



CE//CZ2002: Object-Oriented Design & Programming

## ASSIGNMENT

FEP 1 - Group 1

|                   |           |
|-------------------|-----------|
| ASHLEY NEE        | U1822505K |
| HARIKISHAN KALYAN | U1821462C |
| LAM MAN TING      | N1904534G |
| THIN LAT HAN      | U1822997F |
| TRAN THIEN HA MY  | U1631219K |

2019/2020 SEMESTER 2






SCHOOL OF COMPUTER SCIENCE & ENGINEERING  
NANYANG TECHNOLOGICAL UNIVERSITY

## **PART I: 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<br>(CE2002 or CZ2002) | Lab<br>Group | Signature /Date   |
|-------------------|------------------------------|--------------|---|
| TRAN THIEN HAMY   | CE2002                       | FEP1         | <br>20/04/2020  |
| Thin Lat Han      | CE2002                       | FEP1         | <br>21/04/2020 |
| LAM MAN TING      | CE2002                       | FEP1         | <br>20/04/2020 |
| ASHLEY NEE        | CE2002                       | FEP1         | <br>21/04/2020 |
| HARIKISHAN KALYAN | CE2002                       | FEP1         | <br>21/04/2020 |

## **PART II: Design Considerations**

This project is going to implement a Hotel Reservation and Payment System (HRPS). There are a total of 12 classes in this project. And 'test' is the controlling class which contains the main method. From the 'test' class, the messages are sending to the other 11 classes to complete the required functions of HRPS.

### **Assumptions Made:**

We made the below assumptions before developing our application:

- This program is designed specifically for the hotel in question (the number of rooms is fixed, and the number of floors and room numbers are also pre-fixed). To get the application more universal, we can have a separate method which asks for the number of floors, the types of rooms they have, the number of rooms of each type and the number of rooms. Then, room arrays will be customized to the needs of the hotel more.
- The system is solely used by the hotel staff. Customers will only require some functions that are dedicated for them (e.g. reservation, ordering) by asking the hotel staff and don't tamper with other admin-related functions (e.g. update room details) because there are no log-ins of users to grant specific access.
- Time is entered correctly. (when in reservation class, wrongly entered time are still taken.)
- The room number will be entered as a string.

### **Approaches Taken:**

We adopted the below approaches to implement HRPS:

- We used 4 different arrays for each type of room (Single, Double, Deluxe and VIP). This makes it easier to manage in terms of room creating, searching and updating. However, it reduces user-friendliness as it requires the user to enter one more variable that is the type of the room (instead of only the room number). For more user friendly, we can take in only the room number variable and search which type matches that room number, instead of asking the user to enter the room type themselves.
- The main objectives of the *roomServiceOrder* class are to create an order for a guest and calculate the total bill for that order
- Objects are defined in the constructor with a unique combination of the guest's ID and order number
- The class:
  - Creates a static array (allocated length 20) containing objects of the Item class by obtaining these Items from the Menu class
  - Contains functions to display certain information like order confirmation for clarity purposes
  - Can update the status of the order
- A layered approach was considered during the design of the classes to help improve the flexibility, reusability while becoming more extensible and allowing room to further improvisations. For example, the Guest class is primarily used for entering and updating the guest's details while the *GuestDB* class is responsible for searching and finding a valid guest that has checked in while the *GuestDBStorage* class functions as the data storage class, storing all the details of the guests.
- Storing of data into text files is needed so that even after closing the console/Eclipse, the next time the program is used, the data will still be stored. This fits reality where guests often stay more than a day and at the end of each day, staff will close the program while expecting the data to be still stored when opening the next day.

- *BufferedReader* and *BufferWriter* are used to ensure that the text file produced is readable. Additionally, *toString()* function is overridden for classes which data need to be stored so that *BufferWriter* can be used to write data in the text file.
  - Example: In *GuestDBStorage* class of *GuestDBStorage.java*, *toString()* method that has been overridden is used to write each guest object in *GuestDB* into the text file.
- Use of *toString()* method in *GuestDBStorage.java*

```
for (int i = 0; i < guestDB.getCount(); i++) {  
    bw.newLine();  
    guest = gList[i];  
    bw.write(guest.toString());  
    bw.newLine();  
}  
bw.close();
```

- *toString()* method in *Guest.java*

```
public String toString() {  
    return "guestID=" + this.guest_ID + "\nname=" + this.name + "\nccNo=" + this.ccNo + "\naddress=" + this.address  
        + "\ncountry=" + this.country + "\nnationality=" + this.nationality + "\ngender=" + this.gender  
        + "\nidentity=" + this.identity + "\ncontact=" + this.contact + "\nnumChildren=" + this.numChildren  
        + "\nnumAdults=" + this.numAdults + "\nmax_ID=" + this.max_ID + "\ntotalGuest=" + this.totalGuest  
        + "\ncheck=" + this.check + "\nempty=" + this.empty;  
}
```

## Principles Used:

We apply the following principles to develop the system:

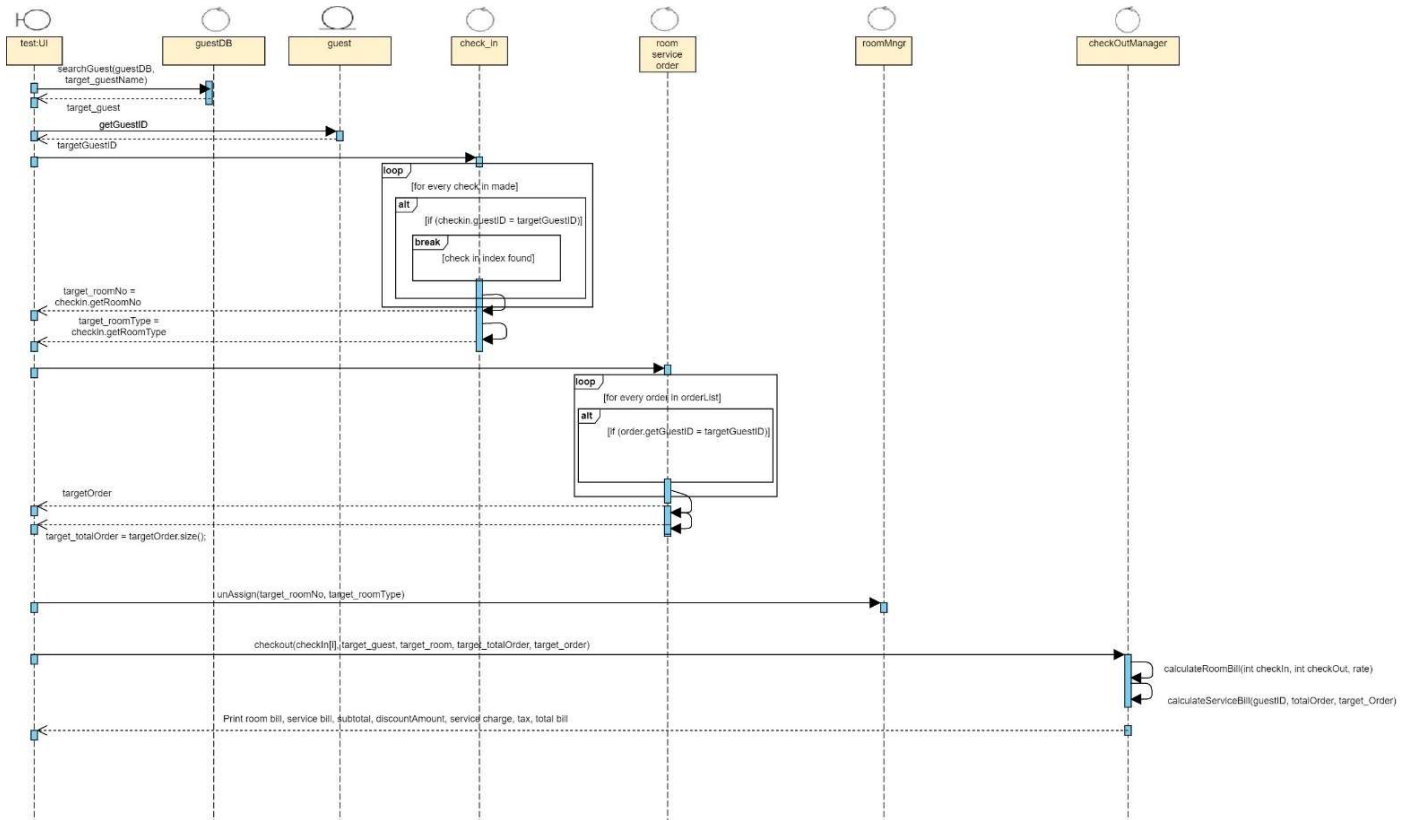
- Importance of privacy, accessors and mutators: The attributes in each class are set as *private*, which means it could be accessible only within that class's implementation. Therefore, we use methods *get* and *set*, so that the other classes would be able to access the value and modify them.
- Method overloading:
  - Example 1: In the *Menu* class, the user could update price, description, or both. Therefore, we apply method overloading to allow methods to execute similar tasks.
  - Example 2: Multiple constructors
- Method overriding of inheritance: The class *Reservation* is the subclass of class *Check\_in*. *Reservation* and *CheckIn* are similar, but *Reservation* has more methods like checking expiry. Hence, *Reservation* extends from *CheckIn* as *CheckIn* is more general.

- Visibility modifiers: From the UML Class Diagram, we could identify that the attributes of each class are accessible only within that class's implementation, while the methods are visible to anywhere in the application. Besides, one class cannot access those declared as private from other classes, so it is necessary to create a method 'get' for accessing the attributes from other classes.
- Delegation principles: Several classes implemented the use of Strings, Arrays, Serialization, and ArrayList. To improve the logic flow of the program, several methods of Serialisation, ArrayList, Arrays (equals(), contains(), get()) were delegated and were crucial in the overall implementation of the program.
- Java Packages: Organize java classes into one package 'registration' according to its functionality
- Message sending and method calling from other classes
- Single Responsibility Principle (SRP): to read and write each class in a text file, new classes are created.  
execution.
  - Example, the class *Guest* is implemented to create, update and store the details of the guest to be checked in while the *GuestDB* class functions as a control class, interacting with both the *Guest* class and the test class (the UI component), updating both simultaneously. Similarly, when checking out the *roomManager* is responsible for unassigning the room of the guest to be checked out while the *checkOutManager* facilitates the payment aspect of the check-out.
- File I/O
  - *FileWriter*, *FileReader*, *BufferWriter*, *BufferedReader*
  - Serialisable classes (For classes that read and write data)



## PART IV: UML Sequence Diagram

Note: For the clear version, please refer to the image file *Sequence Diagram - FEP1 - grp1* in the folder.





## **PART V: Testing**

Note: The demonstration video available on Youtube

(<https://youtu.be/vPWDG2E25f8>) with English subtitles.

### **Test case 1: Room service menu item**

1. Create 10 room service items as the following:
  - To create a room service item, its name, description and price are needed

| Name                                  | Description  | Price |
|---------------------------------------|--|-------|
| Canned Beverages                      | Coke/Sprite/Fanta/A&W  | 6     |
| Bottled Water                         | Ice Mountain   | 5     |
| Fish 'n' Chips                        | dory fillet and crinkle-cut fries with a side of sour chives sauce   | 20    |
| House Made Potato<br>Saltbush Gnocchi | charred broccoli, blue cheese, creme fraiche, saltbush &<br>macadamia crumble                                    | 28    |
| Roasted Free Range<br>Chicken Breast  | hand cut chips, Mont Priscilla, broccolini, Wildfire cream sauce   | 34    |
| Lamb Duo                              | smoked dukkah lamb cutlets, braised lamb shank pie, pumpkin &<br>maple purée, truffled peas, yoghurt, lamb glaze | 39    |
| Cold Smoked Beef<br>Fillet            | charred corn, cauliflower purée, glazed shallot  | 43    |
| Sautéed Greens                        | native garlic butter   | 9.5   |
| Rich Chocolate<br>Cheesecake          | dark chocolate ganache, vanilla bean ice cream   | 15    |
| Liqueur                               | Frangelico/Baileys/Kahlua  | 5     |

2. Change the details of a room service item
  - Update price of Liquor to 10.
3. Remove a room service item
  - Remove “Fish ‘n’ Chips from the Room Service order

## Test case 2: Reservations

### 1. Creating 5 reservations as the following:

- To create a reservation, the guest's particulars are needed. This includes name, credit card number, address, country, gender, identity number (passport number/ license number) and contact number.
- The guest also needs to indicate the number of adults and children will be coming and expected date and time of check-in
- Additionally, the guest needs to choose the desired room type, desired room number.
- Finally, he will be given a reservation acknowledge receipt containing his assigned reservation code and reservation details.
- Note: If the reservation has not expired (one hour after the expected date and time of check-in), the room will remain reserved. When it expires, the room will become valid immediately.
- Note: *GuestID* is unique to each guest and is used to distinguish between guests with the same name.

| Guest ID | Name    | No. of adults | No. of children | Credit card no. | Address   | Country | Nationality | Gender | (1)<br>Passport<br>no. (2)<br>License no. | Contact no.           | Desired room<br>type<br>(Single/Doubl<br>e/Deluxe/VIP) | Desired<br>room no. | Expected<br>date of check<br>in<br>(YYYYMMDD) | Expected<br>time of<br>check in<br>(HHMMSS) |
|----------|---------|---------------|-----------------|-----------------|---|---------|-------------|--------|---|-----------------------|--|---------------------|---|---|
| 1        | Lenard  | 2             | 0               | 23579327459832  | 12 Walnut Drive,<br>Oakland, CA<br>94603                          | USA     | American    | Male   | (1)<br>000901324<br>8                     | +1<br>293571921       | Double   | 1-Feb               | 20200422                                      | 200000                                      |
| 2        | Mickey  | 3             | 2               | 32954925892451  | 9144 Dogwood<br>Court,<br>Parkersburg, WV<br>26101                | USA     | American    | Male   | (1)<br>037245871<br>2                     | +1<br>23959237        | Double   | 1-Apr               | 20200422                                      | 200000                                      |
| 3        | Heather | 1             | 8               | 932472361989234 | 102 Jain<br>Mansion,<br>Bhavani Street,<br>Pydhonie, IN<br>400003 | India   | Indian      | Female | (1)<br>031749236<br>4                     | +91<br>001034983<br>7 | VIP  | 7-Apr               | 20200808                                      | 120000                                      |
| 4        | Dave    | 4             | 0               | 459632948923749 | 53 Snake Hill<br>Street, Salisbury,<br>MD 21801                   | USA     | American    | Male   | (2)<br>93271592                           | +1<br>23957945        | Deluxe   | 6-Jun               | 20200422                                      | 200000                                      |
| 5        | Polly   | 2             | 3               | 394297459723952 | 204 Heritage<br>Drive, Calgary,<br>Alberta                        | Canada  | Canadian    | Female | (2)<br>31023715                           | +1<br>38479231        | Deluxe   | 6-Jul               | 20200721                                      | 120000                                      |

### 2. Print out all reservations by pressing “11”

- These reservations will not include those that are expired/check-in.
- Note: Updating of reservations can be done by pressing “3” if guest realizes they enter one of the details wrongly.

### Test case 3: Check-in

Guests can check-in by having a reservation beforehand or by walk-in.

1. Create check-in for 3 guests with reservations made beforehand. (Guest IDs 1, 2 and 4 with reservation codes 0, 1 and 3 respectively)
2. Create check-in for 2 walk-in guests. The guests will be required to fill in the following details (similar to the information reservation guests need to make when making a reservation but without expected date and time).

| Guest ID | Name    | No. of adults | No. of children | Credit card no. | Address                        | Country    | Nationality | Gender | (1) Passport no. (2) License no. | Contact no.  | Desired room type (Single/Double/Deluxe/VIP) | Desired room no. |
|----------|---------|---------------|-----------------|-----------------|--------------------------------|------------|-------------|--------|----------------------------------|--------------|--|------------------|
| 6        | Terry   | 3             | 1               | 3184963894689   | 34 Berry Ville Lane, Tristate  | Germany    | German      | Male   | 3792364                          | +49 37303274 | Deluxe                                       | 07-01            |
| 7        | Magdene | 1             | 1               | 329823989892    | Ratcity, 56 Dorman Drive, Yale | Costa Rica | Italian     | Female | 47932695                         | +23 02375892 | Single                                       | 03-07            |

### Test case 4: Update guest

1. Change contact number of the guest with name Lenard (with guestID 1).

### Test case 5: Room service order

1. Create room service order: Guest ID 1 ordered items 9 and 3, indicating in remarks “deliver in 45 mins”
  - o Then, he can view the bill by doing the following:

```
Room Service Ordering System
Enter guest ID: 1
Choose your option
1. Create new order
2. Update order status
3. Print bill
3
Enter order number: 0
Item(s) purchased:
Liqueur ----- $12.0
House Made Potato Saltbush Gnocchi ----- $28.0
Total: $40.0/-
Billed on 20200422 at 234501
```
2. Updated room service order's status as the order gets created, processed etc.

- When there is an update of the order's status by the staff, the following will be printed if the status is updated to processing or delivered.

Status updated from: Creating order  
to: Processing order

Status updated from: Processing order  
to: Delivered

#### **Test case 6: Update room details**

1. Update room status of single room 01-01 and 01-03 to 'Under maintenance'
  - Note: When room statistics report is printed by status shows that they are no longer vacant and is now under "Under Maintenance"
  - Note: When room statistics report is printed by room type, they are not available as they are under maintenance.

#### **Test case 7: Search guest**

1. Search for guest detail. The name of the guest to be searched is Lenard.
  - The guest's details will be printed.

#### **Test case 8: Check-out**

1. Check out made by 3 guests who have discounts. The guestID are 1,2 and 4.
  - Guests will receive a discount on their total bill.
  - Guests will be shown their total bill, including room bill, order bill, tax and service charge.
  - Then the guests will need to make payment for the bill using a credit card number that is previously given or by cash.
  - After successful payment, the room that is assigned to the guests will be unassigned and become "Vacant" again.
2. Check out made by 2 guests who do not have discounts. The guestID are 6 and 7.
  - Same as 1, but there is no discount on their total bill.