Midterm Paired Task 1.

Object Oriented Analysis and Design

1. **Following the OO workflow as discussed in class,** you are task to design the OO Model of the given problem (use draw.io) of the scenario below:

   Problem Statement. Tiny Hospital keeps information on **patients** and **hospital rooms**. The system assigns each patient a patient ID number. In addition, the patient's name and date of birth are recorded. Some patients are resident patients (they spend at least one night in the hospital) and others are outpatients (they are treated and released). Resident patients are assigned to a room. Each room is identified by a room number. The **Tiny hospital system** also stores the room type (private or semi-private) and room fee. Overtime, each room will have many patients who stay in it. Each resident patient will stay in only one room. The hospital system has features that can view patient information and view whether a room is occupied or not. Both patient and room entities must have features that allows adding, updating and searching of records.

   STEP1. IDENTIFY all the necessary **OBJECT** within the problem domain
   STEP 2. IDENTIFY all the properties and methods/behaviors in the problem statement
   STEP 3. Design the MODEL using a Class Diagram (You may use draw.io to represent the Blueprint of all the class that you need to create)
   STEP 4. Implement the **class using Java code** construct of each interacting entities that you have identified.

Note: *Highlight all the outputs following the example from STEP 1 to STEP 4 as shown in the lecture*

Angeles, Gabriel Elmo L.
BSCS C204

STEP 1: Identify Objects

- Patient (general class)
- ResidentPatient (subclass)
- OutPatient (subclass)
- Room

STEP 2: Properties and Methods

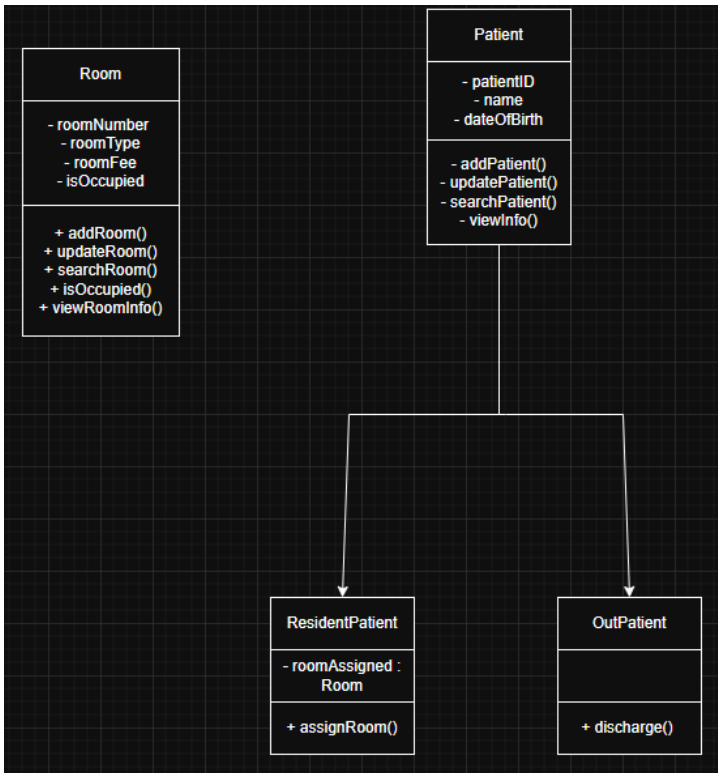
Patient:
- Properties: patientID, name, dateOfBirth
- Methods: addPatient(), updatePatient(), searchPatient(), viewInfo()

ResidentPatient:
- Properties: roomAssigned (Room)
- Methods: assignRoom()

OutPatient: - Methods: dischargePatient()

Room:
- Properties: roomNumber, roomType, roomFee, isOccupied
- Methods: addRoom(), updateRoom(), searchRoom(), isOccupied(), viewRoomInfo()

STEP 3: UML Class Diagram



STEP 4: Java Implementation

```
class Patient {
    String patientID;
    String name;
    String dateOfBirth;

    public Patient(String patientID, String name, String dateOfBirth) {
        this.patientID = patientID;
        this.name = name;
        this.dateOfBirth = dateOfBirth;
    }

    public void addPatient() {
        System.out.println("Patient added: " + name);
    }

    public void updatePatient() {
        System.out.println("Updating patient record for " + name);
    }

    public void searchPatient(String id) {
        if (id.equals(patientID)) {
            System.out.println("Patient found: " + name);
        } else {
            System.out.println("Patient not found.");
        }
    }

    public void viewInfo() {
        System.out.println("Patient ID: " + patientID);
        System.out.println("Name: " + name);
        System.out.println("Date of Birth: " + dateOfBirth);
    }
}

// Subclass for resident patients
class ResidentPatient extends Patient {
    Room roomAssigned;

    public ResidentPatient(String patientID, String name, String dateOfBirth) {
        super(patientID, name, dateOfBirth);
    }
    public void assignRoom(Room room) {
        this.roomAssigned = room;
        room.isOccupied = true;
        System.out.println(name + " assigned to Room " + room.roomNumber);
    }

    @Override
    public void viewInfo() {
        super.viewInfo();
        if (roomAssigned != null) {
            System.out.println("Room Assigned: " + roomAssigned.roomNumber);
        }
    }
}

// Subclass for outpatients
class OutPatient extends Patient {
    public OutPatient(String patientID, String name, String dateOfBirth) {
        super(patientID, name, dateOfBirth);
    }

    public void dischargePatient() {
        System.out.println(name + " has been treated and discharged.");
    }
}

// Room class
class Room {
    String roomNumber;
    String roomType;
    double roomFee;
    boolean isOccupied;

    public Room(String roomNumber, String roomType, double roomFee) {
        this.roomNumber = roomNumber;
        this.roomType = roomType;
        this.roomFee = roomFee;
        this.isOccupied = false;
    }

    public void addRoom() {
        System.out.println("Room added: " + roomNumber);
    }
    public void updateRoom() {
        System.out.println("Room " + roomNumber + " updated.");
    }

    public void searchRoom(String number) {
        if (number.equals(roomNumber)) {
            System.out.println("Room found: " + roomNumber);
        } else {
            System.out.println("Room not found.");
        }
    }

    public void viewRoomInfo() {
        System.out.println("Room Number: " + roomNumber);
        System.out.println("Room Type: " + roomType);
        System.out.println("Room Fee: " + roomFee);
        System.out.println("Occupied: " + (isOccupied ? "Yes" : "No"));
    }
}

// Main class
public class TinyHospitalSystem {
    public static void main(String[] args) {
        // Create a room
        Room room1 = new Room("101", "Private", 1500);

        // Create patients
        ResidentPatient rp = new ResidentPatient("P001", "Alice", "1990-05-12");
        OutPatient op = new OutPatient("P002", "Bob", "1985-10-22");

        // Add records
        room1.addRoom();
        rp.addPatient();
        op.addPatient();

        // Assign room to resident patient
        rp.assignRoom(room1);

        // Show details
        rp.viewInfo();
        op.dischargePatient();
        room1.viewRoomInfo();
    }
}
```