

Criterion C: Development

Techniques used:

The following techniques were used to develop the final product:

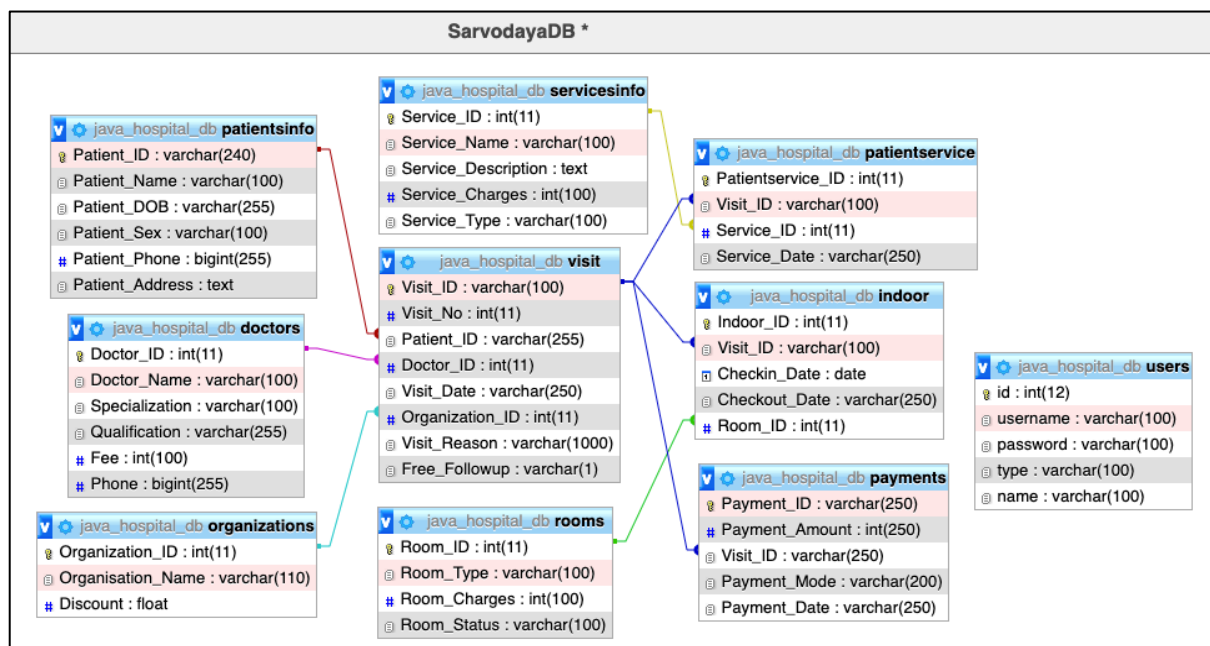
- Creating a locally hosted database and connecting it to the program
- Developing a user-friendly GUI Interface
- Providing security and authentication to the product
- Writing algorithms and methods using Java
 - Importing Java libraries
 - Defining variables and functions
 - Inheritance from javax.swing.JFrame class
 - Using the event listener interface
 - Modular Programming
 - File writing
- Creating SQL queries to extract, add, update, and delete information from the database

I will now lay out the step by step development process of the product while highlighting the techniques mentioned above.

Creating the database and connecting it to the program

Creation of the database:

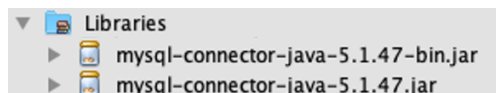
XAMPP is an open source web server solution which was used to host the MySQL Database and the Apache Web Server. This allowed me to use the GUI tool, phpMyAdmin, to create and manage the MySQL database. This relational database created was in 3NF to reduce data redundancy and avoid insertion, update and delete anomalies. The following tables and relationships were created:



Primary keys and foreign keys were created to establish the relationships and each table has its own primary key. The most important table in the database is the visit table since it provides the unique Visit ID for each patient visit which is used to track opd patients, indoor patients, service records and payment records. The users table is created for the users of the product to store the login information.

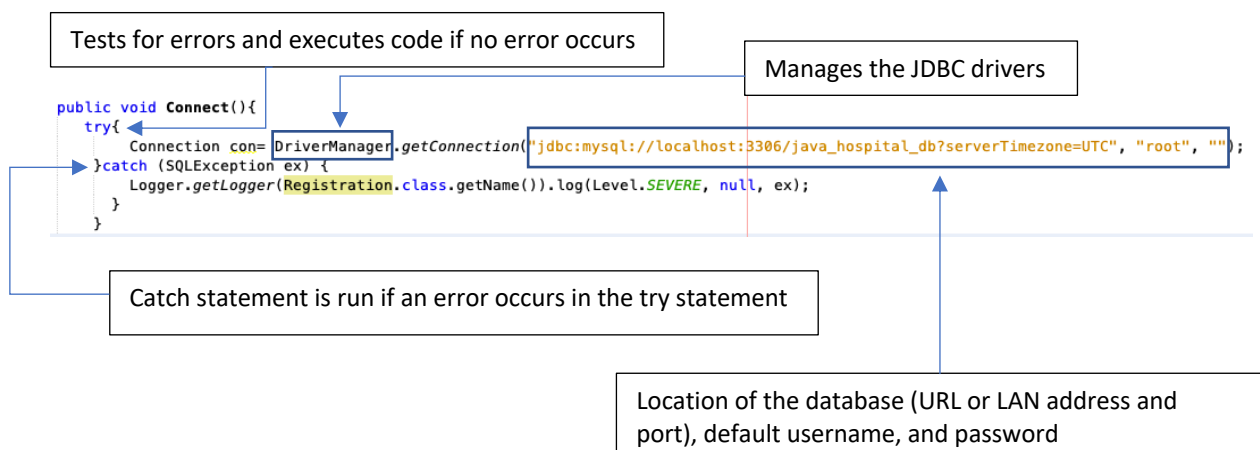
Connecting the database to the program:

In order to connect the java program to the MySQL database, the connector was added to the libraries and java.sql.Connection and java.sql.DriverManager libraries were imported.



```
import java.sql.Connection;
import java.sql.DriverManager;
```

The following code was written to establish the connection. The try-catch statement ensures that the exceptions are handled and the data isn't lost.



This connection is used throughout the program to ensure that the database is connected and is also used to create queries which will be discussed later.

Developing the user-friendly GUI Interface

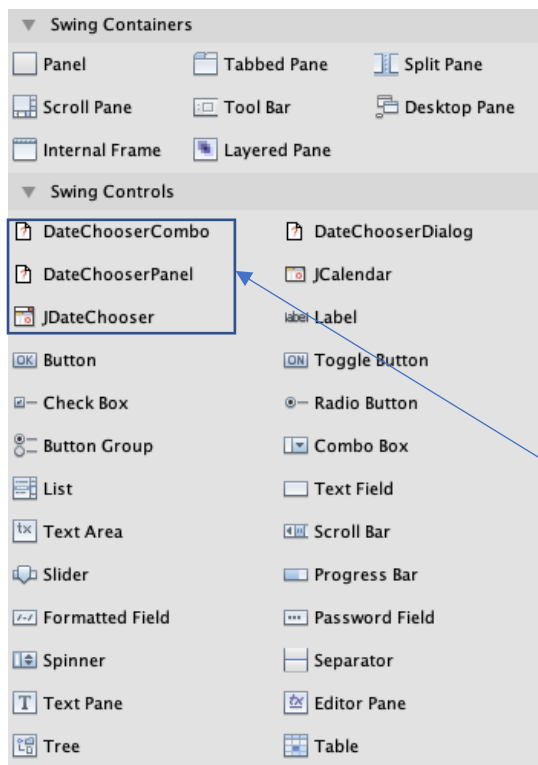
The Java Swing Library was used to create the GUI interface for my product. This allowed me to add components like buttons, text fields, combo boxes, check boxes, labels and lists to the product to make it interactive. NetBeans IDE provides several inbuilt GUI functions which allowed me to customise my product and all the swing components were automatically added to the code. All the classes were extended to the class `javax.swing.JFrame`. Thus, inheritance was used here as all classes inherited this class which allowed them to access the GUI components.

```
// Variables declaration - do not modify
private javax.swing.JLabel ID_Label;
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.ButtonGroup buttonGroup2;
private javax.swing.ButtonGroup buttonGroup3;
private javax.swing.JButton jButtonClear;
private javax.swing.JButton jButtonDelete;
private javax.swing.JButton jButtonSave;
```

Example of swing components declaration

```
*/
public class Registration extends javax.swing.JFrame
```

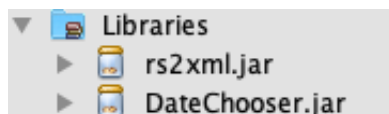
Class inherits javax.swing.JFrame class



GUI components in NetBeans' Palette which can be added to the graphical design

JDateChooser downloaded and added to the swing controls.

Apart from the existing components, the JDateChooser was added to the libraries to add a Date chooser component to allow the receptionist to choose the date.

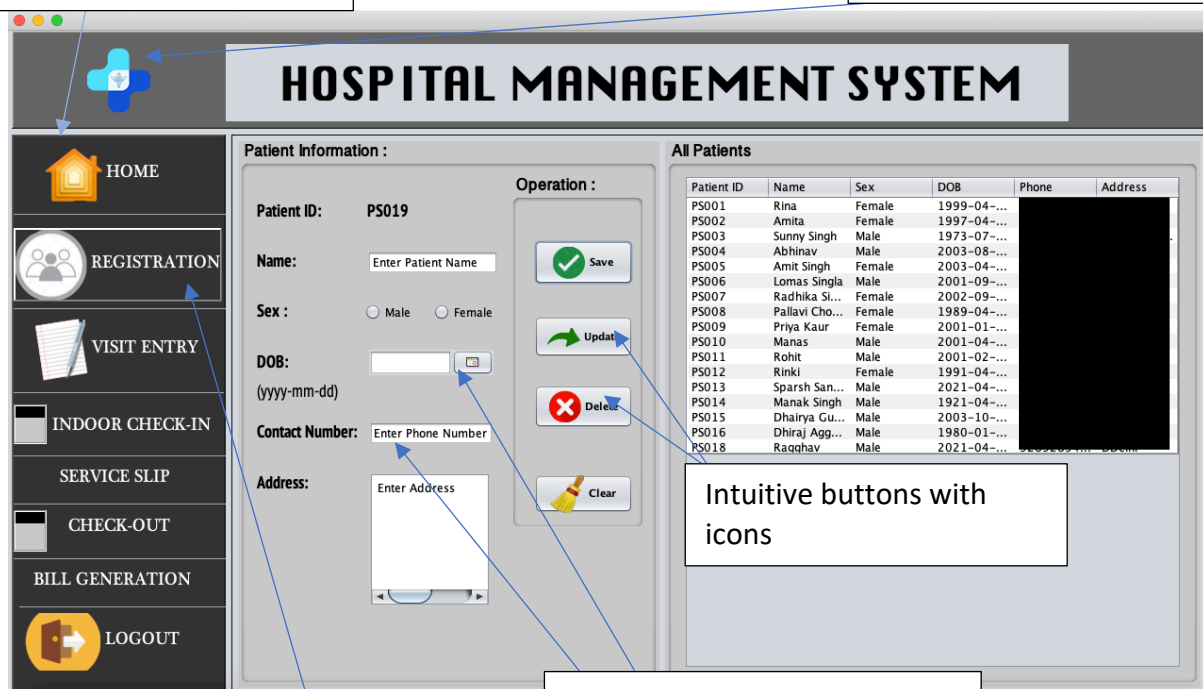


JDateChooser in libraries

The graphical design of the user interface is shown below

Navigation pane to allow user to switch between windows

Hospital logo displayed



Intuitive buttons with icons

Highlighting the page that is open

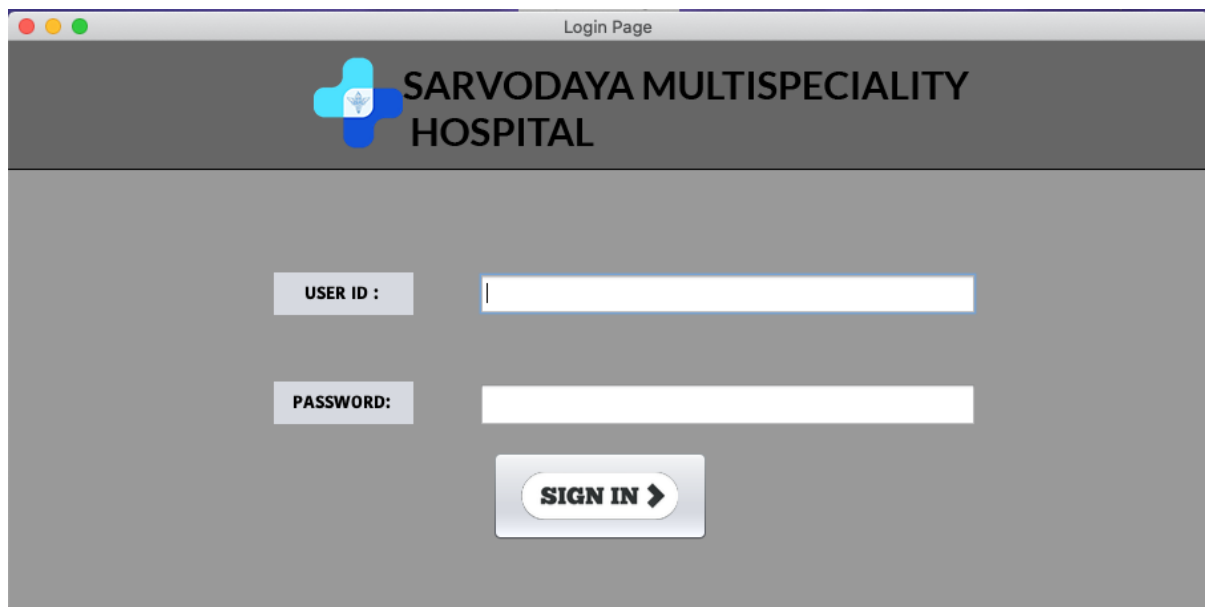
Text fields, date chooser and buttons for easy data entry

Developing all the forms and classes

Several classes were required for the development of the product as outlined in the design overview. The design overview made it easy for me to write the code in Java.

1) Login Page (Providing security)

The security of the data and the functions of the product was essential to the client. A login page was specifically requested with different credentials for the receptionist and the director.



Final Login Page Developed

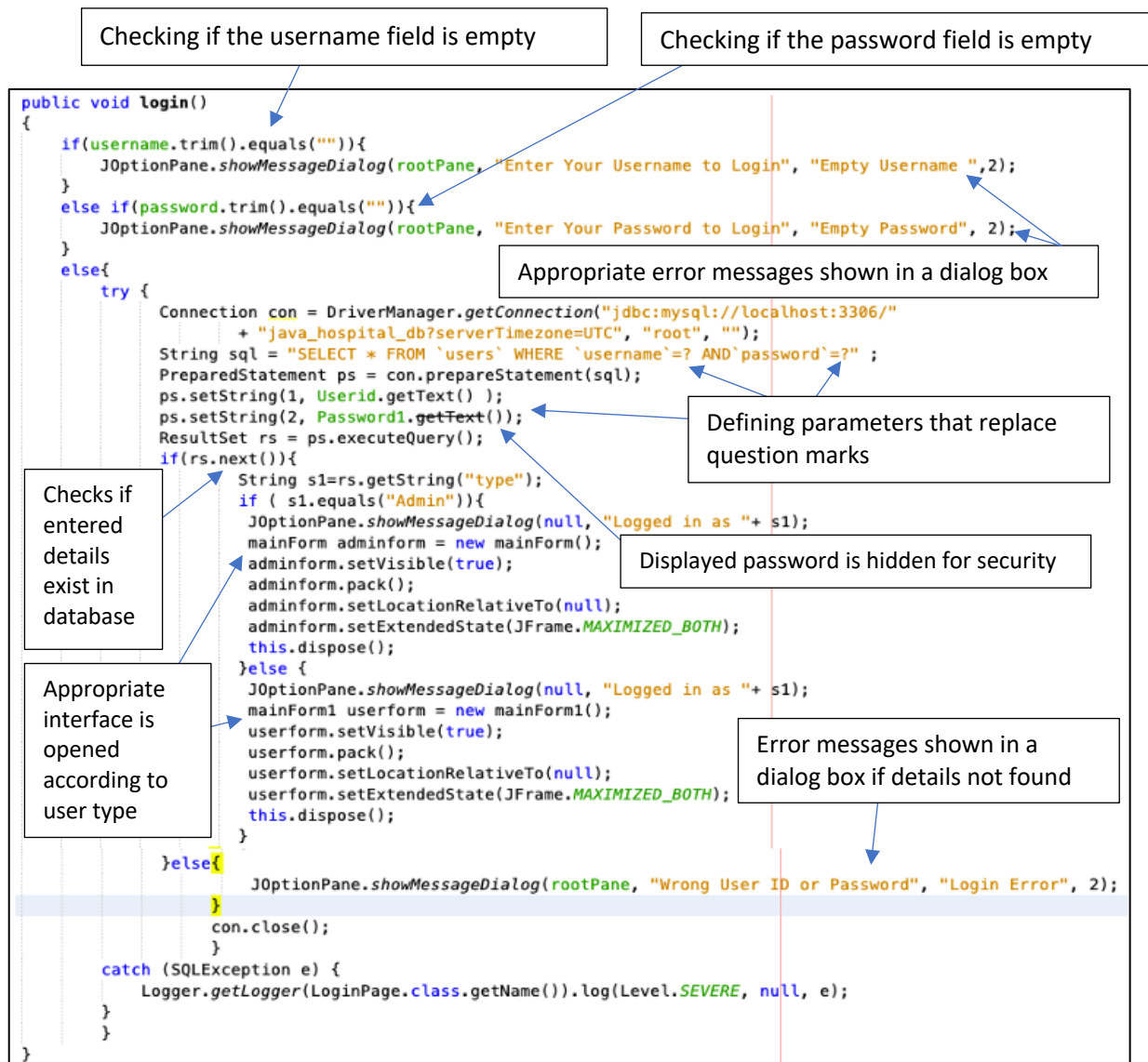
The users table in the database stored the usernames and passwords and the type of user.

username	password	type	name
reception	reception	User	Receptionist
director	director	Admin	Dr. Ajay Gupta

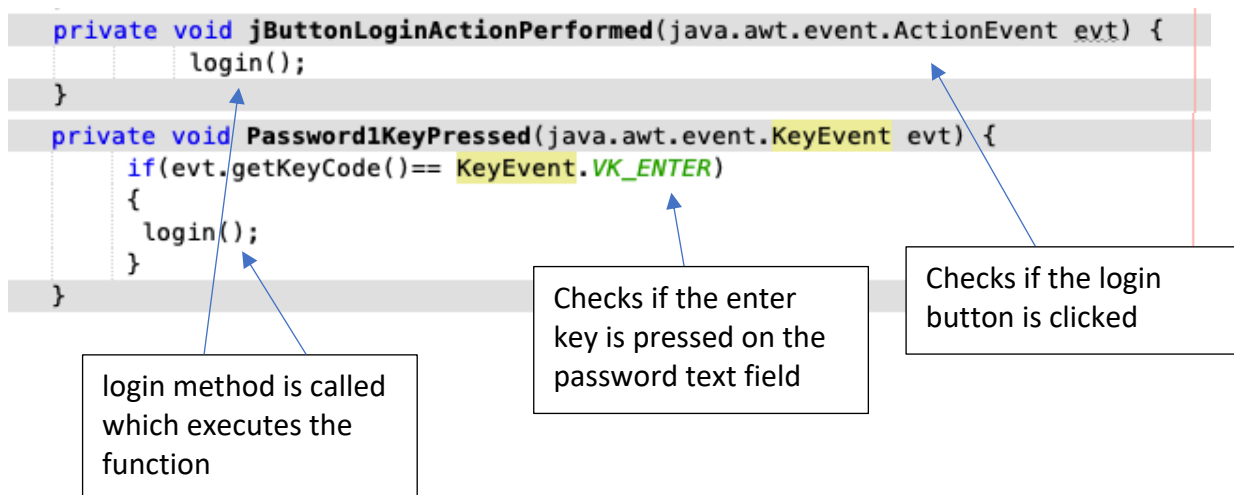
← users table

SQL is a higher level programming language used to manage data in relational databases. As mentioned in the techniques, SQL was used throughout the program. In the LoginPage Class, an SQL Statement allows the program to retrieve the username and password from the database and compare it to the entered credentials.

The following method was created to allow the user to login and be redirected to the respective interfaces:



The following event listeners were added to the login button and the password text field so that the login method is executed when the button is clicked or when the enter key is pressed.



The following libraries were imported for this class:

```

import java.awt.event.KeyEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
  
```

Allows the enter key event

Allows dynamic and parametric queries

Allows the dialogue boxes to be displayed

2) Registration Form Class

This class allows the receptionist to add, update, delete or view patients. Complex SQL queries and java methods and algorithms were used here.

- ❖ Firstly, a function was created which displays a JTable with all the records. `javax.swing.table.DefaultTableModel` was imported to define the table model. A `Vector` was initialized in order to store the data to be entered into the table. A vector is essentially a dynamic array with unlimited size. This is important because we do not know the number of rows here.

```

private void patient_table(){
    try {
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/java_hospital_db?serverTimezone=UTC", "root", "");
        PreparedStatement pst;
        ResultSet rs;
        pst=con.prepareStatement("select * from patientsinfo ");
        rs=pst.executeQuery();
        ResultSetMetaData Rsm= rs.getMetaData();
        int c;
        c=Rsm.getColumnCount();
        DefaultTableModel df = (DefaultTableModel)jTable1.getModel();
        df.setRowCount(0);
        while (rs.next()){
            Vector v2= new Vector();
            for(int i = 1; i<=c; i++)
            {
                v2.add(rs.getString("Patient_ID"));
                v2.add(rs.getString("Patient_Name"));
                v2.add(rs.getString("Patient_Sex"));
                v2.add(rs.getString("Patient_DOB"));
                v2.add(rs.getString("Patient_Phone"));
                v2.add(rs.getString("Patient_Address"));
            }
            df.addRow(v2);
        }
        catch (SQLException ex) {
            Logger.getLogger(Registration.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
  
```

Query to extract data from patientsinfo table in the database

Retrieving information about the table then assigning the number of columns in the table to a variable 'c'

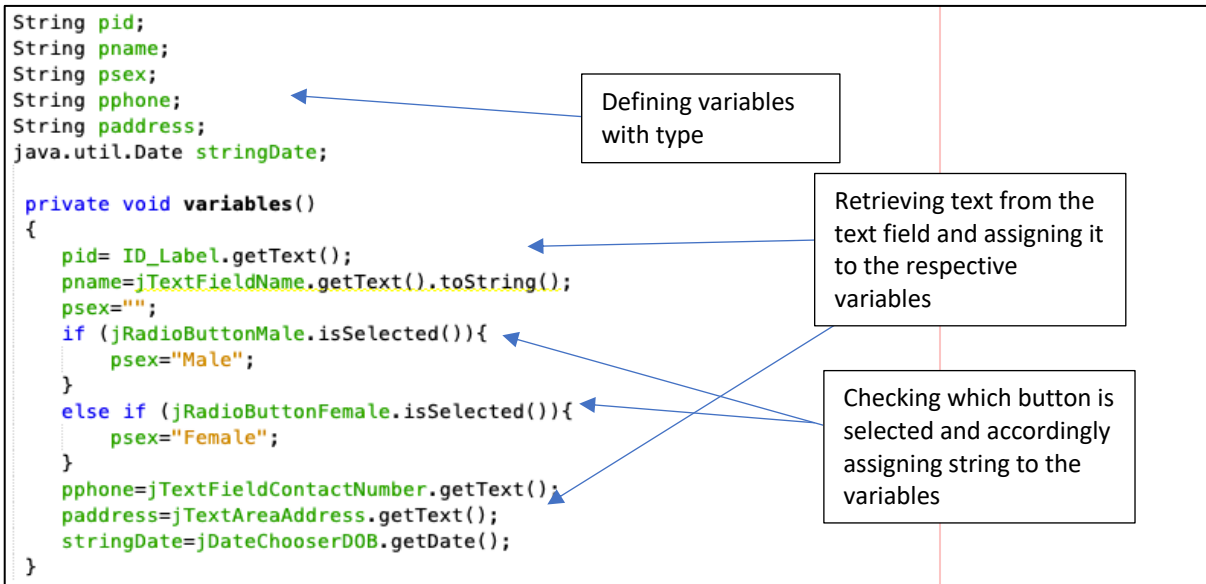
Inserts the data into the vector 'v2'

Initializing the table and passing the data and column name to vector type

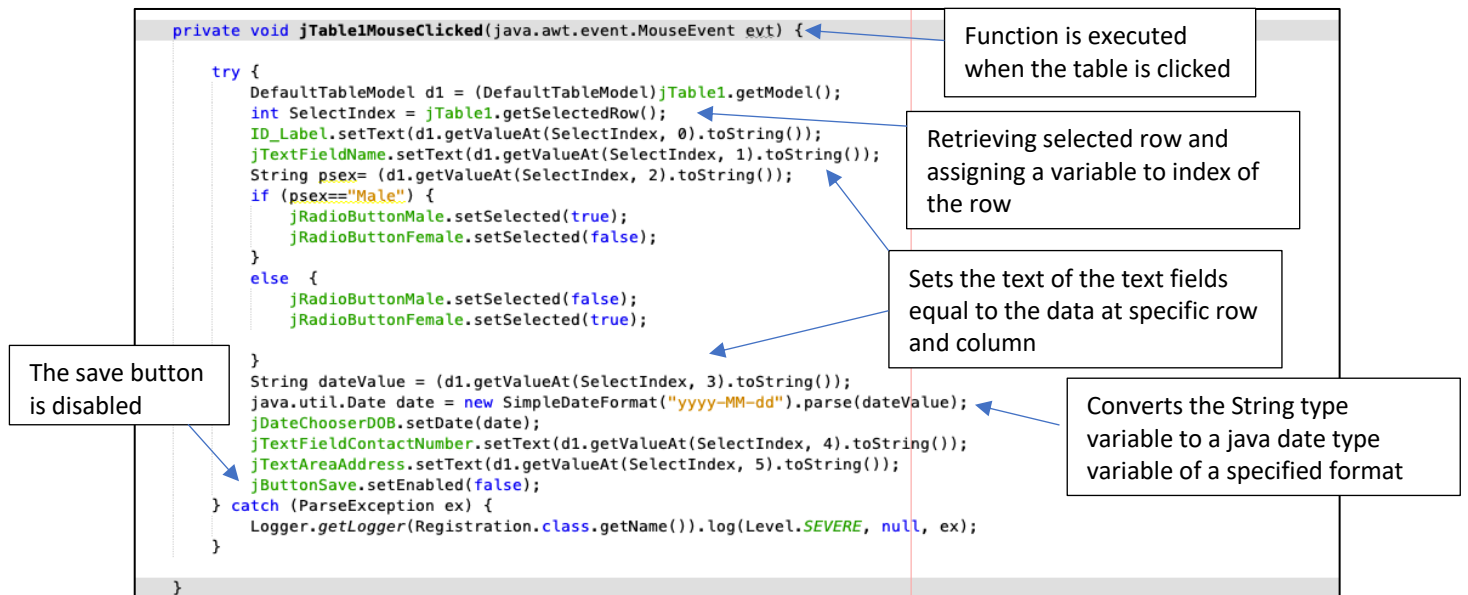
Inserts the data from the vector 'v2' into the table's row

Loop till a record exists in the database

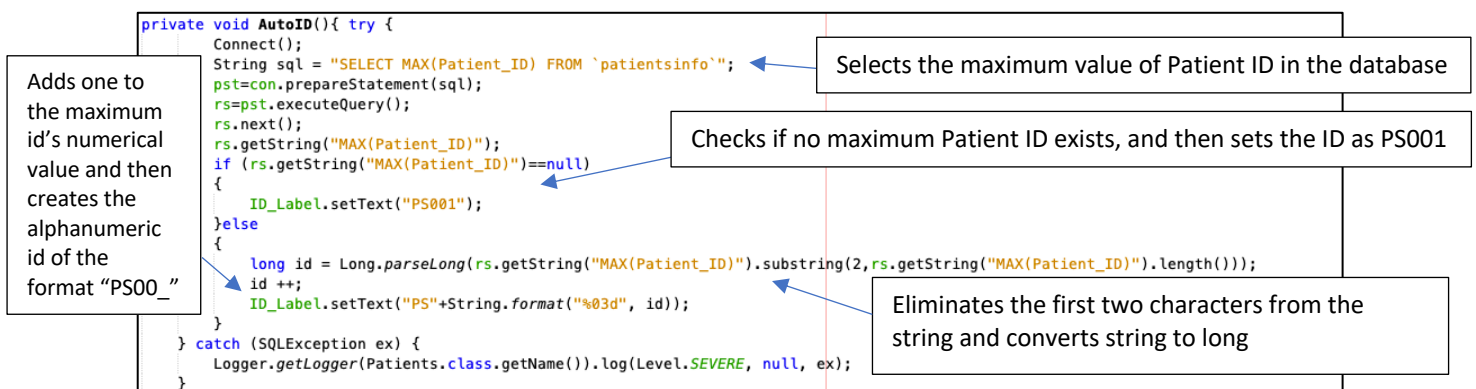
Loop till all columns are added



- ❖ When a row is selected from the table, the data should be displayed in the text fields to be edited.



- ❖ Generate a new patient's ID:



❖ The following method was created to input the entered data into the database:

```
private void save(){
    variables();
    if(
        pname.trim().equals("") || pname.trim().equals("Enter Patient Name") ||
        psex.trim().equals("") || pphone.trim().equals("Enter Phone Number") ||
        pphone.trim().equals("") || paddress.trim().equals("") )
    {
        JOptionPane.showMessageDialog(null, "All fields are compulsory "
            + " Kindly check all fields again","Error",JOptionPane.ERROR_MESSAGE);
    }
    else if (stringDate==null)
    {
        JOptionPane.showMessageDialog(null, "All fields are compulsory "
            + " Choose a valid date","Error",JOptionPane.ERROR_MESSAGE);
    }
    else
    {
        try {
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/"
                + "java_hospital_db?serverTimezone=UTC", "root", "");
            PreparedStatement pst;
            ResultSet rs;
            java.sql.Date sqlDate=new java.sql.Date(jDateChooserDOB.getDate().getTime());
            pst=con.prepareStatement("INSERT INTO `patientsinfo`(`Patient_ID`,`Patient_Name`,`Patient_Sex`,`Patient_DOB`,`Patient_Phone`,`Patient_Address`) VALUES (?, ?, ?, ?, ?, ?)");
            pst.setString(1, ID_Label.getText());
            pst.setString(2, jTextFieldName.getText());
            pst.setString(3, psex);
            pst.setDate(4, sqlDate);
            pst.setString(5, jTextFieldContactNumber.getText());
            pst.setString(6, paddress);
            int i = pst.executeUpdate();
            if(i>0){
                JOptionPane.showMessageDialog(null,"New Patient "+ pname+ " Saved Successfully");
                patient_table();
                AutoID();
                clear();
            }
            else {
                JOptionPane.showMessageDialog(null, pname+" 's Information not saved successfully");
            }
        } catch (SQLException e) {
            Logger.getLogger(Patients.class.getName()).log(Level.SEVERE, null, e);
            JOptionPane.showMessageDialog(null, pname+" 's Information not saved successfully");
        }
    }
}
```

Checking if no data is entered

Error message is shown asking user to try again

Query to insert data into the table

Appropriate message is shown

The displayed table is updated, automatic id is generated, and text fields are cleared.

❖ The following method was created to clear the text fields:

```
private void clear()
{
    AutoID();
    jButtonSave.setEnabled(true);
    jTextFieldName.setText("");
    buttonGroup1.clearSelection();
    jTextFieldContactNumber.setText("");
    jTextAreaAddress.setText("");
    jDateChooserDOB.setDate(null);
}
```

Save button is enabled.

All text fields are set to blank, and the selected date and sex are removed

❖ The following method was created to update the data in the database:

```
private void update(){
    variables();
    if(
        pname.trim().equals("") || pname.trim().equals("Enter Patient Name") ||
        psex.trim().equals("") || pphone.trim().equals("Enter Phone Number") ||
        pphone.trim().equals("") || paddress.trim().equals("") )
    {
        JOptionPane.showMessageDialog(null, "All fields are compulsory "
            + " Kindly check all fields again","Error",JOptionPane.ERROR_MESSAGE);
    }else if (stringDate==null)
    {
        JOptionPane.showMessageDialog(null, "All fields are compulsory "+
            " Choose a valid date","Error",JOptionPane.ERROR_MESSAGE);
    }else
    {
        try {
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/"
                + "java_hospital_db?serverTimezone=UTC", "root", "");
            PreparedStatement pst;
            ResultSet rs;
            java.sql.Date sqlDate=new java.sql.Date(jDateChooserDOB.getDate().getTime());
            pst=con.prepareStatement("Update `patientsinfo` set Patient_Name = ?, "
                + " `Patient_Sex` = ?, `Patient_DOB`=?, `Patient_Phone`=?, `Patient_Address`=? where Patient_ID=?");
            pst.setString(1, pname);
            pst.setString(2, psex);
            pst.setDate(3, sqlDate);
            pst.setString(4, pphone);
            pst.setString(5, paddress);
            pst.setString(6, pid);
            int i = pst.executeUpdate();
            if(i>0){
                JOptionPane.showMessageDialog(null,"Patient "+ pname+ " Information Updated Successfully");
                patient_table();
                AutoID();
                clear();
            }else {
                JOptionPane.showMessageDialog(null, pname+ " 's Information not updated successfully");
            }
        } catch (SQLException e) {
            Logger.getLogger(Patients.class.getName()).log(Level.SEVERE, null, e);
            JOptionPane.showMessageDialog(null, pname+" 's Information not updated successfully");
        }
    }
}
```

Query to update database

❖ The following method was created to delete the selected record:

```
private void delete(){
    String pid=ID_Label.getText();
    String pname= jTextFieldName.getText();
    try {
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/java_hospital_db?serverTimezone=UTC", "root", "");
        PreparedStatement pst;
        ResultSet rs;
        java.sql.Date sqlDate=new java.sql.Date(jDateChooserDOB.getDate().getTime());
        pst=con.prepareStatement("delete from `patientsinfo` where Patient_ID=?");
        pst.setString(1, pid);
        int i = pst.executeUpdate();
        if(i>0){
            JOptionPane.showMessageDialog(null,"Patient "+ pname+ " Information Deleted Successfully");
            patient_table();
            AutoID();
            clear();
        }else {
            JOptionPane.showMessageDialog(null, "Information not deleted successfully");
        }
    } catch (SQLException e) {
        Logger.getLogger(Patients.class.getName()).log(Level.SEVERE, null, e);
        JOptionPane.showMessageDialog(null, "Information not deleted successfully");
    }
}
```

Query to delete the entry from database

- ❖ The event listeners `MouseClicked` and `ActionPerformed` were used to detect when the following buttons were clicked to call the required method to be executed:

```
private void jButtonUpdateMouseClicked(java.awt.event.MouseEvent evt) {
    update();
}

private void jButtonDeleteMouseClicked(java.awt.event.MouseEvent evt) {
    delete();
}

private void jButtonSaveActionPerformed(java.awt.event.ActionEvent evt) {
    save();
}

private void jButtonClearActionPerformed(java.awt.event.ActionEvent evt) {
    clear();
}
```

It must be evident by now that modular programming was used in the development of the code as different parts of the code were developed individually and methods that were created were then called by other parts of the code when needed. This allowed me to write all the methods separately and in order to change one function, only a block of code needed to be edited, thus it was a time-saving and efficient method.

3) Visit Entry Page

Visit Number	Patient	Doctor	Organization	Reason
1	Rina	Pradeep TI...	ICICI	Fever

```

public void visit_table(){
    try {
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/java_hospital_db?serverTimezone=UTC", "root", "");
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        String date = dateFormat.format(jDateChooser.getDate());
        PreparedStatement pst= con.prepareStatement("SELECT * FROM `visit` where Visit_Date= ?");
        pst.setString(1, date);
        ResultSet rs=pst.executeQuery();
        ResultSetMetaData Rsm= rs.getMetaData();
        int c;
        c=Rsm.getColumnCount();
        DefaultTableModel df = (DefaultTableModel)jTable1.getModel();
        df.setRowCount(0);
        while (rs.next())
        {
            Vector v2= new Vector();
            for(int i = 1; i<=c; i++)
            {
                v2.add(rs.getString("Visit_No"));
                String pid=rs.getString("Patient_ID");
                pst1= con.prepareStatement("SELECT * FROM `patientsinfo` WHERE Patient_ID= ?");
                pst1.setString(1, pid);
                rs1=pst1.executeQuery();
                if (rs1.next()==true)
                {
                    v2.add(rs1.getString("Patient_Name"));
                }
                String did=rs.getString("Doctor_ID");
                pst2= con.prepareStatement("SELECT * FROM `doctors` WHERE Doctor_ID= ?");
                pst2.setString(1, did);
                rs2=pst2.executeQuery();
                if (rs2.next()==true)
                {
                    v2.add(rs2.getString("Doctor_Name"));
                }
                String oid = rs.getString("Organization_ID");
                pst3= con.prepareStatement("SELECT * FROM organizations where Organization_ID= ?");
                pst3.setString(1, oid);
                rs3=pst3.executeQuery();
                if (rs3.next()==true)
                {
                    v2.add(rs3.getString("Organisation_Name"));
                }
                v2.add(rs.getString("Visit_Reason"));
            }
            df.addRow(v2);
        }
    } catch (SQLException ex) {
        Logger.getLogger(Registration.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Retrieving patient id, doctor id, organization id from the visit table

Query to retrieve visit data for the date

Looping till visits exist in the database for the date entered

Query to retrieve patient information data for the patient

Query to retrieve doctor information for the doctor

Query to organization information for the organization

View Visit Table

```

public void AutoNo(){
    try {
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/java_hospital_db?serverTimezone=UTC", "root", "");
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        String date = dateFormat.format(jDateChooser.getDate());
        String sql = "SELECT MAX(Visit_No) FROM `visit` where Visit_Date= ?";
        pst=con.prepareStatement(sql);
        pst.setString(1, date);
        rs=pst.executeQuery();
        rs.next();
        rs.getString("MAX(Visit_No)");
        if (rs.getString("MAX(Visit_No)")==null)
        {
            NO_Label.setText("1");
        }else
        {
            int number = rs.getInt("MAX(Visit_No)");
            number++;
            NO_Label.setText(String.valueOf(number));
        }
    } catch (SQLException ex) {
        Logger.getLogger(Patients.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public void AutoID(){
    try {
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/java_hospital_db?serverTimezone=UTC", "root", "");
        String sql = "SELECT MAX(Visit_ID) FROM `visit`";
        pst=con.prepareStatement(sql);
        rs=pst.executeQuery();
        rs.next();
        rs.getString("MAX(Visit_ID)");

        if (rs.getString("MAX(Visit_ID)")==null)
        {
            ID_Label.setText("VS001");
        }else
        {
            long id = Long.parseLong(rs.getString("MAX(Visit_ID)").substring(2,rs.getString("MAX(Visit_ID)").length()));
            id ++;
            ID_Label.setText("VS"+String.format("%03d", id));
        }
    } catch (SQLException ex) {
        Logger.getLogger(Patients.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Query to extract maximum visit number for the selected date

Converting integer to string format

Automatically Generate Visit ID and Visit Number

```
String[] idarray = new String [50];
```

An array to store patient id is declared

```
public void LoadPatient()
{
    try {
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/java_hospital_db?serverTimezone=UTC", "root", "");
        PreparedStatement pst;
        ResultSet rs;
        pst=con.prepareStatement("select * from patientsinfo where Patient_name=?");
        pst.setString(1, jTextField1.getText());
        rs= pst.executeQuery();
        PatientList.removeAll();
        PatientList.setModel(new DefaultListModel());
        int x=0;
        while (rs.next())
        {
            DefaultListModel model = (DefaultListModel) PatientList.getModel();
            model.add(x, "Name:"+rs.getString(2)+". ID:"+rs.getString(1));
            idarray[x]=rs.getString(1);
            x=x+1;
        }
    } catch (SQLException ex) {
        Logger.getLogger(Visit.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Query to search for the entered patient name in the database

Creating model for the list of patients

The array stores the patient id at the same location as the index of the list

Searching for a Patient

```
public void LoadOrganization()
{
    try {
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/java_hospital_db?serverTimezone=UTC", "root", "");
        PreparedStatement pst;
        ResultSet rs;
        pst=con.prepareStatement("select*from organizations");
        rs= pst.executeQuery();
        organizationname.removeAll();
        while (rs.next())
        {
            organizationname.addItem(rs.getString(2));
        }
    } catch (SQLException ex) {
        Logger.getLogger(Visit.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Query to retrieve all organizations from database

Loop to add organizations names one by one

```
public void LoadDoctor()
{
    try {
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/java_hospital_db?serverTimezone=UTC", "root", "");
        PreparedStatement pst;
        ResultSet rs;
        pst=con.prepareStatement("select*from doctors");
        rs= pst.executeQuery();
        doctorname.removeAll();
        while (rs.next())
        {
            doctorname.addItem(new Doctor(rs.getString(1),rs.getString(2)));
        }
    } catch (SQLException ex) {
        Logger.getLogger(Visit.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Query to retrieve all doctors from database

Loop to add doctor names one by one

Inserting doctor and organization names into combo boxes

```
public Visit() {
    initComponents();
    Disable();
    LoadDoctor();
    LoadOrganization();
    AutoID();
    jButtononok2.setEnabled(false);
}
```

Disabling the second ok button

Methods called in the main class

```

public void Disable()
{
    jTextFieldDOB.setEnabled(false);
    jTextFieldSex.setEnabled(false);
    jTextFieldContactNumber.setEnabled(false);
    jTextAreaAddress.setEnabled(false);
}

```

Disabling the text fields

```

private void jButtonok2MouseClicked(java.awt.event.MouseEvent evt) {
    try{
        PreparedStatement pst;
        ResultSet rs;
        int x=PatientList.getSelectedIndex();
        String p = idarray[x];

        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/java_hospital_db?serverTimezone=UTC", "root", "");

        pst= con.prepareStatement("SELECT * FROM `patientsinfo` WHERE Patient_ID=?");
        pst.setString(1, p);
        rs=pst.executeQuery();
        rs.next();
        ID_Label1.setText(rs.getString("Patient_ID"));
        jTextFieldSex.setText(rs.getString("Patient_Sex"));
        jTextFieldDOB.setText(rs.getString("Patient_DOB"));
        jTextFieldContactNumber.setText(rs.getString("Patient_Phone"));
        jTextAreaAddress.setText(rs.getString("Patient_Address"));

    } catch (SQLException ex) {
        Logger.getLogger(Visit.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Retrieving the selected patient's index from the list and the stored id in the array

Setting text field data as the data from patientsinfo table in the database

Displaying Patient Information

```

private void jButtonSaveActionPerformed(java.awt.event.ActionEvent evt) {
    jButtonok1.setEnabled(true);
    Doctor d = (Doctor) doctorname.getSelectedItem();
    int x=PatientList.getSelectedIndex();
    String pid = idarray[x];
    Organization o = (Organization)organizationname.getSelectedItem();
    SimpleDateFormat dateformat = new SimpleDateFormat("yyyy-MM-dd");
    String date = dateformat.format(jDateChooser.getDate());
    String vreason=jTextFieldReason.getText();
    if(
        vreason.trim().equals("") )
    {
        JOptionPane.showMessageDialog(null, "All fields are compulsory "+
            " Kindly check all fields again","Error",JOptionPane.ERROR_MESSAGE);
    }else
    {
        try {
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/"
                + "java_hospital_db?serverTimezone=UTC", "root", "");
            pst=con.prepareStatement("INSERT INTO `visit`(`Visit_ID`, `Visit_No`, "
                + "`Patient_ID`, `Doctor_ID`, `Visit_Date`, `Organization_ID`, "
                + "`Visit_Reason`, `Free_Followup`) VALUES (?, ?, ?, ?, ?, ?, ?)");
            pst.setString(1, ID_Label1.getText());
            pst.setString(2, NO_Label1.getText());
            pst.setString(3, pid);
            pst.setString(4, d.id);
            pst.setString(5, date);
            pst.setString(6, o.id);
            pst.setString(7, vreason);
            if (jCheckBox1.isSelected()==true)
            {
                pst.setString(8, "1");
            }else
            {
                pst.setString(8, "0");
            }
        }
    }
}

```

Query to insert data

```

int i = pst.executeUpdate();
if(i>0){
    JOptionPane.showMessageDialog(null,"New Visit Saved Successfully");
    AutoID();
    AutoNo();
    jTextFieldReason.setText("");
    ID_Label1.setText("");
    doctorname.setSelectedIndex(-1);
    PatientList.setSelectedIndex(-1);
    organizationname.setSelectedIndex(-1);
    jCheckBox1.setSelected(false);
    jTextFieldDOB.setText("");
    jTextFieldSex.setText("");
    jTextFieldContactNumber.setText("");
    jTextFieldAddress.setText("");
    visit_table();
}
else {
    JOptionPane.showMessageDialog(null, "Information not saved successfully");
}
} catch (SQLException e) {
    Logger.getLogger(Patients.class.getName()).log(Level.SEVERE, null, e);
    JOptionPane.showMessageDialog(null, " 's Information not saved successfully");
}
}

```

Clearing fields
and updating
table

Saving visit data

4) Check-in Page

Room No.	Room Type	Patient
1	General	No Patient
2	General	No Patient
3	Private	No Patient
4	Private	Spursh Samwaria
5	Premium	No Patient
6	Premium	No Patient

```

public void indoor_table(){
    try {
        Connection con = java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/"
            + "java_hospital_db?serverTimezone=UTC", "root", "");
        PreparedStatement pst;
        ResultSet rs;
        pst=con.prepareStatement("Select* from rooms");
        rs=pst.executeQuery();
        ResultSetMetaData Rsm= rs.getMetaData();
        int c;
        c=Rsm.getColumnCount();
        DefaultTableModel df = (DefaultTableModel)jTable1.getModel();
        df.setRowCount(0);
        while (rs.next())
        {
            Vector v2= new Vector();
            for(int i = 1; i<=1; i++)
            {
                v2.add(rs.getString("Room_ID"));
                v2.add(rs.getString("Room_Type"));

                String Status = rs.getString("Room_Status");
                if (Status=="Vacant")
                {
                    v2.add("No Patient");
                }else
                {
                    PreparedStatement pst1;
                    ResultSet rs1;
                    pst1=con.prepareStatement("SELECT * FROM `indoor` WHERE Room_ID=? AND Checkout_Date=?");
                    pst1.setString(1, rs.getString("Room_ID"));
                    pst1.setString(2, "Pending");
                    rs1=pst1.executeQuery();
                    if (rs1.next()==false)
                    {
                        v2.add("No Patient");
                    }else
                    {

```

Checking if room is vacant and
setting Patient as "No Patient"

Retrieving the patient who is in the
room

Updating indoor table

```

        String pid= rs1.getString("Visit_ID");

        PreparedStatement pst2;
        ResultSet rs2;
        pst2=con.prepareStatement("SELECT * FROM `visit` WHERE Visit_ID=?");
        pst2.setString(1, pid);
        rs2=pst2.executeQuery();
        rs2.next();
        String ppid = rs2.getString("Patient_ID");
        PreparedStatement pst3;
        ResultSet rs3;
        pst3=con.prepareStatement("Select* from patientsinfo where Patient_ID=?");
        pst3.setString(1, ppid);
        rs3=pst3.executeQuery();
        rs3.next();
        v2.add(rs3.getString("Patient_Name"));
    }
    df.addRow(v2);
}
}
} catch (SQLException ex) {
    Logger.getLogger(CheckIN.class.getName()).log(Level.SEVERE, null, ex);
}
}

```

Retrieving patient id from visit table and patient name from patients info table

Updating indoor table (contd)

```

private void jButtonSaveActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Connection con = java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/"
            + "java_hospital_db?serverTimezone=UTC", "root", "");
        PreparedStatement pst;
        ResultSet rs;
        PreparedStatement pst1;
        ResultSet rs1;
        SimpleDateFormat dateformat = new SimpleDateFormat("yyyy-MM-dd");
        String date = dateformat.format(jDateChooser.getDate());
        Room r = (Room)Roombox.getSelectedItem();
        String sql = "SELECT MAX(Indoor_ID) FROM `indoor`";
        pst1=con.prepareStatement(sql);
        rs1=pst1.executeQuery();
        rs1.next();
        rs1.getString("MAX(Indoor_ID)");
        String IndoorID;
        if (rs1.getString("MAX(Indoor_ID)")==null)
        {
            IndoorID ="1";
        }
        else
        {
            int number = rs1.getInt("MAX(Indoor_ID)");
            number++;
            IndoorID= String.valueOf(number);
        }
        pst=con.prepareStatement("INSERT INTO `indoor`(`Indoor_ID`, `Visit_ID`, "
            + "`Checkin_Date`,`Checkout_Date`, `Room_ID`) VALUES (?, ?, ?, ?, ?)");
        pst.setString(1,IndoorID);
        pst.setString(2,ID_Label1.getText());
        pst.setString(3, date);
        pst.setString(4, "Pending");
        pst.setString(5, r.getId());
        int i = pst.executeUpdate();
        if(i>0){
            JOptionPane.showMessageDialog(null,"Check-In Successful");
            ID_Label1.setText("");
            jTextFieldReason.setText("");
            jTextFieldDOB.setText("");
            jTextFieldSex.setText("");
            jTextFieldContactNumber.setText("");
            jTextAreaAddress.setText("");
            jTextFieldDoctor.setText("");
            jTextFieldOrganization.setText("");
            PatientName.setSelectedIndex(-1);
            Roombox.setSelectedIndex(-1);
            PreparedStatement pst2;
            ResultSet rs2;
            pst2=con.prepareStatement("UPDATE `rooms` SET `Room_Status`= ? WHERE Room_ID=?");
            pst2.setString(1, "Active");
            pst2.setString(2, r.getId());
            int j = pst2.executeUpdate();
            if(j>0){
                indoor_table();
            }
        }
        else {
            JOptionPane.showMessageDialog(null, "Information not saved successfully");
        }
    } catch (SQLException e) {
        Logger.getLogger(Patients.class.getName()).log(Level.SEVERE, null, e);
        JOptionPane.showMessageDialog(null, " 's Information not saved successfully");
    }
}

```

Query to insert data into indoor table and setting checkout date as "Pending"

Query to update rooms table

Checking In a patient

5) Checkout Page:

```
private void jTable4MouseClicked(java.awt.event.MouseEvent evt) {
    try {
        Connection con = java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/"
            + "java_hospital_db?serverTimezone=UTC", "root", "");
        PreparedStatement pst;
        ResultSet rs;
        DefaultTableModel d1 = (DefaultTableModel) jTable4.getModel();
        int SelectIndex = jTable4.getSelectedRow();
        String name = d1.getValueAt(SelectIndex, 2).toString();
        String id = d1.getValueAt(SelectIndex, 0).toString();
        jLabel14.setText(name);
        String sql = "Select Visit_ID from indoor where Room_ID=? AND Checkout_Date=?";
        pst = con.prepareStatement(sql);
        pst.setString(1, id);
        pst.setString(2, "Pending");
        rs = pst.executeQuery();
        rs.next();
        String vid = rs.getString("Visit_ID");
        ID_Label12.setText(vid);

        PreparedStatement pst1 = con.prepareStatement("SELECT * FROM `visit` WHERE `Visit_ID`=?");
        pst1.setString(1, vid);
        ResultSet rs1 = pst1.executeQuery();
        String pid = rs1.getString("Patient_ID");
        String did = rs1.getString("Doctor_ID");
        String oid = rs1.getString("Organization_ID");
        Reason.setText(rs1.getString("Visit_Reason"));
        String sql2 = "Select * from patientsinfo where Patient_ID=?";

        PreparedStatement pst2 = con.prepareStatement(sql2);
        pst2.setString(1, pid);
        ResultSet rs2 = pst2.executeQuery();
        rs2.next();
        String sql3 = "Select * from doctors where Doctor_ID=?";

        PreparedStatement pst3 = con.prepareStatement(sql3);
        pst3.setString(1, did);
        ResultSet rs3 = pst3.executeQuery();
        rs3.next();
        String sql4 = "Select * from organizations where Organization_ID=?";

        PreparedStatement pst4 = con.prepareStatement(sql4);
        pst4.setString(1, oid);
        ResultSet rs4 = pst4.executeQuery();
        rs4.next();
        DOB.setText(rs2.getString("Patient_DOB"));
        Phone.setText(rs2.getString("Patient_Phone"));
        Sex.setText(rs2.getString("Patient_Sex"));
        Address.setText(rs2.getString("Patient_Address"));
        Doctor.setText(rs3.getString("Doctor_Name"));
        Org.setText(rs4.getString("Organisation_Name"));
    } catch (SQLException ex) {
        Logger.getLogger(CheckOUT.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Event listener applied when the Table is clicked and retrieving selected row

Retrieving patient's data from different tables

Display patient details when patient is clicked

```

private void jButtonSaveActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Connection con = java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/"
            + "java_hospital_db?serverTimezone=UTC", "root", "");
        PreparedStatement pst;
        ResultSet rs;
        PreparedStatement pst1;
        ResultSet rs1;
        SimpleDateFormat dateformat = new SimpleDateFormat("yyyy-MM-dd");
        String date = dateformat.format(jDateChooser.getDate());
        DefaultTableModel d1 = (DefaultTableModel)jTable4.getModel();
        int SelectIndex = jTable4.getSelectedRow();
        if (jTable4.getSelectedRow() == -1)
        {
            JOptionPane.showMessageDialog(null, "Chose a date and a room", "Error", HEIGHT);
        }
        else {
            String rid = d1.getValueAt(SelectIndex, 0).toString();
            String sql = "Select * from indoor where Room_ID=? AND Checkout_Date=?";
            PreparedStatement pst2 = con.prepareStatement(sql);
            pst2.setString(1, rid);
            pst2.setString(2, "Pending");
            ResultSet rs2 = pst2.executeQuery();
            rs2.next();
            try {
                Date d3 = dateformat.parse(date);
                Date d4 = dateformat.parse(rs2.getString("Checkin_Date"));
                if (d3.compareTo(d4) >= 0) {
                    pst1 = con.prepareStatement("Update indoor set Checkout_Date= ? where Room_ID=? and Checkout_Date=?");
                    pst1.setString(1, date);
                    pst1.setString(2, rid);
                    pst1.setString(3, "Pending");
                    int i = pst1.executeUpdate();
                    pst = con.prepareStatement("UPDATE `rooms` SET `Room_Status`=? WHERE `Room_ID`=?");
                    pst.setString(1, "Vacant");
                    pst.setString(2, rid);
                    int j = pst.executeUpdate();
                    indoor_table();
                    JOptionPane.showMessageDialog(null, "Check-out Successful");
                    DOB.setText("");
                    Phone.setText("");
                    Sex.setText("");
                    Address.setText("");
                    Doctor.setText("");
                    Org.setText("");
                    Reason.setText("");
                    ID_Label12.setText("");
                    ID_Label.setText("");
                    jLabel14.setText("");
                }
                else {
                    JOptionPane.showMessageDialog(null, "Select a date after Check-in Date", "Error", HEIGHT);
                }
            }
            catch (ParseException ex) {
                Logger.getLogger(CheckOUT.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
    catch (SQLException e) {
        Logger.getLogger(Patients.class.getName()).log(Level.SEVERE, null, e);
        JOptionPane.showMessageDialog(null, " 's Information not saved successfully");
    }
}

```

Checking if date entered is after the check-in date of the patient

Queries to update indoor and rooms table

Error message if date is before check-in date

Checkout a Patient

6) ServiceSlip Page: ServiceSlip form class was created to allot services to patients.

- ❖ The patients and the services displayed in the dropdown list must be according to the category of the patient selected (opd or indoor). Thus, two methods for each combo box were created.

```

public void LoadPatient()
{
    try {
        Connection con = java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/"
            + "java_hospital_db?serverTimezone=UTC", "root", "");
        PreparedStatement pst;
        ResultSet rs;
        SimpleDateFormat dateformat = new SimpleDateFormat("yyyy-MM-dd");
        String date = dateformat.format(jDateChooser.getDate());
        pst=con.prepareStatement("select * from visit where Visit_Date=?");
        pst.setString(1, date);
        rs= pst.executeQuery();
        PatientName.removeAllItems();
        while (rs.next())
        {
            String vid=rs.getString("Visit_ID");
            PreparedStatement pst2 = con.prepareStatement("Select * from indoor where Visit_ID=? ");
            pst2.setString(1,vid);
            ResultSet rs2=pst2.executeQuery();
            rs2.next();
            if (rs2.next()==false)
            {
                String pid= rs.getString("Patient_ID");
                PreparedStatement pst1;
                ResultSet rs1;
                pst1= con.prepareStatement("SELECT * FROM `patientsinfo` WHERE Patient_ID=?");
                pst1.setString(1, pid);
                rs1=pst1.executeQuery();
                if (rs1.next()==true)
                {
                    PatientName.addItem(new Patient(rs.getString(1),rs.getString(2)));
                }
            }
        }
    } catch (SQLException ex) {
        Logger.getLogger(Visit.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Checks if the patient's record does not exist in indoor

```

public void LoadPatient1()
{
    try {
        Connection con = java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/java_hospital_db?serverTimezone=UTC", "root",
        PreparedStatement pst;
        ResultSet rs;
        SimpleDateFormat dateformat = new SimpleDateFormat("yyyy-MM-dd");
        String date = dateformat.format(jDateChooser.getDate());
        pst=con.prepareStatement("select * from indoor where Checkout_Date=?");
        pst.setString(1, "Pending");
        rs= pst.executeQuery();
        PatientName.removeAllItems();
        while (rs.next())
        {
            try {
                String date1 = rs.getString("Checkin_Date");
                Date d3 = dateformat.parse(date);
                Date d4 = dateformat.parse(date1);
                if (d3.compareTo(d4)>=0){
                    PreparedStatement pst2 = con.prepareStatement("select * from visit where Visit_ID=?");
                    pst2.setString(1, rs.getString("Visit_ID"));
                    ResultSet rs2= pst2.executeQuery();
                    rs2.next();

                    PreparedStatement pst3 = con.prepareStatement("select * from patientsinfo where Patient_ID=?");
                    pst3.setString(1, rs2.getString("Patient_ID"));
                    ResultSet rs3= pst3.executeQuery();
                    rs3.next();
                    PatientName.addItem(new Patient(rs3.getString("Patient_ID"),rs3.getString("Patient_Name")));
                }
            } catch (ParseException ex) {
                Logger.getLogger(ServiceSlip.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    } catch (SQLException ex) {
        Logger.getLogger(Visit.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Retrieves all patients who are currently in a room

```

public void LoadService()
{
    try {
        Connection con = java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/"
            + "java_hospital_db?serverTimezone=UTC", "root", "");
        PreparedStatement pst;
        ResultSet rs;

        pst=con.prepareStatement("select * from servicesinfo");
        rs= pst.executeQuery();
        ServiceBox.removeAllItems();
        while (rs.next())
        {
            ServiceBox.addItem(rs.getString(2));
        }
    } catch (SQLException ex) {
        Logger.getLogger(Visit.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public void LoadService1()
{
    try {
        Connection con = java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/java_hospital_db?serverTimezone=UTC", "root",
        PreparedStatement pst;
        ResultSet rs;
        pst=con.prepareStatement("select * from servicesinfo where Service_Type= ?");
        pst.setString(1, "General");
        rs= pst.executeQuery();
        ServiceBox.removeAllItems();
        while (rs.next())
        {
            ServiceBox.addItem(rs.getString(2));
        }
    } catch (SQLException ex) {
        Logger.getLogger(Visit.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Only retrieves services with the type, "General"

Using ActionPerformed event listeners:

```

private void jButtonIndoorActionPerformed(java.awt.event.ActionEvent evt) {
    LoadService();
    LoadPatient1();
    jLabel21.setVisible(true);
    jTextFieldRoom.setVisible(true);
    jTextFieldRoom.setEnabled(false);
    jButtonOpd.setEnabled(false);
    jButtonIndoor.setEnabled(false);
}

private void jButtonOpdActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    LoadPatient();
    LoadService1();
    jButtonOpd.setEnabled(false);
    jButtonIndoor.setEnabled(false);
}

```

Disabling the buttons after any one is clicked to avoid any accidental clicks

❖ Allotting service: Using the event listener, the following code was written:

```

private void jButtonSaveActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Connection con = java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/"
            + "java_hospital_db?serverTimezone=UTC", "root", "");
        PreparedStatement pst;
        ResultSet rs;
        PreparedStatement pst1;
        ResultSet rs1;
        SimpleDateFormat dateformat = new SimpleDateFormat("yyyy-MM-dd");
        String date = dateformat.format(jDateChooser.getDate());
        Service s = (Service)ServiceBox.getSelectedItem();
        String sql = "SELECT MAX(PatientService_ID) FROM `patientservice`";
        pst1=con.prepareStatement(sql);
        rs1=pst1.executeQuery();
        rs1.next();
        rs1.getString("MAX(PatientService_ID)");
        String PatientServiceID;
        if (rs1.getString("MAX(PatientService_ID)")==null)
        {
            PatientServiceID = "1";
        }
        else
        {
            int number = rs1.getInt("MAX(PatientService_ID)");
            number++;
            PatientServiceID= String.valueOf(number);
        }

        pst=con.prepareStatement("INSERT INTO `patientservice`(`PatientService_ID`, "
            + "`Visit_ID`, Service_Date, Service_ID) VALUES (?, ?, ?, ?)");
        pst.setString(1, PatientServiceID);
        pst.setString(2, ID_Label1.getText());
        pst.setString(3, date);
        pst.setString(4, s.id);
        int i = pst.executeUpdate();
        if(i>0){
            JOptionPane.showMessageDialog(null, "Service Allotted Successfully");
        }
    }
}

```

Automatically assigning a serial number

Query to insert data into patient service table of the database

- 7) Bill Generation Page: This page is the most important page in the product as it allows the receptionist to generate, save and print patient bills.

Calculating the expenses of a patient: The following code was developed for calculations.

```
int total=0;
int totaldiscount;
int opdcharges;
int indoorcharges;
public void CalculateOPD(){
    try {
        Connect();
        SimpleDateFormat dateformat = new SimpleDateFormat("yyyy-MM-dd");
        String date = dateformat.format(jDateChooser.getDate());
        pst= con.prepareStatement("SELECT * FROM `visit` where Visit_ID= ?");
        pst.setString(1, ID_Label2.getText());
        rs=pst.executeQuery();
        rs.next();
        if ("1".equals(rs.getString("Free_Followup")))
        {
            opdcharges=0;
        }
        else
        {
            String did = rs.getString("Doctor_ID");
            PreparedStatement pst1= con.prepareStatement("Select * from doctors where Doctor_ID=?");
            pst1.setString(1, did);
            ResultSet rs1 = pst1.executeQuery();
            rs1.next();
            opdcharges = Integer.parseInt(rs1.getString("Fee"));
            total = total+opdcharges;
        }
    }catch (SQLException ex) {
        Logger.getLogger(Registration.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public void CalculateIndoor(){
    try{
        Connect();
        PreparedStatement pst1= con.prepareStatement("Select * from indoor where Visit_ID=?");
        pst1.setString(1, ID_Label2.getText());
        ResultSet rs1 = pst1.executeQuery();
        if (rs1.next()==true)
        {
            String rid = rs1.getString("Room_ID");
            PreparedStatement pst2= con.prepareStatement("Select * from rooms where Room_ID=?");
            pst2.setString(1, rid);
            ResultSet rs2 = pst2.executeQuery();
            rs2.next();
            String roomcharges = rs2.getString("Room_Charges");
            DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd");
            String inputString1 = rs1.getString("Checkin_Date");
            String inputString2 = rs1.getString("Checkout_Date");
            LocalDate date1 = LocalDate.parse(inputString1, dtf);
            LocalDate date2 = LocalDate.parse(inputString2, dtf);
            long daysBetween = ChronoUnit.DAYS.between(date1, date2);
            int roomday = Integer.parseInt(roomcharges);
            indoorcharges = (int) (roomday*daysBetween);
            total = total + indoorcharges;
        }
        else {
            indoorcharges=0;
        }
    }catch (SQLException ex) {
        Logger.getLogger(Registration.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

If the patient is a follow up patient, OPD fees is zero

Retrieving doctor's fee and equating to OPD charges

Calculating total charges for indoor

Formatting the date in a pattern

Converting to integer

Using localdate variable type to calculate days between check in and check out

The following libraries were imported to allow the calculation of number of days:

```
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
```

- ❖ A table was used to display all the individual charges. The calculated values of opd charges and indoor charges were used. However, the services charges have to be displayed one by one. The following method was created to view the table:

```
public void bill_table(){
    try {
        Connect();
        pst= con.prepareStatement("SELECT * FROM `visit` where Visit_ID= ?");
        pst.setString(1, ID_Label2.getText());
        rs=pst.executeQuery();
        rs.next();
        ResultSetMetaData Rsm= rs.getMetaData();
        int c;
        c=Rsm.getColumnCount();
        DefaultTableModel df = (DefaultTableModel)jTable1.getModel();
        df.setRowCount(0);
        //opd charges
        Vector v2= new Vector();
        v2.add("1");
        v2.add("Opd");
        String opdcharges1 = Integer.toString(opdcharges);
        v2.add(opdcharges1);
        df.addRow(v2);
        //Indoor charges
        Vector v3= new Vector();
        v3.add("2");
        v3.add("Indoor");
        String indoorcharges1 = Integer.toString(indoorcharges);
        v3.add(indoorcharges1);
        df.addRow(v3);
        // service charges
        int x = 3;
        PreparedStatement pst6 = con.prepareStatement("Select * from patientservice where Visit_ID=?");
        pst6.setString(1, ID_Label2.getText());
        ResultSet rs6 = pst6.executeQuery();
        while (rs6.next())
        {
            Vector v4= new Vector();
            for(int i = 1; i<=1; i++)
            {
                v4.add(x);
                PreparedStatement pst7=con.prepareStatement("Select* from servicesinfo where Service_ID= ?");
                pst7.setString(1, rs6.getString("Service_ID"));
                ResultSet rs7=pst7.executeQuery();
                rs7.next();
                v4.add(rs7.getString("Service_Name"));
                v4.add(rs7.getString("Service_Charges"));
                int service = Integer.parseInt(rs7.getString("Service_Charges"));
                total = total+ service;
                x ++;
            }
            df.addRow(v4);
        }
        //discount
        Vector v5= new Vector();
        v5.add(x);
        v5.add("Discount");
        String oid = rs6.getString("Organization_ID");
        PreparedStatement pst8= con.prepareStatement("Select * from organizations where Organization_ID=?");
        pst8.setString(1, oid);
        ResultSet rs8= pst8.executeQuery();
        rs8.next();
        String discount = rs8.getString("Discount");
        int disc = Integer.parseInt(discount);
        int totaldisc = total*disc;
        totaldiscount = totaldisc/100;
        String discount1 = Integer.toString(totaldiscount);
        v5.add(discount1);
        int amount= total-totaldiscount;
        String amount1 = Integer.toString(amount);
        ID_Label3.setText(amount1);
        total=0;
        totaldiscount=0;
        df.addRow(v5);
    } catch (SQLException ex) {
        Logger.getLogger(Registration.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Converting integer to string

Inserting the calculated values of opd charges and indoor charges

Retrieving services availed by the patient

Adding service name and charges to the row

Increasing serial number

Calculating the discount by multiplying the percent discount/100 to the total amount

Calculating the final amount and displaying it with the label

Resetting values for future calculations

- ❖ The following method was created to insert the data from the table into the text area for the receipt.

```
public void bill(){
    String total = ID_Label3.getText();
    DefaultTableModel model= new DefaultTableModel ();
    model = (DefaultTableModel)jTable1.getModel();
    bill.setText(bill.getText()+"*****\n");
    bill.setText(bill.getText()+"**          Sarvodaya Hospital Bill No. "+ID_Label.getText()+"**\n");
    bill.setText(bill.getText()+"*****\n");
    bill.setText(bill.getText()+"Sno."+"\t"+ "Type"+"\t"+"Charges"+"\n");
    int x= 1;
    for (int i=0; i<model.getRowCount();i++)
    {
        String sno = Integer.toString(x);
        String type = (String)model.getValueAt(i, 1);
        String amount = (String)model.getValueAt(i, 2);
        bill.setText(bill.getText()+sno+"\t"+ type+"\t"+amount+"\n");
        x=x+1;
    }
    bill.setText(bill.getText()+"\n");
    bill.setText(bill.getText()+"Total Amount Rs. "+ID_Label3.getText()+"\n");
    bill.setText(bill.getText()+"Paid Rs. "+ID_Label3.getText()+"\n");
    bill.setText(bill.getText()+"Balance Rs. "+" 0"+ "\n"+ "\n");
    bill.setText(bill.getText()+"*****\n");
    bill.setText(bill.getText()+"          THANK YOU          "+ "\n");
    bill.setText(bill.getText()+"*****\n");
}
```

Using Bill ID

Inserting table data into the text area

- ❖ The client also requested a save and print option. File writing technique was used here and the following libraries were imported for writing a file and for the print function.

```
import java.awt.print.PrinterException;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
```

Libraries that enable creating files and writing

The following functions were created using the event listeners:

```
private void jButtonSave2ActionPerformed(java.awt.event.ActionEvent evt) {
    try
    {
        String s = bill.getText();
        File f = new File("Bill.txt");
        FileWriter fw = new FileWriter(f);
        fw.write(s);
        fw.close();
        JOptionPane.showMessageDialog(null, "Bill Saved in File Bill.txt");
        bill.setText("");
    } catch (IOException ioe) {
        System.out.println("Exception Caught : " +ioe.getMessage());
        JOptionPane.showMessageDialog(null, "Bill NOT Saved");
    }
}

private void jButtonPrintActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        bill.print();
        bill.setText("");
    } catch (PrinterException ex) {
        Logger.getLogger(Bill.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Creating the text file

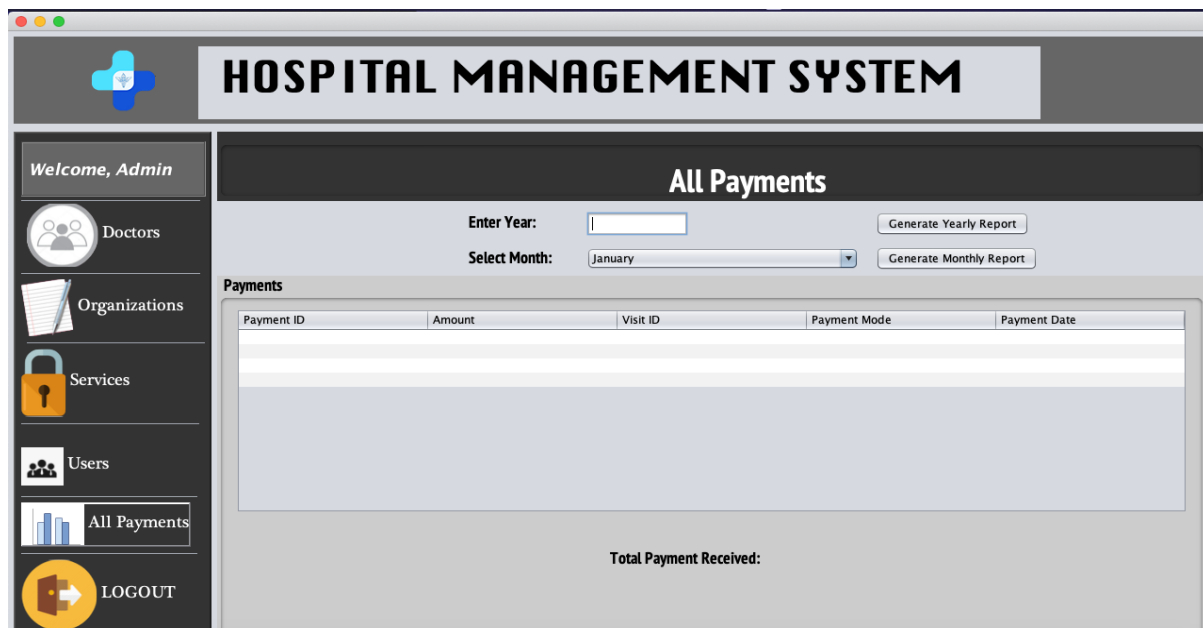
Using predefined function to write the file and insert text from text area

In case an error occurs, the catch statement is given.

Predefined print function

Admin Interface:

One specific feature requested by the client was the ability to view payments sorted according to month and year. These reports were not supposed to be printed or saved, so I decided not to use a report plugin and instead created a table to view the payments and a text box revealing the total payments.



Payments Form Class: Two separate functions were created to view the payments according to the category, i.e, yearly or monthly. Using event listeners, the required functions were called:

Yearly payments:

```
private void jButtonYearlyActionPerformed(java.awt.event.ActionEvent evt) {
    String r = jTextField1.getText();
    if (r=="")
    {
        JOptionPane.showMessageDialog(null, "Enter a valid year", "Error",JOptionPane.ERROR_MESSAGE);
    }
    else {
        indoor_table(r);
    }
}

public void indoor_table(String r){
    try {
        Connection con = java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/"
            + "java_hospital_db?serverTimezone=UTC", "root", "");
        PreparedStatement pst;
        ResultSet rs;
        pst=con.prepareStatement("select * from payments WHERE Payment_DATE >= ? AND Payment_Date <=?");
        pst.setString(1, r+"-01-01");
        pst.setString(2, r+"-12-31");
        rs=pst.executeQuery();
        ResultSetMetaData Rsm= rs.getMetaData();
        int c;
        c=Rsm.getColumnCount();
        DefaultTableModel df = (DefaultTableModel)jTable1.getModel();
        df.setRowCount(0);
        while (rs.next())
        {
            Vector v2= new Vector();
            for(int i = 1; i<=c; i++)
            {
                v2.add(rs.getString("Payment_ID"));
                v2.add(rs.getString("Payment_Amount"));
                v2.add(rs.getString("Visit_ID"));
                v2.add(rs.getString("Payment_Mode"));
                v2.add(rs.getString("Payment_Date"));
            }
            df.addRow(v2);
        }
        int total = 0;
        DefaultTableModel model= new DefaultTableModel ();
        model = (DefaultTableModel)jTable1.getModel();
        for (int o=0; o<model.getRowCount();o++)
        {
            String amount = (String)model.getValueAt(o, 1);
            int amount1 = Integer.parseInt(amount);
            total = total+amount1;
        }
        String totalamount = Integer.toString(total);
        jLabel5.setText("Rs. "+totalamount);
    } catch (SQLException ex) {
        Logger.getLogger(Registration.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Calling the method and providing it the value for the variable that stores the year

Query to retrieve payments between the first day of the year and the last

Inserting data from vector to table

Looping till all rows are considered and calculating total payment by adding the payments

Monthly payments:

```
private void jButtonMonthlyActionPerformed(java.awt.event.ActionEvent evt) {  
    indoor_table1();  
}
```

```
public void indoor_table1(){  
    String month = null;  
    try {  
        int i = jComboBox1.getSelectedIndex();  
        for (int j=0;j<=11;j++)  
        {  
            if (i==j)  
            {  
                int m = i+1;  
                month = String.format("%02d", m);  
            }  
        }  
        String r = jTextField1.getText();  
        Connection con = java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/java_hospital_db?serverTimezone=UTC"  
        PreparedStatement pst;  
        ResultSet rs;  
        pst=con.prepareStatement("select * from payments WHERE Payment_DATE >= ? AND Payment_Date <=?");  
        pst.setString(1, r+"-"+month+"-01");  
        pst.setString(2, r+"-"+month+"-31");  
        rs=pst.executeQuery();  
        ResultSetMetaData Rsm= rs.getMetaData();  
        int c;  
        c=Rsm.getColumnCount();  
        DefaultTableModel df = (DefaultTableModel)jTable1.getModel();  
        df.setRowCount(0);  
        while (rs.next())  
        {  
            Vector v2= new Vector();  
            for(int k = 1; k<=c; k++)  
            {  
                v2.add(rs.getString("Payment_ID"));  
                v2.add(rs.getString("Payment_Amount"));  
                v2.add(rs.getString("Visit_ID"));  
                v2.add(rs.getString("Payment_Mode"));  
                v2.add(rs.getString("Payment_Date"));  
            }  
            df.addRow(v2);  
        }  
        int total = 0;  
        DefaultTableModel model= new DefaultTableModel ();  
        model = (DefaultTableModel)jTable1.getModel();  
        for (int o=0; o<model.getRowCount();o++)  
        {  
            String amount = (String)model.getValueAt(o, 1);  
            int amount1 = Integer.parseInt(amount);  
            total = total+amount1;  
        }  
        String totalamount = Integer.toString(total);  
        jLabel5.setText("Rs. "+totalamount);  
    } catch (SQLException ex) {  
        Logger.getLogger(Registration.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

Setting the month as the index of the combo box for the month plus 1. The format was also created so that the month is in the "0_" format.

Query to select payments between the first date of the month and the last date.

Refer to Appendix C for the complete source code of the product.

Word Count - 1128 (Excluding headings and captions)