 100 marks. Your application will eventually scale the marks to its component weightage percentage)***

| | **Test Case** | **Expected Outcome** |
|---|---|---|
| a | Enter valid coursework mark for course with only 1 main component. | Appropriate message display. |
| b | Enter valid coursework marks for course with 2 sub-components. | Appropriate message display. |
| c | Invalid data entries<br>(eg course code, student ID, mark range, etc ) | Appropriate error message display |
| d | TBA | |

8. Enter exam mark
***(Note : All mark entries should be based on 100 marks. Your application will eventually scale the marks to its component weightage percentage)***

| | **Test Case** | **Expected Outcome** |
|---|---|---|
| a | Enter valid exam mark for the valid course. | Appropriate message display. |
| b | Invalid data entries<br>(eg course code, student ID, mark range, etc ) | Appropriate error message display |
| c | TBA | |

9. Print course statistics
***(Note : You should have at least 15 students' results for 1 course pre-populated)***

| | **Test Case** | **Expected Outcome** |
|---|---|---|
| a | Enter valid course code | Show grade percentage for overall (exam + coursework), exam only and coursework only. [ group to decide other details to display ] |
| b | Invalid data entries (eg course code, mark range, etc ) | Appropriate error message display |
| c | TBA | |

10. Print student transcript.
***(Note : For this function, besides the individual overall course mark and grade, it will also print the student's individual component marks – exam, coursework, subcomponents. The configured weightages should be displayed as well.)***

| | **Test Case** | **Expected Outcome** |
|---|---|---|
| a | Enter valid student ID | Show results (as stated above) for all courses registered by the student. *The overall mark is correctly computed.* |
| b | Invalid data entries (eg student ID) | Appropriate error message display |
| c | TBA | |

**APPENDIX B:**

**Attached a scanned copy with the report with the filled details and signatures.**

## 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 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 (CE2002 or CZ2002) | Lab Group | Signature /Date |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

Important notes:

1. Name must **EXACTLY MATCH** the one printed on your Matriculation Card.

**APPENDIX C:**

## Report requirement:

### 1. Format:

For the main content, please use Times New Roman 12 pt font size and 1.5 line spacing. You may choose to use other fonts (e.g, Courier New) for code segments. Please use the following report structure:

- Cover page: Declaration of original work (Page 10 of the assignment)
- Design Considerations .
    - Approach taken, Principles used, Assumptions made, etc
    - ***Optional*** : You can show the important code segment (e.g, a method or a few lines of code) and necessary illustrations to explain your solution.
- Detailed UML Class Diagram.
    - Further Notes, if needed
- Detailed UML Sequence Diagram of stated function.
    - Further Notes, if needed
- Testing.
    - Test Cases and Results

### 2. Length:

The report should be at most 15 pages from cover to cover including diagrams/Testing results/references/appendix, if there is any. If you could well present your work in fewer than 11 pages, you are encouraged to do so.

DO NOT include source code/Java API doc in the report but stored the them in the CD/DVD.

**APPENDIX D:**

## Creating Javadoc:

Detailed can be found at http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

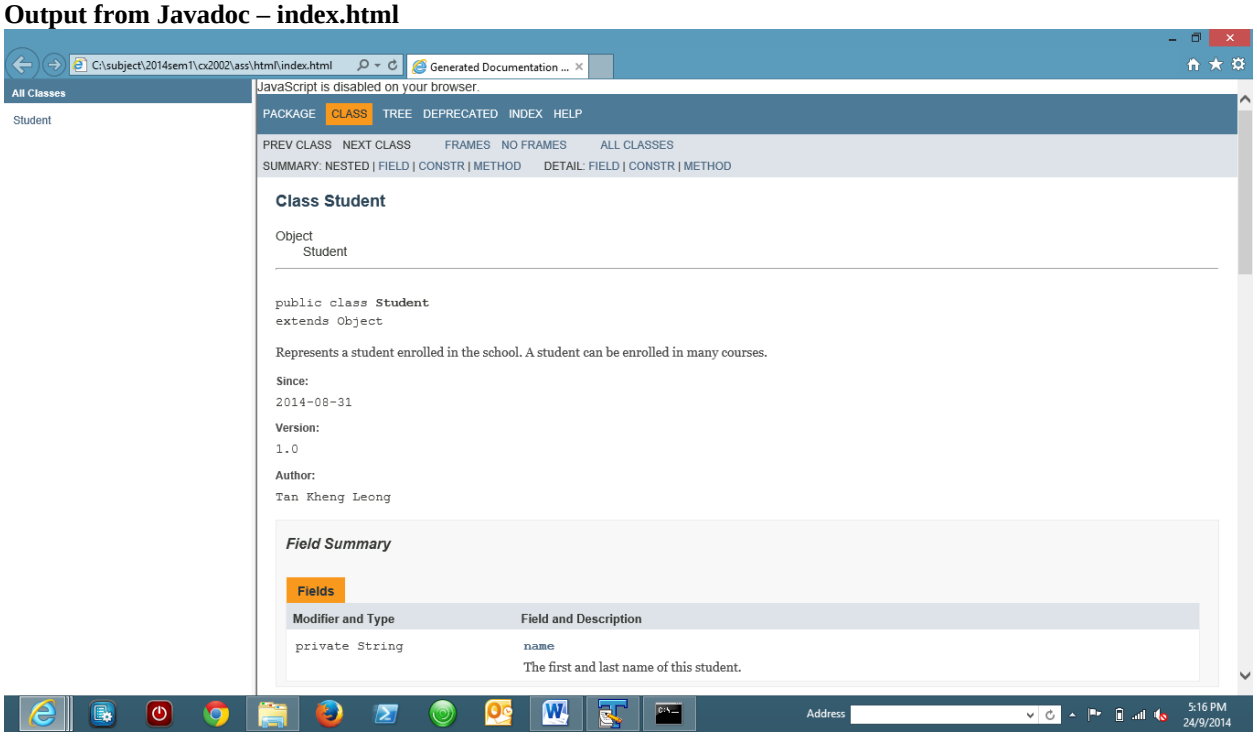
Using Javadoc in Eclipse : Youtube : http://www.youtube.com/watch?v=Hx-8BD_Osdw

Below is a short example :

```
/**
  Represents a student enrolled in the school.
  A student can be enrolled in many courses.
  @author  Tan Kheng Leong
  @version 1.0
  @since   2014-08-31
 */
public class Student {

  /**
   * The first and last name of this student.
   */
  private String name;

  /**
   * The age of this student.
   */
  private int age;

  /**
   * Creates a new Student with the given name.
   * The name should include both first and
   * last name.
   * @param name This Student's name.
   * @param age  This Student's age.
   */
  public Student(String name, int age) {
    this.name = name;
    this.age = age;
  }

  /**
   * Gets the first and last name of this Student.
   * @return this Student's name.
   */
  public String getName() {
    return name;
  }

  /**
   * Changes the name of this Student.
   * This may involve a lengthy legal process.
   * @param newName This Student's new name.
   *                Should include both first
   *                and last name.
   */
  public void setName(String newName) {
    name = newName;
  }
```

```
}
```

**Output from Javadoc – index.html**




**<u>For those familiar with using command prompt :</u>**

Steps to general API doc :

(1) Locate the installed path of JDK (java development kit)
- In Windows, it should be in C:\Program Files\Java\jdk<version>\

(2) Open command prompt

(3) Go to your src directory using cd

(4) At prompt …..src> <path to jdk>\bin\javadoc" -d ./html -author -private -noqualifier all -version <packagename1> <packagename2> <….>

Eg .

**C:\subject\2014sem1\cx2002\src>**"C:\Program Files (x86)\Java\jdk1.8.0_05\bin\javadoc" -d ./html -author -private -noqualifier all -version edu.ntu.sce.cx2002 edu.ntu.sce.cx2003

| **Statement** | **Purpose** |
|---|---|
| **C:\subject\2014sem1\cx2002\src>** | Path to your src root |
| "C:\Program Files (x86)\Java\jdk1.8.0_05\bin\javadoc" | Path to your jdk javadoc.exe [ using double quote if path has space in between, eg Program Files ] |
| -d ./html | -d : specific folder to store html doc<br>Eg ./html means current directory create a html folder to store |
| -author | Include @author in doc, if provided |
| -private | Include all methods and fields |
| --noqualifier all | Omitted all full package name.<br>Eg show **String** instead of **java.lang.String** |
| -version | Include @version in doc, if provided |
| edu.ntu.sce.cx2002 edu.ntu.sce.cx2003 | Different package names |