

# GROUP 29

**HEALTH  
INSURANCE**  
RISK ADVISORY  
FIRST AID  
MONEY MANAGEMENT  
EXAMINATIONS

## PROJECT REPORT

# SELF HEALTHCARE INFORMATION SYSTEM

H.R.A.L.N.Ranasinghe	PS/2020/103
R.Y.W Ekanayaka	PS/2020/150
R.N.R.Fonseka	PS/2020/215
M.R.T Fernando	PS/2020/192
J.A.D.N Jayakody	PS/2020/174



UNIVERSITY OF KELANIYA

# Contents

1.INTRODUCTION .....	4
1.1. Problem .....	4
1.2. Aim of This Project.....	4
1.3. Research Gap .....	5
1.3.1 Overview .....	5
1.3.2 Conclusions .....	6
2. REQUIREMENTS ANALYSIS.....	8
2.1. Functional Requirements .....	8
2.1.1. Patient Package.....	8
2.1.1.1. DBConnection Class.....	8
2.1.1.2. Patient Class.....	8
2.1.1.3. Patientimplement Class .....	9
2.1.1.4. Patientinterface Class.....	9
2.1.2. Doctor Package.....	10
2.1.2.1. DBConnection class.....	10
2.1.2.2. Doctor Class.....	10
2.1.2.3. Doctorimplement Class .....	11
2.1.2.4. Doctorinterface Class.....	12
2.1.3. Appointment Package.....	12
2.1.3.1 Appointment Class.....	12
2.1.4 Billing Package.....	13
2.1.4.1. Billing Class.....	13
2.1.5. Reviews Package.....	13
2.1.5.1 RateReview Class .....	13
2.1.6. Doctor Revenue Package.....	14
2.1.6.1. Revenue Management Class.....	14
2.2. Nonfunctional Requirements .....	15
2.3 Hardware Requirements.....	15
2.4. Software Requirements .....	16
3. Use Cases of OOP Concepts.....	17
3.1. Inheritance.....	17
3.2. Encapsulation .....	18

3.3. Abstraction .....	19
3.4. Polymorphism .....	20
3.4.1. Overloading .....	20
3.4.2. Overriding .....	21
3.4.3. Application of toString() .....	22
4. IMPLEMENTATION OF THE SYSTEM .....	23
4.1. Database .....	23
4.2. Welcome Message .....	23
4.3. Patient Portal .....	24
4.4 Admin Portal .....	24
4.5 Doctor Portal .....	24
4.6. Patient Functionalities .....	25
4.6.1. Patient Registration .....	25
4.6.2. View Patient Profile .....	25
4.6.3. View Doctor List .....	25
4.6.4. View Doctor Profile .....	26
4.6.5. Appointment Placement .....	26
4.6.6. Rate the Doctor .....	27
4.7. Admin Functionalities .....	28
4.7.1. View All Patients .....	28
4.7.2. Delete Patient Profile .....	28
4.7.3. View All Doctors .....	29
4.7.4. Delete Doctor Profile .....	29
4.8. Doctor Functionalities .....	30
4.8.1. Doctor Registration .....	30
4.8.2. View total Revenue of the Doctor .....	30
5. CHALLENGES .....	31
6. SOLUTIONS .....	32
7. FUTURE IMPLEMENTS .....	33
8.CONTRIBUTION .....	34
8.1. Source Code .....	34
9. Teamwork .....	35

# 1.INTRODUCTION

## 1.1. Problem

In the day-to-day world, the field of healthcare is a complicated and significant industry that delivers critical services to individuals and communities all around the world. This industry includes a wide range of services and suppliers, including hospitals and clinics as well as private practices, labs, and medical device makers. The fundamental goal of the healthcare industry is to offer medical care, preventative care, and to enhance patients' general well-being. Healthcare practitioners strive to improve patient outcomes, ease pain, and enhance patients' quality of life. However, the healthcare industry confronts various obstacles, including expanding service demand, restricted resources, and rising costs. Government authorities and professional organizations also regulate and oversee the industry to guarantee the quality and safety of healthcare services.

Though the healthcare industry is a very crucial part of society, it has some major issues while delivering the service to the customers. The main issue is the inconvenience that patients are faced with. Inconvenience can take various forms and have a considerable influence on patients, healthcare professionals, and the broader healthcare system. One typical annoyance is healthcare system inefficiency, which can result in extended wait times, delayed test results, and difficulty getting care. Furthermore, the usage of complicated and out-of-date record-keeping systems can lead to mistakes and misunderstanding among healthcare personnel. Patients may sometimes encounter difficulties in scheduling appointments, particularly with specialists or during peak demand periods. Other drawbacks include the high expense of healthcare, insufficient insurance coverage, and restricted access to preventative treatment.

So, our team designed a Java program to reduce this inconvenience of reservation and channeling. We created a program to make this process more effective and efficient using Java coding.

## 1.2. Aim of This Project

The aim of this project is to develop and implement a comprehensive and efficient Self Healthcare Management System that will streamline and automate the various processes and functions of a hospital. This system will provide a unified platform for managing patient care, financial operations, and administrative tasks, and will improve the quality of care and the overall patient experience, while also increasing operational efficiency and reducing costs. The goal is to create a system that is user-friendly, reliable, and scalable, and that will meet the evolving needs of the hospital and its stakeholders.

## 1.3. Research Gap

### 1.3.1 Overview

The project's goal is to create a hospital system that includes a self-rating and booking system. We have developed a solution based on research conducted in some other countries that suggest the value of having a rating system for doctors because there are numerous indications that patient feedback is significant. (<https://thescript.zocdoc.com/6-reasons-patient-feedback-is-important/>) On the other hand,

- *A recent Johns Hopkins study claims more than 250,000 people in the U.S. die every year from medical errors. Other reports claim the numbers to be as high as 440,000.*
- *Medical errors are the third-leading cause of death after heart disease and cancer.*

<https://www.cnn.com/2018/02/22/medical-errors-third-leading-cause-of-death-in-america.html>

According to the researchers, the third leading cause of death in the US is the medical errors. we discovered that medicals errors and certain medical professionals' negligence may result in the deaths of several patients <https://journals.sagepub.com/doi/full/10.1177/0022018320946498>

With the help of our solution, we can reduce the risk of the case and assign well-performed doctors at a higher rate and enable patients to choose wisely the best one among their ratings from themselves. Moreover, this will provide a thorough overview of doctors' performance and identify any instances of doctoral malpractice.

Further reviewing the assessments, it became clear, this will demonstrate that, instead of focusing on interpersonal skills, we would know the best doctors to channel if we had a decent grading system.

J Med Internet Res. 2019 Jun; 21(6): e11188.

Published online 2019 Jun 28. doi: [10.2196/11188](https://doi.org/10.2196/11188)

The Impact of Web-Based Ratings on Patient Choice of a Primary Care Physician Versus a Specialist: Randomized Controlled Experiment

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6625218/>

*The findings suggest that people place more weight on technical skills than interpersonal skills in their selection of a physician based on their ratings on the Web. Specifically, people are more likely to make a compromise on interpersonal skills in their choice of a specialist compared with a primary care physician. This study emphasizes the importance of examining Web-based physician ratings in a more nuanced way in relation to the selection of different types of physicians.*

Nowadays these types of systems are rare and have caught the attention and are getting popularity and there are However, there have been fewer types of research done in this field, indicating a deficit.



<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3636311/>

Reviewed by Guodong Gao and Felix Greaves

Martin Emmert, MSc, Ph.D,<sup>1</sup> Uwe Sander, MD,<sup>2</sup> and Frank Pisch, B.Sc

J Med Internet Res. 2013 Feb; 15(2): e24.

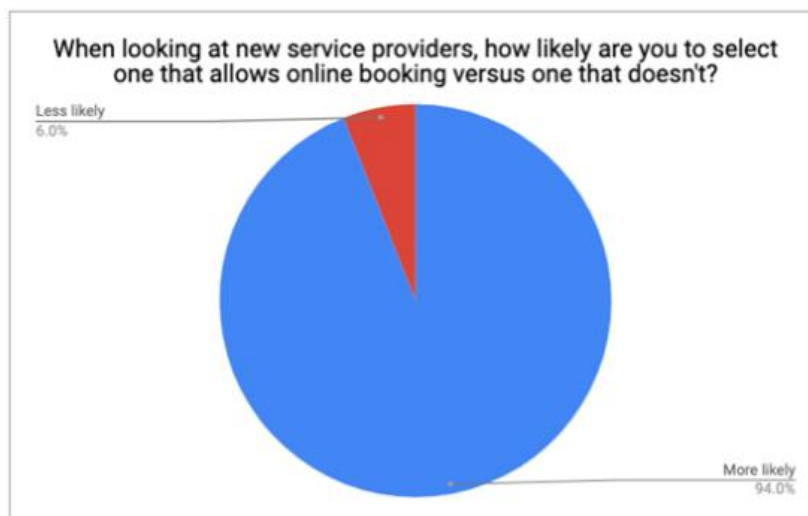
Published online 2013 Feb 1. doi: [10.2196/jmir.2360](https://doi.org/10.2196/jmir.2360)

*Physician-rating websites are currently gaining in popularity because they increase transparency in the health care system. However, research on the characteristics and content of these portals remains limited.*

### 1.3.2 Conclusions

Although the number of publications is still low, physician-rating websites are gaining more attention in research. But the current condition of physician-rating websites is lacking. This is the case both in the United States and in Germany. Further research is necessary to increase the quality of the websites, especially from the patients' perspective.

After conducting a thorough investigation into appointment scheduling systems, we discovered that the complex systems used by the healthcare industry have a great deal of appointment scheduling issues (<https://www.hindawi.com/journals/jhe/2022/5819813/>). As a result, we decided to develop a self-booking service that takes patients' preferences into consideration. This graph will illustrate the likelihood of patients go for an online booking.



Additionally, studies reveal that our primary self-booking and channeling system has some untapped potential benefits.

(<https://www.commusoft.co.uk/online-appointment-system-benefits/>)

and changes may affect the field and have a certain trend.

J Med Internet Res. 2017 Apr; 19(4): e134.

Published online 2017 Apr 26. doi: [10.2196/jmir.6747](https://doi.org/10.2196/jmir.6747)

Web-Based Medical Appointment Systems: A Systematic Review

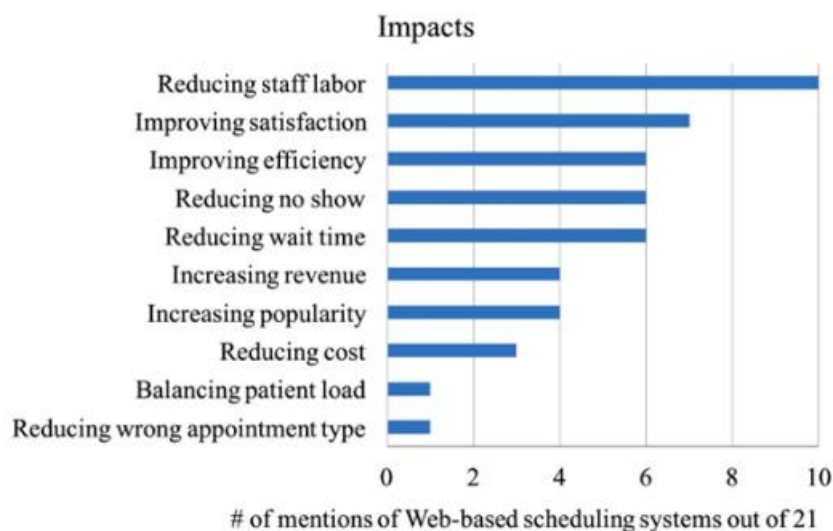
Reviewed by David Miller, Ping Yu, and Xiaojun Zhang

Peng Zhao, MSc,<sup>1</sup> Illhoi Yoo, PhD,<sup>1,2</sup> Jaie Lavoie, PharmD, MS,<sup>3</sup> Beau James Lavoie, PharmD, MS,<sup>4</sup> and Eduardo Simoes, MSc, DLSHTM, MPH, MD<sup>1,2</sup>

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5425771/>

*Health care is changing with a new emphasis on patient-centeredness. Fundamental to this transformation is the increasing recognition of patients' role in health care delivery and design. Medical appointment scheduling, as the starting point of most non-urgent health care services, is undergoing major developments to support active involvement of patients. By using the Internet as a medium, patients are given more freedom in decision making about their preferences for the appointments and have improved access.*

*Overall, the literature suggests a growing trend for the adoption of Web-based appointment systems. The findings of this review suggest that there are benefits to a variety of patient outcomes from Web-based scheduling interventions with the need for further studies.*



## 2. REQUIREMENTS ANALYSIS

### 2.1. Functional Requirements

In this part, let's go through the classes, attributes and the methods use in the system. In our system we used 6 packages which came with classes and interfaces.

#### 2.1.1. Patient Package

This package is used to add personal information about patients, display all the patients' information who have registered with the system, search relevant patient information by patient id, and delete the patient's information after the process. This package consists of 3 classes and one interface.

##### 2.1.1.1. DBConnection Class

###### Methods

- DBConnection() -

This class was created to connect the database for store and use the information of the doctors.

##### 2.1.1.2. Patient Class

###### Attributes

- patientID - an integer represents the ID number the patient has registered to system.
- patientName - a string to represent the name of the patient.
- email - a string to represent the email of the patient.
- address - a string to represent the address of the patient.
- phone - a string to represent the telephone number of the patient.
- age - an integer to represent the age of the patient.

###### Methods

- Patient() - default constructor
- Patient(int patientID, String patientName, String email, String address, String phone, int age) - parameterised constructor in order to add all the details of the doctor object
- Getters and setters – for get and set the values of the information.

```
getPatientID()
setPatientID(int patientID)
getPatientName()
setPatientName(String patientName)
getEmail()
```



```

setEmail(String email)
getAddress()
setAddress(String address)
getPhone()
setPhone(String phone)
getAge()
setAge(int age)

```

### 2.1.1.3. Patientimplement Class

#### Methods

- addPatient(Patient patient) -

Using this method, the database will be updated by setting all the information about the relevant patient details and showing a print statement if the added process is done successfully.

- showAllPatients() -

From this the information of all the patients will be printed on the screen by getting the details from the database.

- showPatientrByID( patientID) -

Getting the doctor ID as a parameter and using data base, the details of the relevant patient will be printed.

- deletePatientByID( patientID) -

Getting only the patient ID as a parameter, the information will be deleted in the database and process to print the text show as successfully deleted.

### 2.1.1.4. Patientinterface Class

Interface only holds the name of the methods.

- addPatient(Patient patient)
- showAllPatients()
- showPatientByID(PatientID)
- deletePatient( PatientID)

### 2.1.2. Doctor Package

This package is used to add personal information about doctors, display all the doctors' information who have registered with the system, search relevant doctor information by doctor id, and delete the doctor's information after the process. There is an additional option to update the doctor's specification. This package consists with 3 classes and one interface.

#### 2.1.2.1. DBConnection Class

##### Methods

- createDBConnection() -

This class was created to connect the database for store and use the information of the doctors.

#### 2.1.2.2. Doctor Class

##### Attributes

- doctorID - an integer for represent the ID number they have registered to the system.
- doctorName - a string to represent the name of the doctor.
- specialization - a string to represent the specialization of the doctor.
- specializationID - an integer for the ID number of the specialization stream
- time - a Time, which is represent, which time the doctor can do the consultation.
- date - a Date for represent the date, when the doctor can do the consultation.
- doctorFee - a double to represent the charge of the doctor.

##### Methods

- Doctor() - default constructor
- Doctor(int doctorID, String doctorName, String specialization, int specializationID, Time time, Date date, double doctorFee) - parameterised constructor in order to add all the details of the doctor object
- Getters and setters - for get and set the values of the information.

```
getDoctorID()
setDoctorID(int doctorID)
```

```
getDoctorName()
setDoctorName(String doctorName)
```

```
getSpecialization()
setSpecialization(String specialization)
```

```
getSpecializationID()
setSpecializationID(int specializationID)
```

```
double getDoctorFee()
setDoctorFee(double doctorFee)
```

```
getTime()
setTime(Time time)
```

```
getDate
setDate(Date date)
```

- toString() - show all the information of the doctor as a print statement

### 2.1.2.3. Doctorimplement Class

#### Methods

- addDoctor(Doctor doctor) -

Using this method, the database will be updated by setting all the information about the relevant doctor details and showing a print statement if the added process is done successfully.

- showAllDoctors() -

From this the informations of all the doctors will printed on the screen with the details from the database.

- showDoctorByID(int doctorID) -

Getting the doctor ID as a parameter and using data base, the details of the relevant doctor will be printed.

- updateDoctor(int doctorID, String doctorName) -

Using the entered doctor name and ID, the information will be updated in the database and process to print the text show as successfully updated.

- deleteDoctor(int doctorID) -

Getting only the doctor ID as a parameter, the information will be deleted in the database and process to print the text show as successfully deleted.

- printDoctorName(int doctorID) -

Getting only the doctor ID as a parameter, using this method the name of the relevant doctor will be printed or print a text as no doctor found with ID.

#### 2.1.2.4. Doctorinterface Class

Interface only holds the name of the methods.

- addDoctor(Doctor doctor)
- showAllDoctors()
- showDoctorByID(int doctorID)
- updateDoctor(int doctorID, String doctorName)
- deleteDoctor(int doctorID)
- printDoctorName(int doctorID)

### 2.1.3. Appointment Package

This package only consists of one class which is used to get the appointment details about the patient's latest appointment. By using the patient ID, they can get the appointment details such as appointment date and time.

#### 2.1.3.1 Appointment Class

##### Methods

- addDoctorPatient (doctorID,patientID) -

used in order to add the appointment details of the patient and doctor details to the database and show a text if the patient has booked the appointment successfully.

- getPatientAndDoctorDetails( patientID,doctorID) -

To get all the details of the appointment using the database and show it as a print statement to the user.

### 2.1.4 Billing Package

This package only consists of one class which is used to get the billing details about the patient's latest appointment after the channeling process. By using the patient ID and the doctor ID, they can get the billing details such as consultation charges, hospital charges and the grand total.

#### 2.1.4.1. Billing Class

##### Attributes

- doctorFee - a double to represent the charge of the doctor.
- hospitalCharges - a double representing the hospital charge with a fixed value.
- medicalCharges - a double representing the medical charge with a fixed value.
- totalCharges - a double representing the total charge with assigning 0.

##### Methods

- getDoctorFee( doctorID, patientID) -

After adding the patient ID and the doctor ID get the doctor fee through the data base and add all the charges to the total payment and print the billing process. Also have the exception handling for if the user added an invalid patient or doctor ID.

### 2.1.5. Reviews Package

We use this package to add a review or rating to the doctor that the patient had channeled. This is one of the main goals of our system. Patients can add a review or a rating using the doctor ID. And doctors can go through their rating and the review using their doctor ID.

#### 2.1.5.1 RateReview Class

##### Methods

- displayReviews(int doctorID) -

Getting only the doctor ID as a parameter, the updated database will be showed the review about the relevant doctor and show a print statement if the review process done successfully. If the ID does not match will be printed a statement as no review found for the doctor.

- addReview(int patientID, int doctorID, String review)

Getting all the parameters of the doctor and patient ID and a string as a review, the database will be updated by the review about the relevant doctor and show a print statement if the review added process done successfully.

### 2.1.6. Doctor Revenue Package

This class is used to get the doctors' revenue using their doctor ids. When a doctor enters his/her id to the system in the revenue section they can get the details about total revenue of them, number of patents that they have examined and their names.

#### 2.1.6.1. Revenue Management Class

##### Attributes

- totalfee - a double which is representing the total amount of charges.

##### Methods

- displayPatientsAndCount(doctorID) -

Get the doctor ID as a parameter and search through the data base in order to find and set the relevant doctor and the fee also the patients names then print the final revenue report which included consultation fee of the doctor, name of all the patient consulted, total revenue of the doctor.



## 2.2. Nonfunctional Requirements

- **Security:** Ensure the confidentiality and availability of sensitive patient information.
- **Usability:** The system should be user-friendly and easy to use for both patients and hospital staff.
- **Performance:** The system should respond quickly to user requests, even during high usage times.
- **Scalability:** The system should be able to accommodate an increasing number of users and handle larger amounts of data.
- **Reliability:** The system should be available and always functioning properly.
- **Compliance:** The system should meet industry standards and regulations.
- **Maintainability:** The system should be easy to maintain, update, and fix when needed.
- **Accessibility:** The system should be accessible to all users, including those with disabilities.
- **Disaster Recovery:** The system should have a plan for disaster recovery in case of system failures or data loss.

## 2.3 Hardware Requirements

- **Servers:** High-performance servers are needed to run the hospital management system, store patient data, and support networked devices.
- **Mobile devices:** Devices such as desktop computers or laptops are needed for hospital staff to access the system and perform their tasks.
- **Network Equipment:** The hospital's network infrastructure should be capable of supporting the hospital management system, by providing a secure, reliable, and fast network.
- **Scanners:** Scanners can be used to digitize paper-based records and integrate them into the hospital management system.
- **Medical Devices:** The hospital management system should be able to integrate with various medical devices, such as X-ray machines, MRI machines, and electrocardiogram (ECG) machines.

## 2.4. Software Requirements

- **Operating System:** The healthcare management system is running on a commonly used Windows operating system.
- **Database Management System:** Java Arrays to store data related to the system and to act as the database.
- **Java Development Kit (JDK):** The JDK is a software development environment that includes the Java Runtime Environment (JRE) and the Java Compiler. It is required for developing, compiling, and executing Java code.
- **Integrated Development Environment (IDE):** An IDE provides a user-friendly interface for writing,
- **Security:** The system uses built-in security features, such as encryption and authentication, to protect sensitive patient information.

## 3. Use Cases of OOP Concepts

### 3.1. Inheritance

In object-oriented programming, inheritance is a key concept that allows for the creation of new classes based on existing ones. Inheritance allows a subclass to inherit properties and methods from a superclass, promoting code reuse and making complex software systems easier to create. The superclass serves as the subclass's blueprint, defining its structure and behavior. By inheriting from a superclass, a subclass can reuse the superclass's code while adding new features and functionality as needed. Inheritance is an extremely useful tool for developing modular, extensible, and maintainable software systems. It aids in the reduction of duplication, the increase of code reuse, and the promotion of consistency and coherence in the design of software applications.

We use inheritance in the Doctor and DoctorImplement class which as DoctorImplement is the parent class and the doctor class as the child class by extending to inherit the methods in the parent class. We used that in our main method to call by the doctor object.

Doctor class extends the DoctorImplement class:

```
package Doctors;

import java.sql.Date;
import java.sql.Time;

public class Doctor extends DoctorImplements{
    private int doctorID;
    private String doctorName;
    private String specialization;
    private int specializationID;
    private Time time;
    private Date date;
    private double doctorFee;
```

Main method:

```
        break;
    case 2:
        doctor.showAllDoctors();
        break;
    case 3:
        System.out.print("Enter Doctor ID to view doctor details: ");
        int docid = sc.nextInt();
        doctor.showDoctorByID(docid);
        break;
```

## 3.2. Encapsulation

Encapsulation is a fundamental concept in object-oriented programming that entails grouping data and methods that operate on that data into a single unit known as a class. Encapsulation allows you to hide a class's implementation details from other parts of the program, promoting modularity, maintainability, and code reuse. Encapsulation also allows for the enforcement of data integrity and security by restricting data access to well-defined interfaces. This means that an object's internal state can only be modified by methods defined in its class, reducing the risk of unintended modifications and improving the software system's reliability and robustness.

In our project data protection is very important when it comes to hiding the personal information from the outsiders. In each of the patient and doctor class have added the private modifier in each attribute and we used getters and setters to call or show these attributes.

Doctor class:

```
public class Doctor{
    private int doctorID;
    private String doctorName;
    private String specialization;
    private int specializationID;
    private Time time;
    private Date date;
    private double doctorFee;
```

Patient class:

```
public class Patient {
    private int patientID;
    private String patientName;
    private String email;
    private String address;
    private String phone;
    private int age;
```

Used getters and setters(in the Doctor class):

```
public int getDoctorID() { return doctorID; }

public void setDoctorID(int doctorID) {
    this.doctorID = doctorID;
}

public String getDoctorName() { return doctorName; }

public void setDoctorName(String doctorName) { this.doctorName = doctorName; }

public String getSpecialization() { return specialization; }

public void setSpecialization(String specialization) { this.specialization = specialization; }

public int getSpecializationID() { return specializationID; }

public void setSpecializationID(int specializationID) { this.specializationID = specializationID; }

public double getDoctorFee() { return doctorFee; }

public void setDoctorFee(double doctorFee) { this.doctorFee = doctorFee; }

public Time getTime() {
```

### 3.3. Abstraction

The creation of simplified models of complex systems is a fundamental concept in object-oriented programming. Abstraction enables programmers to concentrate on the most important aspects of a system while hiding unnecessary details, making it easier to understand, design, and maintain. Abstraction is achieved in object-oriented programming with abstract classes and interfaces, which define a set of properties and methods that must be implemented by the concrete classes that inherit from them. Abstraction enables code reuse and promotes modularity and extensibility in software systems by defining a common set of behaviors and properties. In this essay, we will delve deeper into the concept of abstraction, including its benefits, limitations, and best practices.

There are two methods to use the principle of abstraction, Using interfaces or abstract classes. In our system we have used two interfaces, such as DoctorInterface and the PatientInterface. We used the abstraction principle by implementing the interfaces from the DoctorImplement and the PatientImplement classes.

Interface:

```

1 public interface DoctorInterface {
2
3     //Add doctor
4     public void addDoctor(Doctor doctor);
5
6     //Show all doctors
7     public void showAllDoctors();
8
9     //Show doctor by id
10    public void showDoctorByID(int doctorID);
11
12    //Update doctor
13    public void updateDoctor(int doctorID,String doctorName);
14
15 }

```

DoctorImplement class:

```

1 package Doctors;
2 import java.sql.*;
3
4 public class DoctorImplements implements DoctorInterface{
5     Connection con;
6     @Override
7     public void addDoctor(Doctor doctor) {
8
9         con = DBConnection.createDBConnection();
10        String query = "insert into doctor values(?,?,?, ?, ?, ?, ?)";
11
12    }
13 }

```

### 3.4. Polymorphism

Polymorphism is an important concept in object-oriented programming because it allows objects of different classes to be treated as if they were of the same type. This means that regardless of their specific implementation details, different objects can be manipulated in a consistent manner. Polymorphism can occur via a variety of mechanisms, including inheritance, interfaces, and overloading. Inheritance allows a subclass to inherit methods and properties from a superclass, whereas interfaces define a set of methods that any class that implements the interface must implement. Overloading allows for the definition of methods with the same name but different parameters in the same or different classes. Polymorphism is an effective tool for creating adaptable and extensible software systems that can respond to changing requirements.

#### 3.4.1. Overloading

In our system overloading concept applied in both patient and doctor classes using **constructor overloading**. We use two constructors such as the default constructor and the parameterized constructor in order to create an object from the doctor class and assign the all the information given by the patient and doctor object.

Doctor class:

```
public Doctor() {
}

public Doctor(int doctorID, String doctorName, String specialization, int specializationID, Time time, Date date, double doctorFee) {
    this.doctorID = doctorID;
    this.doctorName = doctorName;
    this.specialization = specialization;
    this.specializationID = specializationID;
    this.time = time;
    this.date = date;
    this.doctorFee = doctorFee;
}
```

Main method:

```
//Object creation from RevenueManagement class
RevenueManagement rev = new RevenueManagement();
//Creating an object from Doctor class
Doctor doctor = new Doctor();
```



```

System.out.println("Enter the date (yyyy-mm-dd):");
String dateString = sc.next();
Date date = Date.valueOf(dateString);

System.out.print("Enter Doctor Fee: ");
double doctorFee=sc.nextDouble();

//Setting the input values gain by the user to the parameterized constructor
doctor = new Doctor(doctorID, doctorName, specialization, specializationID, time, date, doctorFee);

doc.addDoctor(doctor);
break;

```

### 3.4.2. Overriding

When it comes to overriding usually after added an interface, we can only show the methods heading in the interface and there must be classes which shows the body parts of the mentioned methods. In our system we use two interfaces, one is DoctorInterface and the other one is PatientInterface and all the methods we implemented in the classes are named DoctorImplement and PatientImplement.

Interface:

```

//Add patient
public void addPatient(Patient patient);

//Show all patients
public void showAllPatients();

//Show patient by id
public void showPatientByID(int patientID);

```

Doctor class:

```

@Override
public void showAllDoctors() {
    con = DBConnection.createDBConnection();
    String query = "select * from doctor";

@Override
public void showDoctorByID(int doctorID) {
    con=DBConnection.createDBConnection();
    String query = "select * from doctor where
    doctorID=" + doctorID + ";";
}

```

### 3.4.3. Application of toString()

We used toString method to print all of the information about the doctor which are taken from the attributes in the Doctor class.

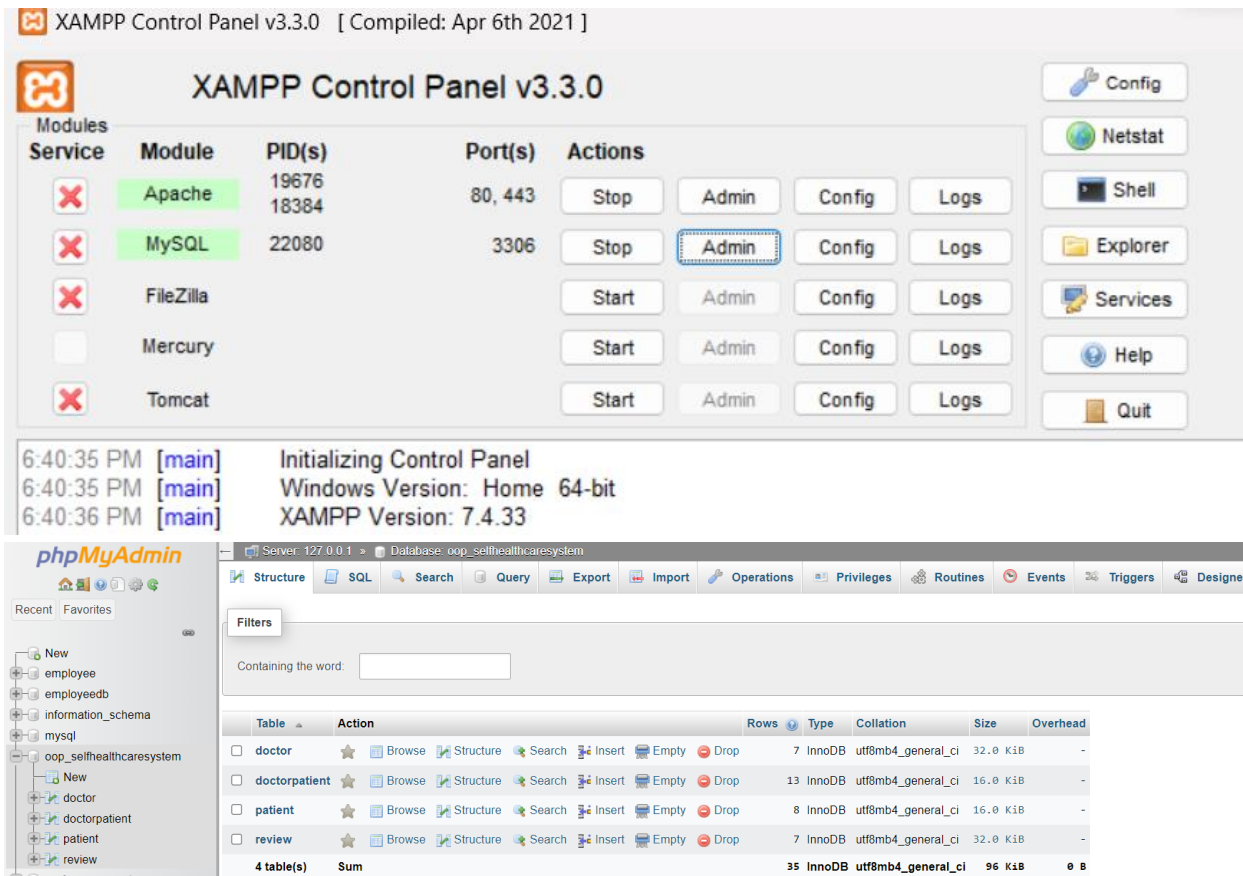
Doctor class:

```
@Override
public String toString() {
    return "Doctor{" +
        "doctorID=" + doctorID +
        ", doctorName='" + doctorName + '\'' +
        ", specialization='" + specialization + '\'' +
        ", specializationID=" + specializationID +
        ", time=" + time +
        ", date=" + date +
        ", doctorFee=" + doctorFee +
        '}';
}
```

## 4. IMPLEMENTATION OF THE SYSTEM

The snapshots of our system implementation.

### 4.1. Database

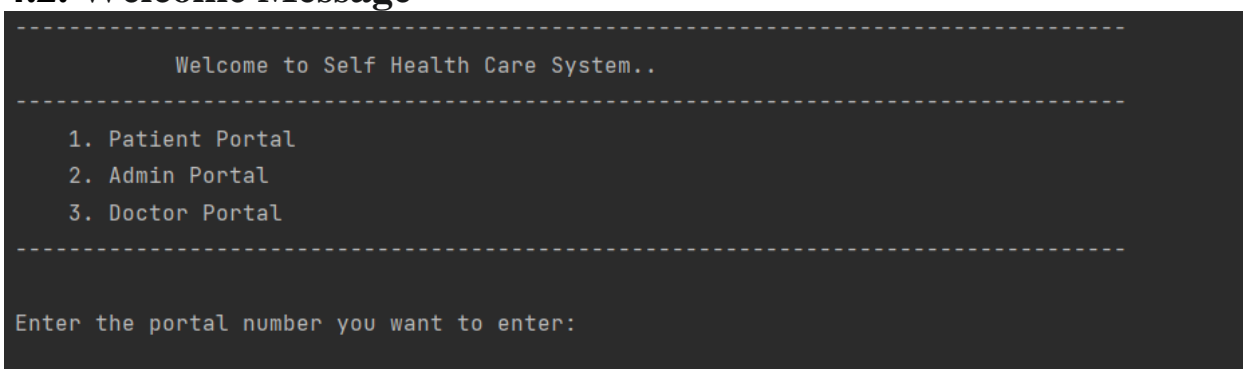


The screenshot displays the XAMPP Control Panel v3.3.0 interface. The top section shows the status of various services: Apache, MySQL, FileZilla, Mercury, and Tomcat. The 'Admin' button for MySQL is highlighted. Below the service list, the console output shows the initialization of the control panel, the Windows version (Home 64-bit), and the XAMPP version (7.4.33).

The bottom section shows the phpMyAdmin interface. The left sidebar lists the database structure, including the 'oop\_selfhealthcaresystem' database. The main area displays a table list for the 'oop\_selfhealthcaresystem' database, showing tables: doctor, doctorpatient, patient, and review. The table list includes columns for Table, Action, Rows, Type, Collation, Size, and Overhead.

Table	Action	Rows	Type	Collation	Size	Overhead
doctor	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_general_ci	32.0 Kib	-
doctorpatient	Browse Structure Search Insert Empty Drop	13	InnoDB	utf8mb4_general_ci	16.0 Kib	-
patient	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_general_ci	16.0 Kib	-
review	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_general_ci	32.0 Kib	-
4 table(s)	Sum	35	InnoDB	utf8mb4_general_ci	96 Kib	0 B

### 4.2. Welcome Message



The screenshot shows a terminal window with the following text:

```

-----
Welcome to Self Health Care System..
-----

1. Patient Portal
2. Admin Portal
3. Doctor Portal
-----

Enter the portal number you want to enter:
  
```

### 4.3. Patient Portal

```
Enter the portal number you want to enter: 1
```

```
-----  
Welcome to Patient Portal  
-----
```

1. Patients Registration
2. View Profile by ID
3. Available Doctor List
4. View Doctor Details and Rates by ID
5. Rating Doctors
6. Place an Appointment

```
Enter the number of the operation to perform: |
```

### 4.4 Admin Portal

```
Enter the portal number you want to enter: 2
```

```
-----  
Welcome to Admin Portal  
-----
```

1. View All Patients
2. Delete Patient Profile
3. View All Doctors
4. Delete Doctor Profile

```
Enter the number of the operation to perform:
```

### 4.5 Doctor Portal

```
Enter the portal number you want to enter: 3
```

```
-----  
Welcome to Doctor Portal  
-----
```

1. Doctor Registration
2. View Total Revenue

```
Enter the number of the operation to perform: |
```

## 4.6. Patient Functionalities

### 4.6.1. Patient Registration

```

Enter the number of the operation to perform: 1
-----
Patient Registration
**Please provide patient details to proceed the registration...
-----

Enter Patient ID: 4848
Enter Patient Name: Saman_Perera
Enter Patient Age: 45
Enter Patient Address: 12/A,Kadawatha
Enter Patient Phone: 0715667878
Enter Patient Email: saman@mail.com

Successfully Registered the Patient!
**Please remember your PatientID for the future proceedings
Your PatientID is: 4848

```

### 4.6.2. View Patient Profile

```

Enter the number of the operation to perform: 2
-----
View Patient Profile
-----

Enter Patient ID to view patient details: 4848
-----
4848          Saman_Perera          saman@mail.com          12/A,Kadawatha          715667878 45
-----

```

### 4.6.3. View Doctor List

```

-----
Available Doctors and Time Schedules
-----

```

Doctor ID	Doctor Name	Specialization	Specialization ID	Available Time	Date	Doctor Charges
2132	S.K.Nayakkara	Cadiology	211	13:20:00	2023-03-04	5500.0
2222	Amal	Surgeon	888	12:20:12	2000-12-23	2300.0
5555	D.S.Kamal	Surgeon	666	12:20:00	2023-05-25	2500.0
6655	A.J.Ekanayake	Surgeon	222	14:20:00	2023-05-25	6000.0
8871	T.S.K.Perera	Cardiologists	2	11:36:24	2023-05-03	5000.0
9911	A.S.A.Ranasinghe	Gynecologist	6	11:36:24	2023-05-03	98000.0
9990	Rushini_Fonseka	Surgeon	999	18:20:00	2020-05-08	5400.0

```

-----

```

#### 4.6.4. View Doctor Profile

```

Enter the number of the operation to perform: 4
-----
View Doctor Profile
-----

Enter Doctor ID to view Doctor details: 6655
-----
6655          A.J.Ekanayake      Surgeon          222          14:20:00          2023-05-25          6000.0
-----
Reviews for Doctor ID: 6655
-----
Review 1: Highly_recommended
-----
Review 2: Treatments_are_superb
-----
Review 3: Good_service
-----
Review 4: Very_good_treatments
-----

```

#### 4.6.5. Appointment Placement

```

Enter the number of the operation to perform: 5
-----
Appointment Placement
-----

Please enter your patient ID: 4848
Enter ID of the Doctor you want to consult: 6655
Doctor you choose: A.J.Ekanayake
-----
You have successfully placed the Appointment
-----

Do you want to get your appointment receipt? (y/n)y
-----

YOUR APPOINTMENT DETAILS
-----

PATIENT DETAILS
Patient Name: Saman_Perera
Patient Email: saman@mail.com
Patient Address: 12/A,Kadawatha
Patient PhoneNumber: 715667878

DOCTOR DETAILS
Doctor Reserved: A.J.Ekanayake
Doctor Specialization: Surgeon
Doctor Consultation Fee: 6000.0
-----

APPOINTMENT DETAILS
Appointment Time: 14:20:00
Appointment Date: 2023-05-25
-----

```



Do you want to view your Final Bill? (y/n) **y**

-----  
 YOUR FINAL BILL  
 -----

Doctor Consultation Fee: 6000.0  
 Hospital Charges: 3000.0  
 Medical Tests Charges: 1000.0  
 -----

Total Charges: 10000.0  
 -----

#### 4.6.6. Rate the Doctor

Enter the number of the operation to perform: **5**

-----  
 Rate Doctors  
 -----

You should know Doctor ID before reviewing the doctor

Enter Your ID: **4848**

Enter Doctor ID: **6655**

Doctor you choose: A.J.Ekanayake

Enter Your Review: **Excellent\_Treatments**

Review added successfully!

## 4.7. Admin Functionalities

### 4.7.1. View All Patients

```

Enter the portal number you want to enter: 2
-----
Welcome to Admin Portal
-----
1. View All Patients
2. Delete Patient Profile
3. View All Doctors
4. Delete Doctor Profile

Enter the number of the operation to perform: 1
-----
Patient Details
-----
Patient ID      Patient Name      Email              Address            Phone      Age
999             S.D.Gamage        gamage@gmail.com   Wattala            718998989  45
1005            Saman              saman@mail.com     Colombo2           789009090  34
2343            Lochani_Ranasinghe loch@mail.com      Ragama             778558585  23
2345            Wasana             wasana@gmail.com   Nuwaraeliya        712345678  12
2434            Ashen_Fonseka      ashen@mail.com     Gampaha            712523612  45
4343            Kusum_Ranasinghe   kusum@mail.com     35/B,MainRd,Kadawatha 717887878  54
4848            Saman_Perera       saman@mail.com     12/A,Kadawatha     715667878  45
6399            Sunera_Akash       sunera@mail.com    Colombo            717889878  34
7421            Rushini_Gamage     rushi@mail.com     Ragama             767889098  34
7899            Amila_Perera       amila@mail.com     Gampaha            712346789  56
-----

```

### 4.7.2. Delete Patient Profile

```

-----
Welcome to Admin Portal
-----
1. View All Patients
2. Delete Patient Profile
3. View All Doctors
4. Delete Doctor Profile

Enter the number of the operation to perform: 2
-----
Delete Patient Profiles
-----

Enter the ID of the patient to delete: 7899
Successfully Deleted the Patient

```

### 4.7.3. View All Doctors

```

Enter the number of the operation to perform: 1
-----
                Available Doctors and Time Schedules
-----
Doctor ID      Doctor Name      Specialization      Specialization ID      Available Time      Date      Doctor Charges
-----
2132           S.K.Nayakkara      Cadiology           211                    13:20:00           2023-03-04      5500.0
2222           Amal               Surgeon             888                    12:20:12           2000-12-23      2300.0
5555           D.S.Kamal          Surgeon             666                    12:20:00           2023-05-25      2500.0
6655           A.J.Ekanayake      Surgeon             222                    14:20:00           2023-05-25      6000.0
8871           T.S.K.Perera       Cardiologists       2                      11:36:24           2023-05-03      5000.0
9911           A.S.A.Ranasinghe   Gynecologist        6                      11:36:24           2023-05-03      98000.0
9990           Rushini_Fonseka    Surgeon             999                    18:20:00           2020-05-08      5400.0
-----

```

### 4.7.4. Delete Doctor Profile

```

-----
                Welcome to Admin Portal
-----

1. View All Patients
2. Delete Patient Profile
3. View All Doctors
4. Delete Doctor Profile

Enter the number of the operation to perform: 4
-----

                Delete Doctor Profiles
-----

Enter the ID of the doctor to delete: 9990
Doctor with ID: 9990 Deleted Successfully!

```

## 4.8. Doctor Functionalities

### 4.8.1. Doctor Registration

```

-----
Welcome to Doctor Portal
-----

1. Doctor Registration
2. View Total Revenue

Enter the number of the operation to perform: 1
-----

Doctor Registration
**Please provide doctor details to proceed the registration...
-----

Enter Doctor ID: 5558
Enter Doctor Name: D.M.Ramanayake
Enter Doctor Specialization: Dermatologist
Enter Doctor SpecializationID: 666
Enter the available time slot (hh:mm:ss):
18:25:00
Enter the date (yyyy-mm-dd):
2023-06-16
Enter Doctor Fee: 5000.00
Successfully Added the Doctor
-----

```

### 4.8.2. View total Revenue of the Doctor

```

-----
Welcome to Doctor Portal
-----

1. Doctor Registration
2. View Total Revenue

Enter the number of the operation to perform: 2
Enter Doctor ID: 6655
-----

Final Revenue Report
-----

***** Consultation fee of Doctor per patient: 6000 *****

Names of all Patients Consulted:

Wasana
Amila_Perera
Sunera_Akash
Rushini_Gamage
Saman_Perera

-----

Total Number of Patients Consulted: 5
Total Revenue of the Doctor: 30000.0
-----

```

## 5. CHALLENGES

The challenges that we faced during our Java development are as follows,

- **Debugging:** One of the biggest challenges in our Java development was debugging. Finding and fixing errors was more time consuming and frustrating.
- **Performance:** Java is known for its performance, but poorly optimized code can lead to slow and inefficient programs. Optimizing code for speed and memory usage was also another challenge that we faced during our coding. This caused us a system breakdown in our computer while the code was running.
- **Maintaining Code Quality:** Maintaining code quality was also another major challenge that we faced, especially when working as a team. Each team member had their own perspectives and we managed to include almost all, while maintaining the code quality.
- **Security:** Java is also known for its security, but vulnerabilities can still exist in code if it is not properly secured. Ensuring that code is secure was also challenging, especially when dealing with patients' private details.

## 6. SOLUTIONS

As solutions to these challenges, we took several measures to overcome them. We tried debugging tools, such as Eclipse's integrated debugger, to help identify and fix errors. That was a new experience for our team. Also, we realized the importance of writing clean and readable code with clear comments to make it easier to debug and maintain the quality of the code. By using code reviews, automated testing, and other quality control measures, we ensured that code remains maintainable and reliable. These are just a few strategies to overcome the challenges that were raised during our Java coding. Apart from these solutions we also took a little advice from the experts as well.



## 7. FUTURE IMPLEMENTS

- In order to make this system more user friendly adding a Graphical user interface would be more beneficial. There are many Java GUI frameworks available, such as Swing, JavaFX, and SWT, we realized that would help us create a professional-looking GUI quickly and easily.
- Converting our Java code to a web application can allow users to access it from anywhere with an internet connection. There are many Java web frameworks available, such as Spring, Struts, and Play, that would build a web application quickly and easily.
- To make the Java code accessible on mobile devices, creating a mobile app is also another way of modification for our system. There are many frameworks available for creating cross-platform apps, such as React Native, Flutter, and Xamarin, that would help us to create an app that works on both iOS and Android.
- Also, the Java code can be upgraded to make it more accessible to users who speak different languages by adding support for multiple languages. There are many Java internationalization frameworks available, such as Resource Bundle and Message Format, that can help to create multilingual applications.
- Improving the security of our system can help to protect it from unauthorized access or malicious attacks. Adding security features such as encryption, authentication, and access control to our code can be done as well.
- Adding an appointment reminder system can help reduce no-shows and improve customer satisfaction. We can consider integrating our Java code with a third-party messaging service, such as Twilio or SendGrid, that can help us send automated SMS or email reminders to our users.

## 8.CONTRIBUTION

### 8.1. Source Code

Our source code has been posted to GitHub and is available to the public via the following link:

[https://github.com/LochaniRanasinghe/OOP\\_SelfHealthCareSystem.git](https://github.com/LochaniRanasinghe/OOP_SelfHealthCareSystem.git)

Our source code is also available via the drive link.

[https://drive.google.com/file/d/1Y9VIDu5neLtL8NillwjOqJvb-y6Q5Ph8/view?usp=share\\_link](https://drive.google.com/file/d/1Y9VIDu5neLtL8NillwjOqJvb-y6Q5Ph8/view?usp=share_link)

Anyone can view the code and contribute to the project if anyone wishes to do so.

## 9. Teamwork

PS Numbers	Name	Work
PS/2020/103	H.R.A.L.N.Ranasinghe	Main programmer Contribution to the presentation
PS/2020/150	R.Y.W Ekanayaka	Supportive Programmer Contribution to the presentation Research on the system
PS/2020/215	R.N.R.Fonseka	Supportive Programmer Contribution to the presentation Create the content of the report
PS/2020/192	M.R.T Fernando	Supportive Programmer Contribution to the presentation Create the content of the report
PS/2020/174	J.A.D.N Jayakody	Supportive Programmer Design the presentation and report