DOCUMENTATION ABOUT SMALL CLINIC MANAGEMENT SYSTEM

OBJECTED-ORIENTED ANALYSIS (OOA) MODEL:

When applying the Object-Oriented Analysis (OOA) model to the small clinic management system, I identified the objects, attributes, methods, and inheritance relationships as follows:

- Objects: include

+ Appointment

+ Patient

+ ChronicPatient

+ Doctor

+ Room

- Attributes:

+ For Class Appointment: date, time, reason, status

+ For Class Patient: name, ID, age, medicalHistory

+ For Class ChronicPatient: inherits all attributes from Patient and extends with conditionType, lastCheckUpDate, frequency

+ For Class Doctor: name, ID, specialty, assignedAppointments

+ For Class Room: roomNumber, department, available

- Methods:

+ For Appointment: displayAppointment, getDate, setStatus

+ For Patient: displayPatientInfo, scheduleAppointment, viewMedicalHistory, updateMedicalHistory

+ For ChronicPatient: inherits all methods from Patient and extends with checkFrequency

+ For Doctor: displayDoctorInfo, viewAssignedAppointments, updateStatus

+ For Room: displayRoomInfo, setAvailability

- Inheritance Relationships: ChronicPatient inherits from Patient.

Class Design Explanation

In the design, I created classes based on real entities of a clinic:

- Appointment: Represents a specific medical appointment with details such as date, time, reason, and status. This is a core data unit as it connects both patients and doctors.
- Patient: Stores the basic information of a patient. The medicalHistory attribute is implemented as a vector<Appointment> to manage all past appointments. Methods include scheduling appointments, viewing history, and updating history.
- ChronicPatient: Inherits from Patient and adds attributes such as conditionType, lastCheckUpDate, and frequency. Inheritance is used here because a chronic patient is still a patient, but with additional needs like periodic checkups and condition tracking. To avoid code duplication, I overrode scheduleAppointment() so that chronic patients are reminded of their follow-ups more frequently.
- Doctor: Manages information about the doctor, including specialty and assigned appointments. Doctors can view their schedule and update the status of appointments.
- Room: Represents clinic rooms. Each room is associated with a department and has an availability status.

Code Walkthrough

Code implementation is divided into several parts:

- Class Appointment:

+ displayAppointment() prints appointment details.

+ setStatus() updates the appointment status (Example: change from Scheduled to Completed).

- Class Patient:

+ displayPatientInfo() prints patient details.

+ scheduleAppointment() allows entering a new appointment and adds it to the patient’s medical history.

+ viewMedicalHistory() displays all past appointments.

+ updateMedicalHistory() removes an appointment from the history.

- Class ChronicPatient:

+ Overrides displayPatientInfo() to include conditionType and lastCheckUpDate.

+ Overrides scheduleAppointment() to also update the chronic condition status and remind about periodic checkups.

+ checkFrequency() shows the regular checkup frequency.

- Class Doctor:

+ displayDoctorInfo() prints doctor information.

+ viewAssignedAppointments() lists all appointments assigned to the doctor.

+ updateStatus() allows updating the status of appointments.

- Class Room:

+ displayRoomInfo() prints room details.

+ setAvailability() updates the availability of the room.

- Main function:

+ A role selection menu is displayed, where the user can choose to log in as Patient, Chronic Patient, or Doctor.

+ Each role has its own menu of available operations.

Test Result

Test case 1:

- Input:

+ For patient1:

1 (Role selection)

2 (Function Selection)

+ For patient2:

1 (Role selection)

9/9/2025

14:00

Check-up

- Output:

Choose your role:

1. Patient

2. Chronic Patient

3. Doctor

Your selection: 1

Name: Nguyen Van An

ID: P001

Age: 25

Choose function:

1. Schedule appointment

2. View medical history

3. Update medical history

4. View room

Your selection: 2

Medical history:

Appointment 1:

Date: 7/9/2025

Time: 10:00

Reason: Fever

Status: Completed

Appointment 2:

Date: 8/9/2025

Time: 09:00

Reason: Flu

Status: Scheduled

Appointment 3:

Date: 14/8/2025

Time: 15:00

Reason: Headache

Status: Completed

Name: Le Thi Binh

ID: P002

Age: 30

Choose function:

1. Schedule appointment

2. View medical history

3. Update medical history

4. View room

Your selection: 1

Input informations below:

Date: 9/9/2025

Time: 14:00

Reason: Check-up

Appointment scheduled successfully!

Medical history:

Appointment 1:

Date: 6/9/2025

Time: 14:00

Reason: Cough

Status: Scheduled

Appointment 2:

Date: 10/9/2025

Time: 11:00

Reason: Fever

Status: Completed

Appointment 3:

Date: 12/9/2025

Time: 16:00

Reason: Back Pain

Status: Scheduled

Appointment 4:

Date: 9/9/2025

Time: 14:00

Reason: Check-up

Status: Scheduled

- Explain about test case 1:

+ First, the role selected is Patient

+ For patient 1:

- The program creates patient Nguyen Van An with 3 existing appointments (initialized in the code).
- The user selects function 2, which is to view the medical history.
- Result: The program iterates through the vector `medicalHistory` and prints out all 3 previous appointments (Fever, Flu, Headache).

+ For patient 2:

- The program creates patient *Le Thi Binh* with 3 existing appointments (Cough, Fever, Back Pain).
- The user selects function 1 to create a new appointment.
- The user inputs the new appointment:
  Date: 9/9/2025
  Time: 14:00
  Reason: Check-up
- The program adds this appointment to the vector `medicalHistory`.
- Then it prints the entire medical history again, now including 4 appointments (the 3 old ones + the newly added appointment).

About Using LLM Models

During the coding process, I used ChatGPT to explore what a real-world clinic usually has and then recreated those elements in the code through classes and methods. In addition, when coding scheduleAppointment function in the ChronicPatient class, after finishing the line that displays the success message for scheduling an appointment and the reminder about regular checkups, I wanted to update the variable lastCheckUpDate with the newly scheduled date. However, I could not access the variable date, so I asked ChatGPT to point out the error so I was able to fix it.