****

**Course No.: CSE 2104**

**Course Title: Object Oriented Programming Lab**

**Submitted to**

**Shakib Mahmud Dipto**

**Department of Computer Science and Engineering**

**Submitted by:**

**Meharab Hossen (212014025)**

**Hospital Management system**

The hospital management system is a popular and useful area to show off programming skills and ideas. Taking care of patients, doctors, and meetings is part of it. Object-oriented programming can be used to model real-world things and how they interact with each other. In a real hospital management system, people sign up, make appointments with doctors, see a list of their appointments, and get information about their doctors. Doctors can make changes to schedules and see a list of them. The goal of this method is to make running a hospital more efficient and to improve interactions between patients and doctors. The idea is just a basic plan for making a hospital management system. To work in the real world, it will need to be changed and improved in many ways.

**Setting up a class:**

Person Class: Both doctors and patients use this class as their base class. It includes common attributes like name, gender, and phone number, and provides methods for creating and displaying person details. The Person class gives rise to the Patient class. Adds attributes like ins (insurance), Pid (Patient ID), and keeps an array of patient objects. Provides a way for creating patient details.

**Doctor Class**

Also inherits from the Person class. Adds attributes like splz (specialization), fee (consultation fees), and keeps an array of doctor objects. Provides a way forcreating doctor details.

**Visit Class**: Represents a visit with a doctor.   
Contains attributes for doctor, patient, date, time, and an array to keep appointments. Provides ways for presenting appointments, showing doctor-specific appointments, showing patient-specific appointments, cancelling appointments, and rescheduling appointments. Hospital Class: Contains methods for the menu-driven processes of the system. Handles the flow between doctor and patient interfaces. Offers menus for registration, appointment booking, showing appointments, and more.

**Code Flow:** The program starts by asking the user to select their designation (Patient or Doctor).   
Depending on the designation chosen, the program enters either the patient or doctor portal:   
Patient Portal: Allows patients to register, make appointments, check their appointments, read doctor details, and see available doctors.

**Doctor Portal**: Allows doctors to register, view planned appointments for the day, get a list of doctors, cancel appointments, and reschedule appointments.   
In both portals, users can navigate the menu options using a menu-driven method. The program leads users through registration, appointment booking, appointment viewing, and more.   
The Appointment class handles the booking, viewing, and management of meetings between patients and doctors.   
Data for patients and doctors are stored in arrays within their respective classes.   
  
**Flow of the Code:**1. The app starts by asking the user for their designation (patient or doctor).   
2. The program joins the respective portal (patient or doctor).   
3. In each portal, users can choose from various menu-driven choices to perform actions such as registration, appointment booking, appointment viewing, or getting information.   
4. The program uses a set of classes to model patients, doctors, and meetings, allowing for data storage and management.   
5. Both the portals have a bridge in between so that they can switch in between and work properly.   
6. Error handling is introduced with try-catch blocks to handle exceptions gracefully.

**Source Code**

import java.io.\*;

import java.util.\*;

import java.util.Calendar;

class staff

{

String sid, sname, desg, sex;

int salary;

void new\_staff()

{

Scanner input = new Scanner(System.in);

System.out.print("id:-");

sid = input.nextLine();

System.out.print("name:-");

sname = input.nextLine();

System.out.print("desigination:-");

desg = input.nextLine();

System.out.print("sex:-");

sex = input.nextLine();

System.out.print("salary:-");

salary = input.nextInt();

}

void staff\_info()

{

System.out.println(sid + "\t" + sname + "\t" + sex + "\t" + salary);

}

}

class doctor

{

String did, dname, specilist, appoint, doc\_qual;

int droom;

void new\_doctor()

{

Scanner input = new Scanner(System.in);

System.out.print("id:-");

did = input.nextLine();

System.out.print("name:-");

dname = input.nextLine();

System.out.print("specilization:-");

specilist = input.nextLine();

System.out.print("work time:-");

appoint = input.nextLine();

System.out.print("qualification:-");

doc\_qual = input.nextLine();

System.out.print("room no.:-");

droom = input.nextInt();

}

void doctor\_info()

{

System.out.println(did + "\t" + dname + " \t" + specilist + " \t" + appoint + " \t" + doc\_qual + " \t" + droom);

}

}

class patient

{

String pid, pname, disease, sex, admit\_status;

int age;

void new\_patient()

{

Scanner input = new Scanner(System.in);

System.out.print("id:-");

pid = input.nextLine();

System.out.print("name:-");

pname = input.nextLine();

System.out.print("disease:-");

disease = input.nextLine();

System.out.print("sex:-");

sex = input.nextLine();

System.out.print("admit\_status:-");

admit\_status = input.nextLine();

System.out.print("age:-");

age = input.nextInt();

}

void patient\_info()

{

System.out.println(pid + "\t" + pname + " \t" + disease + " \t" + sex + " \t" + admit\_status + "\t" + age);

}

}

class medical

{

String med\_name, med\_comp, exp\_date;

int med\_cost, count;

void new\_medi()

{

Scanner input = new Scanner(System.in);

System.out.print("name:-");

med\_name = input.nextLine();

System.out.print("comp:-");

med\_comp = input.nextLine();

System.out.print("exp\_date:-");

exp\_date = input.nextLine();

System.out.print("cost:-");

med\_cost = input.nextInt();

System.out.print("no of unit:-");

count = input.nextInt();

}

void find\_medi()

{

System.out.println(med\_name + " \t" + med\_comp + " \t" + exp\_date + " \t" + med\_cost);

}

}

class lab

{

String fecility;

int lab\_cost;

void new\_feci()

{

Scanner input = new Scanner(System.in);

System.out.print("fecility:-");

fecility = input.nextLine();

System.out.print("cost:-");

lab\_cost = input.nextInt();

}

void feci\_list()

{

System.out.println(fecility + "\t\t" + lab\_cost);

}

}

class fecility

{

String fec\_name;

void add\_feci()

{

Scanner input = new Scanner(System.in);

System.out.print("fecility:-");

fec\_name = input.nextLine();

}

void show\_feci()

{

System.out.println(fec\_name);

}

}

public class HospitalManagement

{

public static void main(String args[])

{

String months[] = {

"Jan",

"Feb",

"Mar",

"Apr",

"May",

"Jun",

"Jul",

"Aug",

"Sep",

"Oct",

"Nov",

"Dec"

};

Calendar calendar = Calendar.getInstance();

int count1 = 4, count2 = 4, count3 = 4, count4 = 4, count5 = 4, count6 = 4;

System.out.println("Welcome to Hospital and Health Care Management System Project in Java ");

System.out.print("Date: " + months[calendar.get(Calendar.MONTH)] + " " + calendar.get(Calendar.DATE) + " " + calendar.get(Calendar.YEAR));

System.out.println("\t\t\t\t\t\tTime: " + calendar.get(Calendar.HOUR) + ":" + calendar.get(Calendar.MINUTE) + ":" + calendar.get(Calendar.SECOND));

doctor[] d = new doctor[25];

patient[] p = new patient[100];

lab[] l = new lab[20];

fecility[] f = new fecility[20];

medical[] m = new medical[100];

staff[] s = new staff[100];

int i;

for (i = 0; i < 25; i++)

d[i] = new doctor();

for (i = 0; i < 100; i++)

p[i] = new patient();

for (i = 0; i < 20; i++)

l[i] = new lab();

for (i = 0; i < 20; i++)

f[i] = new fecility();

for (i = 0; i < 100; i++)

m[i] = new medical();

for (i = 0; i < 100; i++)

s[i] = new staff();

d[0].did = "21";

d[0].dname = "Dr.Beenish";

d[0].specilist = "ENT";

d[0].appoint = "5-11AM";

d[0].doc\_qual = "MBBS,MD";

d[0].droom = 17;

d[1].did = "32";

d[1].dname = "Dr.Sara";

d[1].specilist = "Physician";

d[1].appoint = "10-3AM";

d[1].doc\_qual = "MBBS,MD";

d[1].droom = 45;

d[2].did = "17";

d[2].dname = "Dr.Sehresh";

d[2].specilist = "Surgeon";

d[2].appoint = "8-2AM";

d[2].doc\_qual = "BDM";

d[2].droom = 8;

d[3].did = "33";

d[3].dname = "Dr.Rayan";

d[3].specilist = "Artho";

d[3].appoint = "10-4PM";

d[3].doc\_qual = "MBBS,MS";

d[3].droom = 40;

p[0].pid = "12";

p[0].pname = "Paras";

p[0].disease = "Cancer";

p[0].sex = "Male";

p[0].admit\_status = "y";

p[0].age = 30;

p[1].pid = "13";

p[1].pname = "Sumit";

p[1].disease = "Cold";

p[1].sex = "Male";

p[1].admit\_status = "y";

p[1].age = 23;

p[2].pid = "14";

p[2].pname = "Ali";

p[2].disease = "Maleriya";

p[2].sex = "Male";

p[2].admit\_status = "y";

p[2].age = 45;

p[3].pid = "15";

p[3].pname = "Roshan";

p[3].disease = "Diabetes";

p[3].sex = "Male";

p[3].admit\_status = "y";

p[3].age = 25;

m[0].med\_name = "Ciprofloxacin";

m[0].med\_comp = "Bayer";

m[0].exp\_date = "9-5-16";

m[0].med\_cost = 55;

m[0].count = 8;

m[1].med\_name = "COLOFAC";

m[1].med\_comp = " Abbot";

m[1].exp\_date = "4-4-15";

m[1].med\_cost = 500;

m[1].count = 5;

m[2].med\_name = "Panadol";

m[2].med\_comp = " GSK PLC";

m[2].exp\_date = "12-7-17";

m[2].med\_cost = 50;

m[2].count = 56;

m[3].med\_name = "Movax";

m[3].med\_comp = " SAMI pvt";

m[3].exp\_date = "12-4-12";

m[3].med\_cost = 1100;

m[3].count = 100;

l[0].fecility = "X-ray ";

l[0].lab\_cost = 800;

l[1].fecility = "CT Scan ";

l[1].lab\_cost = 12000;

l[2].fecility = "OR Scan ";

l[2].lab\_cost = 5000;

l[3].fecility = "Blood Bank";

l[3].lab\_cost = 100;

f[0].fec\_name = "Ambulance";

f[1].fec\_name = "Admit Facility ";

f[2].fec\_name = "Canteen";

f[3].fec\_name = "Emergency";

s[0].sid = "22";

s[0].sname = "Prakash";

s[0].desg = "Worker";

s[0].sex = "Male";

s[0].salary = 5000;

s[1].sid = "23";

s[1].sname = "Komal";

s[1].desg = "Nurse";

s[1].sex = "Female";

s[1].salary = 2000;

s[2].sid = "24";

s[2].sname = "Raju";

s[2].desg = "Worker";

s[2].sex = "Male";

s[2].salary = 500;

s[3].sid = "25";

s[3].sname = "Rani";

s[3].desg = "Nurse";

s[3].sex = "Female";

s[3].salary = 20000;

Scanner input = new Scanner(System.in);

int choice, j, c1, status = 1, s1 = 1, s2 = 1, s3 = 1, s4 = 1, s5 = 1, s6 = 1;

while (status == 1)

{

System.out.println("\n MAIN MENU");

System.out.println("1.Doctors 2. Patients 3.Medicines 4.Laboratories 5. Facilities 6. Staff ");

choice = input.nextInt();

switch (choice)

{

case 1:

{

System.out.println(" DOCTOR SECTION");

System.out.println("--------------------------------------------------------------------------------");

s1 = 1;

while (s1 == 1)

{

System.out.println("1.Add New Entry\n2.Existing Doctors List");

c1 = input.nextInt();

switch (c1)

{

case 1:

{

d[count1].new\_doctor();count1++;

break;

}

case 2:

{

System.out.println("--------------------------------------------------------------------------------");

System.out.println("id \t Name\t Specilist \t Timing \t Qualification \t Room No.");

System.out.println("--------------------------------------------------------------------------------");

for (j = 0; j < count1; j++)

{

d[j].doctor\_info();

}

break;

}

}

System.out.println("\nReturn to Back Press 1 and for Main Menu Press 0");

s1 = input.nextInt();

}

break;

}

case 2:

{

System.out.println("--------------------------------------------------------------------------------");

System.out.println(" PATIENT SECTION");

System.out.println("--------------------------------------------------------------------------------");

s2 = 1;

while (s2 == 1)

{

System.out.println("1.Add New Entry\n2.Existing Patients List");

c1 = input.nextInt();

switch (c1)

{

case 1:

{

p[count2].new\_patient();count2++;

break;

}

case 2:

{

System.out.println("--------------------------------------------------------------------------------");

System.out.println("id \t Name \t Disease \t Gender \t Admit Status \t Age");

System.out.println("--------------------------------------------------------------------------------");

for (j = 0; j < count2; j++) {

p[j].patient\_info();

}

break;

}

}

System.out.println("\nReturn to Back Press 1 and for Main Menu Press 0");

s2 = input.nextInt();

}

break;

}

case 3:

{

s3 = 1;

System.out.println("--------------------------------------------------------------------------------");

System.out.println(" MEDICINE SECTION");

System.out.println("--------------------------------------------------------------------------------");

while (s3 == 1)

{

System.out.println("1.Add New Entry\n2. Existing Medicines List");

c1 = input.nextInt();

switch (c1)

{

case 1:

{

m[count3].new\_medi();count3++;

break;

}

case 2:

{

System.out.println("--------------------------------------------------------------------------------");

System.out.println("Name \t Company \t Expiry Date \t Cost");

System.out.println("--------------------------------------------------------------------------------");

for (j = 0; j < count3; j++) {

m[j].find\_medi();

}

break;

}

}

System.out.println("\nReturn to Back Press 1 and for Main Menu Press 0");

s3 = input.nextInt();

}

break;

}

case 4:

{

s4 = 1;

System.out.println("--------------------------------------------------------------------------------");

System.out.println(" LABORATORY SECTION");

System.out.println("--------------------------------------------------------------------------------");

while (s4 == 1)

{

System.out.println("1.Add New Entry \n2.Existing Laboratories List");

c1 = input.nextInt();

switch (c1)

{

case 1:

{

l[count4].new\_feci();count4++;

break;

}

case 2:

{

System.out.println("--------------------------------------------------------------------------------");

System.out.println("Fecilities\t\t Cost");

System.out.println("--------------------------------------------------------------------------------");

for (j = 0; j < count4; j++) {

l[j].feci\_list();

}

break;

}

}

System.out.println("\nReturn to Back Press 1 and for Main Menu Press 0");

s4 = input.nextInt();

}

break;

}

case 5:

{

s5 = 1;

System.out.println("--------------------------------------------------------------------------------");

System.out.println("HOSPITAL FACILITY SECTION");

System.out.println("--------------------------------------------------------------------------------");

while (s5 == 1)

{

System.out.println("1.Add New Facility\n2.Existing Fecilities List");

c1 = input.nextInt();

switch (c1)

{

case 1:

{

f[count5].add\_feci();count5++;

break;

}

case 2:

{

System.out.println("--------------------------------------------------------------------------------");

System.out.println("Hospital Facility are:");

System.out.println("--------------------------------------------------------------------------------");

for (j = 0; j < count5; j++) {

f[j].show\_feci();

}

break;

}

}

System.out.println("\nReturn to Back Press 1 and for Main Menu Press 0");

s5 = input.nextInt();

}

break;

}

case 6:

{

s6 = 1;

System.out.println("--------------------------------------------------------------------------------");

System.out.println(" STAFF SECTION");

System.out.println("--------------------------------------------------------------------------------");

while (s6 == 1)

{

String a = "nurse", b = "worker", c = "security";

System.out.println("1.Add New Entry \n2.Existing Nurses List\n3.Existing Workers List \n4.Existing Security List");

c1 = input.nextInt();

switch (c1)

{

case 1:

{

s[count6].new\_staff();count6++;

break;

}

case 2:

{

System.out.println("--------------------------------------------------------------------------------");

System.out.println("id \t Name \t Gender \t Salary");

System.out.println("--------------------------------------------------------------------------------");

for (j = 0; j < count6; j++)

{

if (a.equals(s[j].desg))

s[j].staff\_info();

}

break;

}

case 3:

{

System.out.println("--------------------------------------------------------------------------------");

System.out.println("id \t Name \t Gender \t Salary");

System.out.println("--------------------------------------------------------------------------------");

for (j = 0; j < count6; j++)

{

if (b.equals(s[j].desg))

s[j].staff\_info();

}

break;

}

case 4:

{

System.out.println("--------------------------------------------------------------------------------");

System.out.println("id \t Name \t Gender \t Salary");

System.out.println("--------------------------------------------------------------------------------");

for (j = 0; j < count6; j++)

{

if (c.equals(s[j].desg))

s[j].staff\_info();

}

break;

}

}

System.out.println("\nReturn to Back Press 1 and for Main Menu Press 0");

s6 = input.nextInt();

}

break;

}

default:

{

System.out.println(" You Have Enter Wrong Choice!!!");

}

}

System.out.println("\nReturn to MAIN MENU Press 1");

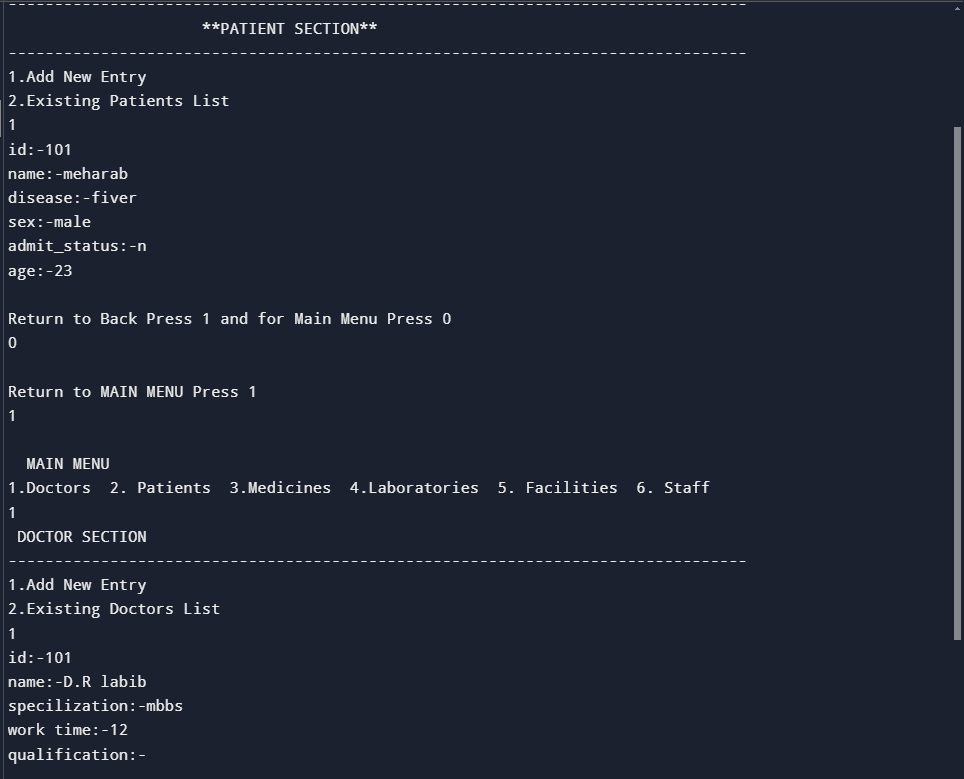
status = input.nextInt();

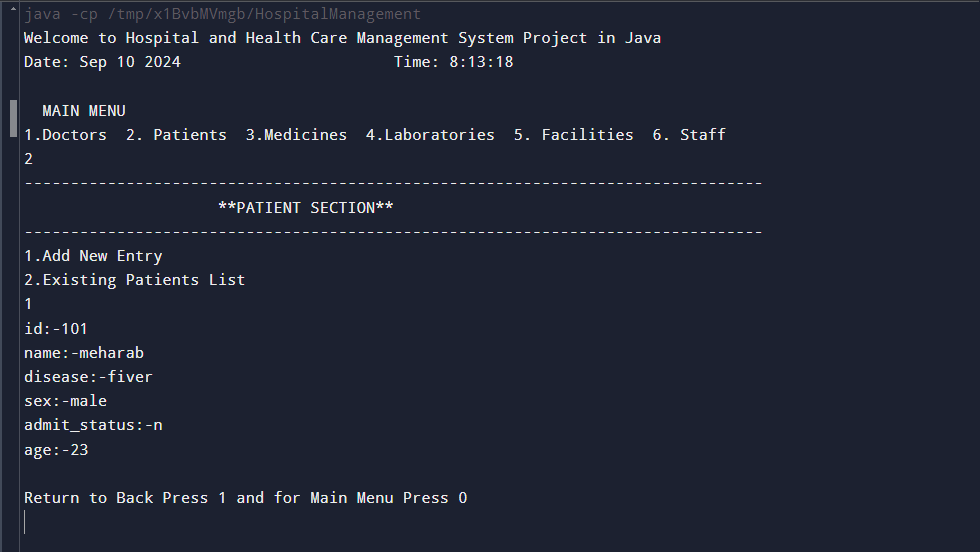
}

}

}

**Outputs**





**Explanation of concepts Implemented in code**

1. Import Statements:   
- `import java.util.Formatter;`: Importing the `Formatter` class from the `java.util` package.   
- `import javax.print.Doc;`: Importing the `Doc` class from the `javax.print` package.   
- `import java.nio.ByteBuffer;` and `import java.nio.CharBuffer;`: Importing classes related to byte and character buffers from the `java.nio` package.   
  
2. **Class Definitions** :   
- The code describes several classes: `Person`, `Patient`, `Doctor`, and `Appointment`. These classes are used to model a hospital management system.   
  
3**. Inheritance**: - The `Patient` and `Doctor` classes both expand the `Person` class. This shows an inheritance relationship, where `Patient` and `Doctor` inherit attributes and methods from the `Person` class.   
  
4. **Instance Variables and Methods ():**   
- The `Person` class includes instance variables like `name`, `gender`, `phn`, and methods like `createPerson()` and `getDetail()` for creating and displaying person details.   
  
5. Static Arrays:   
- The `patients` and `doctors` arrays are used to store examples of patients and doctors, respectively. These are defined as static arrays with a maximum size of 20.   
  
6. **Constructors**:   
- The `Doctor` and `Patient` classes have constructors `createDoctor()` and `createPatient()` that initialize their properties by calling the `createPerson()` method from the base `Person` class.   
  
**7. Appointment Class**- The `Appointment` class is used for booking meetings. It includes instance variables like `doctor`, `patient`, `date`, and `time`.   
- The constructor `public Appointment(Doctor doctor, Patient patient, String date, String time)` is used to make appointments.   
- Several methods like `showAppointments()`, `showDoctorApp()`, `showPatientApp()`, `DenyApp()`, and `rescheApp()` are used to handle appointments.   
  
8**. Menu-Driven Operations**   
- The `showDocMenu()` and `showPatMenu()` methods present menu options for the doctor and patient portals, respectively.   
- The `docOperations()` and `patOperations()` methods handle the menu-driven operations for doctors and patients.   
- These processes include registering new doctors/patients, booking appointments, displaying appointments, watching doctor details, and more.   
  
9. **Main Method :** - The `main` method is the entry place of the program.   
- It starts by asking the user to select their designation (patient or doctor) and then calls the associated portal functions (e.g., `patOperations()` for patients and `docOperations()` for doctors).   
- Exception handling is included to handle any errors that may come during program execution.   
  
10**. Flow Control** : The code uses different control structures like loops (e.g., `while` loops for menu-driven options), conditionals (e.g., `switch` statements for menu choices), and conditional statements to control the flow of the program.   
  
11. Input/Output : - The `Scanner` class is used to take user input for different attributes.   
- `System.out.println()` is used to show messages and results to the user.   
  
12. Data Storage : - The program uses arrays (`patients`, `doctors`, and `appointments`) to store and handle data about patients, doctors, and appointments.   
  
13. Constructor Overriding: - In the `Doctor` and `Patient` classes, the constructors `createDoctor()` and `createPatient()` are used to create instances of doctors and patients, respectively.   
- These constructors override the normal constructor of the base class, `Person`. By calling `super.createPerson()`, they run the constructor of the base class and initialize the common attributes (`name`, `gender`, and `phn`).   
- This is an example of constructor overriding, where a subclass gives its own implementation of a constructor, but still invokes the constructor of the base class.   
  
14. **Method Overriding** : - In the `Person`, `Patient`, and `Doctor` classes, the `getDetail()` method is defined.   
- The `getDetail()` method in the `Person` class is overridden in both the `Patient` and `Doctor` classes to provide customized versions.   
- In the `Appointment` class, several methods like `showAppointments()`, `showDoctorApp()`, `showPatientApp()`, `DenyApp()`, and `rescheApp()` are defined, and these methods are called with the relevant objects.   
- This is an example of method overriding, where a subclass gives its own implementation of a method that is already defined in the base class. It lets each subclass to have its version of the method.   
Certainly, here are the key points to highlight what you've learned from the given Java code:

**CONCLUSION**

Basic Concepts and Syntax of Java: - I have gotten a knowledge of the fundamental syntax and structure of the Java programming language.   
 Inheritance:- Inheritance concepts have been learned and applied, including constructor overriding and method overriding, showing how classes can inherit and extend functionality from other classes.   
Exception Handling:- You have met exception handling in Java, which is crucial for managing and responding to errors during program execution.   
String Handling: - The code includes string handling, such as capturing and displaying user input using the `Scanner` class, and manipulating strings to display information to users.   
. Arrays: - You have worked with arrays, especially static arrays, to store and manage data about patients, doctors, and appointments.   
These ideas provide a solid foundation for understanding object-oriented programming in Java and can be extended to create more complex and feature-rich applications. To continue improving MY Java skills, consider studying more advanced topics like data structures, design patterns, and database integration

Code Organization: The code could benefit from better organization, with the division of classes into different files or packages for improved maintainability.   
 Exception Handling: While the code includes a simple exception handling mechanism, it could be enhanced to handle specific exceptions more gracefully. For example, addressing input validation errors or database-related exceptions.   
 User Experience: The program's user interface is text-based and may not provide the best user experience. Consider using a graphical user interface (GUI) or web-based system to make it more user-friendly.   
Data Persistence: The code currently uses static vectors to store data. In a real-world application, a database or some form of data persistence mechanism would be more ideal.