**CZ2002 Object Oriented Design Programming**

Assignment

FSP1 Group 5

Done By: Yong Xin, Brenda, Justin Grace, Jun Long

**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** |
| Ang Yong Xin | CZ2002 | FSP1 Group 5 |  |
| Brenda Ng Xin En | CZ2002 | FSP1 Group 5 |  |
| Justin Grace | CZ2002 | FSP1 Group 5 | 19/4/19 |
| Tan Jun Long | CZ2002 | FSP1 Group 5 |  |

**Important notes:**

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

**Contents**

1. **Design Considerations 3**
   1. Approach taken 3
   2. Principle used 3
   3. Assumptions made 3
   4. Two New Design Features 4
2. **Detailed UML Class Diagram 4**
3. **Detailed UML Sequence Diagram 5**

3.1 Sequence Diagram of “Print Bill Invoice” 5

1. **Testing 6**

**1 Design Considerations**

**1.1 Approach taken**

Object Oriented Concepts used:

* Inheritance
  + Inheritance was used in MenuItem and Promotion class.
  + MenuItem is the super class whereas Promotion inherits from the MenuItem super class.
* Encapsulation
  + Encapsulation was used in all classes.
  + All the details or instance variables of these classes will be hidden from user by using “private”.
  + Accessor and mutator methods were used to get and set the values of these variables.

**1.2 Principle used**

* Single Responsibility Principle (SRP):
  + Every class has its own specific responsibility and roles, there’s no overlapping of function and conflicts from other classes.
  + Eg: The method of ArrayList<Orders> RetrieveOrder(int Id) only retrieve the order from the class.

**1.3 Assumptions made**

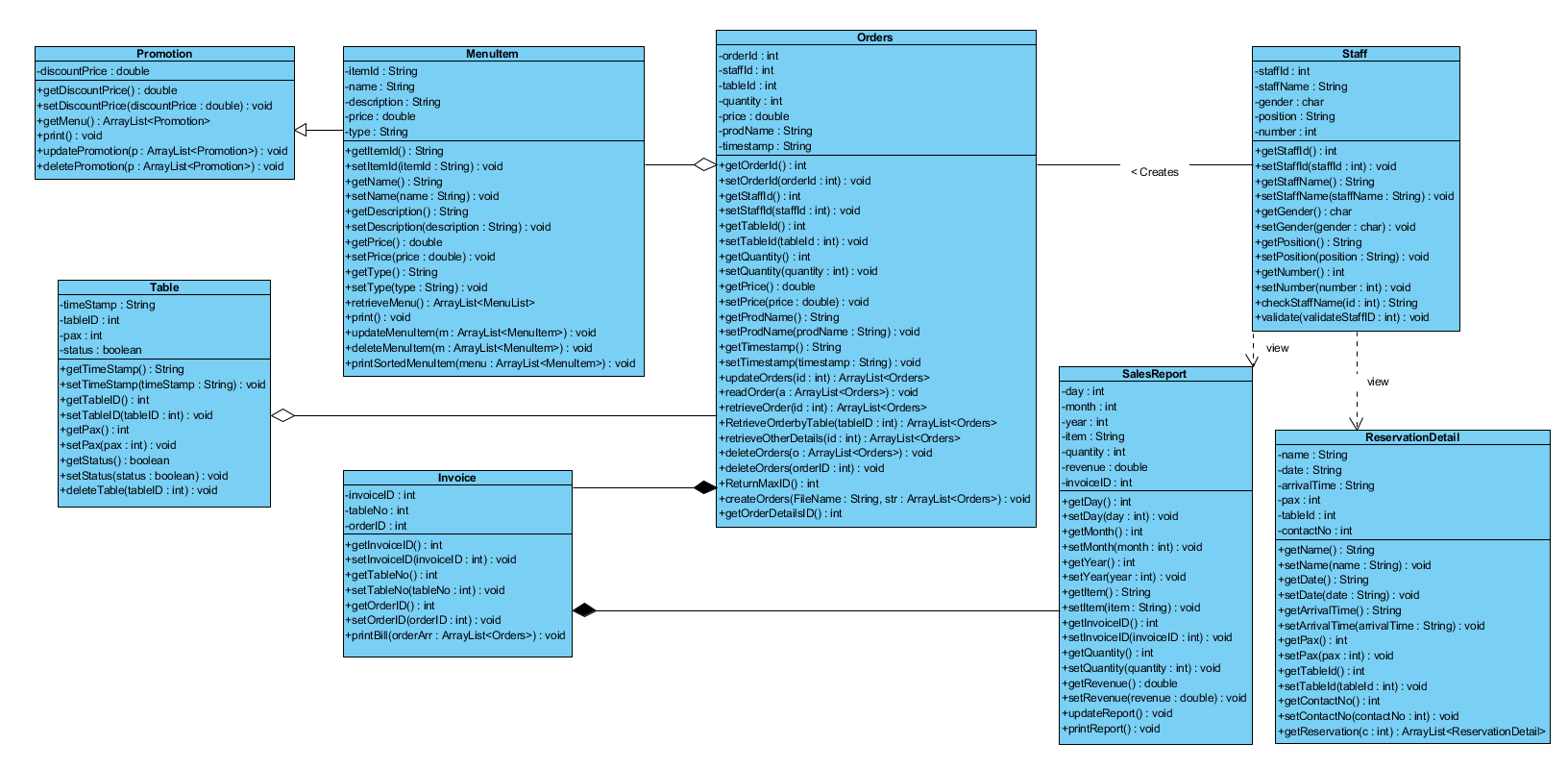
Here are the assumptions made:

* The text files required exist in the project directory and will not be deleted.
* Reservation will be automatically removed if the guest does not arrive within 30 minutes after the stated arrival time.
* Reservation can only be made in at most 1 month in advance.
* The currency will be in Singapore Dollar (SGD) and Good and Services Tax (GST) and service charge must be included in the order invoice.
* There is no need to interface with external system, eg Payment, printer, etc. Payment via credit card will always be successful.
* The restaurant has 30 tables – 5 x 10-seats, 5 x 8-seats, 10 x 4-seats, 10 x 2-seats.
* Tables cannot be combined/joined (eg, join 3 x 2-seats table to form 6-seats table) and the table ID is fixed.
* The application will not be used by staff when the restaurant is closed.

**1.4 Two New Design Features**

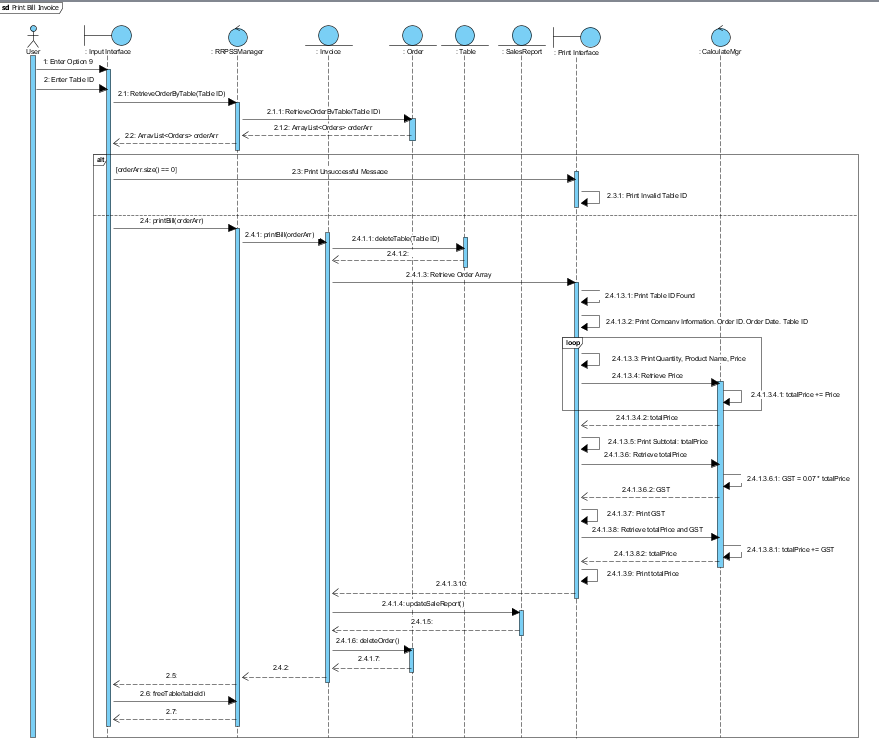
* Generate a report based on the number of sales of each product
  + This allows the staff to know which product is the most popular and what can be recommended to the customer.
  + Design Principle: It will be a class that extends from the sales report class, and use each product individual sales from the sales report class and generate a product popularity report (Reusability & extensibility)
* Reservation counter
  + Display the total number of reservation for that specific date and time when the staff login.
  + Design Principle: It automatically show all the reservation that has been made on that day, this will allow staff to be prepared for business of that day. (Reusability & extensibility)

**2 Detailed UML Class Diagram (Extract from Visual Paradigm)**

****

**3 Detailed UML Sequence Diagram (Extract from Visual Paradigm)**

**3.1 Sequence Diagram of “Print Bill Invoice”**

****

**4 Testing**

Staff ID

|  |  |
| --- | --- |
| Test Scenario | Output |
| Check Staff ID with valid data |  |
| Check Staff ID with invalid data |  |

Menu Item

|  |  |
| --- | --- |
| Test Scenario | Output |
| **Creating/**  **Updating Menu Item**  Check price and type of Menu Item with valid data |  |
| **Creating/**  **Updating Menu Item**  Check price and type of Menu Item with invalid data |  |
| **Removing Menu Item**  Check Item ID of item to be deleted with valid data |  |
| **Removing Menu Item**  Check Item ID of item to be deleted with invalid data |  |

Orders

|  |  |
| --- | --- |
| Test Scenario | Output |
| Check Pax with valid data |  |
| Check Pax with invalid data |  |
| Check Quantity with valid data |  |
| Enter Quantity with  Invalid input |  |
| View Order with valid Order ID |  |
| View Order with invalid Order ID for the first and second time |  |

Invoice

|  |  |
| --- | --- |
| Test Scenario | Output |
| Print bill invoice with valid Order ID | Null pointer exception error |
| Print bill invoice with invalid Order ID |  |
| Print sale revenue report by period with Staff ID that is a manager |  |
| Print sale revenue report by period with Staff ID that is not a manager |  |
| Print sale revenue report by period with valid date |  |
| Print sale revenue report by period with invalid date |  |