Hospital Management System (Mini Project)

Project Overview

The **Hospital Management System** is a mini project developed using Java to provide practical exposure to Object-Oriented Programming (OOP) concepts and data structures, specifically the dynamic usage of ArrayList. This system allows users to perform CRUD (Create, Read, Update, Delete) operations on patient records via a simple console-based menu-driven interface. The project simulates a small-scale hospital management scenario where patient details such as ID, Name, Age, Gender, and Disease can be efficiently managed.

The primary goal of this project is to demonstrate the effective implementation of OOP principles such as encapsulation, abstraction, and modularity, while also focusing on code reusability and data management. This project helps beginners gain hands-on experience with designing classes, handling user inputs, and managing collections dynamically.

Key Features

- **Add Patient Details:** Enter new patient information and store it dynamically.
- **View Patient Details:** Display all stored patient records in a readable format.
- **Update Patient Details:** Modify existing patient information by referencing their unique ID.
- **Delete Patient Details:** Remove patient records using their ID.
- **Dynamic Storage:** Use of ArrayList allows the system to handle any number of patient records without fixed size limitations.
- **User-Friendly Menu:** Easy-to-navigate text-based menu for interacting with the system.

Technologies Used

- **Java Programming Language:** Core language used to develop the entire project.
- **Object-Oriented Programming (OOP):** Concepts like classes, objects, encapsulation, and methods are extensively used.
- **Data Structures:**
    - **ArrayList:** To dynamically store patient objects, enabling flexible addition and removal of records.
- **Java Scanner Class:** For taking user input from the console.

- **IDE (Optional):** Any Java IDE such as Eclipse, IntelliJ IDEA, or NetBeans can be used to develop and run the project.

Key Skills Developed

- **Object-Oriented Design:** Creating classes and using encapsulation to protect data.
- **CRUD Operations:** Implementing Create, Read, Update, and Delete functionalities.
- **Dynamic Data Handling:** Managing data using Java's dynamic ArrayList collection.
- **User Input Handling:** Efficient use of Java Scanner class for interactive console input.
- **Code Modularity and Reusability:** Designing methods to separate different functionalities and avoid code duplication.
- **Basic Exception Handling Awareness:** Handling input buffers and user choices gracefully.
- **Problem-Solving:** Understanding and implementing system requirements logically.

Use Case / Practical Application

This project simulates a basic hospital patient record management system which can be extended and adapted for real-world hospital or clinic management. Hospitals often need efficient systems to keep track of patients' demographic details and medical conditions. Although this mini project is console-based, the concepts can be scaled to graphical user interfaces or web applications with database integration for better functionality.

Some practical uses include:

- Quickly adding and managing patient data during hospital admissions.
- Updating patient information during treatment.
- Maintaining historical records for patients.
- Deleting records once patients are discharged or records are obsolete.

Conclusion

The **Hospital Management System** mini project successfully demonstrates the application of core Java programming skills and OOP principles in a real-world-inspired problem. By implementing dynamic storage and CRUD operations, the project offers a foundational understanding of how to manage data efficiently. It highlights the importance of writing reusable and modular code, which is crucial for scalable software development.

This project lays the groundwork for more advanced hospital management systems incorporating databases, user authentication, report generation, and other complex features. As a fresher, working on such a project enhances programming skills, logical thinking, and prepares one for professional software development challenges.

Source Code :

```java
import java.util.ArrayList;

import java.util.Scanner;


// Patient class representing patient details

class Patient {

    private int id;

    private String name;

    private int age;

    private String gender;

    private String disease;


    public Patient(int id, String name, int age, String gender, String disease) {

        this.id = id;

        this.name = name;

        this.age = age;

        this.gender = gender;

        this.disease = disease;

    }


    // Getters and Setters
    public int getId() { return id; }
    public void setName(String name) { this.name = name; }
    public String getName() { return name; }
    public void setAge(int age) { this.age = age; }
    public int getAge() { return age; }
    public void setGender(String gender) { this.gender = gender; }
    public String getGender() { return gender; }
    public void setDisease(String disease) { this.disease = disease; }
    public String getDisease() { return disease; }

    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Age: " + age + ", Gender: " + gender +
", Disease: " + disease;
    }
}

// Hospital Management System Class
public class HospitalManagementSystem {
    private static ArrayList<Patient> patientList = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    // Method to add patient
    private static void addPatient() {
        System.out.print("Enter Patient ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();  // consume newline
        System.out.print("Enter Patient Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Patient Age: ");
        int age = scanner.nextInt();
        scanner.nextLine();  // consume newline
        System.out.print("Enter Patient Gender: ");
        String gender = scanner.nextLine();
        System.out.print("Enter Disease: ");
        String disease = scanner.nextLine();

        Patient patient = new Patient(id, name, age, gender, disease);
        patientList.add(patient);
        System.out.println("Patient added successfully!\n");
    }

    // Method to view all patients
    private static void viewPatients() {
        if (patientList.isEmpty()) {
            System.out.println("No patient records found.\n");
        } else {
            System.out.println("Patient Records:");
            for (Patient p : patientList) {
                System.out.println(p);
            }
            System.out.println();
        }
    }

    // Method to update patient details by ID
    private static void updatePatient() {
        System.out.print("Enter Patient ID to update: ");
```

```java
int id = scanner.nextInt();
scanner.nextLine();  // consume newline

boolean found = false;
for (Patient p : patientList) {
    if (p.getId() == id) {
        found = true;
        System.out.print("Enter new Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter new Age: ");
        int age = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter new Gender: ");
        String gender = scanner.nextLine();
        System.out.print("Enter new Disease: ");
        String disease = scanner.nextLine();

        p.setName(name);
        p.setAge(age);
        p.setGender(gender);
        p.setDisease(disease);

        System.out.println("Patient details updated successfully!\n");
        break;
    }
}
if (!found) {
    System.out.println("Patient with ID " + id + " not found.\n");
}
```

```
}

// Method to delete patient by ID
private static void deletePatient() {
    System.out.print("Enter Patient ID to delete: ");
    int id = scanner.nextInt();
    scanner.nextLine();  // consume newline

    boolean removed = patientList.removeIf(p -> p.getId() == id);
    if (removed) {
        System.out.println("Patient record deleted successfully!\n");
    } else {
        System.out.println("Patient with ID " + id + " not found.\n");
    }
}

// Main menu
public static void main(String[] args) {
    int choice;
    do {
        System.out.println("---- Hospital Management System ----");
        System.out.println("1. Add Patient Details");
        System.out.println("2. View Patient Details");
        System.out.println("3. Update Patient Details");
        System.out.println("4. Delete Patient Details");
        System.out.println("5. Exit");
        System.out.print("Enter your choice: ");
        choice = scanner.nextInt();
        scanner.nextLine(); // consume newline
```

```
            switch (choice) {
                case 1:
                    addPatient();
                    break;
                case 2:
                    viewPatients();
                    break;
                case 3:
                    updatePatient();
                    break;
                case 4:
                    deletePatient();
                    break;
                case 5:
                    System.out.println("Exiting the system. Thank you!");
                    break;
                default:
                    System.out.println("Invalid choice! Please try again.\n");
            }
        } while (choice != 5);
    }
}
```

Output

```
---- Hospital Management System ----
1. Add Patient Details
2. View Patient Details
3. Update Patient Details
4. Delete Patient Details
```

```
5. Exit
Enter your choice: 1
Enter Patient ID: 101
Enter Patient Name: John Doe
Enter Patient Age: 30
Enter Patient Gender: Male
Enter Disease: Flu
Patient added successfully!

---- Hospital Management System ----
1. Add Patient Details
2. View Patient Details
3. Update Patient Details
4. Delete Patient Details
5. Exit
Enter your choice: 2
Patient Records:
ID: 101, Name: John Doe, Age: 30, Gender: Male, Disease: Flu

---- Hospital Management System ----
1. Add Patient Details
2. View Patient Details
3. Update Patient Details
4. Delete Patient Details
5. Exit
Enter your choice: 3
Enter Patient ID to update: 101
Enter new Name: John Smith
Enter new Age: 31
Enter new Gender: Male
Enter new Disease: Cold
Patient details updated successfully!

---- Hospital Management System ----
1. Add Patient Details
2. View Patient Details
3. Update Patient Details
4. Delete Patient Details
5. Exit
Enter your choice: 2
Patient Records:
ID: 101, Name: John Smith, Age: 31, Gender: Male, Disease: Cold

---- Hospital Management System ----
1. Add Patient Details
2. View Patient Details
3. Update Patient Details
4. Delete Patient Details
5. Exit
Enter your choice: 4
Enter Patient ID to delete: 101
Patient record deleted successfully!

---- Hospital Management System ----
1. Add Patient Details
2. View Patient Details
3. Update Patient Details
4. Delete Patient Details
```

```
5. Exit
Enter your choice: 2
No patient records found.

---- Hospital Management System ----
1. Add Patient Details
2. View Patient Details
3. Update Patient Details
4. Delete Patient Details
5. Exit
Enter your choice: 5
Exiting the system. Thank you!
```