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.

This design keeps the system realistic, extendable, and maintainable while reflecting real-world clinic operations.

Code Walkthrough

The 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.