

Project on

OBJECT ORIENTED PROGRAMMING

Project Title

**MedCare Healthcare Management System**

Submitted by

Name: Pratapkumar Bhajantri

USN: 01FE21BCS299

1.Introduction:

In today’s world, efficient and reliable healthcare systems are essential for quality medical services. The Healthcare Management System (HMS) for MedCare Pvt. Ltd. is designed to streamline hospital operations through integrated modules including Patient Management, Doctor Scheduling, Pharmacy, Laboratory Services, Billing, Emergency Services, Rooms Management, and Feedback Handling. With intuitive interfaces and a user-friendly design, it ensures smooth navigation for all users. The system features robust error handling and exception management, maintaining data integrity during faults. A built-in Help Menu offers guidance and support. This HMS improves efficiency, reduces manual effort, and enhances patient care through digital transformation.

2.Problem Definition:

### Develop a comprehensive Healthcare Management System (HMS) for MedCare Pvt. Ltd., encompassing modules for Hospital Administration, Patient Management, Doctor Scheduling, Pharmacy Inventory, Laboratory Services, Billing and Emergency Services, Rooms Management, and Feedback and.,Error Exception ,Help Menu Each module should feature intuitive interfaces and , facilitating easy navigation and usage. The system should prioritize robust error handling, ensuring graceful recovery from common issues while maintaining data integrity and user convenience.

### 3. List of objects identified

* + - * LABORATORY
      * PHARMACY
      * EMERGENCY SERVICE
      * ROOM
      * HOSPITAL
      * PATIENT
      * BILLING
      * DOCTOR
      * STAFF
      * NURSE
      * EXCEPTION
      * DEPARTMENT

### 4.Concepts of Oops Used:

### • Classes And Objects

### • Constructors

### • Destructors

### • Inheritance

### • Polymorphism

### • Function Overloading

### • Function Overriding

### • Virtual Function

### • Friend Function

### • Abstract Base Class

### • Templates

### • Exception Handling

# 5.Description of each class

##### MyException

* + A custom exception class derived from runtime\_error.
  + Used to throw specific error messages in the program.
  + Contains a constructor to initialize the error message.
  + Simplifies exception handling with a custom message.

##### Staff

* + An abstract base class representing hospital staff.
  + Contains protected members for id, name, role, department, and contactInfo.
  + Declares a pure virtual method calculateSalary for derived classes to implement.
  + Provides methods to update contact info and assign staff to a department.
  + Represents a hospital with staff, appointments, and room bookings.
  + Contains members for name, address, staffMembers, appointments, and bookedRooms.
  + Provides methods to add staff, make/cancel appointments, and book/cancel room bookings.
  + Includes a utility method for case-insensitive string comparison.
  + Simulates a successful payment transaction.
  + Used for handling billing operations in the hospital.

##### Doctor

* + Inherits from Staff and represents doctors.
  + Adds a specialization member to the base class attributes.
  + Implements calculateSalary to return a fixed salary.
  + Provides a method to get the doctor's specialization.

##### Nurse

* + Inherits from Staff and represents nurses.
  + Adds a duty member to specify the nurse's duty.
  + Implements calculateSalary to return a fixed salary.
  + Utilizes the base class constructor to initialize common attributes.

##### Hospital

* + Represents a hospital with staff, appointments, and room bookings.
  + Contains members for name, address, staffMembers, appointments, and bookedRooms.
  + Provides methods to add staff, make/cancel appointments, and book/cancel room bookings.
  + Includes a utility method for case-insensitive string comparison.

##### Appointment

* + Represents an appointment with a patient.
  + Contains members for id, patientName, doctorId, date, and time.
  + Provides methods to get details of the appointment.
  + Used to manage patient appointments in the hospital.

##### Billing

* + Handles payment processing for the hospital.
  + Contains a method to process payment with a given amount.
  + Simulates a successful payment transaction.
  + Used for handling billing operations in the hospital.

##### Pharmacy

* + Manages medication orders for the hospital.
  + Contains a method to order medication by specifying the name and quantity.
  + Simulates ordering medication from the pharmacy.
  + Used to handle pharmacy operations in the hospital.

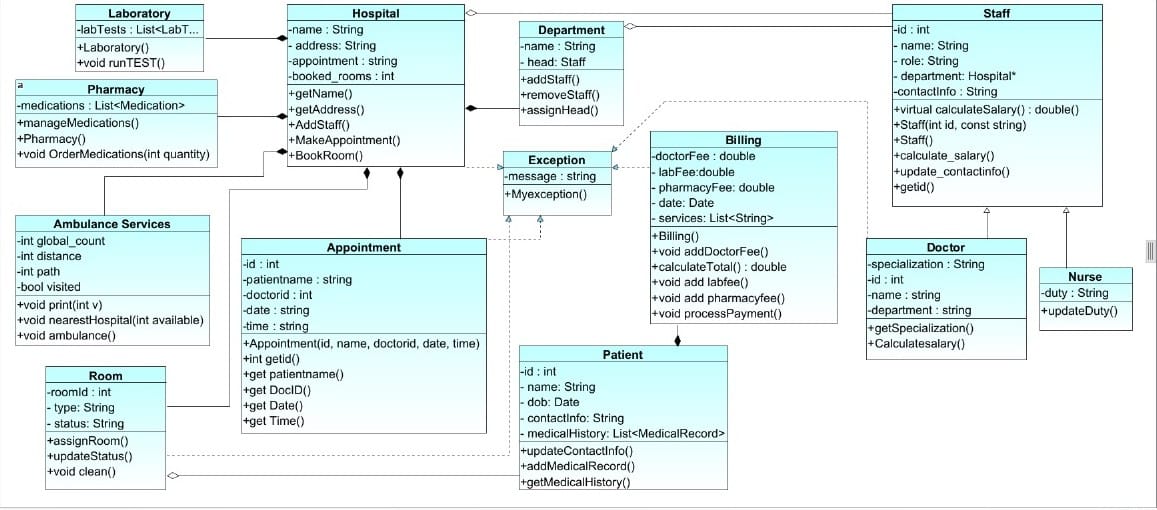
#### 9.Laboratory

* + - Handles laboratory test operations for the hospital.
    - Contains a method to run specified tests.
    - Simulates running laboratory tests.
    - Used to manage laboratory operations in the hospital.

#### Room

* + Represents a hospital room.
  + Contains a method to clean the room.
  + Simulates the cleaning operation of a room.
  + Used to manage room-related tasks in the hospital.

6.Class Diagram



## 7.Design pattern

###### Abstract Factory Pattern

* + Used in creating Staff objects (Doctor and Nurse).
  + The Staff class serves as an abstract base class with a factory method (calculateSalary) that derived classes (Doctor and Nurse) implement.
  + This pattern provides an interface for creating families of related or dependent objects without specifying their concrete classes.

###### Singleton Pattern

* + Not explicitly used in the given code, but it could be implied for Hospital to ensure that only one instance of a hospital

exists throughout the application.

###### Factory Method Pattern

* + Used when creating specific types of staff members like Doctor and Nurse.
  + Each specific staff class implements the calculateSalary method defined in the Staff base class.
  + This pattern allows a class to delegate instantiation to subclasses.

1. **Strategy Pattern**
   * Implemented through the calculateSalary method in the Staff class.
   * Each subclass (Doctor, Nurse) has its own strategy for calculating the salary.
   * This pattern defines a family of algorithms and makes them interchangeable.
2. **Exception Handling Pattern**
   * The MyException class represents a custom exception handling pattern.
   * It encapsulates exception details and provides a specific error message for handling errors.
3. **Composite Pattern**
   * Used in the Hospital class to manage a collection of Staff objects.
   * The staffMembers vector holds various Staff objects (e.g., Doctor and Nurse).
   * This pattern allows individual objects and compositions of objects to be treated uniformly.
4. **Command Pattern**
   * Implemented in the Hospital class for actions like makeAppointment, cancelAppointment, bookRoom, and cancelRoomBooking.
   * Each method represents a command that encapsulates a request as an object.
5. **Iterator Pattern**
   * Used implicitly in the searchDoctorBySpecialization method.
   * Iterates through the staffMembers vector to find doctors with the specified specialization.
   * This pattern provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation.

##### Facade Pattern

* + The Hospital class acts as a facade to provide a simplified interface for managing staff, appointments, and room bookings.
  + This pattern hides the complexities of the subsystem (various operations) and provides a unified interface.

##### Observer Pattern

* + Not explicitly used, but it could be implemented to observe changes in appointments or room bookings.
  + For instance, updating a UI or notifying staff members when an appointment is made or cancelled.

8.Conclusion:

In conclusion, the Healthcare Management System (HMS) for MedCare effectively integrates key hospital functions into a single, modular, and user-friendly platform. Utilizing core object-oriented programming concepts such as inheritance, polymorphism, abstract classes, templates, and exception handling, the system ensures scalability, maintainability, and robust performance. Each class, from Patient and Staff to Billing, Pharmacy, Laboratory, and Emergency Services, plays a crucial role in replicating real-world healthcare operations. With strong error handling through the custom MyException class and a dedicated Help Menu, the HMS guarantees reliability, data integrity, and ease of use, enhancing the overall healthcare experience.