Small Clinic Management System – Documentation

Name: Ly Tran Gia Khang

ID: 24110098

---

1. Object-Oriented Analysis (OOA)

Following the 4-step OOA model, the system has the following objects:

1.1 Objects (Nouns)

- Patient
- ChronicPatient (inherits from Patient)
- Doctor
- Appointment
- Prescription
- Staff (abstract class)
- SecurityGuard (inherits from Staff)
- Clinic

1.2 Attributes (Descriptive Nouns)

- Patient: name, ID, age, medicalHistory
- ChronicPatient: (inherits Patient) + conditionType, lastCheckup
- Doctor: name, ID, specialty
- Appointment: appID, date, time, reason, status, doctor, patient
- Prescription: presID, date, patient, doctor, medicines
- Staff: name, ID
- SecurityGuard: (inherits Staff) + logs
- Clinic: list of patients, doctors, appointments, prescriptions, staff

1.3 Methods (Verbs)

- Patient: displayInfo(), addHistory(), removeHistory(), showHistory(), virtual scheduleAppointment()
- ChronicPatient: override scheduleAppointment()
- Doctor: displayInfo(), getters/setters
- Appointment: cancel(), complete(), displayInfo()
- Prescription: addMedicine(), displayInfo()
- Staff: displayInfo(), getRole() (pure virtual)
- SecurityGuard: addLogs(), showLogs(), override displayInfo(), getRole()
- Clinic: addPatient(), addDoctor(), addAppointment(), addPrescription(), addStaff(), display lists, choosePatient(), chooseDoctor(), chooseAppointment(), chooseStaff(), scheduleConflict()

1.4 Inheritance Relationships

- ChronicPatient : public Patient
- Inherits all attributes and methods from Patient.
- Overrides scheduleAppointment() to enforce frequent check-ups and add notes in the history.
- SecurityGuard : public Staff
- Inherits all attributes and methods from Staff.

2. Class Design Explanation

The system is designed around five main classes and extra 2 classes:

- Doctor: stores doctor information, includes getters/setters and displayInfo().
- Patient: manages patient data and medical history. Includes a virtual method scheduleAppointment().

- ChronicPatient: extends Patient with chronic condition attributes. Overrides scheduleAppointment() to highlight condition-specific appointments.
- Appointment: represents a link between Doctor and Patient, containing date, time, reason, and status (Scheduled, Cancelled, Completed).
- Prescription: stores prescriptions created by doctors for patients.
- Staff (abstract class): represents general staff members.
- SecurityGuard: extends Staff, records activity logs and has a distinct role.
- Clinic: central manager of patients, doctors, and appointments. Provides methods for adding, displaying, and selecting objects.

Why inheritance?

- Patient → ChronicPatient: demonstrates inheritance and polymorphism via overridden scheduleAppointment().
- Staff → SecurityGuard: demonstrates abstraction and polymorphism by forcing subclasses to define getRole().

3. Code Walkthrough

Example 1 – Override in ChronicPatient:

```
void scheduleAppointment(string appID, string date, string time,
string reason) override{

    cout << "Chronic patient requires regular check-up...!" <<
endl;

    cout << "Appointment set on " << date << " at " << time

        << " for " << reason << " (Condition: " << conditionType
<< ")." << endl;
```

```
      addHistory(date + " - " + time + ": " + reason + "
[Chronic]");

}
```

-> Chronic patients log their appointments differently, marking them with [Chronic].

Example 2 – Clinic manages all objects:

```
void addPatient(Patient *_patient){patients.push_back(_patient);}

void addDoctor(Doctor *_doctor){doctors.push_back(_doctor);}

void addAppointment(Appointment
*_appointment){appointments.push_back(_appointment);}
```

-> The Clinic class uses vectors to store patients, doctors, and appointments, providing centralized management.

4. Testing & Output

4.1 Test Cases

The main() function demonstrates the following:

- Registering regular and chronic patients.
- Registering doctors.
- Scheduling appointments.
- Canceling appointments.
- Completing appointments.
- Viewing patient medical history.
- Displaying lists of doctors, patients, and appointments.
- Create prescription
- Displaying list of prescription
- Show patient's prescription
- Register a security guard
- Show staff list
- Show guard logs

4.2 Sample Output (excerpt)

```
========== Clinic Manager ==========
1. Register the patient
2. Register the doctor
= = = = = = = = = = = = = = = = = = = =
3. Schedule the appointment
4. Cancel the appointment
5. Attend the appointment
= = = = = = = = = = = = = = = = = = = =
6. Check the medical history
7. Show the Doctors' list
8. Show the Patients' list
9. Show the Appointments' list
= = = = = = = = = = = = = = = = = = = =
10. Create Prescription
11. Show Prescriptions' list
12. Show Patient's prescriptions
= = = = = = = = = = = = = = = = = = = =
13. Register a Security Guard
14. Show Staff list
15. Show Guard logs
= = = = = = = = = = = = = = = = = = = =
0. Exit the Clinic Manager
===================================
Choose your choice: _
```

```
Choose your choice: 3
Enter the date: 09-09-2025
Enter the time: 14:00
Enter the reason: General checking
Choose the patient will attend the appointment:

========== Patients List ==========
#1. Patient's Information:
Name: Nguyen Van A
ID: BN001
Age: 25

#2. Patient's Information:
Name: Tran Van B
ID: BN002
Age: 28

#3. Patient's Information:
Name: Le Van C
ID: BN003
Age: 55

Enter the number: 1
```

```
Choose the doctor will examine the patient:

========== Doctors List ==========
#1. Doctor's Information:
Name: Le Van X
ID: DR001
Specialty: Cardiology

#2. Doctor's Information:
Name: Tran Thi Y
ID: DR002
Specialty: Chiropractic

Enter the number: 1
Patient Nguyen Van A scheduled appointment on 09-09-2025 at 14:00 for General checking.
Appointment scheduled successfully...!
```

-> Output proves the system works for both regular and chronic patients.

5. Use of LLM (ChatGPT)

During development, I used ChatGPT as a support tool, not as a replacement.

How I used it:

- Brainstorming possible methods for the Appointment class.
- Asking for clarification on how to override a virtual method in C++.
- Getting sample ideas for menu-driven programs.

Example Prompt:

"Suggest methods for an Appointment class in a small clinic management system."

Response:

ChatGPT suggested methods like cancel(), complete(), displayInfo(). I adapted these to my final design.

Note: The final code and report are my own work. The LLM was used only for brainstorming and debugging support.

6. Conclusion

The Small Clinic Management System successfully demonstrates key OOP concepts:

- Encapsulation: private attributes with getters/setters.

- Inheritance & Polymorphism: ChronicPatient inherits from Patient and overrides scheduleAppointment; SecurityGuard inherits from Staff and overrides getRole().
- Abstraction: Staff as an abstract class forces subclasses to implement their role.

-> The program compiles and runs correctly, supports all required operations, and even extends functionality with prescriptions and staff management. This project is a practical application of OOP for real-world clinic management.