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

ASSIGNMENT

Building an OO Application

2023/2024 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,
- 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

Assigned SCSE lab.

3. EQUIPMENT

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

4. THE ASSIGNMENT

The group assignment involves designing and developing a:

Fastfood ordering and management System (FOMS)

The FOMS is an application aimed at streamlining the process of ordering, payment, and order management for both customers and staff at Fastfood restaurants. The system is expected to facilitate easy staff management, menu browsing, order customization, and payment processing.

Functional Requirements:

1. **Menu Management:**
   - Ability to add, update, or remove menu items, including prices, descriptions, and item categories.
2. **Order Placement:**
   - Functionality for customers to place orders, customize their food items, and choose between takeaway or dine-in options.
3. **Payment Integration:**
   - Secure payment processing capabilities for various payment methods, including credit/debit cards and online payment platforms.
4. **Order Tracking:**
   - Real-time updates for customers on the status of their orders, from new order to completed order.

Actors and Interaction:

1. **Customers:**
   - Can browse the menu, customize orders, make payments, and track order status, collect food after the order status becomes “ready to pickup”.
2. **Staff:**
   - Processes orders.
3. **Manager:**
   - Manages menu items.
4. **Admin:**
   - Responsible for company management and staff management.

Assumptions:

1. Orders are automatically canceled and removed from the " ready to pickup" list if not picked up within a specified timeframe.

Login System:

- **Customer:**
  - Customer has the access to ordering system without login.
- **Staff:**
  - Choose staff option.
  - Use an account-based system for staff, Branch Manager, Admin to log in.
  - Default password 'password' for the first login, with the option to change it.
  - A staff list can be initiated through files uploaded into the system at initialization.
  - Three types of staff: staff (S), Branch Manager (M), Admin (A).

Staff:

- There are multiple staff members including manager in a branch.
- **Staff actions:**
  - Display the new orders.
  - View the details of a particular order.
  - Process order: select order to process, update the status of the processed order from a new order to be “Ready to pickup”.

Branch Manager:

- A branch with 1-4 staffs (excluding managers) has 1 manager.
- A branch with 5-8 staffs (excluding managers) has 2 managers.

- A branch with 9-15 staffs (excluding managers) has 3 managers.
- **Manager actions:**
    - Everything a staff is able to do.
    - Display the staff list in the branch supervised by the manager.
    - Add, edit, or remove menu items, price, and availability. The menu items and prices might vary among branches.

Admin:

- Add, edit, or remove Staff accounts.
- Display staff list (filter: branch, role, gender, age).
- Assign managers to each branch within the quota constraint.
- Promote a staff to a Branch manager.
- Transfer a staff/manager among branches.
- Add/remove payment method.
- Open/close branch.

Customer Interface:

- **Customer actions:**
    - Select “customer” between “customer” and “staff” options.
    - Select branch.
    - Check placed order status using order ID.
    - **New order:**
        - **Menu Browsing:**
            - Organized display of menu items.
            - Add, edit, delete menu items from the cart.
            - Choose either takeaway or dine-in for the order.
            - Check out cart.
            - Make payment for the order (no need to implement, just have an option for them to simulate payment).
            - Print receipt with order ID.
            - Collect food to make the order status change from “Ready to pickup” to “completed”.

Order Placement:

- Facilitate the selection of items, customization, and confirmation of orders.
- Item, quantity, price calculation.
- Payment

Technical Requirements:

- **Data Persistence:**

- Implement data storage to maintain the state between sessions, using serialization.

- **Error Handling:**
  - Ensure the system handles invalid inputs and exceptions.

The application is to be developed as a Command Line Interface (CLI) application (non-Graphical User Interface).

This assignment requires students to apply object-oriented design principles, focusing on modularity, reusability, and scalability of the system components.

Extensibility (Minimum impact if there is any of the following changes):

1. New payment method.
2. New branch.

The sample data file for staff list, menu item list, and branch list are given in Excel in the assignment folder. You can use them directly, or copy the content to a text file if you plan to read from a text file, or make your own data files.

But *No database application (eg MySQL, MS Access, etc) is to be used.*
No JSON or XML is to be used.

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 **write-up** on your **design considerations** and use of OO concepts in your current design, extensibility and maintainability of your design.

c) Reflection: The difficulties encountered and the ways in which they were overcome, the knowledge learnt from this course, and further improvement suggestions. Strong demonstration of understanding of learning points and insights of good design and implementation practices, based on experience gained from doing the assignment.

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 & Presentation (Deadline: Week 14 Wednesday, 11.59pm.)**

Your group is required to present your work to your TA to demonstrate the working of the application – presenting ALL the required functionalities of the application. It is advised that you planned your demonstration in a story-boarding flow to facilitate understanding of your application. *Please introduce your members and group number at the start of presentation, all the group members must take turn to present. The presenter should show his/her face while presenting.*

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 presentation duration must not exceed 15 minutes in total.***
- ***The font size used must be large enough to be readable and viewable.***
- ***The demo of the application is to done in real-time and NOT pre-run display.***
- ***The presentation can be conducted either through online meeting or physically.***

You will create your own test cases and data to test your application thoroughly. *Refer to **Appendix A for reference.*

7. **THE DELIVERABLE (Deadline: Wednesday of Week 14)**

Your group submission should include the following:

a. The report (separate diagram file if diagram is unclear in report)
b. All implementation codes and java documentation (javadoc).
c. Other relevant files (eg data files, setup instruction, etc)

8. **ASSESSMENT WEIGHTAGE**

UML Class Diagram [**25 Marks**]

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

Design Consideration [**20 Marks**]

- Usage of OO concepts and principle - correctness and appropriateness
- Explanation of design choices and how it fits the project requirements
- Coupling and cohesion of classes

Implementation Code [**30 Marks**]

- Diagram to Code correctness, readability, Java naming

convention, exception handling, completeness of Java Doc and overall quality.

- A Java API HTML documentation of **ALL** your defined classes using Javadoc mustbe submitted. The use of javadoc feature is documented in Appendix D.

Demonstration and report [**25 Marks**]

- Coverage of application essentials and functionalities, user friendliness, demoflow, innovation.
- Report structure and reflection

9. SUBMISSION

This is a **group assignment**, and one submission from each group.

Report format guidelines are provided in the Appendix C below.

1. Soft copy of your deliverables to be **uploaded** to your individual SC/CE/CZ2002 **LAB site** (eg FEP1, FSP1, etc) in **NTULearn**. The link is provided on the left panel "Assignment Submission".
   File name convention : <lab_grp>-grp<assignment_grp#>.<ext> Eg, FEP2-grp3.pdf [<ext> can be pdf, doc, zip,]
2. **DEADLINE** : **Week 14 Wednesday, 11.59pm.**

Important:

Note that THREE (3) marks will be deducted for the delay submission of each calendar day. Lateness is based on the date the captured in NTULearn or subsequent resubmissions (whichever is later). So check your work before submitting.

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
  [You can also try with the video recording feature for gaming in Windows 10 – press ‘Windows key + G’]

APPENDIX A:

1. Manager's action: Menu Management:

- **Test Case 1:**
  - Add a new menu item with a unique name, price, description, and category.
  - Verify that the menu item is successfully added.
- **Test Case 2:**
  - Update the price and description of an existing menu item.
  - Verify that the changes are reflected in the menu.
- **Test Case 3:**
  - Remove an existing menu item.
  - Verify that the menu item is no longer available.

2. Order Processing:

- **Test Case 4:**
  - Place a new order with multiple food items, customize some items, and choose takeaway option.
  - Verify that the order is created successfully.
- **Test Case 5:**
  - Place a new order with dine-in option.
  - Verify that the order is created with the correct preferences.

3. Payment Integration:

- **Test Case 6:**
  - Simulate a payment for using a credit/debit card.
  - Verify that the payment is processed successfully.
- **Test Case 7:**
  - Simulate a payment using an online payment platform (e.g., PayPal).
  - Verify that the payment is processed successfully.

4. Order Tracking:

- **Test Case 8:**
  - Track the status of an existing order using the order ID.
  - Verify that the correct status is displayed.

5. Staff Actions:

- **Test Case 9:**
  - Login as a staff member and display new orders.

- Verify that the staff can see all the new orders in the branch he/she is working.

- **Test Case 10:**
  - Process a new order, updating its status to "Ready to pickup."
  - Verify that the order status is updated correctly.

6. Manager Actions:

- **Test Case 11:**
  - Login as a manager and display the staff list in the manager's branch.
  - Verify that the staff list is correctly displayed.
- **Test Case 12:**
  - **A manager should be able to process order as described in Test Case 9**

7. Admin Actions:

- **Test Case 13:**
  - Close a branch.
  - Verify that the branch does not display in Customer's Interface anymore.

- **Test Case 14:**
  - Login as an admin and display the staff list with filters (branch, role, gender, age).
  - Verify that the staff list is correctly filtered.
- **Test Case 15:**
  - Assign managers to branches with the quota/ratio constraint.
  - Verify that managers are assigned correctly.
- **Test Case 16:**
  - Promote a staff to a Branch Manager.
  - Verify that the staff is promoted successfully.
- **Test Case 17:**
  - Transfer a staff/manager among branches.
  - Verify that the transfer is reflected in the system.

8. Customer Interface:

- **Test Case 18:**
  - Place a new order, check the order status using the order ID, and collect the food.
  - Verify that the order status changes from "Ready to pickup" to "completed."

9. Error Handling:

- **Test Case 19:**
  - Attempt to add a menu item with a duplicate name.
  - Verify that an appropriate error message is displayed.
- **Test Case 20:**
  - Attempt to process an order without selecting any items.
  - Verify that an error message prompts the user to select items.

10. Extensibility:

- **Test Case 21:**
  - Add a new payment method.
  - Verify that the new payment method is successfully added.
- **Test Case 22:**
  - Open a new branch.
  - Verify that the new branch is added without affecting existing functionalities.

11. Order Cancellation:

- **Test Case 23:**
  - Place a new order and let it remain uncollected beyond the specified timeframe.
  - Verify that the order is automatically canceled and removed from the " Ready to pickup " list.

13. Login System:

- **Test Case 24:**
  - Attempt to log in with incorrect credentials as a staff member.
  - Verify that an appropriate error message is displayed.
- **Test Case 25:**
  - Log in as a staff member, change the default password, and log in again with the new password.
  - Verify that the password change functionality works as expected.

14. Staff List Initialization:

- **Test Case 26:**
  - Upload a staff list file during system initialization.
  - Verify that the staff list is correctly initialized based on the uploaded file.

15. Data Persistence:

- **Test Case 27:**
  - Perform multiple sessions of the application, adding, updating, and removing menu items.
  - Verify that changes made in one session persist and are visible in subsequent sessions.

These test cases cover a range of scenarios and functionalities within the Popeyes Ordering System, ensuring that different components and interactions are thoroughly tested. Depending on the specific implementation details, additional test cases may be necessary.

APPENDIX B:

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 beawarded 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.
2. Student Code of Academic Conduct includes the latest guidelines on usage of Generative AI and any other guidelines as released by NTU.

APPENDIX C:

Report requirement:

1. **Format**:

For the main content, please use Times New Roman 12 pt font size and 1.5 linespacing. 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 (Appendix B)
- 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 toexplain your solution.
- Detailed UML Class Diagram.
  - Further Notes, if needed
- Testing.
  - Test Cases and Results
- Reflection.
  - The difficulties encountered and the way to conquer, theknowledge learnt from this course, further improvement suggestion.

2. **Length**:

The report should be at most 12 pages from cover to cover including diagrams/Testing results/references/appendix, if there is any. If you could wellpresent your work in fewer than 12 pages, you are encouraged to do so.

DO NOT include source code in the report but stored the source code in a folder. You are to ensure that the diagrams are readable and clear to the reader. [You can save the diagrams as image files and include in a folder]

APPENDIX D:

Creating Javadoc:

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

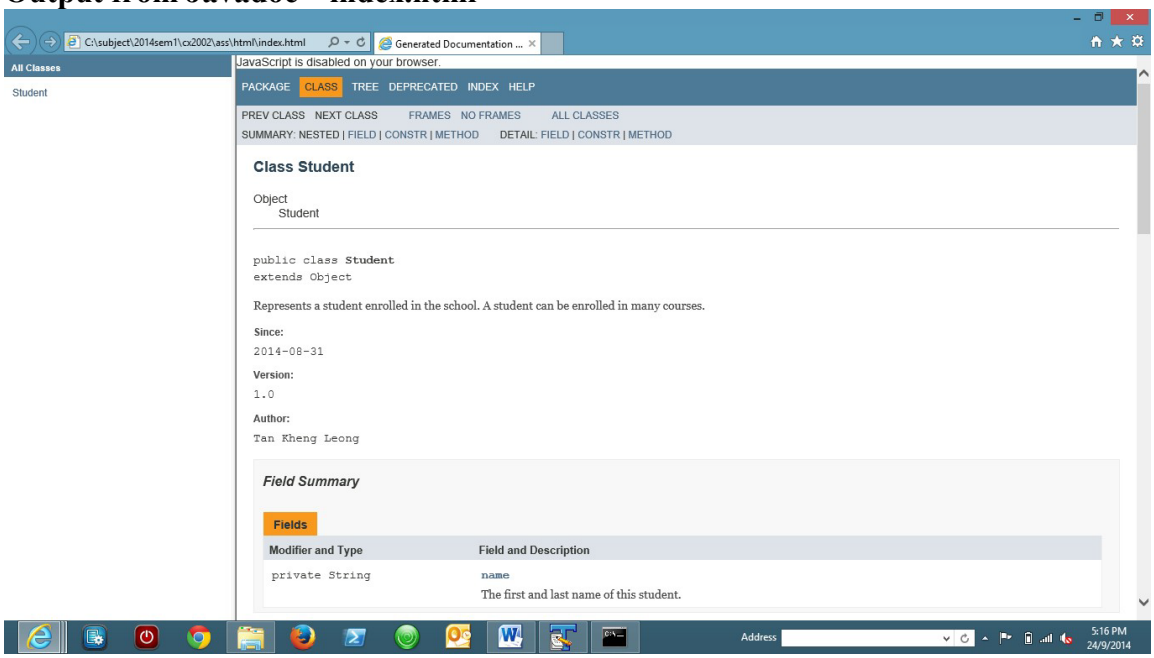
Using Javadoc in Eclipse : Youtube : http://www.youtube.com/watch?v=Hx-8BD_Osdw

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



For those familiar with using command prompt :

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 folderto 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