

Small Clinic Management System

Github: [htung0403/ClinicManagementSystem](https://github.com/htung0403/ClinicManagementSystem)

1. OOA analysis

1.1. Identify objects (nouns):

- **Patient**
- **ChronicPatient** (inherits from Patient)
- **Doctor**
- **Appointment**
- **ClinicManager**

1.2. Identify attributes for each object (descriptive nouns):

- **Patient**: patientId, name, age, address, phoneNumber, medicalHistory.
- **ChronicPatient**: patientId, name, age, address, phoneNumber, medicalHistory, conditionType, lastCheckupDate, nextRequiredCheckupDate, medicationList.
- **Doctor**: doctorId, name, specialty, phoneNumber, appointmentList.
- **Appointment**: appointmentId, patientId, doctorId, date, time, reason, status.
- **ClinicManager**: patients, doctors, appointments.

1.3. Identify methods for each object (verbs):

- **Patient**: scheduleAppointment(), showInfo(), addMedicalHistory().
- **ChronicPatient**: scheduleAppointment(), showInfo(), addMedication(), removeMedication().
- **Doctor**: assignAppointment(), showInfo().
- **Appointment**: updateStatus(), reschedule(), cancel(), complete(), showInfo().
- **ClinicManager**: addRegularPatient(), addChronicPatient(), addDoctor(), scheduleAppointment.

1.4. Identify inheritance relationships, if any.

ChronicPatient inherits from **Patient**

ChronicPatient is the subclass, while **Patient** is the superclass. **ChronicPatient** reuses the attributes and methods of **Patient**, and also adds its own specific attributes and methods tailored for chronic patients.

2. Class design and how inheritance is used and why

2.1. Class design

- **Patient**: Represents a general patient with attributes such as name, patientId, age, address, phoneNumber, and medicalHistory. The class provides methods to schedule appointments, update medical history, and display patient information.
- **ChronicPatient**: Inherits from **Patient** and represents patients with chronic conditions. This class adds attributes like conditionType, lastCheckupDate, and a list of medications. It overrides methods such as scheduleAppointment() and showInfo() to provide specialized behavior for chronic patients.
- **Doctor**: Represents a doctor with attributes like name, ID, specialty, phone number, and a list of assigned appointments. Methods allow assigning appointments and displaying doctor information.
- **Appointment**: Represents an appointment, linking a patient and a doctor by their IDs, and storing details such as date, time, reason, and status. Methods allow updating status, rescheduling, and displaying appointment information.
- **ClinicManager**: Manages collections of patients, doctors, and appointments. It provides methods to add new entities, schedule appointments, and display lists.

2.2. How inheritance is used and why?

ChronicPatient inherits from **Patient**, reusing source code and adding chronic-specific ones (conditionType, lastCheckupDate, medicationList). This avoids code duplication, simplifies maintenance, increase software development efficiency and models the real-world fact that a chronic patient is a specialized type of patient.

3. Code walkthrough

- **Patient**: this class holds basic details (ID, name, age, address, phoneNumber) and a vector for medical history. A static counter generates unique IDs. Core methods handle appointment scheduling, history updates, and info display. Virtual methods allow subclasses to adjust these behaviors.

- **ChronicPatient:** this class extends Patient with fields for condition type, last check-up date, and medications. It overrides scheduling and info display to reflect chronic care needs, and adds functions to manage medication and follow-up dates.
- **Doctor:** The Doctor class manages doctor info (ID, name, specialty, phone) and tracks assigned appointments. It supports adding appointments and showing doctor details. IDs are auto-generated.
- **Appointment:** Appointments link a patient and doctor, storing date, time, reason, and status. They can be rescheduled, canceled, or completed. Each appointment also has a unique auto-incremented ID.
- **ClinicManager:** this class manages the system, storing patients, doctors, and appointments in vectors. It provides functions to add entities, schedule visits, and list all records.

4. Test results

4.1. Patient and ChronicPatient

```
PS D:\OOP-2025\ClinicManagementSystem\tests> ./test_Patient
Patient Info:
--- Patient Information ---
ID: 1
Name: Vo Hoang Tung
Age: 22
Address: 01 Vo Van Ngan
Phone: 0901 340 403

Chronic Patient Info:
--- Patient Information ---
ID: 2
Name: Le Thi B
Age: 40
Address: 10 Tran Hung Dao
Phone: 0909 888 777
Chronic Condition: Diabetes
Last Checkup Date:
Medications:

[Patient] Appointment has been scheduled for Vo Hoang Tung
[ChronicPatient] Regular checkup appointment scheduled for Le Thi B

Patient Medical History:
- Visited for flu symptoms
- Annual checkup

Chronic Patient Medical History:
- Diabetes checkup
- Blood sugar test

Updated Patient Info:
--- Patient Information ---
ID: 1
Name: Vo Hoang Tung
Age: 23
Address: 01 Le Van Viet
Phone: 0123 456 789

Updated Chronic Patient Info:
--- Patient Information ---
ID: 2
Name: Le Thi B
Age: 41
Address: 20 Nguyen Trai
Phone: 0999 888 777
Chronic Condition: Diabetes
Last Checkup Date: 10/09/2025
Medications: Metformin Insulin
```

Explain:

- Shows create Patient and ChronicPatient objects with auto-incrementing patientId and update info.
- Displays medical history and medication management.

4.2. Doctor

```
PS D:\OOP-2025\ClinicManagementSystem\tests> ./test_Doctor
Doctor 1 Info:
--- Doctor Information ---
ID: 101
Name: Dr. Hehe
Specialty: Cardiology
Phone: 0123 456 789

Doctor 2 Info:
--- Doctor Information ---
ID: 102
Name: Dr. Huynh Xuan Phung
Specialty: OOP
Phone: 0987 654 321

Doctor 1 Appointments:
- 101
- 102

Doctor 2 Appointments:
- 201

Updated Doctor 1 Info:
--- Doctor Information ---
ID: 101
Name: Dr. Hehe
Specialty: General Medicine
Phone: 0111 222 333
```

Explain:

- Shows create Doctor objects with auto-incrementing doctorId, appointment assignment and update doctor info.

4.3. Appointment

```
PS D:\OOP-2025\ClinicManagementSystem\tests> ./test_Appointment
Appointment 1 Info:
--- Appointment Information ---
ID: 10001
Patient ID: 1
Doctor ID: 101
Date: 2025-09-15
Time: 09:00
Reason: General Checkup
Status: Scheduled

Appointment 2 Info:
--- Appointment Information ---
ID: 10002
Patient ID: 2
Doctor ID: 102
Date: 2025-09-16
Time: 10:30
Reason: Follow-up
Status: Scheduled

Rescheduled Appointment 1 Info:
--- Appointment Information ---
ID: 10001
Patient ID: 1
Doctor ID: 101
Date: 2025-09-17
Time: 11:00
Reason: General Checkup
Status: Scheduled

Updated Appointment 1 Status:
--- Appointment Information ---
ID: 10001
Patient ID: 1
Doctor ID: 101
Date: 2025-09-17
Time: 11:00
Reason: General Checkup
Status: Completed

Updated Appointment 2 Status:
--- Appointment Information ---
ID: 10002
Patient ID: 2
Doctor ID: 102
Date: 2025-09-16
Time: 10:30
Reason: Follow-up
Status: Canceled
```

```
Updated Appointment 2 Reason:  
--- Appointment Information ---  
ID: 10002  
Patient ID: 2  
Doctor ID: 102  
Date: 2025-09-16  
Time: 10:30  
Reason: Consultation  
Status: Canceled
```

Explain:

- Shows Appointment creation, rescheduling, status updates, and reason changes.

4.4. ClinicManager

```
PS D:\OOP-2025\ClinicManagementSystem\tests> ./test_ClinicManager
--- Patient List ---
--- Patient Information ---
ID: 1
Name: Vo Hoang Tung
Age: 22
Address: 01 Vo Van Ngan
Phone: 0901 340 403
--- Patient Information ---
ID: 2
Name: Le Thi B
Age: 40
Address: 10 Tran Hung Dao
Phone: 0909 888 777
Chronic Condition: Diabetes
Last Checkup Date:
Medications:

--- Doctor List ---
--- Doctor Information ---
ID: 101
Name: Dr. Hehe
Specialty: Cardiology
Phone: 0123 456 789
--- Doctor Information ---
ID: 102
Name: Dr. Huynh Xuan Phung
Specialty: OOP
Phone: 0987 654 321

--- Appointment List ---
--- Appointment Information ---
ID: 10001
Patient ID: 1
Doctor ID: 101
Date: 15/09/2025
Time: 09:00
Reason: General Checkup
Status: Scheduled
--- Appointment Information ---
ID: 10002
Patient ID: 2
Doctor ID: 102
Date: 16/09/2025
Time: 10:30
Reason: Blood Pressure Follow-up
Status: Scheduled
```

Explain:

- Shows adding patients/doctors, scheduling appointments, and displaying all entities.

5. LLM Usage

I used Github Copilot to brainstorm methods for suggest attributes for each class in a simple Small Clinic Management.

Suggest attributes for each class

Real-World Example Scenario

You are tasked with building a basic Small Clinic Management System for a local family clinic. The system should handle:

Patients: Each patient has a name, ID, age, and medical history (a list of past visits). Patients can schedule appointments or update their history. **Appointments:** Appointments include date, time, reason, and status (scheduled, completed, or canceled). They are linked to patients and doctors.

Specialized Patients: The clinic treats regular patients and chronic patients (e.g., those with ongoing conditions like diabetes). Chronic patients inherit from the general Patient class but include additional attributes like condition type and last check-up date. They override methods, such as requiring more frequent appointments (e.g., every 3 months vs. as-needed for regular patients).

Doctors: Doctors have a name, ID, specialty, and a list of assigned appointments. They can view and update appointment statuses.

Operations: The system should allow creating patients and doctors, scheduling/canceling appointments, and displaying information about patients, doctors, and appointments.

.... (using LLM to get more requirements)

This scenario draws from real-world clinic operations, where inheritance can be used to extend base classes (e.g., Patient as base, ChronicPatient as derived) to handle variations in patient care without redundant code.

Here are suggested attributes for each class based on your scenario:

Patient

- **name:** string
- **id:** string or int
- **age:** int
- **medicalHistory:** list of past visits (could be a list of strings or Appointment objects)
- **appointments:** list of Appointment objects

ChronicPatient (inherits from Patient)

- **conditionType:** string (e.g., "Diabetes", "Hypertension")
- **lastCheckupDate:** date
- (inherits all Patient attributes)
- (may override scheduling logic for frequent appointments)

Doctor

- **name:** string
- **id:** string or int
- **specialty:** string
- **appointments:** list of Appointment objects

Appointment

- **date:** date
- **time:** time
- **reason:** string
- **status:** enum/string ("scheduled", "completed", "canceled")
- **patient:** reference to Patient object
- **doctor:** reference to Doctor object

These attributes cover the requirements for patient management, appointment scheduling, chronic care, and doctor operations. Let me know if you want method suggestions or class skeletons.

And used prompt “suggest one more class for system” for one more class needed.

```

suggest one more class for system

A useful additional class for your system would be ClinicManager.

ClinicManager
• Manages collections of patients, doctors, and appointments.
• Handles operations like creating patients/doctors, scheduling/canceling appointments, and displaying information.
• Attributes:
  ◦ patients: list of Patient objects
  ◦ doctors: list of Doctor objects
  ◦ appointments: list of Appointment objects

This class acts as the central controller for the clinic, coordinating interactions between patients, doctors, and appointments.
```

In addition, I used LLM model to write test cases for the program and to fix errors that appeared during execution.