Project: Hotel Reservation System

Due Date: April 9th, 2025

Introduction:

Hotel ABC is one of the famous tourist hotels in XYZ city. But this hotel's current reservation is based on a manual system. When guests come in to make a reservation, their details are recorded in a file and then those files are stored in special cabinets. Also, the billing system is manual too.

As the current system is manual based, the management of the hotel must put extra efforts to keep the data secured. Records can easily be destroyed in case of fire, disaster or even possible to be stolen. Additionally, storing files requires extra cabinet space, and searching for a record is difficult, resulting in significant manpower hours. As well as the billing system, the system is manually handled so having an error in calculation is also at high risk. The management is also looking for a way to get the customer feedback after the stay.

Since the pandemic hits now the management is also looking to implement 2 kiosks as well, so the guests can book their room with no contact or interaction with anyone.

We have decided to develop their Reservation system as a computer-based system and hotel can give quick service to the guests.

Objectives:

- Computerized system to replace the manual reservation system.
- Save time of hotel employees and guests.
- Develop a data structure which stores guest's details when they arrive for a reservation.
- Create an easy-to-understand, user-friendly environment.
- The customers can book the room via phone (an authorized person must do it) or in person (using the kiosk).
- The billing system must provide a bill against each reservation made at check out.
- No contact-based kiosk systems.
- Integrate a Guest Feedback System to gather guest feedback on their stay and the booking process.
    - Feedback should be only available once the guest checks out and clears up the bills.

- A reminder to the admin who is clearing the bills should be populated as a reminder to let the guest know that they can use the KIOSK to leave a feedback on their stay.

Functional Requirements:

Using KIOSK or SELF-SERVICE

- Welcome page with appropriate message, small video to guide the customer about how the self-kiosk system works. (choose appropriate welcome message, it has marks)
- Moving forward from the welcome message, your system must take multiple easy-to-explained steps to finish the booking and enjoy the stay.

Some examples of these steps are the following: (You can add more steps to make the system user-friendly)

- The system must ask for the number of people going to check in.
- System must ask for check-in/ check-out dates.
- Based on the information above the system must suggest how many rooms the guests are going to need. (Have a rules and regulations button on the side available all the time)
- The system must not accept more than 2 adult guests per room. (Assuming single room is equipped with a queen bed)
- The system must accept the booking if a single person wants to book a room as well.
- The system must ask the guests' details for room booking registration if all the fields are entered correctly.
    - Proper validation is required and make sure that your form indicates to the user which are required fields.
- The system must store the user/ guest data into an appropriate database.
- The system must ask for confirmation with appropriate message and showing the booking details, estimate price and tax.
- The system must store the booking in the database.
- The system must show the conformation of the room and welcome the guest.
- The KIOSK is not going to generate a bill for the guest, but a reminder is needed once the booking is completed to let the guest know that it can check with the front desk (Admin) for the billing details.

Using phone booking (Admin)

- The system main page must provide a link/ button/ menu option to login the admin. (Menu is more appropriate)
- The system must display a message the login page is for admin only.
- Repeat the same steps as the kiosk system for booking.
  Extra functionalities for the admin only
  - The system must allow the admin to search for guests by their name/ phone number.
  - Upon check-out the system must allow the admin to offer a discount as well. (Every guest must check out via admin; the kiosk system allows the guest to book contactless booking but check out needs to be done via front desk.)
  - Admin must be able to cancel the booking as well.
  - Admin must be able to check out the guest properly once all the bills are settled.
  - Upon confirmation of the check-out, the room must be available for booking again.

Proposed Classes:

These are some proposed sets of classes required to complete the project. You can add more classes, interfaces or abstract classes as needed and add more functionalities to it. All inputs needed to have get and set methods. Your model class must use the proper binding properties with proper types. All information to be stored in the database regarding guests, bills, room information etc.

Guest:

- Properties: GuestID, Name, PhoneNumber, Email, Address, Feedback.
- Methods: GetGuestDetails, SetGuestDetails, ValidateGuestDetails.

Reservation:

- Properties: ReservationID, GuestID, RoomID, CheckInDate, CheckOutDate, NumberOfGuests, Status.
- Methods: CreateReservation, CancelReservation, GetReservationDetails, ConfirmReservation.

RoomType (Enum):

- SINGLE, DOUBLE, DELUX, PENT HOUSE

Room:

- Properties: RoomID, RoomType, NumberOfBeds, Price, Status.
- Methods: GetRoomDetails, SetRoomDetails, CheckRoomAvailability.

Billing:

- Properties: BillID, ReservationID, Amount, Tax, TotalAmount, Discount.
- Methods: GenerateBill, ApplyDiscount, CalculateTotal, PrintBill.

Admin:

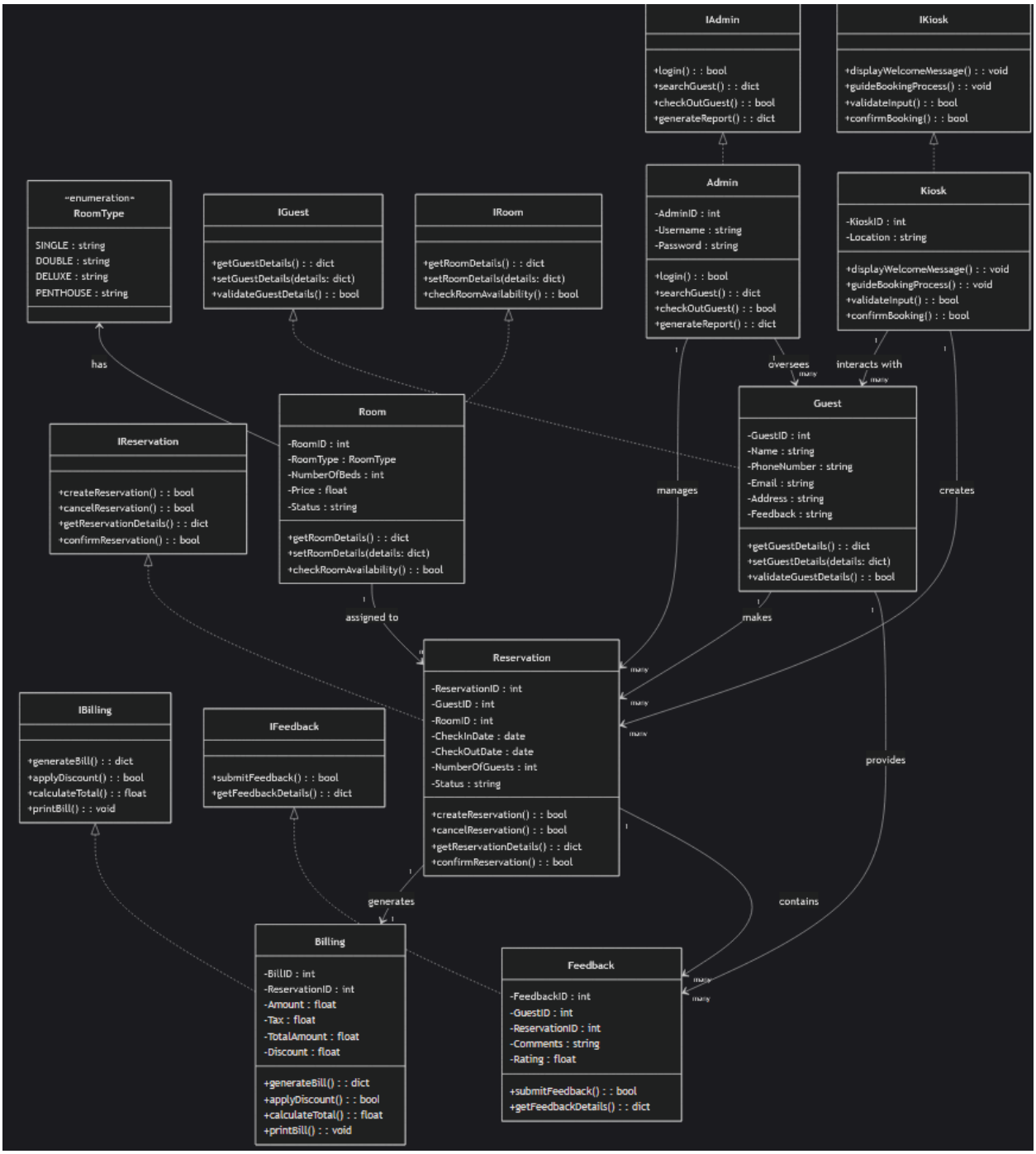
- Properties: AdminID, Username, Password.
- Methods: Login, SearchGuest, CheckOutGuest, GenerateReport.

Feedback:

- Properties: FeedbackID, GuestID, ReservationID, Comments, Rating.
- Methods: SubmitFeedback, GetFeedbackDetails.

Kiosk:

- Properties: KioskID, Location.
- Methods: DisplayWelcomeMessage, GuideBookingProcess, ValidateInput, ConfirmBooking.



Rules

- Single room: Max two people.
- Double room: Max 4 people.
- Deluxe and Pent rooms: Max two people but the prices are higher.

- More than 2 adults less than 5 can have Double room or two single rooms will be offered.
- More than 4 adults will have multiple Double or combination of Double and single rooms.

Logging and Exception Handling Requirements

To ensure proper debugging, tracking, and security in the system, students **must implement Java Logging** to record important events such as exceptions and admin activity.

1. **Exception Logging**
    - All exceptions (e.g., invalid input, database errors, failed reservations) must be logged using the **Java Logger**.
    - The log should include **a timestamp, the type of exception, and a descriptive message** to help identify and resolve issues efficiently.

    Example:

```
Logger logger = Logger.getLogger(HotelReservation.class.getName());
try {
    // Some code that may throw an exception
} catch (Exception e) {
    logger.log(Level.SEVERE, "An error occurred: " + e.getMessage(), e);
}
```

2. **Admin Activity Logging**
    - All admin logins, guest searches, booking modifications, and check-outs must be **recorded in a log file**.
    - This improves security and accountability by tracking who accessed or modified guest records.

    Example log entry:

```
[2025-02-26 14:05:32] INFO: Admin 'admin123' logged in successfully.
[2025-02-26 14:07:12] INFO: Admin 'admin123' modified reservation ID 1045.
```

3. **Log File Setup and Rotation**

   - Students should configure logging to store logs in a **separate log file**.
   - Consider implementing **log rotation** to avoid excessive file size growth.

   Example configuration and rotation (using FileHandler):

```
try {
    FileHandler fileHandler = new FileHandler("system_logs.%g.log", 1024 * 1024, 10, true);
    logger.addHandler(fileHandler);
    SimpleFormatter formatter = new SimpleFormatter();
    fileHandler.setFormatter(formatter);
} catch (IOException e) {
    logger.log(Level.SEVERE, "Failed to initialize logger", e);
}
```

   - "system_logs.%g.log" is a pattern where %g is a placeholder for the generation number of the log file.
   - 1024 * 1024 sets the limit to 1MB for each log file.
   - 10 specifies the maximum number of log files to keep.
   - This setup will create a new log file after the current file reaches 1MB and will maintain up to 10 log files. Anything beyond that will overwrite the oldest log file.

Submission deadlines:

- **March 19th, 2025,** 3%
  - The first deadline is a check point in which everyone must participate it will be a short 3 to 5 minutes meeting in which you will present your **final front end designs**, class diagrams, UML etc. or any other diagram that can

support your project structure, data flow (**Final Screen Shots of all the front ends)**

- Separate submission will be open where you need to submit all UML, front end screen shots, database designs, or any other design diagrams that you have shown in the lab. (Only those submissions will be accepted which are approved during the lab time).
- Attendance is mandatory for this milestone.
- Students can show their design in earlier labs as well.

- **April 9th, 2025,** 12%
  - The second deadline is your final submission in which you will record a video to show the full working project that you have prepared with an explanation of your challenges only faced. The video must be minimum 5 - 7 minutes but you can use more time to explain if you want. There is no deduction for extra minutes of video.
  - Submit your full project.
  - Submit your database scripts (if you are using sql, mysql - must).
  - Reflection must include,
    - Challenges you faced and finding the solutions.
    - Learnings during the project.

Important pointers:

- Save your database scripts in a text file, you need them with your submissions.
- **Every Wednesday lab students can approach and discuss the design of the project, their progress and if there are any issues.**
- **Your whole project must follow the MVC approach with proper packages of their names, classes with proper names following java naming conventions.**
- **The system must have minimum two administration Id's and passwords stored in the database.**
- All the fields must have proper validation, for example if you are taking input as an integer then I mustn't be able to enter string or double values, like no of guests.
- All exceptional handling must be done properly.
- You are allowed to add more functionality or menu items or classes.

Optional Requirement: Implementing a Multithreaded Server for Admin Access

In a real-world hotel reservation system, multiple admins might need to log in and manage bookings at the same time. To simulate this, you must implement a multithreaded server that allows multiple admin sessions to run simultaneously.

What This Means:

- The system should allow at least two admins to log in and use the system at the same time via separate command prompt windows.
- Each admin should be able to search guests, modify reservations, process checkouts, and apply discounts independently.
- The server should handle multiple admin requests without conflicts or crashes.

How to Achieve This?

1. **Use a Multi-Threaded Server Approach**
    - The system should have a server-side application that listens for admin connections.
    - When an admin logs in, the server should create a new thread to handle that admin's session.

    Example:

```
class AdminHandler extends Thread {
    private Socket socket;
    public AdminHandler(Socket socket) {
        this.socket = socket;
    }
    public void run() {
        // Handle admin requests here
    }
}
```

    - This ensures that multiple admins can work independently without affecting each other.

2. **Client-Server Communication**
    - The admin interface should act as a client, sending requests to the server.
    - The server will process requests and respond to each admin individually.
3. **Testing the Multithreaded Functionality**
    - Open two command prompt windows and start an admin session in each.
    - Verify that both admins can log in, search for guests, and process reservations simultaneously.

Good Luck and Happy Codding….