UNIVARSITY OF SCIENCE & TECHNOLOGY CHITTAGONG

**Faculty of Science Engineering & Technology (FSET)**

**Department of Computer Science & Engineering (CSE)**

---

# Project Title: Hospital Management System

**Submitted by**

Name: **S.M. ALI AKBAR KHALED**

Roll: **24070103**

Section: **"A"** Semester: 2nd

Date: 28. 05. 2025

Dept.: **Computer Science & Engineering (CSE)**

**Submitted to**

**DEBARATA MALLICK**

Lecturer

CSE, FSET, USTC

## Contents

[illegible]

22.05.2025

*List of figures: -*

# Project Title: Hospital Management System (HMS)

## ***1. Motivation***

Seeing the system work and handle real input gives me confidence to enhance this project further. I plan to implement database support, create more interactive UI designs, and convert this into a complete desktop solution

## ***2. Project Description***

### 2.1 Purpose:

To develop a simple, user-friendly hospital management system that allows basic management of patient records, appointments, and payments using a graphical interface.

### 2.2 Problem Statement:

Manual handling of hospital records is prone to human error, inefficiency, and data loss. There is a need for an automated system to manage patient information and appointments effectively.

### 2.3 Relevance of OOP Concepts:

Using Object-Oriented Programming enables structured, reusable, and maintainable code that aligns well with real-world entities such as **patients**, **doctors**, and **appointments**.

## ***3. Project Goals & Functionalities***

### 3.1 Main Goals:

- Implement an intuitive GUI for managing hospital operations.
- Ensure patient data is stored temporarily and easily manipulated.
- Enable payment and appointment booking features.

### 3.2 Key Functionalities Implemented:

- Secure login using password
- Add, view, search, and delete patients
- Payment options (bKash/Nagad)
- Appointment booking with file logging
- Encapsulation of patient data
- Simple GUI using Java Swing

## ***4. Technology Stack***

| Component | Tool/Language |
|---|---|
| **Programming Language** | Java (OOP) |
| **GUI Framework** | Java Swing |
| **File Handling** | Java IO (FileWriter) |
| **Data Persistence** | In-Memory ArrayList (for patients) |
| **Optional DB** | Not used in current phase |

## *5. Use of OOP Principles*

| Principle | Usage |
|---|---|
| **Encapsulation** | Private fields in Individual and HospitalPatient classes with getters |
| **Inheritance** | HospitalPatient inherits from abstract class Individual |
| **Polymorphism** | Method overriding with display() and toString() in subclass |
| **Abstraction** | Abstract Individual class hides implementation details |

## *6. Project Phases & Timeline*

| Phase | Time Estimate |
|---|---|
| **Requirements & Planning** | 2 weeks |
| **GUI Design (Swing)** | 1 weeks |
| **Core Logic (Patient, Payment)** | 3 days |
| **File Handling (Appointment)** | 1 week |
| **Testing & Debugging** | 2 days |
| **Documentation & Finalization** | 1 day |

## *7. Expected Output*

- A **working hospital management GUI application**
- Ability to manage patient data efficiently
- Easy and quick **appointment booking**
- Simulated **payment interface**
- Simple and educational demonstration of **OOP principles in Java**

## *8. Implementation section*

### 8.1 Class diagram

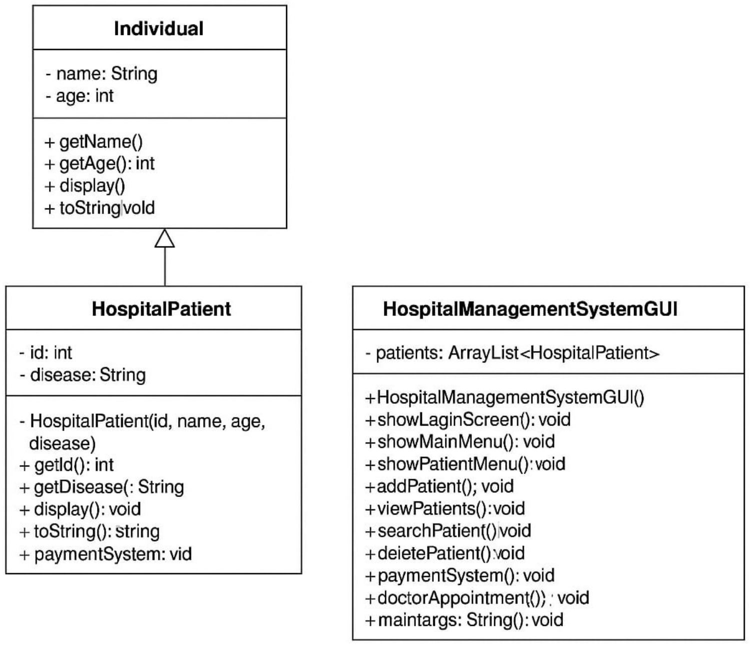
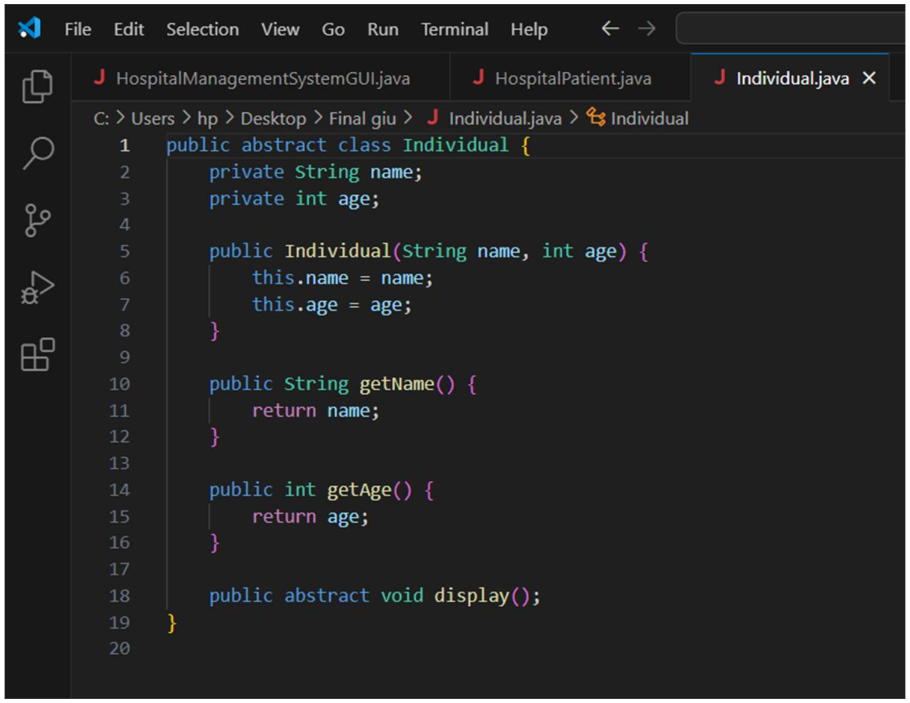


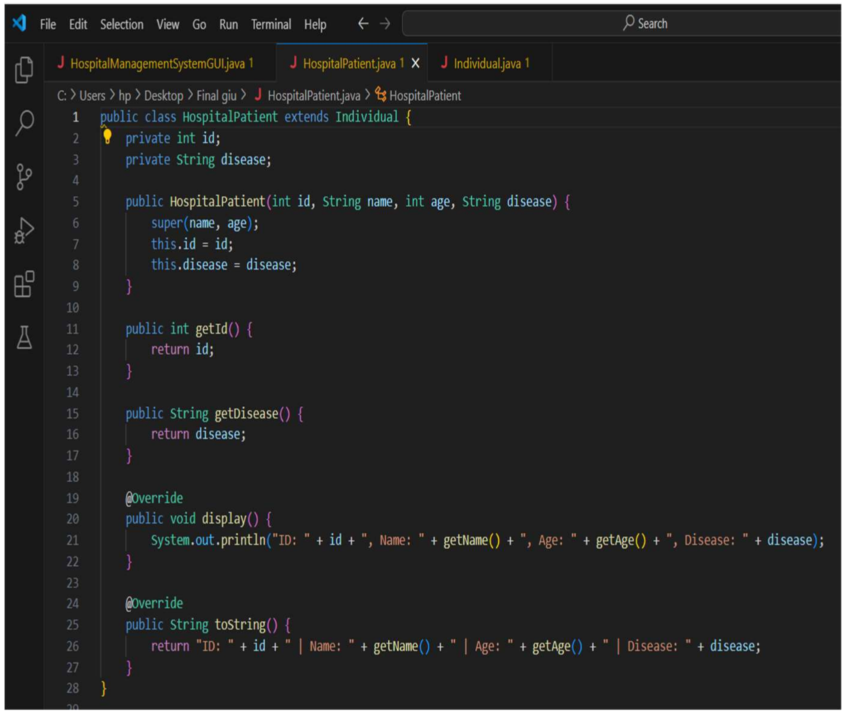
**Figure 01 – Class diagram**

## 8.2 Parent class and child class



```java
public abstract class Individual {
    private String name;
    private int age;

    public Individual(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public abstract void display();
}
```

**Figure 2- : Parent class named – "Individual"**

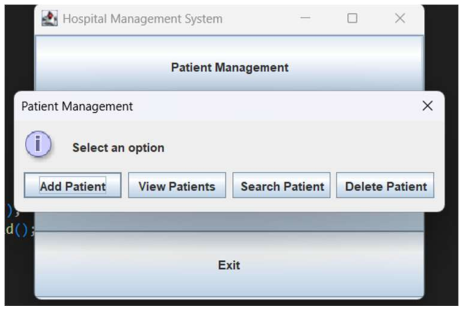


```java
public class HospitalPatient extends Individual {
    private int id;
    private String disease;

    public HospitalPatient(int id, String name, int age, String disease) {
        super(name, age);
        this.id = id;
        this.disease = disease;
    }

    public int getId() {
        return id;
    }

    public String getDisease() {
        return disease;
    }

    @Override
    public void display() {
        System.out.println("ID: " + id + ", Name: " + getName() + ", Age: " + getAge() + ", Disease: " + disease);
    }

    @Override
    public String toString() {
        return "ID: " + id + " | Name: " + getName() + " | Age: " + getAge() + " | Disease: " + disease;
    }
}
```

**Figure 3- : child class named – "HospitalPatient"**

## 8.3 Current Progress Screenshots:

Fig 4- log in page

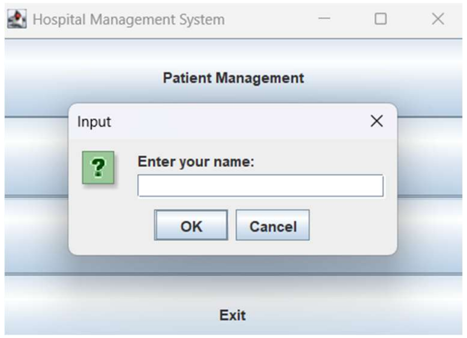


Hospital Management System

Patient Management

Select an option

Fig 6- Patient management system

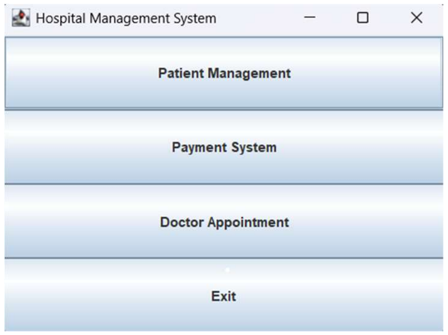


Hospital Management System

Input

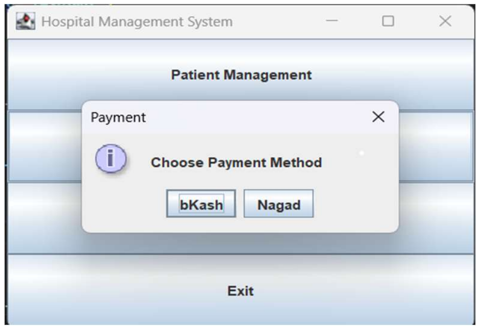
Enter your name:

Fig 8- Doctor appointment



Hospital Management System

Fig 5– Main home page



Hospital Management System

Payment

Choose Payment Method

Fig 7– Payment method

## *9. Remaining Features:*

- Database integration (optional)
- Export patient list to CSV
- Add doctor scheduling module
- GUI enhancements with better layout
- Role-based access (Admin vs User)

## *10. Problems Faced:*

- Initial trouble aligning layout with GridLayout
- Handling input exceptions in GUI
- Understanding abstract class implementation in GUI context

## *11. New Learnings:*

- Event-driven programming with Java Swing
- Use of JOptionPane for input/output dialogs
- File I/O using FileWriter
- Building object-oriented hierarchy for real-world systems

## ***12. Project Summary & Impact***

This system showcases how **object-oriented programming** can effectively solve real-life problems such as managing hospital operations. It improves operational efficiency, user experience, and **encourages structured software development** practices.

This project lays the foundation for:

- Extending into database integration
- User authentication systems
- Reporting and analytics modules
- Deploying as a desktop or web-based application

## ***13. References***

- Java Official Documentation: https://docs.oracle.com/javase/8/docs/
- Oracle Swing Tutorial: https://docs.oracle.com/javase/tutorial/uiswing/
- File I/O in Java: https://www.geeksforgeeks.org/file-handling-in-java/
- Object-Oriented Principles: https://www.tutorialspoint.com/java/java_object_classes.htm
- Overall basics of java - Anisul Islam
- Graphical user interface (GUI) lerning - LoveExtendsCode
- Additional help - Bro code
- Programming with JAVA by – Balagurusamy (5th edition)

.

## *14.Appendix*

### A. Technology Stack

| Component | Details |
|---|---|
| **Programming Language** | Java |
| **GUI Framework** | Java Swing |
| **File Handling** | FileWriter for appointment storage |
| **Development IDE** | Any Java-supporting IDE (e.g., IntelliJ, Eclipse, NetBeans) |
| **OS Compatibility** | Cross-platform (Windows, macOS, Linux) |

### B. Class Overview

#### *1. HospitalManagementSystemGUI*

- **Purpose**: Main GUI class handling user interactions and navigation.
- **Key Methods**:
    - showLoginScreen() – Displays password-protected login screen.
    - showMainMenu() – Displays main menu with navigation buttons.
    - showPatientMenu() – Handles CRUD operations on patient records.
    - paymentSystem() – Simulates payment via mobile services.
    - doctorAppointment() – Books and stores doctor appointments in a file.

#### *2. HospitalPatient* (extends Individual)

- **Purpose**: Model class representing a patient.
- **Attributes**:
    - id: Unique patient ID
    - disease: Current ailment/disease of the patient
- **Methods**:
    - getId(), getDisease() – Accessors
    - display() – Displays patient data in console

    - toString() – Returns formatted patient information

### *3. Individual* (abstract)

- **Purpose**: Abstract base class for entities with name and age.
- **Attributes**:
    - name: Name of the individual
    - age: Age of the individual
- **Method**:
    - display() – Abstract method to be implemented by subclasses

## C. Functional Features

| Feature | Description |
|---|---|
| **Login Security** | Password validation required to access the system (default: 24070103) |
| **Patient Management** | Add, view, search, and delete patient records dynamically |
| **Appointment Booking** | Stores booking information in appointments.txt |
| **Payment Processing** | Simulated payment via mobile wallets (bKash, Nagad) |
| **Data Display** | Uses JOptionPane and JTextArea for formatted display of records |

## D. Sample Data Flow

1. **Login**: User inputs password → If correct → Show main menu
2. **Patient Management**:
    - Add: Collects ID, name, age, and disease → Adds to ArrayList
    - View: Displays all patients
    - Search: Finds by ID
    - Delete: Removes by ID
3. **Appointment**: Takes name, doctor, date → Appends to file
4. **Payment**: User selects method → Enters amount → Simulated confirmation

## E. Limitations and Future Improvements

| Current Limitation | Suggested Improvement |
|---|---|
| **No persistent patient storage** | Integrate file or database backend (e.g., SQLite) |
| **No input validation (e.g., name/disease empty)** | Add form validation and error prompts |
| **Plain-text password storage** | Implement encryption/hashing for login credentials |
| **Limited doctor/patient detail scope** | Add more comprehensive health record and scheduling system |

## F. Files and Structure

| File Name | Description |
|---|---|
| **HospitalManagementSystemGUI.java** | Main application class with GUI |
| **HospitalPatient.java** | Data model for a patient |
| **Individual.java** | Abstract base class for person-like data |
| **appointments.txt** | File storing booked appointments |