## **1. OOA Analysis for Small Clinic Management System (based on your code)**

## **1. Objects (Nouns)**

From your code, the main objects are:

* **Appointments**
* **Patients** (general patients)
* **ChronicPatient** (chronic patients)
* **AcutePatient** (acute patients)
* **Doctor**

## **2. Attributes (Descriptive Nouns)**

* **Appointments**:
  + date
  + time
  + IDDoctor (doctor’s ID)
  + namePatient (patient’s name)
  + reason (appointment reason)
  + status (scheduled, completed, canceled)
* **Patients**:
  + name
  + ID (patient ID)
  + age
  + medicalHistory (vector<string> of past illnesses)
* **ChronicPatient** (inherits from Patients):
  + same attributes as Patients (represents chronic patients)
* **AcutePatient** (inherits from Patients):
  + symptom (acute symptom)
* **Doctor**:
  + name
  + ID (doctor ID)
  + specialty
  + apm (doctor’s personal appointment list, vector<Appointments>)

## **3. Methods (Verbs)**

* **Appointments**:
  + displayInfo() → show appointment details
  + getIDDoctor() → return doctor ID
  + setStatus() → update appointment status
* **Patients**:
  + displayInfo() → show patient details
  + getName(), getAge(), getID() → return patient info
  + updateMH() → update medical history
  + scheduleAppointment() → create a new appointment
  + showMH() → print out patient’s medical history
* **ChronicPatient**:
  + displayInfo() → override to display chronic patient info
* **AcutePatient**:
  + displayInfo() → override to display acute patient info (with symptom)
* **Doctor**:
  + loadAppointment() → reload appointment list from global list
  + showAppointments() → display doctor’s appointments
  + updateStatus() → update appointment status
  + displayInfo() → show doctor details

## **4. Inheritance Relationships**

* **Patients** is the base class.
  + **ChronicPatient** inherits from Patients.
  + **AcutePatient** inherits from Patients.

## **2. Explanation of Class Design**

The system is structured around **three main domains**: patients, doctors, and appointments.

* **Appointments**:  
  Represents a scheduled meeting between a patient and a doctor. Each appointment stores date, time, doctor ID, patient name, reason, and status.
* **Patients (Base Class)**:  
  Holds general information for any patient (name, ID, age, medical history). It also provides methods for scheduling appointments and updating medical history.
* **ChronicPatient & AcutePatient (Inheritance)**:
  + ChronicPatient extends Patients but doesn’t add new attributes in the current design — it specializes the type of patient (for long-term conditions).
  + AcutePatient extends Patients by adding the **symptom** attribute, representing temporary or urgent medical conditions.
  + **Why inheritance?**  
    Inheritance allows you to create specialized types of patients that share the same basic data and methods (ID, age, history) but differ in behavior or attributes. This avoids code duplication and makes the system more extensible.
* **Doctor**:  
  Stores doctor details (name, ID, specialty) and manages their appointments. Doctors can display and update appointment statuses.

## **3. Code Walkthrough**

Let’s go through the **key parts of your code**:

1. **Appointments Class**
   * Manages appointment details.
   * Key methods:
     + displayInfo() → print appointment information.
     + setStatus() → update appointment status.
2. **Patients Class**
   * Stores patient info (name, ID, age, medicalHistory).
   * Key methods:
     + displayInfo() → print patient details.
     + updateMH() → add new medical history entries.
     + scheduleAppointment() → create new appointment and add to global list.
     + showMH() → display past medical records.
3. **ChronicPatient & AcutePatient**
   * Both inherit from Patients.
   * ChronicPatient::displayInfo() overrides to indicate it’s a chronic patient.
   * AcutePatient::displayInfo() overrides and adds symptom output.
4. **Doctor Class**
   * Manages doctor info and their appointment list.
   * Key methods:
     + showAppointments() → reloads and displays appointments for this doctor.
     + updateStatus() → update appointment status by selecting an order number.
5. **Main Function (Testing)**
   * Creates different patient types (normal, chronic, acute).
   * Updates medical histories.
   * Schedules appointments.
   * Creates doctors and shows their appointments.
   * Demonstrates updating appointment statuses.

## **4. Test Results**

When you run the program, the sample output looks like this (simplified for clarity):

Patient:

Name: Nhiep

ID: BN001

Age: 20

Updated medical history for Nhiep: Đau đầu

History: Sốt, Nôn, Đau đầu

--------------------------

Chronic Patient:

Name: Phat

ID: BN002

Age: 45

Updated medical history for Phat: Huyết áp cao

History: Tiểu đường, Huyết áp cao

--------------------------

Acute Patient:

Name: Trang

ID: BN003

Age: 30

Symptom: Sốt cao

Updated medical history for Trang: Viêm họng

History: Viêm họng

--------------------------

Scheduled appointment for Nhiep on 12/09/2025 at 09:00

Scheduled appointment for Phat on 13/09/2025 at 10:00

Scheduled appointment for Trang on 14/09/2025 at 14:00

--------------------------

Doctor: Nguyen Van A

ID: BS01

Specialty: Nội khoa

Appointments:

## 1. Patient Nhiep on 12/09/2025 at 09:00, reason: Khám định kỳ

## 2. Patient Phat on 13/09/2025 at 10:00, reason: Khám tiểu đường

--------------------------

Doctor: Le Thi B

ID: BS02

Specialty: Nhi khoa

Appointments:

## 1. Patient Trang on 14/09/2025 at 14:00, reason: Khám sốt

--------------------------

>>> Doctor 1 updates appointment status:

Enter appointment number: 1

Updated status: hoàn thành

...