Coursework

Programming III

STUDENT NAME : SHIFAN SAMSUDEEN

ESOFT ID : E216444

KU ID : 2377544

Introduction

1. Scenario
2. Class Diagram
3. Software Development
    a. User interface development (GUI)
    b. Modules of Application
    c. Use of OOP in Application
    d. Data structures in Application
    e. Algorithms in Application
    f. Test Cases for “Tooth Care” Application
        i. LoginForm Test Case
        ii. DashboardForm Test Case
        iii. TreatmentForm Test Case
        iv. AppointmentForm Test Case
        v. PaymentForm Test Case
        vi. SettingForm Test Case
4. Programming Explained
    a. Clarity of code
    b. Naming method used (variables, classes, properties, and methods)
    c. Comments
5. **Google Drive Link - Source Code**

Scenario

“Tooth Care” Management System

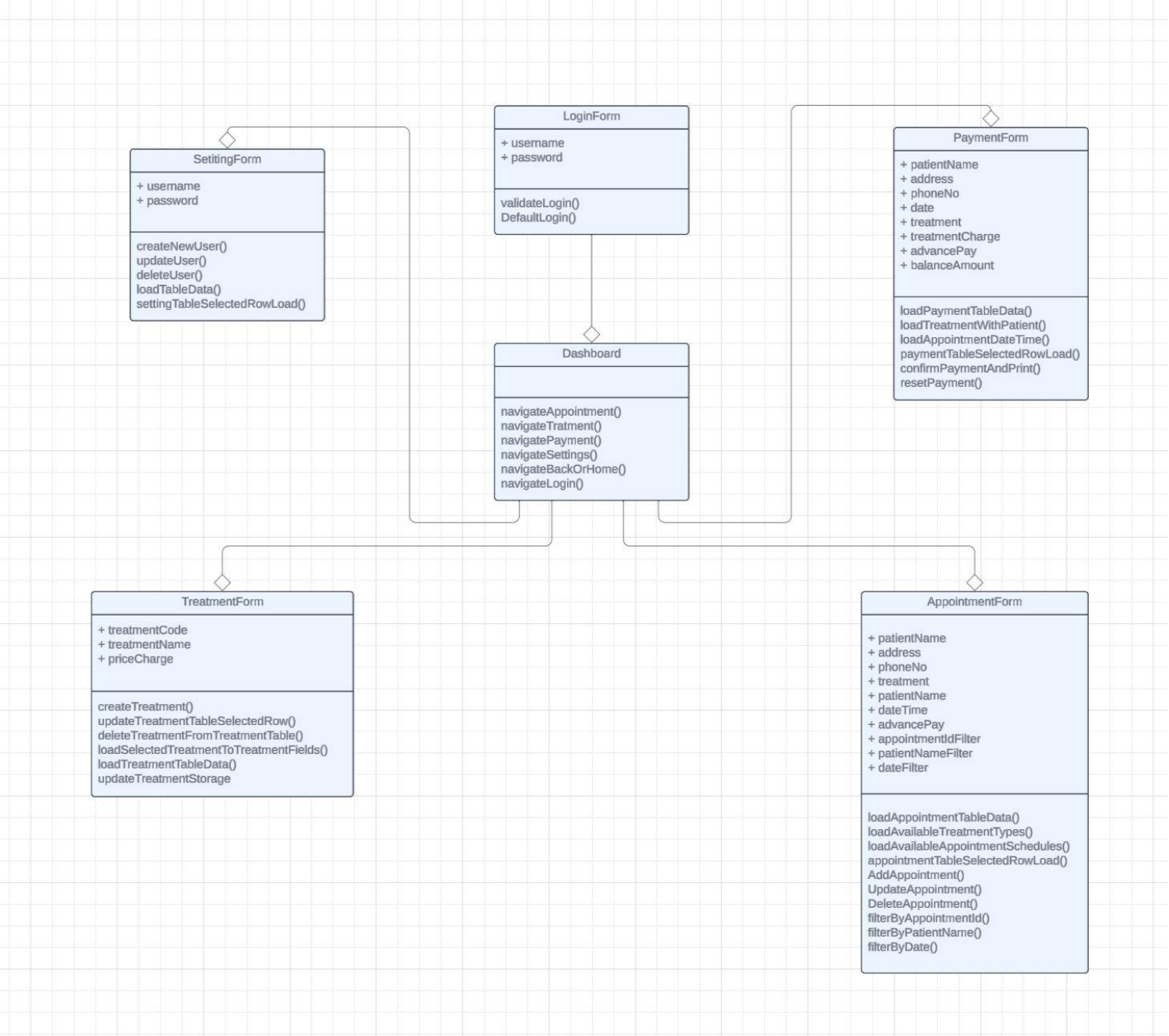
Situated in Nugegoda city, the "Tooth Care" is a leading medical facility specializing in providing comprehensive healthcare services. This individual project aims to design and implement a software program tailored to enhance ToothCare Clinic's operational efficiency. The primary objective is to develop a system that streamlines appointment scheduling, patient records management, and billing processes.

Tooth Care's renowned dental surgeon, Mr. A.D. Ranasinghe, schedules consultations at certain times and dates every week. The hours that are open for channeling appointments are 3:00 pm to 10:00 pm on Saturday and Sunday, and 6:00 pm to 9:00 pm on Monday and Wednesday. Ms. Gayani, the front office operator, is in charge of scheduling these appointments. She checks the doctor's consultation schedule and works with patients to set up convenient times for visits.

The goal of the suggested software system is to offer a complete remedy for optimizing Tooth Care's workflow. This covers features like making an appointment, updating appointment data, , searching appointments using appointment IDs, collecting registration fees, calculating treatment fees, and processing payments while allowing invoice viewing.

The system will use Object-Oriented Programming (OOP) ideas, which emphasize the construction of classes and objects to reflect real-world phenomena, to accomplish these goals. In order to guarantee an effective and scalable solution, the implementation will also include pertinent data structures, algorithms, and design patterns. The suggested system's architecture and implementation will be explained in this paper, together with test cases that illustrate the software application's resilience and a class diagram and some sample source code. The ultimate objective is to improve Tooth Care's patients' entire experience by implementing a well-organized, state-of-the-art scheduling and invoicing system.

Class Diagram



Software Development

Have your students learned everything they need to know in order to complete this lesson? This might be a good time to review some previous lessons so that they feel prepared to learn something exciting and new!

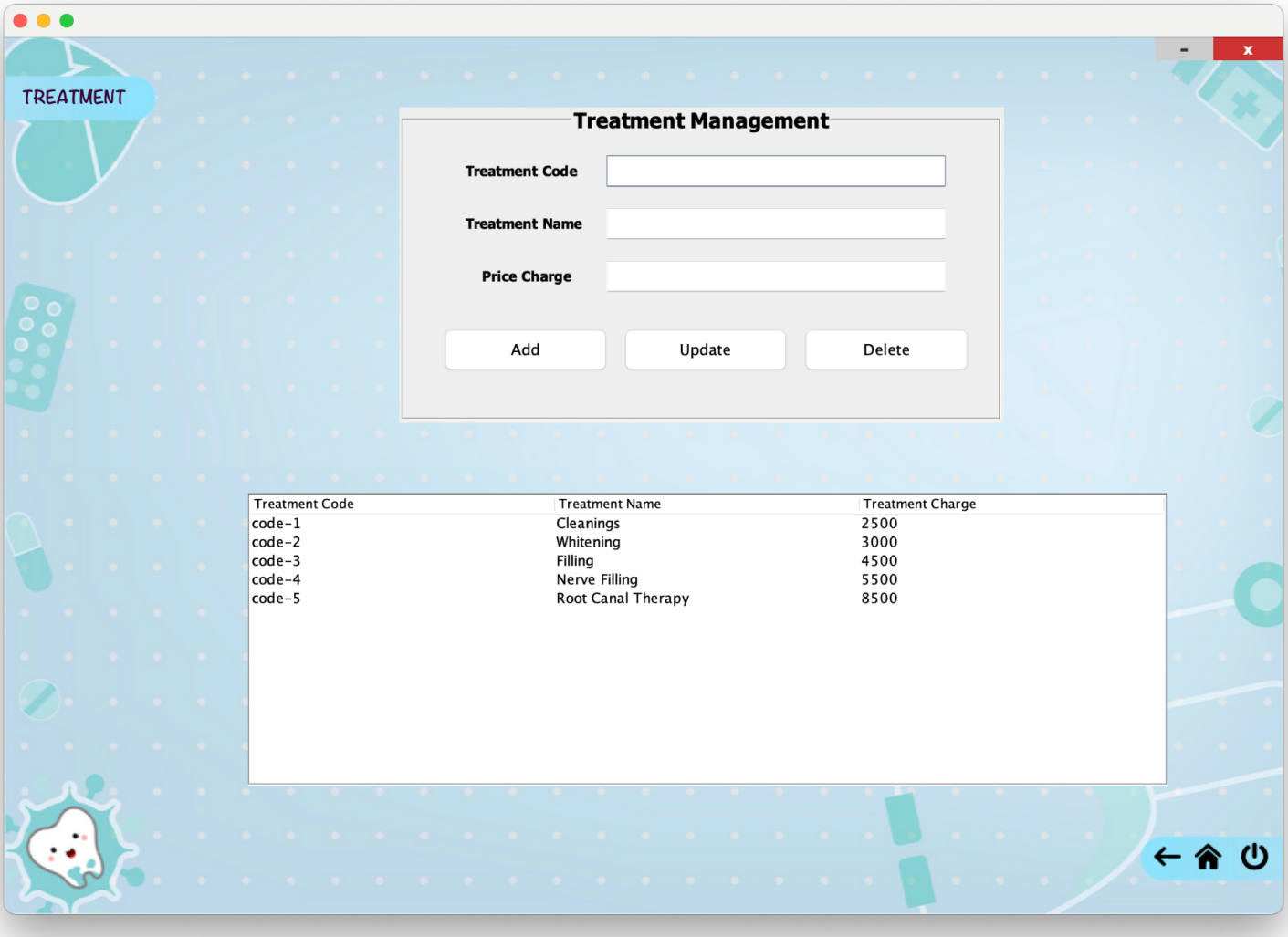
User interface development

1. Login View

When Application starts to run then the first login view will load, for ease of access login validation removed from system.Reset button allows resetting all saved data and gives fresh raw data without closing the application.

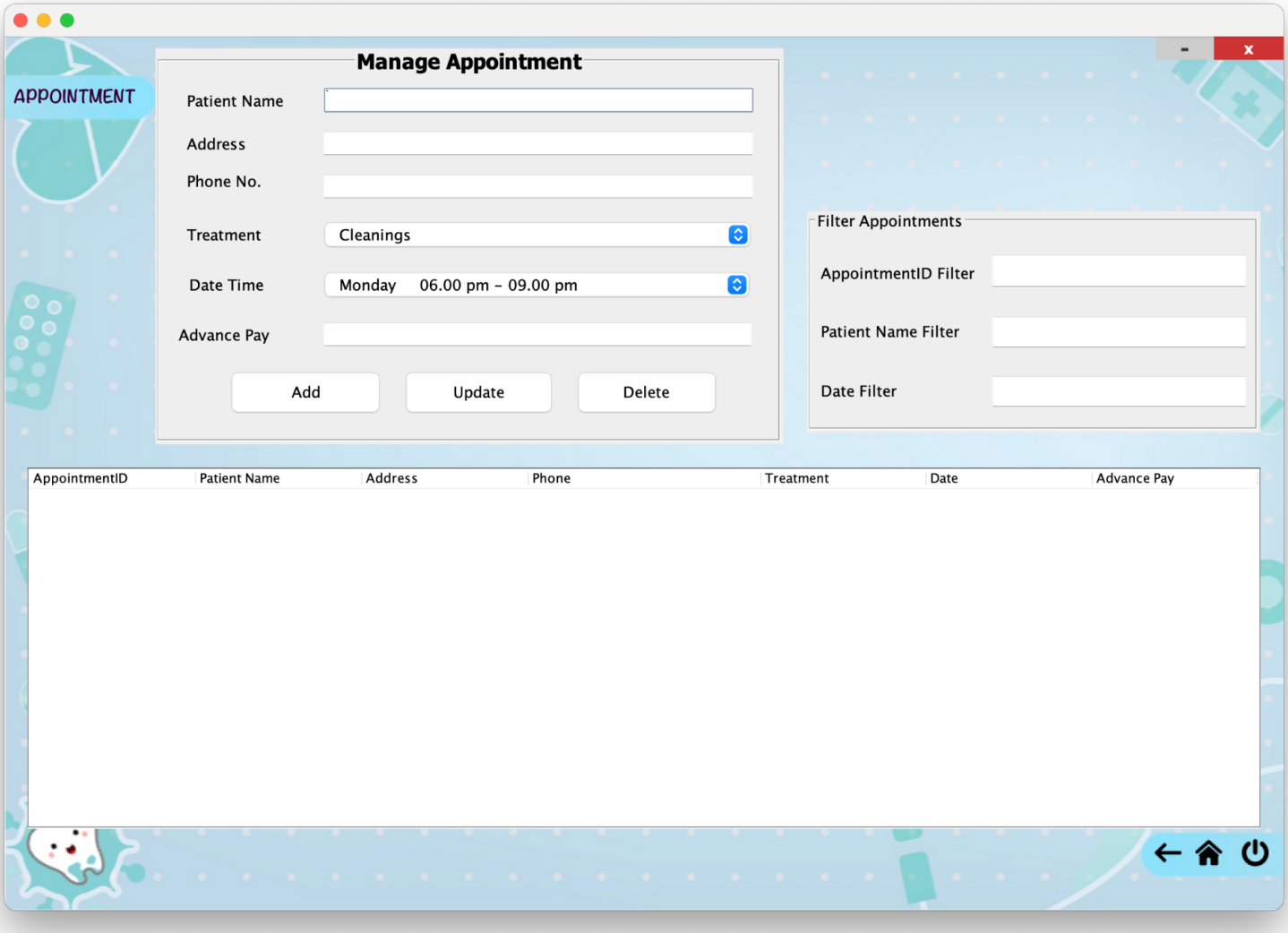
2. Treatment View

Treatment view will automatically load the default treatments (Cleanings,Whitening,Filling,Nerve Filling,Root Canal Therapy - mentioned in coursework) and also we can create new treatments,modify treatment,and also be able to delete treatments.



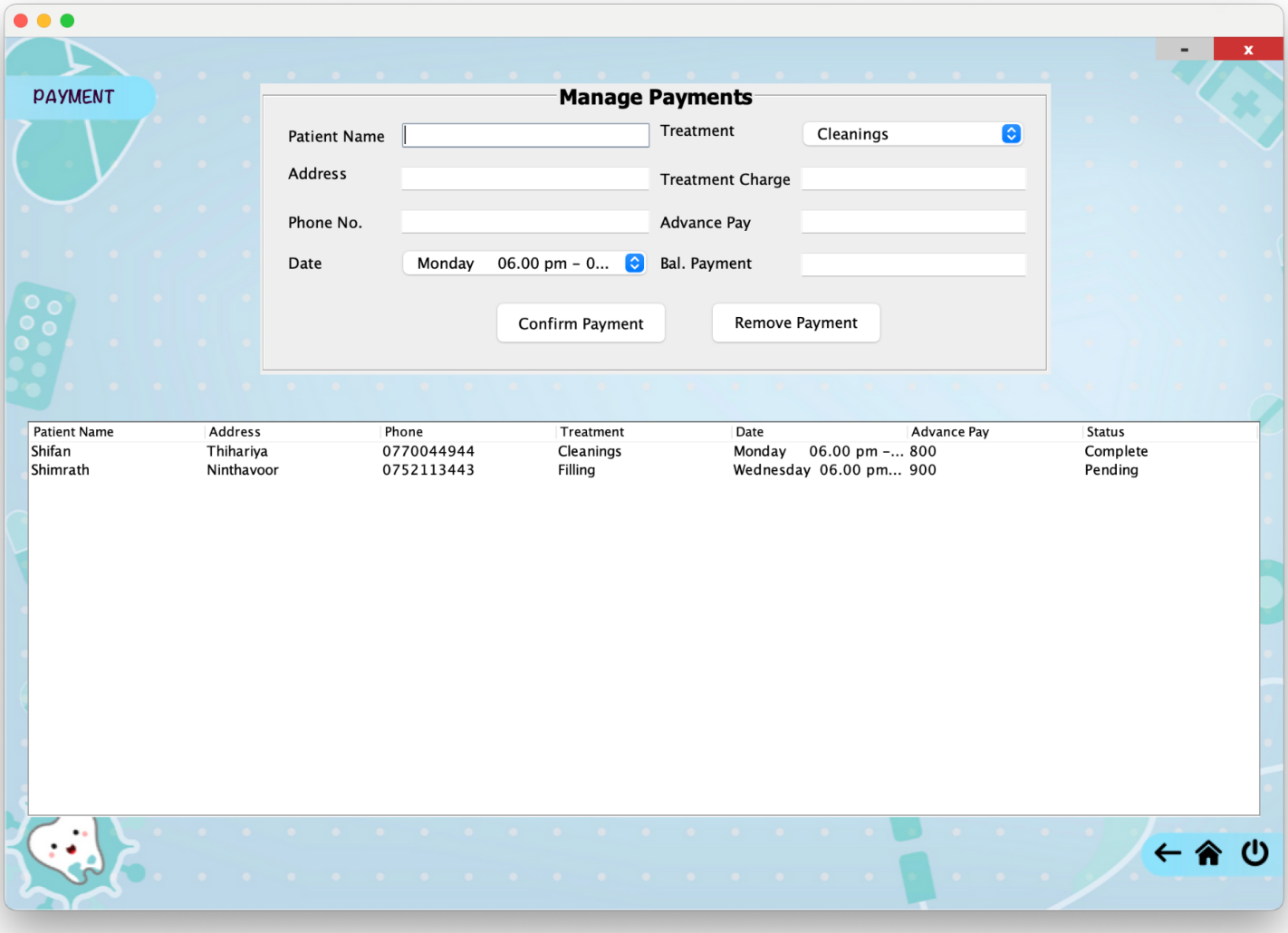
3. Appointment View

In the appointment view we can create appointments for patients, already created and available treatments will load into a comboBox for ease of selection. And according to the doctor's visiting schedule the Date Time ComboBox showing.



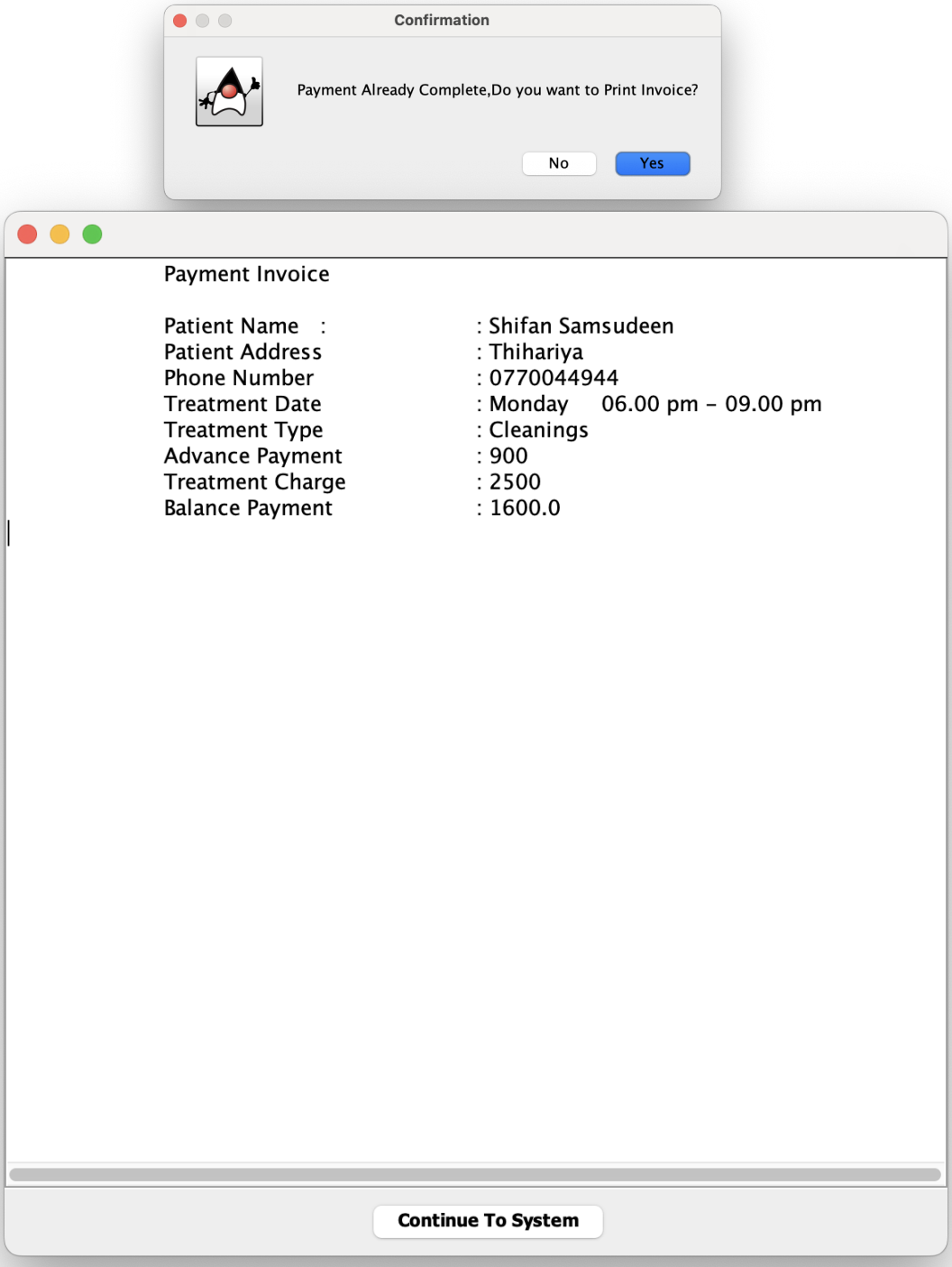
4. Payment View

In the payment view appointments will be loaded with the status, fully paid appointments will show status as complete and if not paid completely then it will show as pending as below.



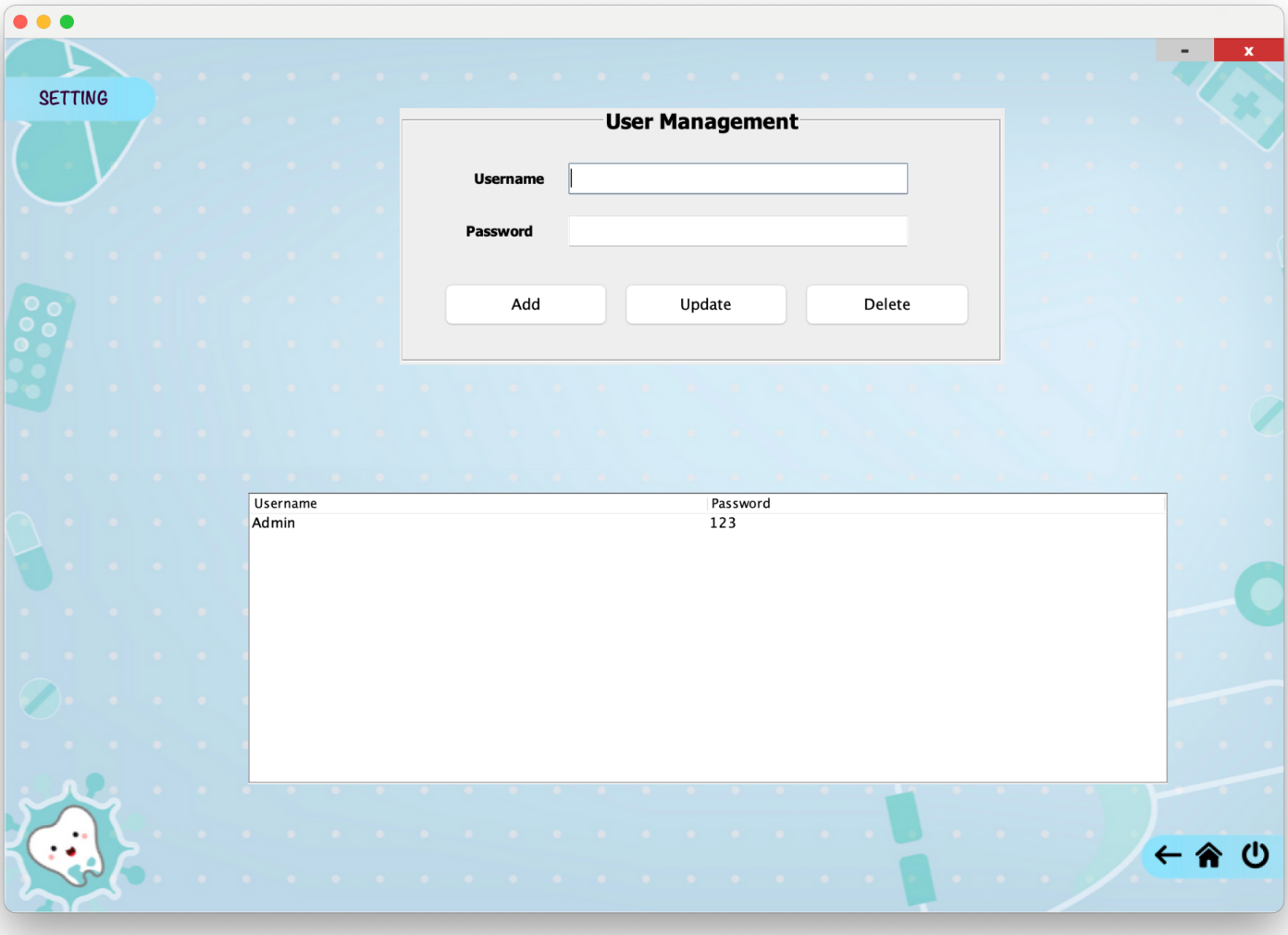
5. Payment Invoice View

In the payment view when we confirm an appointment by selecting the table it will generate a payment invoice , if the payment completes then it will ask for confirmation!



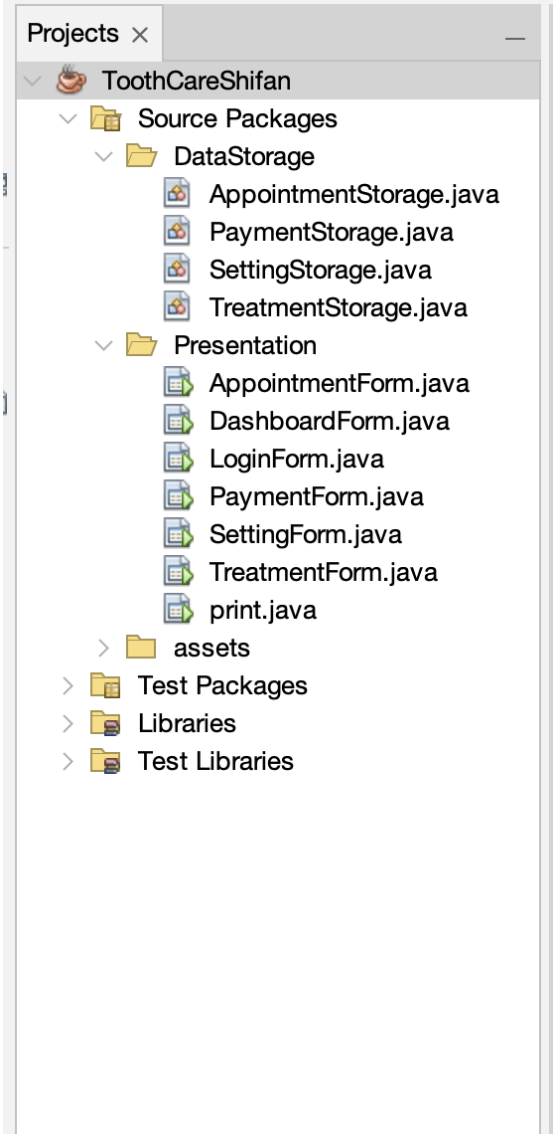
6. Setting View

In the settings view we can create users for the system and maintain.



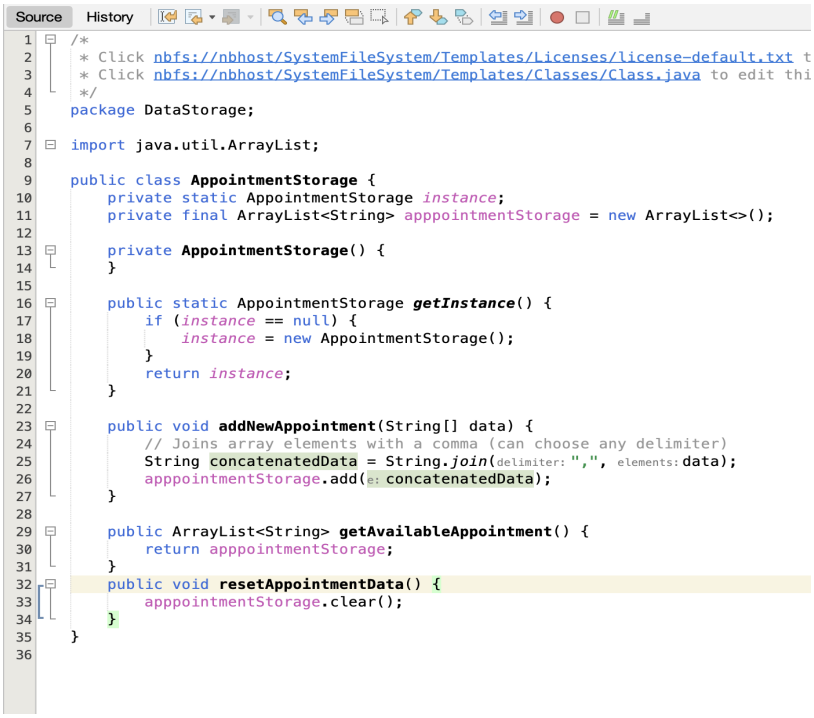
Modules of “Tooth Care”

- In this Application two java packages are available.
- DataStorage
    - Act like singleton
    - (No Database used)
- Presentation Layer
    - These are the jFrames used
- Mainly we can mention as modules
    - Appointment
    - Treatment
    - Payment
    - Settings
- Assets
    - Images stored



Use of OOP

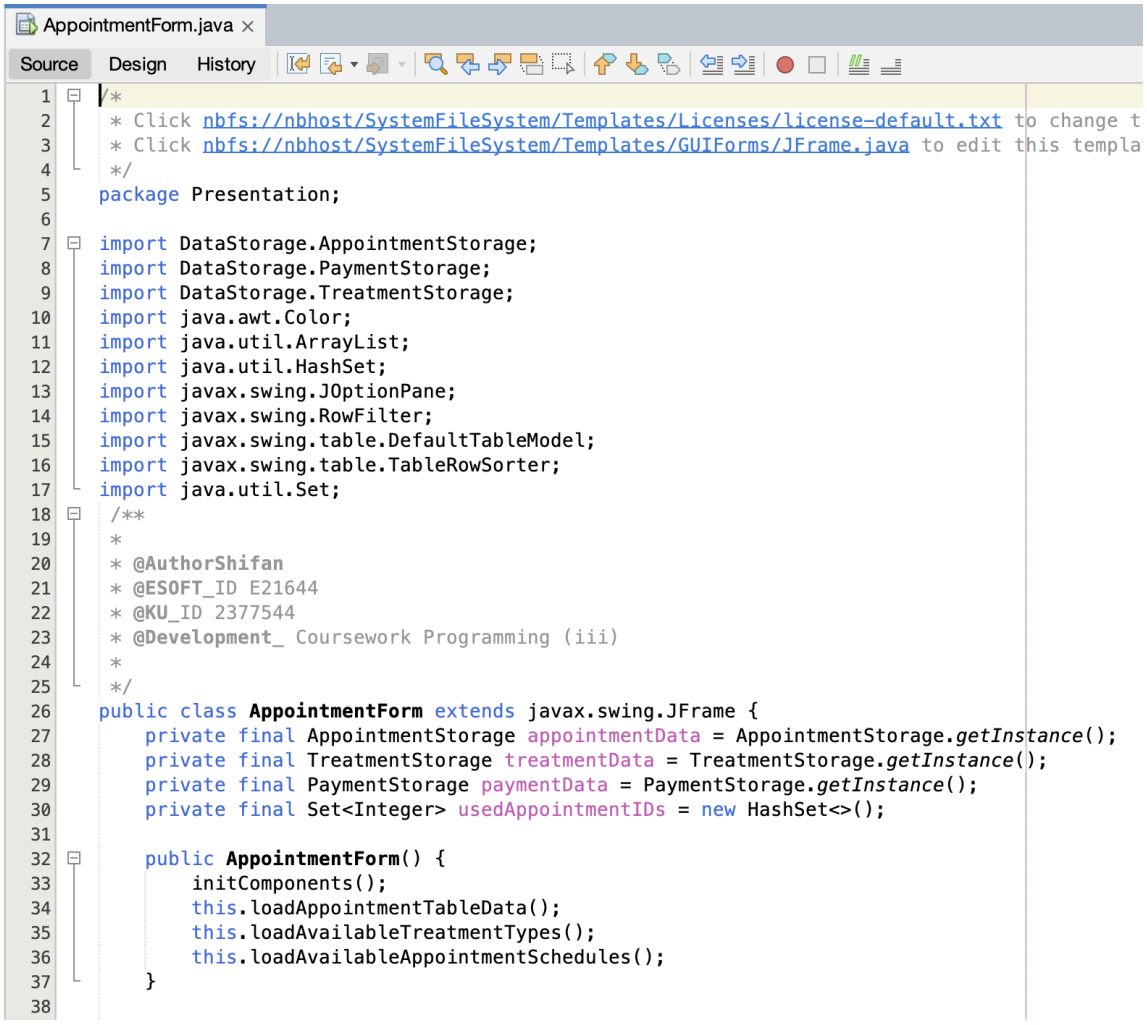
1. Class AppointmentStorage()



```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt t
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit thi
 */
package DataStorage;

import java.util.ArrayList;

public class AppointmentStorage {
    private static AppointmentStorage instance;
    private final ArrayList<String> apppointmentStorage = new ArrayList<>();

    private AppointmentStorage() {
    }

    public static AppointmentStorage getInstance() {
        if (instance == null) {
            instance = new AppointmentStorage();
        }
        return instance;
    }

    public void addNewAppointment(String[] data) {
        // Joins array elements with a comma (can choose any delimiter)
        String concatenatedData = String.join(delimiter: ",", elements: data);
        apppointmentStorage.add(e: concatenatedData);
    }

    public ArrayList<String> getAvailableAppointment() {
        return apppointmentStorage;
    }
    public void resetAppointmentData() {
        apppointmentStorage.clear();
    }
}
```

- Encapsulation
    - The class AppointmentStorage encapsulates the data storage and manipulation related to appointments. The internal details of how appointments are stored and managed are hidden from the external code that uses this class.
- Singleton Pattern
    - The class uses the Singleton pattern by ensuring that only one instance of AppointmentStorage can exist. The getInstance method provides a global point of access to the instance, and if an instance doesn't exist, it creates one. This ensures a single point of control for managing appointment data.
- Method Overloading
    - The addNewAppointment method demonstrates method overloading by accepting different types of parameters. One version takes an array of strings, and another version could potentially take individual parameters.
- Data Hiding
    - The internal storage structure (apppointmentStorage) is hidden from the external code. The class provides public methods (addNewAppointment, getAvailableAppointment, resetAppointmentData) to interact with the data, abstracting away the details of data storage..

2. Class AppointmentForm()



- Constructor
    - The constructor public AppointmentForm() is responsible for initializing the form, loading initial data, and setting up the user interface.
- Inheritance
    - The class extends javax.swing.JFrame, inheriting functionality from the Swing framework to create a graphical user interface (GUI) window.
- Composition
    - The class composes instances of AppointmentStorage, TreatmentStorage, and PaymentStorage to manage appointment, treatment, and payment data, respectively.

NOTE => The OOP concepts mentioned above are samples from two classes.

Data structures

Class AppointmentForm()

AppointmentForm.java

Source | Design | History

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change t
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this templa
 */
package Presentation;

import DataStorage.AppointmentStorage;
import DataStorage.PaymentStorage;
import DataStorage.TreatmentStorage;
import java.awt.Color;
import java.util.ArrayList;
import java.util.HashSet;
import javax.swing.JOptionPane;
import javax.swing.RowFilter;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
import java.util.Set;
/**
 *
 * @AuthorShifan
 * @ESOFT_ID E21644
 * @KU_ID 2377544
 * @Development_ Coursework Programming (iii)
 *
 */
public class AppointmentForm extends javax.swing.JFrame {
    private final AppointmentStorage appointmentData = AppointmentStorage.getInstance();
    private final TreatmentStorage treatmentData = TreatmentStorage.getInstance();
    private final PaymentStorage paymentData = PaymentStorage.getInstance();
    private final Set<Integer> usedAppointmentIDs = new HashSet<>();

    public AppointmentForm() {
        initComponents();
        this.loadAppointmentTableData();
        this.loadAvailableTreatmentTypes();
        this.loadAvailableAppointmentSchedules();
    }

```

- Instance Variables
    - AppointmentStorage appointmentData: An instance of the AppointmentStorage class, which appears to be a singleton class managing the storage of appointment data.
    - Set<Integer> usedAppointmentIDs: A HashSet used to keep track of used appointment IDs to ensure uniqueness.
- Local Variables
    - DefaultTableModel: This is part of Swing and Java GUI programming. It represents the data model for a JTable and is used for managing data in a tabular form
- Data Structures in AppointmentStorage, TreatmentStorage, and PaymentStorage Classes
    - They use ArrayLists to store data
- Swing Components.

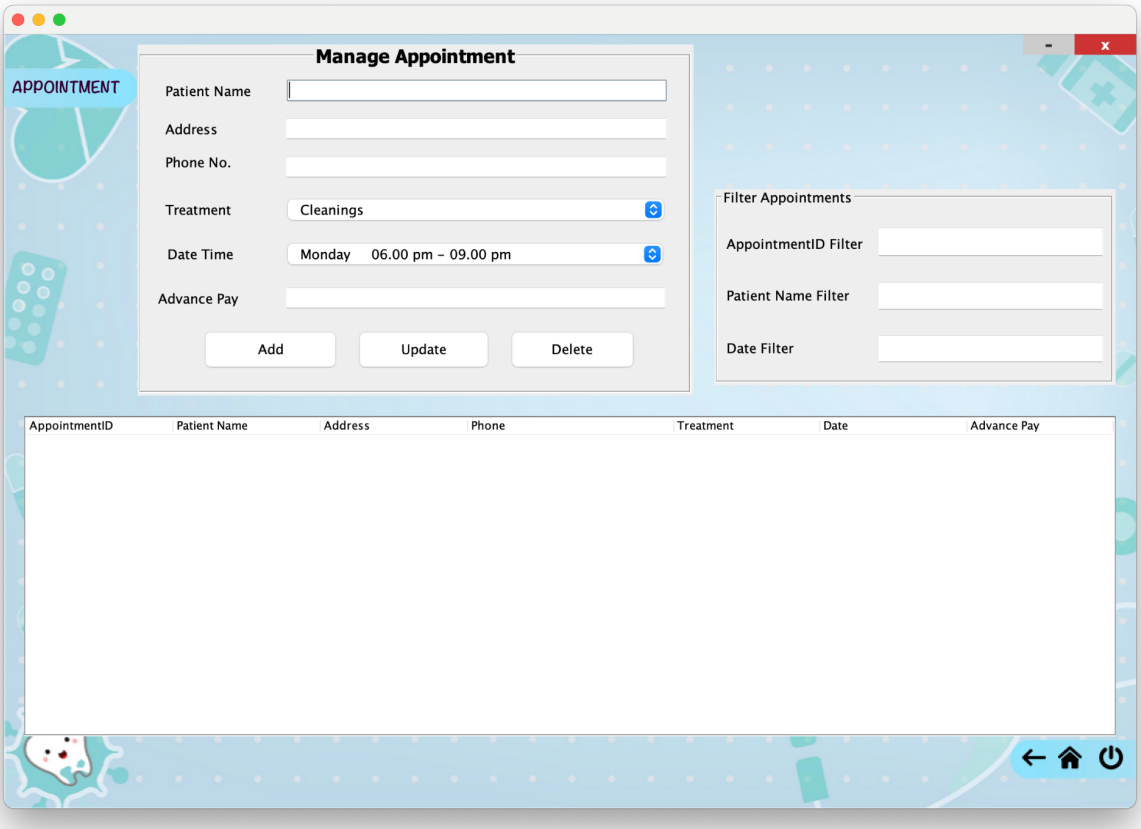
Algorithms

```
804     private void rearangeTableData()
805     {
806         appointmentData.resetAppointmentData();
807         DefaultTableModel tableModel = (DefaultTableModel)appointmentTable.getModel();
808         int rows = tableModel.getRowCount();
809         int cols = tableModel.getColumnCount();
810         Object[][] rowsData = new Object[rows][cols];
811         for (int i = 0; i < rows; i++) {
812             for (int j = 0; j < cols; j++) {
813                 rowsData[i][j] = tableModel.getValueAt(row:i, column: j);
814             }
815         }
816         for (int i = 0; i < rows; i++) {
817         StringBuilder rowData = new StringBuilder();
818 
819         for (int j = 0; j < cols; j++) {
820             rowData.append(rowsData[i][j]);
821 
822             if (j < cols - 1) {
823                 rowData.append(str:", ");
824             }
825         }
826 
827             String[] data = rowData.toString().split(regex: ", ");
828             appointmentData.addNewAppointment(data);
829         }
830     }
```

```
845     private void filterTable(Integer ColumnIndex,String ColumnValue) {
846         DefaultTableModel tableModel = (DefaultTableModel)appointmentTable.getModel();
847         TableRowSorter<DefaultTableModel> sorter = new TableRowSorter<>(model: tableModel);
848         appointmentTable.setRowSorter(sorter);
849 
850         RowFilter<DefaultTableModel, Object> rowFilter = RowFilter.regexFilter("(?i)" + ColumnValue, indices:ColumnIndex);
851         sorter.setRowFilter(filter: rowFilter);
852     }
```

```
655 
656     private int getNextUniqueAppointmentID() {
657         int nextAppointmentID = 1;
658 
659         // Keep generating IDs until a unique one is found
660         while (usedAppointmentIDs.contains(o: nextAppointmentID)) {
661             nextAppointmentID++;
662         }
663 
664         return nextAppointmentID;
665     }
666 
```

- Next Unique Appointment ID Calculation (**getNextUniqueAppointmentID()**)
    - This method uses a simple loop to find the next unique appointment ID by iterating until a unique ID is found. It relies on a HashSet (usedAppointmentIDs) to keep track of used IDs.
- Data Rearrangement (**rearangeTableData()**)
    - These methods involve updating and rearranging data in the JTable and the associated data storage classes (AppointmentStorage). The code iterates through the rows and columns of the JTable to gather and update data.
- Filtering Rows in JTable (**filterTable(Integer ColumnIndex, String ColumnValue)**)
    - This method uses a RowFilter to dynamically filter rows in the JTable based on a specified column index and filter value. It leverages regular expressions for case-insensitive matching.



Appointment fllter used in three ways

1. Filter By Appointment ID
2. Filter By Patient Name
3. Filter By Date

Test Cases

1. LoginForm Test Case

Test case #: 1 System: Tooth care Designed by: Shifan Samsudeen Executed by: Shifan Samsudeen Short Description: validation removed for ease access	Test Case Name: Login Test Case Test Name : Log in

Pre-conditions : Run “Tooth Care” application and should be in LoginForm

Step	Action	Expected System Response	Pass/Fail	Comment
1	Enter username	Able to type inside username field	pass	
2	Enter password	Able to type inside password field	pass	
3	Press login button	Navigate to Dashboard	pass	

2. DashboardForm Test Cases

Test case #: 2 System: Tooth care Designed by: Shifan Samsudeen Executed by: Shifan Samsudeen Short Description:	Test Case Name: Dashboard Test Case Test Name : Dashboard Test

Pre-conditions : Should be in DashboardForm

Step	Action	Expected System Response	Pass/Fail	Comment
1	Press Treatment	Navigate to TreatmentForm	pass	
2	Press Appointment	Navigate to AppointmentForm	pass	
3	Press Payment	Navigate to PaymentForm	pass	
4	Press Setting	Navigate to SettingForm	pass	
5	Press Left Arrow Icon	Navigate to LoginForm	pass	
6	Press Power Icon	Navigate to LoginForm	pass	
7	Press Home Icon	Reload to DashboardForm	pass	

3. Treatment Test Cases

Test case #: 3 System: Tooth care Designed by: Shifan Samsudeen Executed by: Shifan Samsudeen Short Description:	Test Case Name: Treatment Test Case Test Name : Treatment Test

Pre-conditions : Should be in TreatmentForm

Step	Action	Expected System Response	Pass/Fail	Comment
1	Initialize Load	Expected Treatments should load with price charges inside table (Cleaning, whitening, Filling, Nerve Filling, Root Canal Therapy)	pass	
2	Add New Treatment (enter treatment code, treatment name, treatment charge and press Add button)	Data should add to the treatment table and fields should get clear	pass	
3	Click /Select a treatment in Treatment table	According to data in selected row, treatment code , treatment name and treatment charge should load	pass	
4	Update Treatment (After Select a treatment make some changes and Press Update Button)	According to the changes data should modified in selected row and input fields should clear	pass	
5	Delete Treatment (After Select a treatment from table Press Delete Button)	Selected treatment row should remove from treatment table and fields should clear	pass	
6	Press Left Arrow Icon	Navigate to DashboardForm	pass	
7	Press Home Icon	Navigate to DashboardForm	pass	
8	Press Power Icon	Navigate to LoginForm	pass	

4. Appointment Test Cases

Test case #: 4 System: Tooth care Designed by: Shifan Samsudeen Executed by: Shifan Samsudeen Short Description:	Test Case Name: Appointment Test Case Test Name : Appointment Test

Pre-conditions : Should be in AppointmentForm

Step	Action	Expected System Response	Pass/Fail	Comment
1	Initialize Load	Expected Appointments should load (if already made any appointments)	pass	
2	Add New Appointment (enter PatientName, treatment name, Address, Phone number, treatment, Date time, advance pay and Press Add button)	Data should add to the Appointment table with an AppointmentID and fields should get clear	pass	
3	Click /Select a appointment in Appointment table	According to data in selected row, appointment should load into fields	pass	
4	Update Appointment (After Select a appointment make some changes and Press Update Button)	According to the changes data should modified in selected row and input fields should clear	pass	
5	Delete Appointment (After Select a appointment from table and Press Delete Button)	Selected appointment row should remove from appointment table and fields should clear	pass	
5	Filter by AppointmentID	Table should filter by Id	pass	
6	Filter by PatientName	Table should filter by Patient Name	pass	
7	Filter by Date	Table should filter by Date	pass	
8	Press Left Arrow Icon	Navigate to DashboardForm	pass	
9	Press Home Icon	Navigate to DashboardForm	pass	
10	Press Power Icon	Navigate to LoginForm	pass	

5. Payment Test Cases

Test case #: 4 System: Tooth care Designed by: Shifan Samsudeen Executed by: Shifan Samsudeen Short Description:	Test Case Name: Payment Test Case Test Name : Payment Test

Pre-conditions : Should be in PaymentForm

Step	Action	Expected System Response	Pass/Fail	Comment
1	Initialize Load	Expected Appointments should load (if already made any payments should load status as complete otherwise as pending)	pass	
2	Click /Select a appointment from table to continue its payment	According to data in selected row, appointment should load into fields (Treatment charge, Advance pay and Balance payment should be loaded correctly) Bal. Payment = Treatment charge - Advance pay	pass	
3	Confirm Payment (After Select a appointment from table Press Confirm Payment Button)	If the appointment status is "Pending," change it to "Complete" and print. Otherwise, display a popup indicating that the payment has already been completed. If the user wishes to print the selection, they should be able to do so.	pass	
4	Remove Payment (After selecting a appointment from table Press Remove Payment Button)	Payment Details loaded to fields should clear	pass	
5	Press Left Arrow Icon	Navigate to DashboardForm	pass	
6	Press Home Icon	Navigate to DashboardForm	pass	
7	Press Power Icon	Navigate to LoginForm	pass	

6. Setting Test Cases

Test case #: 3 System: Tooth care Designed by: Shifan Samsudeen Executed by: Shifan Samsudeen Short Description:	Test Case Name: Setting Test Case Test Name : Setting Test

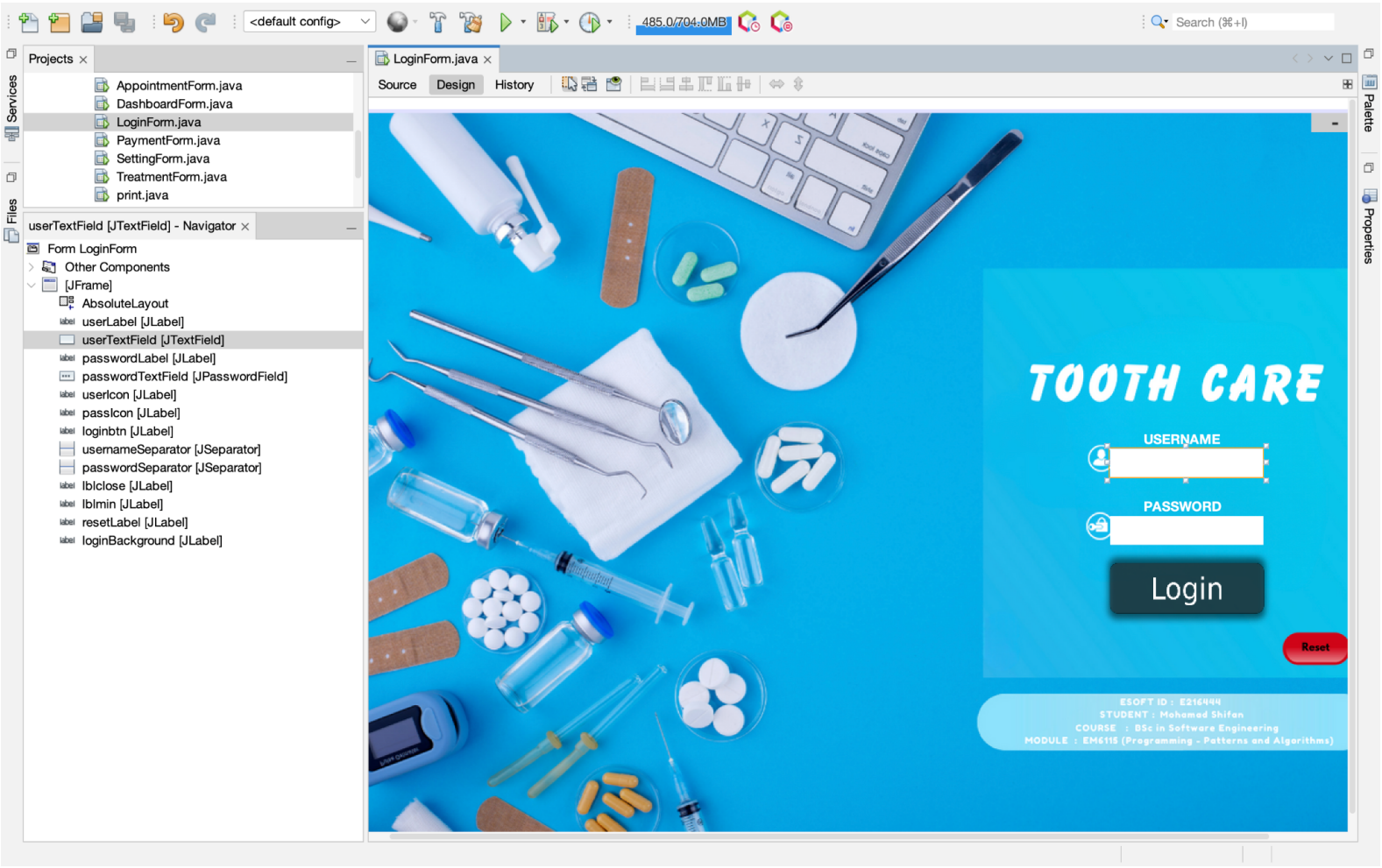
Pre-conditions : Should be in SettingForm

Step	Action	Expected System Response	Pass/Fail	Comment
1	Initialize Load	Expected users should load (if any users already created)	pass	
2	Add New User (enter username,password and press Add button)	Data should add to the user table and fields should get clear	pass	
3	Click /Select a user from table	According to data in selected row,user fields should load	pass	
4	Update User (After Select a user make some changes and Press Update Button)	According to the changes data should modified in selected row and input fields should clear	pass	
5	Delete User (After Select a user from table Press Delete Button)	Selected user row should remove from table and fields should clear	pass	
6	Press Left Arrow Icon	Navigate to DashboardForm	pass	
7	Press Home Icon	Navigate to DashboardForm	pass	
8	Press Power Icon	Reload to LoginForm	pass	

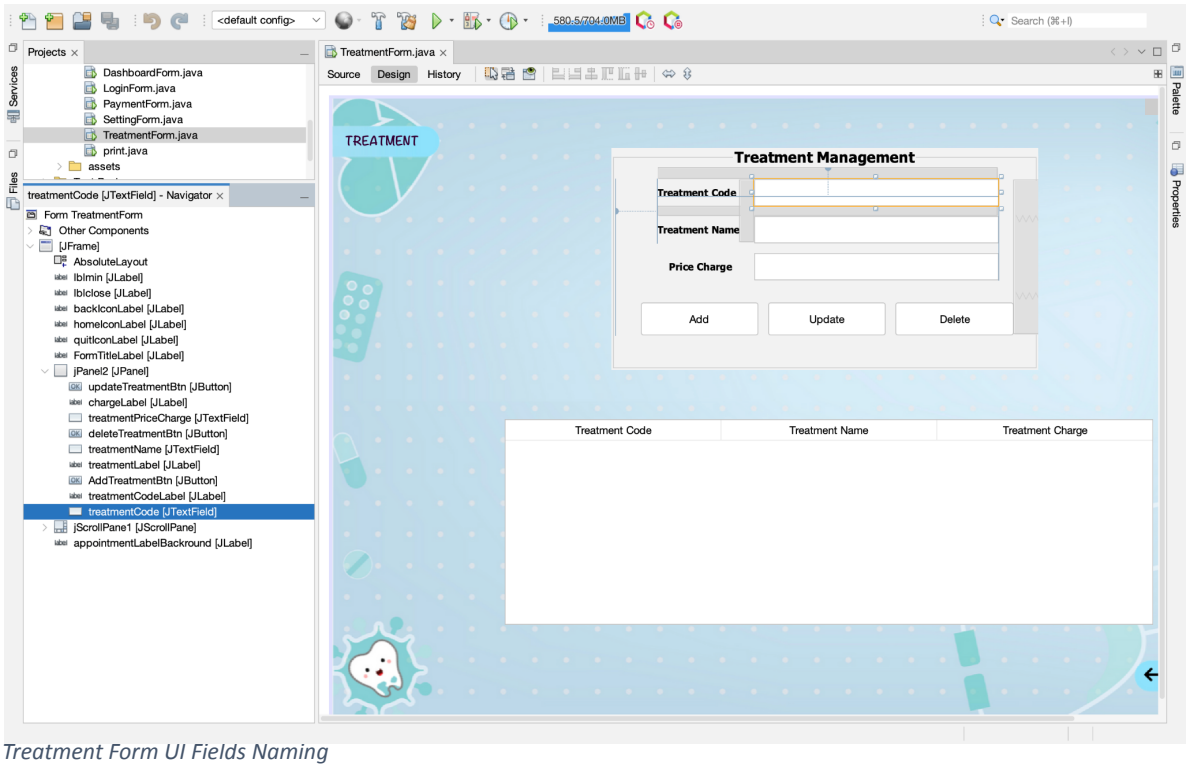
Programming Explained

Naming Methods used

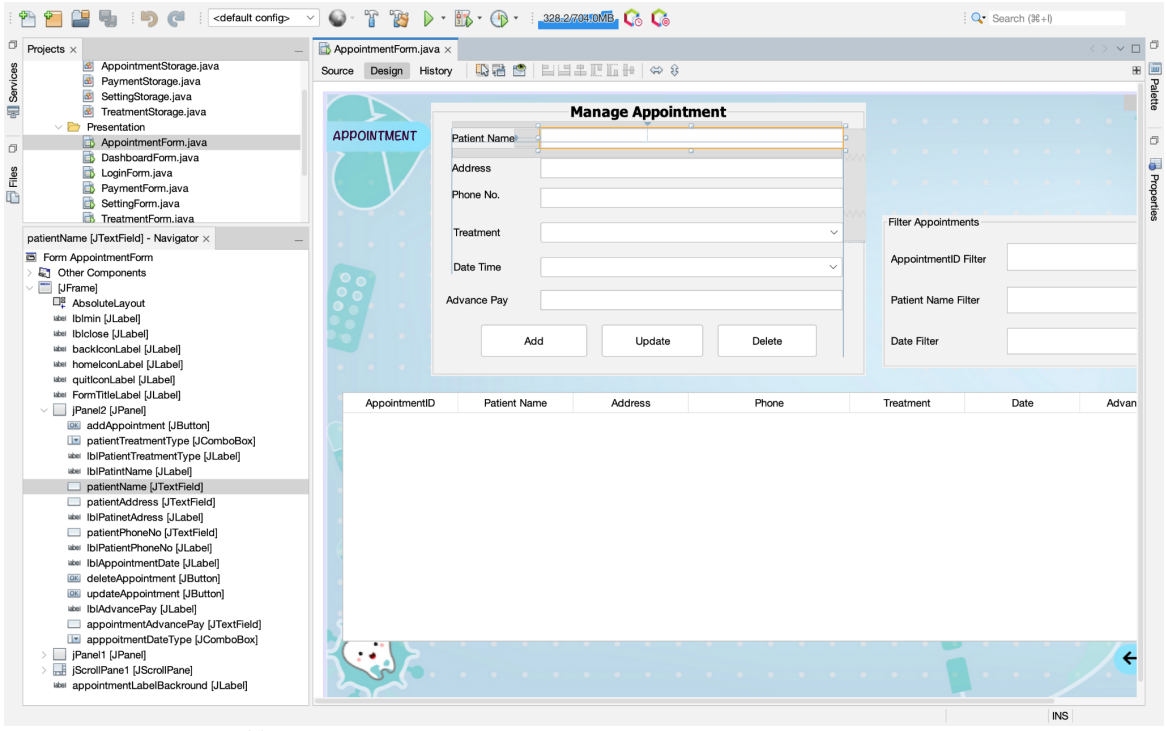
- UI Fields Naming



Login Form UI Fields Naming

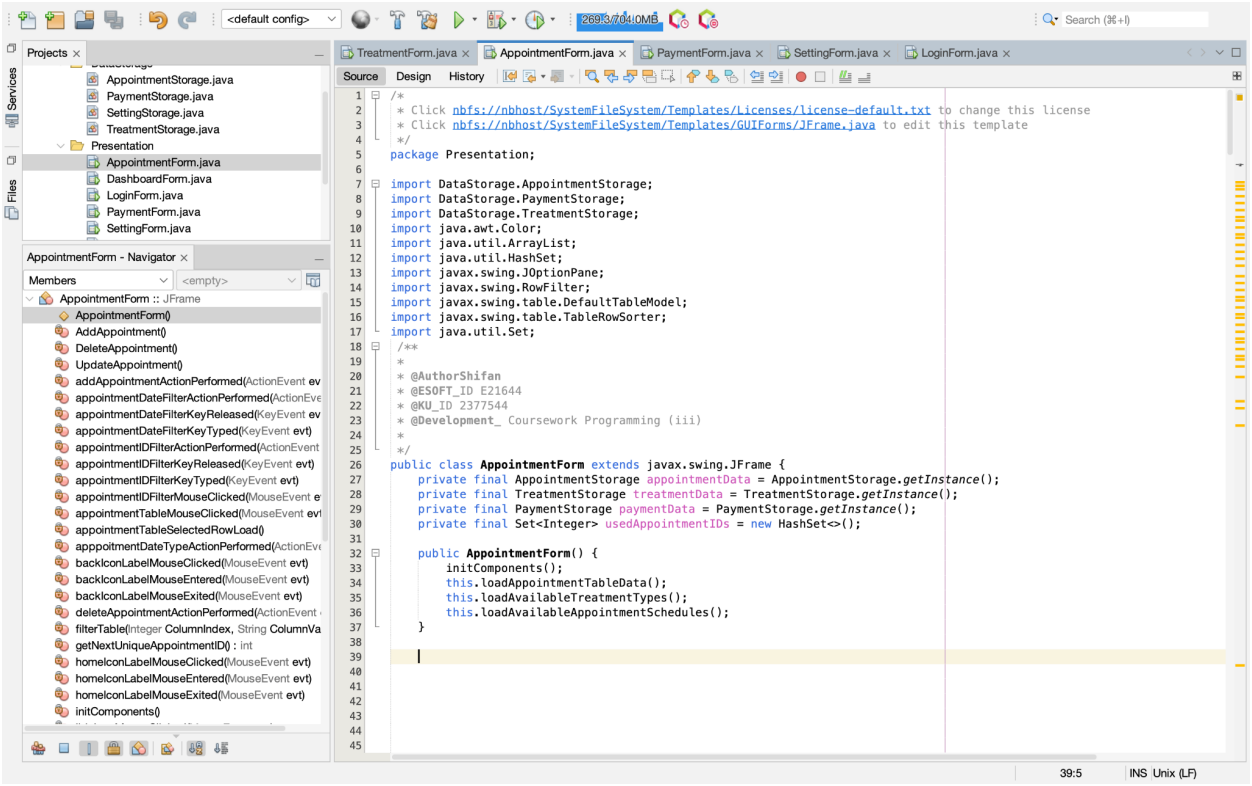


Treatment Form UI Fields Naming

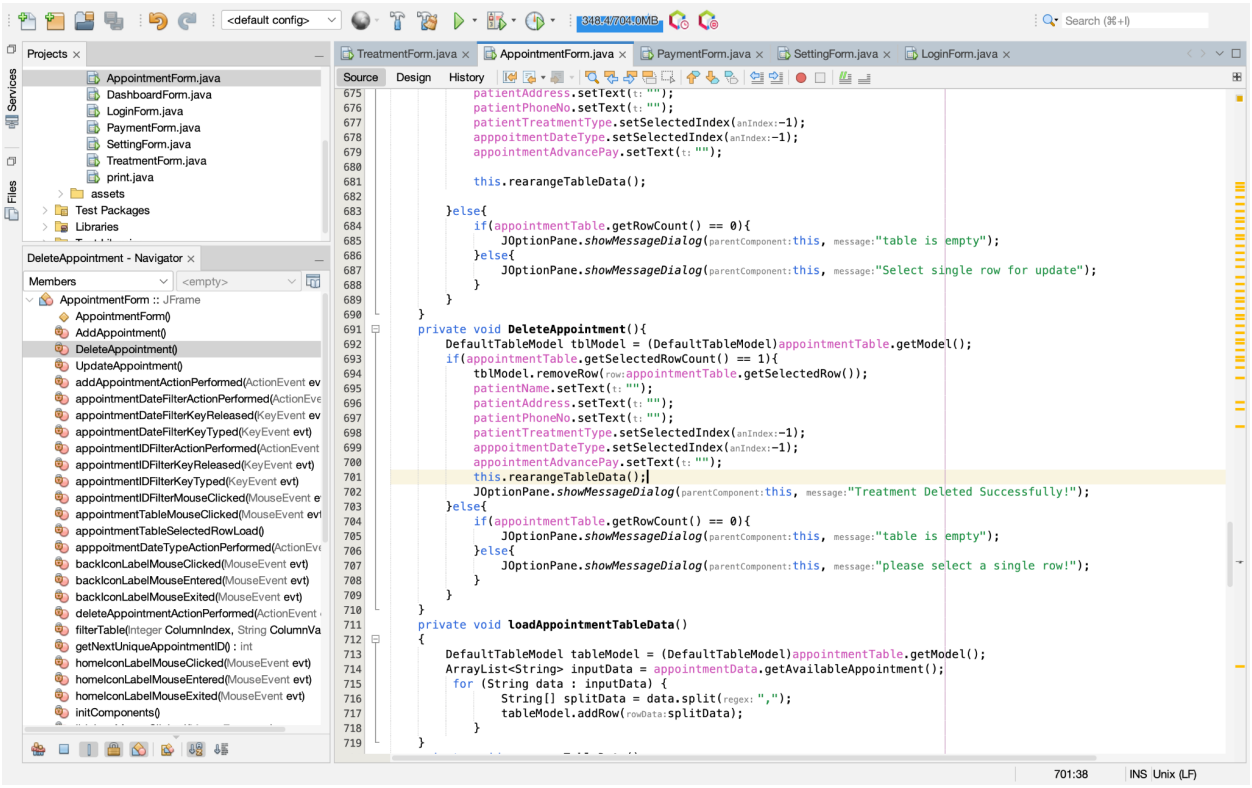


Appointment Form UI Fields Naming

Clarity of code



```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */
package Presentation;

import DataStorage.AppointmentStorage;
import DataStorage.PaymentStorage;
import DataStorage.TreatmentStorage;
import java.awt.Color;
import java.util.ArrayList;
import java.util.HashSet;
import javax.swing.JOptionPane;
import javax.swing.RowFilter;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
import java.util.Set;
/**
 *
 * @AuthorShifan
 * @ESOFT_ID E21644
 * @KU_ID 2377544
 * @Development_ Coursework Programming (iii)
 *
 */
public class AppointmentForm extends javax.swing.JFrame {
    private final AppointmentStorage appointmentData = AppointmentStorage.getInstance();
    private final TreatmentStorage treatmentData = TreatmentStorage.getInstance();
    private final PaymentStorage paymentData = PaymentStorage.getInstance();
    private final Set<Integer> usedAppointmentIDs = new HashSet<>();

    public AppointmentForm() {
        initComponents();
        this.loadAppointmentTableData();
        this.loadAvailableTreatmentTypes();
        this.loadAvailableAppointmentSchedules();
    }
```



```
                patientAddress.setText(t: "");
                patientPhoneNo.setText(t: "");
                patientTreatmentType.setSelectedIndex(anIndex:-1);
                apppoitmentDateType.setSelectedIndex(anIndex:-1);
                appointmentAdvancePay.setText(t: "");

                this.rearangeTableData();

            }else{
                if(appointmentTable.getRowCount() == 0){
                    JOptionPane.showMessageDialog(parentComponent:this, message:"table is empty");
                }else{
                    JOptionPane.showMessageDialog(parentComponent:this, message:"Select single row for update");
                }
            }
        }
    private void DeleteAppointment(){
        DefaultTableModel tblModel = (DefaultTableModel)appointmentTable.getModel();
        if(appointmentTable.getSelectedRowCount() == 1){
            tblModel.removeRow(row:appointmentTable.getSelectedRow());
            patientName.setText(t: "");
            patientAddress.setText(t: "");
            patientPhoneNo.setText(t: "");
            patientTreatmentType.setSelectedIndex(anIndex:-1);
            apppoitmentDateType.setSelectedIndex(anIndex:-1);
            appointmentAdvancePay.setText(t: "");
            this.rearangeTableData();
            JOptionPane.showMessageDialog(parentComponent:this, message:"Treatment Deleted Successfully!");
        }else{
            if(appointmentTable.getRowCount() == 0){
                JOptionPane.showMessageDialog(parentComponent:this, message:"table is empty");
            }else{
                JOptionPane.showMessageDialog(parentComponent:this, message:"please select a single row!");
            }
        }
    }
    private void loadAppointmentTableData()
    {
        DefaultTableModel tableModel = (DefaultTableModel)appointmentTable.getModel();
        ArrayList<String> inputData = appointmentData.getAvailableAppointment();
         for (String data : inputData) {
                String[] splitData = data.split(regex: ",");
                tableModel.addRow(rowData:splitData);
            }
    }
```

Comments

```
655
656     private int getNextUniqueAppointmentID() {
657         int nextAppointmentID = 1;
658
659         // Keep generating IDs until a unique one is found
660         while (usedAppointmentIDs.contains(o: nextAppointmentID)) {
661             nextAppointmentID++;
662         }
663
664         return nextAppointmentID;
665     }
666
```

```
433
434     private void updateTreatmentTableSelectedRow(){
435         DefaultTableModel tblModel = (DefaultTableModel)treatmentTable.getModel();
436         if(treatmentTable.getSelectedRowCount() == 1){
437             String Code = treatmentCode.getText();
438             String Name = treatmentName.getText();
439             String PriceCharge = treatmentPriceCharge.getText();
440             int selectedRow = treatmentTable.getSelectedRow();
441             tblModel.setValueAt(aValue: Code, row:treatmentTable.getSelectedRow(), column: 0);
442             tblModel.setValueAt(aValue: Name, row:treatmentTable.getSelectedRow(), column: 1);
443             tblModel.setValueAt(aValue: PriceCharge, row:treatmentTable.getSelectedRow(), column: 2);
444             JOptionPane.showMessageDialog(parentComponent:this, message:"Treatment Updated successfully!");
445             treatmentCode.setText(t: "");
446             treatmentName.setText(t: "");
447             treatmentPriceCharge.setText(t: "");
448             // Update TreatmentStorage array data by separate function
449             updateTreatmentStorage(row:selectedRow, code:Code, name:Name, priceCharge:PriceCharge);
450         }else{
451             if(treatmentTable.getRowCount() == 0){
452                 JOptionPane.showMessageDialog(parentComponent:this, message:"table is empty");
453             }else{
454                 JOptionPane.showMessageDialog(parentComponent:this, message:"Select single row for update");
455             }
456         }
457     }
458
```

```
403
404 private void deleteTreatmentFromTreatmentTable(){
405     DefaultTableModel tblModel = (DefaultTableModel)treatmentTable.getModel();
406     //make sure selected row count 1
407     if(treatmentTable.getSelectedRowCount() == 1){
408         int selectedRow = treatmentTable.getSelectedRow();
409         tblModel.removeRow(row:treatmentTable.getSelectedRow());
410         treatmentCode.setText(t: "");
411         treatmentName.setText(t: "");
412         treatmentPriceCharge.setText(t: "");
413         treatmentData.getAvailableTreatments().remove(index: selectedRow);
414         JOptionPane.showMessageDialog(parentComponent:this, message:"Treatment Deleted Successfully!");
415     }else{
416         if(treatmentTable.getRowCount() == 0){
417             JOptionPane.showMessageDialog(parentComponent:this, message:"table is empty");
418         }else{
419             JOptionPane.showMessageDialog(parentComponent:this, message:"please select a single row!");
420         }
421     }
422 }
423
```

Google Drive Link

https://drive.google.com/drive/folders/1QZ8GrLSDxOfyLOxNUh7FUVmkZtk2jGE_?usp=sharing