Course: Object Oriented Programming

Title: Design Activity document of Open Ended Assessment

"Yatri Nivas"

Team members:

Name	USN	Roll NO
Veerraj Chitragar	**01FE22BCS164**	**427**
Avinash Nayak	**01FE22BCS204**	**431**

Guided By:
M . K . Gonal

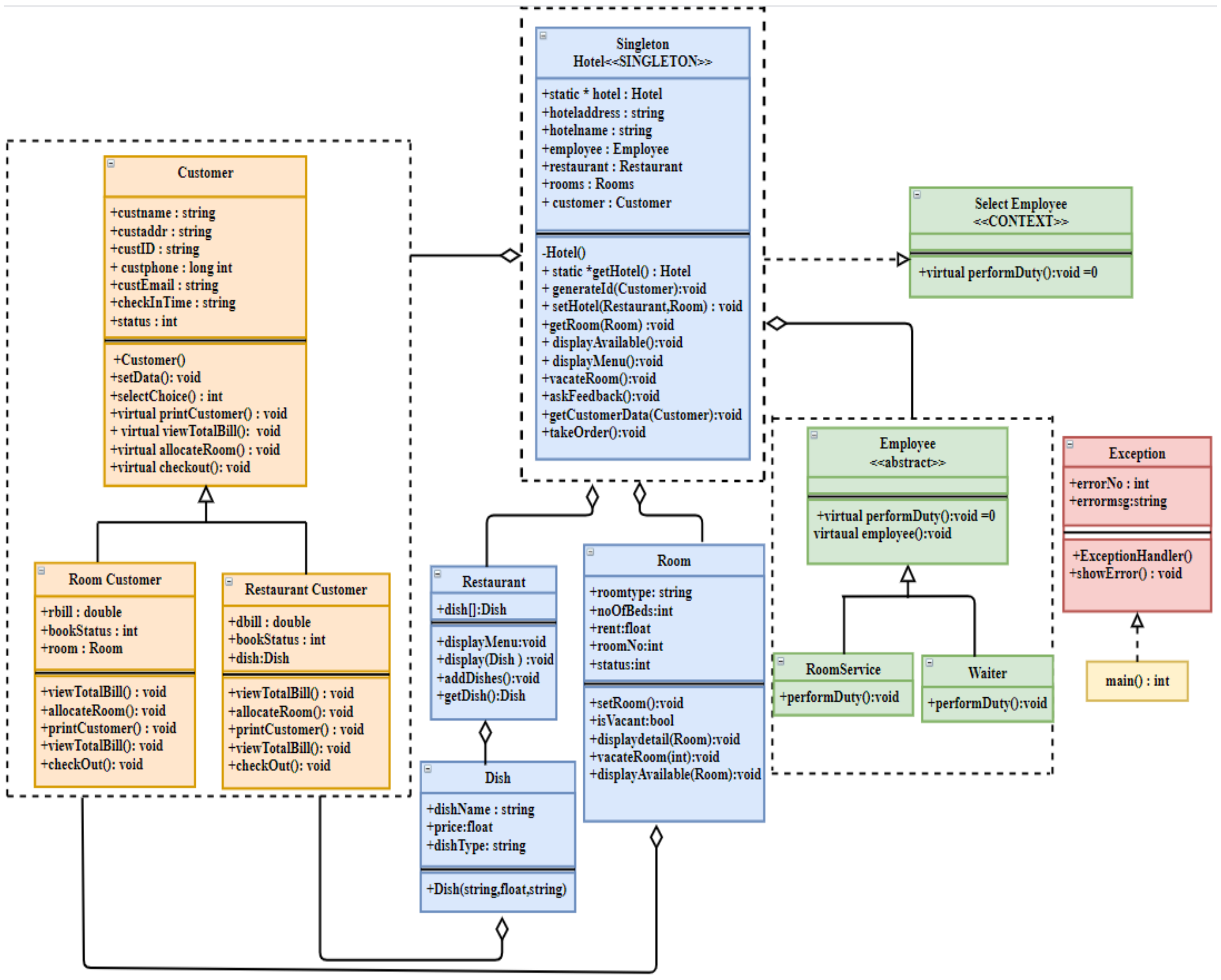
Problem Description:

Residence Serenity is a newly opened accommodation known for its exquisite location amidst manicured gardens and a freshwater lake. It provides standard amenities and facilities, offering well-furnished rooms and delicious meals. The accommodation system needs to efficiently handle room reservations, room services, and dining services for its customers. The system should be able to manage customer information, generate unique customer IDs, handle room and dish allocations, and provide invoices.

Objects Identified:

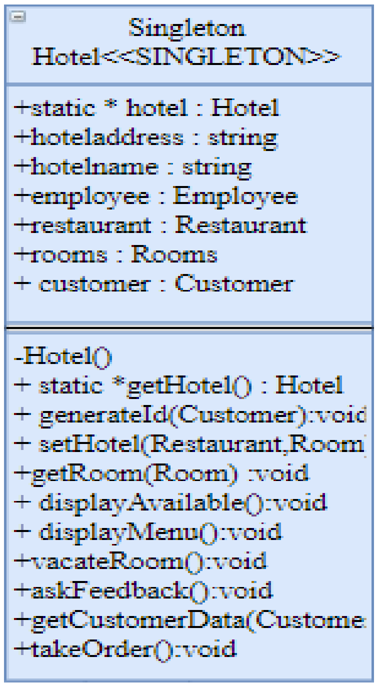
- Customer
- Room Customer
- Employee
- Dish
- Restaurant Customer
- Restaurant
- Hotel
- Room
- Room Service

Class Diagram:



Implementation Details:

1. Class : Hotel



Hotel is the main class which follows singleton design pattern. It consists of static *hotel to create an instance of the class. The other attributes include the name and the address of the hotel.

Functions:

generateID() which generates a unique ID for every customer.

displayAvailable(): displays all the available rooms in the hotel.

displayMenu(): displays the menu available in the restaurant.

guestSummary(): It gives the whole summary of the details of the customer.

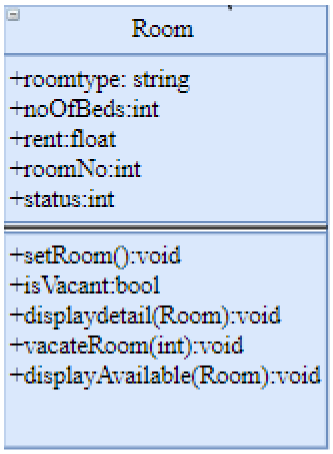
askFeedback() : It asks the feedback of the customer about the hotel service.

vacateRoom(): to vacant the room that has been allotted.

getCustomerData() : to get the data from customer.

takeOrder() : to take the order from the customer.

2. Class : Room



It has five attributes where roomType describes the type of room i.e., Deluxe, AC, Non-AC , General, Suite. The attribute status tells whether the room is occupied or no.

Functions:

setRoom() : to book a room.

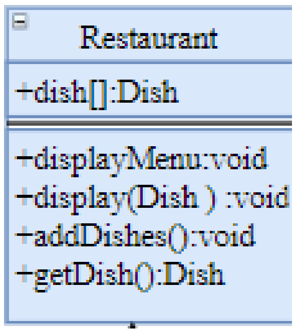
isVacant() : to know the status whether the room is booked or no.

displayDetail(): to display the details of all rooms

vacateRoom(): to vacate the room.

displayAvailable() : to display the details of the rooms which are available.

3.Class : Restaurant



It consists of array all the dishes that are available.

Functions:

displayMenu() : to display the menu for all dishes available.

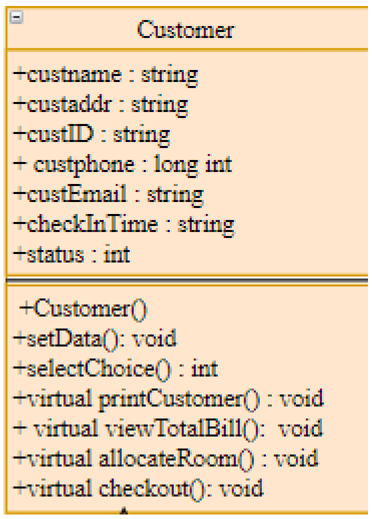
display() : To display the details of a particular dish

addDish(): to add a dish into the menu.

getDish() : to get the dish.

3. Class: Customer

The class Customer has two inherited classes restaurantCustomer and roomCustomer. It follows factory design pattern



Functions:

Setdata() : to enter all the customer details.

selectChoice() : To select the choice between booking a room and placing an order for a dish.

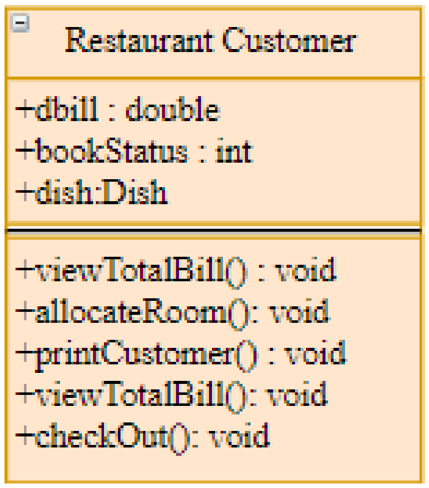
callRoomService(): To call the room service.

viewTotalBill() : to view the bill of the service service provided to him.

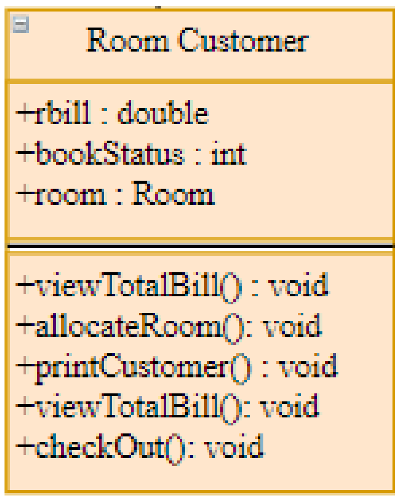
Checkout() : to checkout from the hotel.

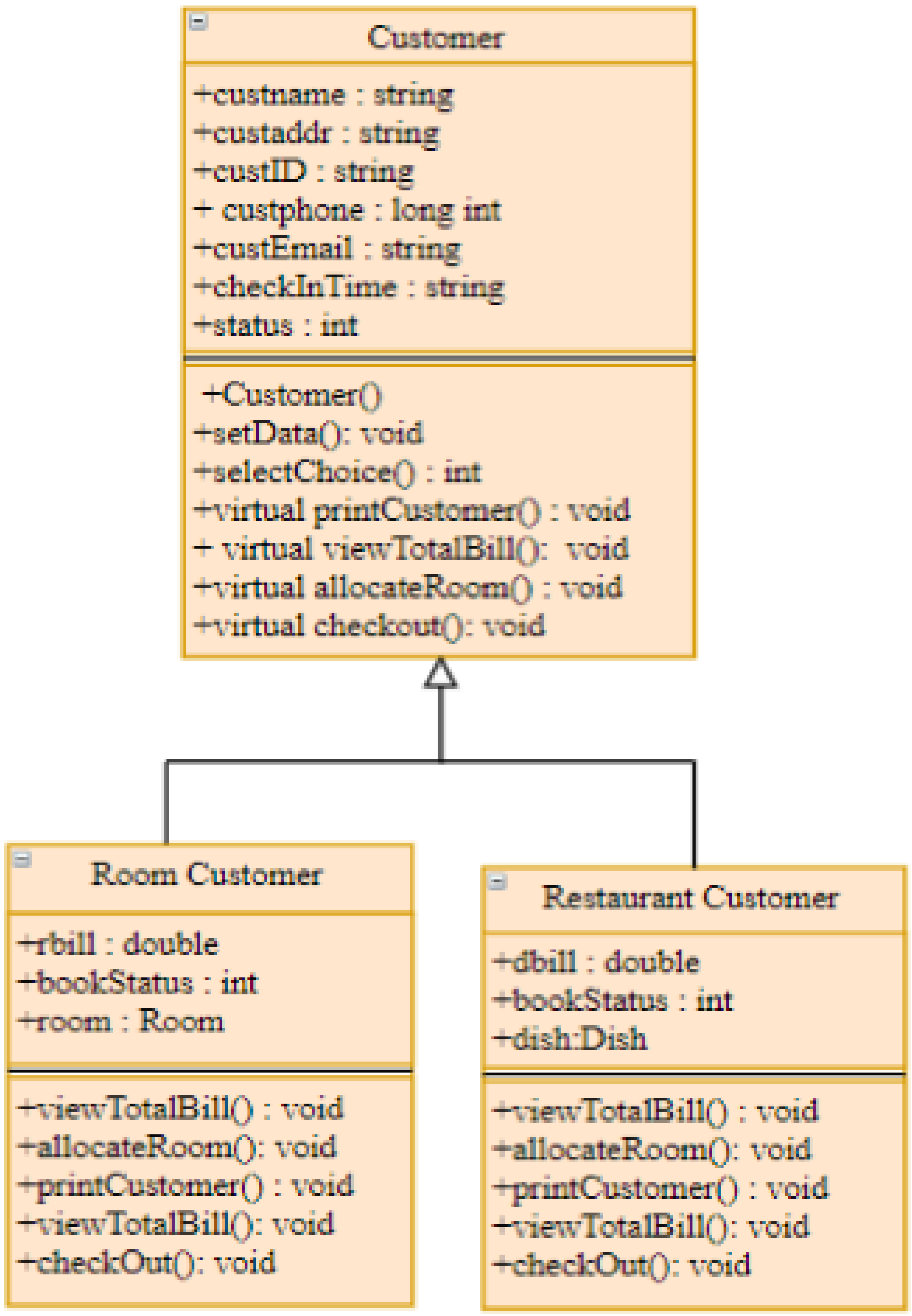
printCustomer() : to print the customer details who is in the room.

Class RestaurantCustomer:



Class RoomCustomer:

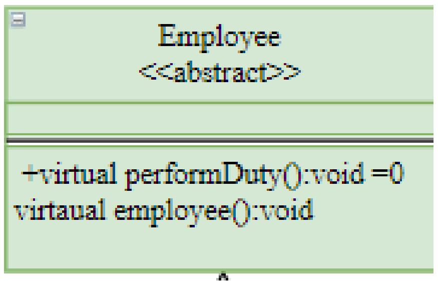




The customer class follows factory design pattern.

4. Class : Employee

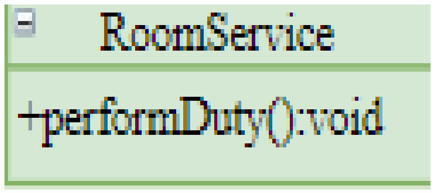
The class employee inherits two classes namely RoomService and Waiter who perform their respective duties on customer demand. It follows strategy design pattern.



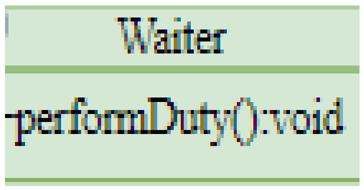
Functions:

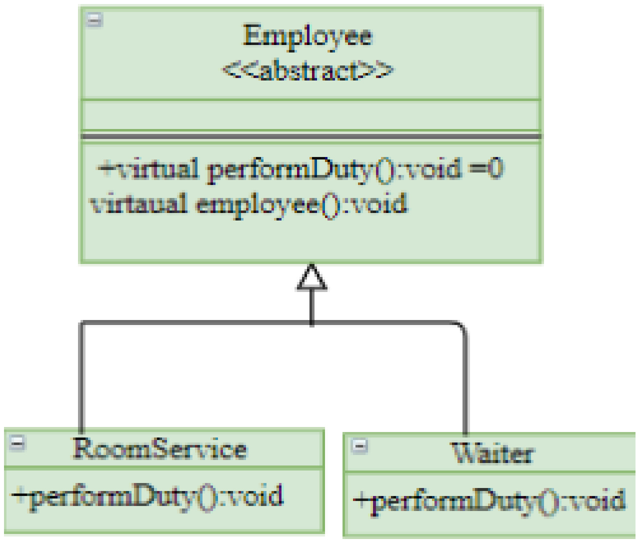
performDuty() : To display all the employees working in the hotel

Class: Roomservice



Class : Waiter

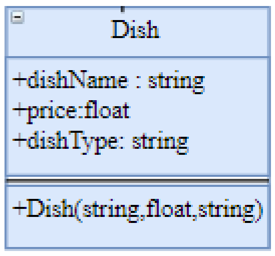




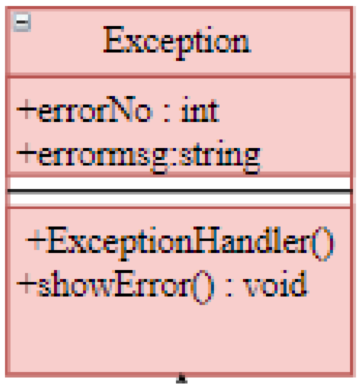
5. Class: Dish

This class has attributes such as dishname, price and dish type. The dishtype specifies the type to which the given dish belongs i.e., main course, starters, desserts, beverages etc.

The parameterized constructor creates new dishes.



6. Class: Exception



The following exception class comes into picture if:

- any customer asks for a room which is already occupied.
- any customer orders for a dish which is not available at that instance of time.

In the above cases Exception Handler will throw an appropriate error.