ASSIGNMENT

CE/CZ2002: Object-Oriented Design & Programming

Building an OO Application

2020/2021 SEMESTER 1

SCHOOL OF COMPUTER SCIENCE AND 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,
- to model, design and develop an OO application.
- to gain familiarity with using Java as an object oriented programming language.
- to 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).** :

My STudent Automated Registration System (MySTARS)

The following are information about the application:

a) It is a university application meant for each School's academic staff and undergraduate students.
b) The application allows the creation of courses and adding of student records as well as registration of courses and students. There will be an administrator mode for academic staff and user mode for students.
c) At the start of each semester registration period, students will be required to register for their courses. Each course (subject) will have its course code, its corresponding index number information, the class schedules and venue, and available vacancy.
d) Courses may have lectures only, lectures and tutorial only or lectures, tutorial and laboratory sessions.
*e) *Students on waitlist will be placed in a queue whereby when there is available slot, the first in queue will be allocated the slot. A notification will be sent to the student.*

Functional Requirements:

1. Login/Logout – upon success login, different menu (Student/Admin) will be display. [The password should not be displayed on the console when entry]

Student (Access is only allowed within pre-defined period)

1. *Add Course
2. Drop Course
3. Check/Print Courses Registered
4. Check Vacancies Available
5. Change Index Number of Course
6. Swop Index Number with Another Student
   [refer to ***STARSPlannerSTARSUser Guidev1_extracted.pdf*** for details of functions - *ignore the Graphical display*]

Admin

1. Edit student access period
2. Add a student (name, matric number, gender, nationality, etc)
3. Add/Update a course (course code, school, its index numbers and vacancy).

Refer to https://wish.wis.ntu.edu.sg/webexe/owa/aus_schedule.main for details of each course's index numbers.

4. Check available slot for an index number (vacancy in a class)
5. Print student list by index number.
6. Print student list by course (all students registered for the selected course).
   [print only student's name, gender and nationality]

(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 using NTU STARS system to recommend and implement additional admin function/s in support of the student function/s. State your additional function/s in your report.

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, how the waitlist is handled and time schedule clashed alert.

Assumptions :

(1) Need not consider multi-users concurrent login.
(2) Need not consider pre-requisite conditions when registering course.
(3) Notification will be via external API which will be available later. System will just show a message that a notification is sent.
[Refer to announcement made on the new notification requirements]
(4) The passwords for login will be stored in a flat file in hashed format and not clear text.
(5) External source implies pre-existing records and can be loaded from the file/s.
(6) Courses and Students records are to be stored in files. The format of storage is up to individual group's considerations.

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)
   - showing the flow of the **"Print student list by course"** function.
   - *The diagram should show clearly all participating objects involved with sufficient detailed flow and relevant interaction fragments.*

c) 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. ***The group must also email the WBS.xls to the course-coordinator with ALL members cc 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 and audio 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 include the following:

a. The report and clear image files of the diagrams.
b. Upload your video and audio recording of the demonstration to **YouTube** and provide the link in the report. The link has to be directly accessible without the need of login.
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 (multiplicity, association name, rolename, etc) and relationship.

UML Sequence Diagram [**20 Marks**]

- Show only the sequence Diagram mentioned in 5(b)
- Clarity, Correctness and Completeness of flow and object interactions details (Boundary-Control-Entit**ies**).

Design Consideration [**15 Marks**]

- Usage of OO concepts and principle - correctness and appropriateness

Implementation Code [**20 Marks**]

- Include 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.
- You are assessed on Diagram to Code correctness, readability, Java naming convention, exception handling, completeness of Java Doc and overall quality.

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 set of submission from each group [***follow strictly the file naming conventions***]. Report format guidelines are provided in the Appendix C below.

1. Soft copy of the **report and code to be uploaded** as a zipped file to your individual CE/CZ2002 **LAB site** (eg FEP1, FSP1, etc). The link is provided on the left panel "Assignment Submission".
   File name convention : <lab_grp>-grp<assignment_grp#>.zip
   Eg, FEP2-grp3.zip

2. **SUBMISSION DEADLINE** : **15th November 2020, 11.59pm**.

Important:
Note that **THREE (3)** marks will be deducted for the delay submission of each **calendar day**.

10. REFERENCES & TOOLS

- UML Diagrams tool - Visual Paradigm http://www.visual-paradigm.com/
- Edventure Cx2002 main course site content
- Edventure 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 MySTARS 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 and 4 courses (pre-populated prior to running test cases).
ii. Refer to **https://wish.wis.ntu.edu.sg/webexe/owa/AUS_SCHEDULE.main** for details. You may change Day/Time to suit your test cases, eg day/time clash.
iii. To have each index class size of 10 vacancies each (slightly more than original lecture size) – for testing of zero vacancy and waitlist.
iv. Need not have to follow functionalities sequentially but follow a storyboard scenario.

(TBA : to be added by your group)

1. Student Login

	Test Case	**Expected Outcome**
a	Login before allowed period (dates)	Appropriate message display
b	Login after allowed period (dates)	Appropriate message display
c	Wrong password	Appropriate error message display

2. Add a student

	Test Case	**Expected Outcome**
a	Add a new student	the listing of **all** students should be displayed after the addition [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	

3. Add a course

	Test Case	**Expected Outcome**
a	Add a new course (with combination of (ii) from above)	listing of **all** courses should be displayed after the addition. [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	

4. Register student for a course

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

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

	Test Case	**Expected Outcome**

a	Check for vacancy in course index	Appropriate message display, eg 3/10 [vacancy/total size]
b	Invalid data entries (eg course code, class code etc)	Appropriate error message display
c	TBA	

6. Day/Time clash with other course

	Test Case	**Expected Outcome**
a	Add a student to a course index with available vacancies.	Appropriate message display, eg day/time clash

7. Waitlist notification

	Test Case	**Expected Outcome**
a(i)	Add studentA to a course index with 0 vacancies	Appropriate message display to inform student on waitlist
a(ii)	Drop studentB from the same course index	StudentB successfully dropped and studentA successfully added. Simulate a notification sent.
a(iii)	Display studentA timetable	Show the course on waitlist added.

8. Print student list by index number, course

	Test Case	**Expected Outcome**
a	Print list by (i) Course (ii) index	Appropriate display. [group to decide the amount of details to display]
b	Invalid data entries (eg course code, index code etc)	Appropriate error message display
c	TBA	

Further test cases to be added by your group to further demonstrated the full functionalities of MySTAR.

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

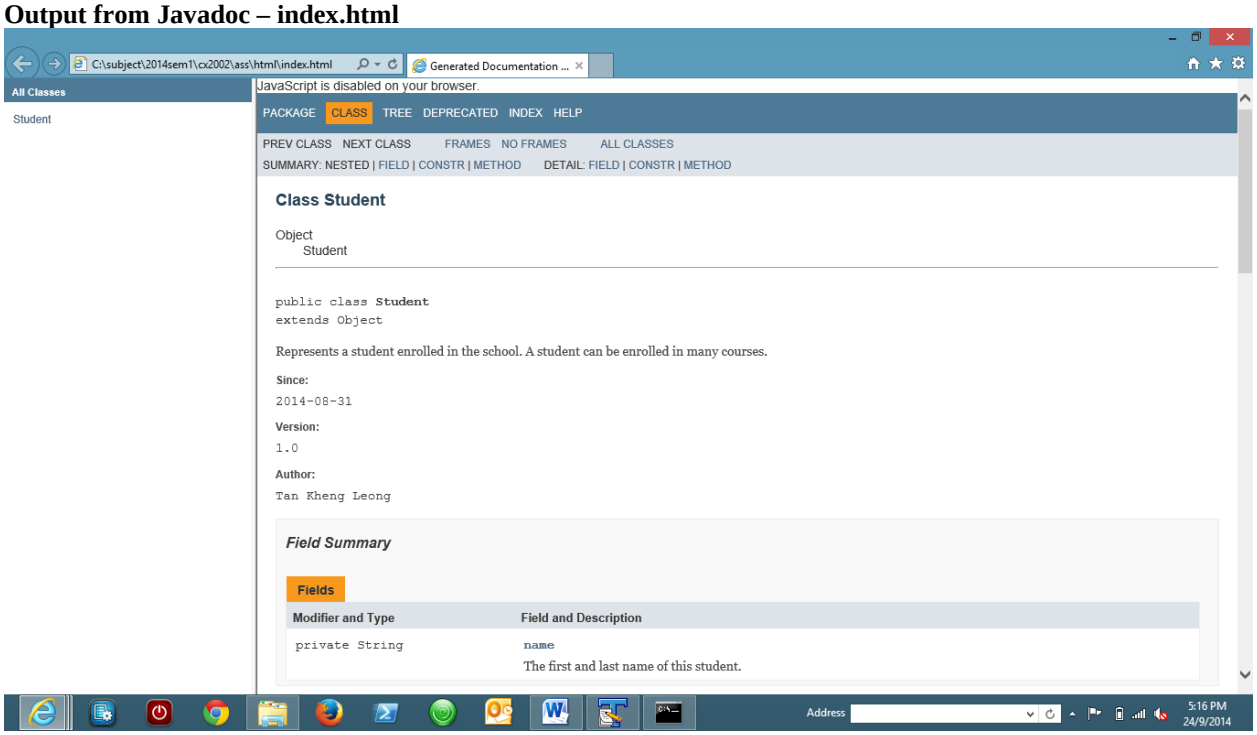
Using Javadoc in Eclipse : Youtube : http://www.youtube.com/watch?v=Hx-8BD_Osdw

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 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. 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