**ADVANCED JAVA PROGRAMMING – PROJECT PHASE 1**

**Tejeswi Devi Priya Pillarisetty N01654962**

This document contains the following

* Assignment and student details
* Completion Plan
* Core concepts used in the project
* Reflection on the concepts learnt
* Links to the GitHub repository
* Source Code
* Instructions to run the code
* Output Screenshots
* Conclusion

# Assignment and student details:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Project Phase #1**

**Course:** *Advanced Java Programming - ITE-5215-0NB - Semester 2*

**Last Name:** *Pillarisetty*

**First Name:** *Tejeswi Devi Priya*

**ID:** *N01654962*

**Section:** *0NB*

*This assignment represents my own work in accordance with Humber Academic Policy.*

***Signature:*** *Tejeswi Devi Priya Pillarisetty*

**Date:** *22nd July 2024*

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Project details:**

**Description:**  
You are tasked with developing a comprehensive medical healthcare application that leverages  
JavaFX for its user interface. The application will allow healthcare professionals to manage patient records, appointments, and medical history. The project will incorporate key programming concepts such as event-driven programming, file handling (both text and binary), lambda expressions, and exception handling.  
**Task 1:**  
Design a user-friendly GUI using JavaFX  
The details for the task are as follows,

• A login screen for healthcare professionals.  
• A dashboard displaying patient information, appointments, and medical history.  
• Forms for adding and updating patient records and appointment details.  
• Navigation controls to switch between different sections of the application.

**Task2 :**  
Implement event-driven programming to handle user interactions.  
Utilize lambda expressions for event handling.  
The details for the task are as follows,

• Handle user actions such as logging in, navigating between screens, and submitting forms.  
• Use lambda expressions to manage event handlers for buttons and other interactive  
elements.

**Task3:**  
Read from and write to text files and binary files for data persistence.  
The details for the task are as follows,

• Implement methods to save patient records and appointment details to text files and binary  
files.  
• Implement methods to read patient data and appointment details from these files and  
display them in the application.

**Task4:**  
Handle any exceptions  
The details for the task are as follows,

• Handle file I/O exceptions to ensure the application can recover from errors during file  
operations.  
• Display user-friendly error messages using JavaFX dialogs when exceptions occur.

**Detailed Requirements:  
Login Screen**

* Fields for username and password.
* A login button that validates credentials and directs the user to the main dashboard.

**Dashboard:**

* Display a summary of patient records and upcoming appointments.
* Provide navigation options to view detailed patient records and appointment schedules.

**Patient Records:**

* Form to add new patient records, including fields for name, age, medical history, and contact information.
* Option to update existing patient records.
* Save patient data to text and binary files.
* Load patient data from files when the application starts.

**Appointments:**

* Form to schedule new appointments, including fields for patient name, date, time, and reason for visit.
* Option to update existing appointment details.
* Save appointment data to text and binary files.
* Load appointment data from files when the application starts.

**Error Handling:**

* Implement try-catch blocks for all file I/O operations.

**Hint :**  
**Define the screens** : Each screen can be represented by s scene object  
**Switching screens**: use a stage object to manage and switch between different scenes.

**Project Header**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Project Phase #  
Course:<subject type> -  
Semester Last  
Name(s) :<student last name>  
First Name(s):<student first  
name> ID:<student ID>  
Section:<section name>  
This project represents our own work in accordance with Humber Academic Policy.  
Signature  
Date:<submission date>  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
**Code Submission Criteria:**  
Please note that you should have:

• Appropriate indentation.  
• Proper file structure  
• Follow java naming convention  
• Follow OOP style  
• Document all the classes properly  
• Do Not have any debug/ useless code and/ or files in the assignment  
• Do not have everything in the main method.  
• Have a separate TestClass with the main method in it.  
• Check your inputs if the user is not entering garbage inputs.  
• Use exceptional handling or other methods to let the user know if the inputs  
are incorrect.

# Completion plan:

This assignment completion plan is as follows:

**Task 1:** Done by Tejeswi Devi Priya Pillarisetty

**Task 2:** Done by Tejeswi Devi Priya Pillarisetty & Devwrat Raval

**Task 3:** Done by Dewrat Raval

**Task 4:** Done by Shubham Sangwan

# Core concepts used in the project:

* JavaFX GUI Design
* Event-Driven Programming
* File Handling (Text and Binary)
* Exception Handling

# Reflection on the concepts learnt:

1. **JavaFX GUI Design**

JavaFX is a framework for building rich user interfaces in Java. It provides a wide range of UI controls and layout options to design a visually appealing and interactive application.

In this project, JavaFX was used to create a login screen, a dashboard, forms for managing patient records and appointments. Components like ‘TextField’, ‘PasswordField’, ‘Button’ and ‘ListView’ were used to collect user input and display data.

1. **Event Driven Programming**

Event Driven Programming determines the flow of program by events such as user actions.

In this project, lambda expressions and methods references were used to handle user interactions with various GUI elements. For example, clicking a submit button triggers an event handler that processes form data and provides a responsive user experience.

1. **File Handling (Text and Binary)**

File Handling is essential for data persistence allowing the application to save and retrieve information between sessions.

In this project, we used file handling techniques for reading from and writing to both text and binary files. Methods implemented in ‘ReadTxtNBinary.java’ and ‘WriteTxtNBinary.java’ managed to use serialization and deserialization processes for patient and appointment data ensuring that user inputs are preserved and loaded correctly.

1. **Exception Handling**

Exception Handling provides mechanisms to gracefully handle errors and prevent application crashes.

In this project, try catch blocks were used to manage file I/O exceptions, ensuring the application can recover from issues like missing or corrupt files. User friendly error messages were displayed using JavaFX dialog boxes to inform users of problems in a clear and understandable manner.

# Links to the GitHub repository:

# <https://github.com/Priya-2705/MyCare-Phase1.git>

# Project Hierarchy

A screen shot of a computer

Description automatically generated

# Source Code and Output Screenshots:

The following are the steps followed in completing this assignment:

**Creating a java project in Eclipse**

* Open Eclipse IDE.
* Go to File 🡪 New 🡪 Others 🡪 Java Project.
* Give the project name ‘MyCare’ and click Finish.
* Create a package named ‘com.humber.project in src folder of the project.
* Then, create the following class files in it.

**Login.java**

package com.humber.project;

import javafx.application.Application;

import javafx.geometry.HPos;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.GridPane;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.stage.Stage;

public class Login extends Application {

@Override

public void start(Stage primaryStage) {

// Create form elements

// Heading label for the login form

Label headingLabel = new Label("MyCare");

headingLabel.setFont(Font.font("Arial", FontWeight.BOLD, 36));

headingLabel.setTextFill(Color.web("#FFFFFF"));

headingLabel.setPadding(new Insets(60, 0, -20, 0));

// Label and text field for username input

Label userNameLabel = new Label("Username:");

userNameLabel.setFont(Font.font("Arial", FontWeight.BOLD, 14));

userNameLabel.setTextFill(Color.web("#FFFFFF"));

TextField userNameField = new TextField();

userNameField.setPromptText("Enter username");

userNameField.setPrefWidth(200);

userNameField.setStyle("-fx-background-color: #E0E0E0;");

// Label and password field for password input

Label passwordLabel = new Label("Password:");

passwordLabel.setFont(Font.font("Arial", FontWeight.BOLD, 14));

passwordLabel.setTextFill(Color.web("#FFFFFF"));

PasswordField passwordField = new PasswordField();

passwordField.setPromptText("Enter password");

passwordField.setPrefWidth(200);

passwordField.setStyle("-fx-background-color: #E0E0E0;");

// Login button with styling and hover effects

Button loginButton = new Button("Login");

loginButton.setFont(Font.font("Arial", FontWeight.BOLD, 14));

loginButton.setStyle("-fx-background-color: #FF5722; -fx-text-fill: white;");

loginButton.setOnMouseEntered(e -> loginButton.setStyle("-fx-background-color: #E64A19; -fx-text-fill: white;"));

loginButton.setOnMouseExited(e -> loginButton.setStyle("-fx-background-color: #FF5722; -fx-text-fill: white;"));

// Create layout for form with labels and fields beside each other

GridPane formLayout = new GridPane();

formLayout.setHgap(10);

formLayout.setVgap(20);

formLayout.setPadding(new Insets(20));

formLayout.setAlignment(Pos.CENTER);

// Add form elements to the grid

formLayout.add(userNameLabel, 0, 0);

formLayout.add(userNameField, 1, 0);

formLayout.add(passwordLabel, 0, 1);

formLayout.add(passwordField, 1, 1);

formLayout.add(loginButton, 1, 2);

GridPane.setHalignment(loginButton, HPos.CENTER);

// Create main layout

BorderPane mainLayout = new BorderPane();

mainLayout.setTop(headingLabel);

BorderPane.setAlignment(headingLabel, Pos.CENTER);

mainLayout.setCenter(formLayout);

mainLayout.setStyle("-fx-background-color: #009688;");

// Set button action

loginButton.setOnAction(e -> handleLogin(userNameField.getText(), passwordField.getText(), primaryStage));

// Create scene with consistent screen size and set the stage

Scene scene = new Scene(mainLayout, 700, 400);

primaryStage.setTitle("Login");

primaryStage.setScene(scene);

primaryStage.show();

}

// Handle login action

private void handleLogin(String username, String password, Stage primaryStage) {

// Check if the username and password match the required values

if ((username.equals("Priya") || username.equals("Shubham") || username.equals("Devwrat")) && "123456".equals(password)) {

Dashboard dashboard = new Dashboard();

try {

dashboard.start(primaryStage);

} catch (Exception ex) {

ex.printStackTrace();

}

} else {

showErrorDialog("Login Failed", "Invalid username or password. Please try again.");

}

}

// Show an error dialog if login fails

private void showErrorDialog(String title, String message) {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.showAndWait();

}

// Main method to launch the application

public static void main(String[] args) {

launch(args);

}

}

**Dashboard.java**

package com.humber.project;

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.ListView;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.stage.Stage;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

public class Dashboard extends Application {

private static Dashboard instance;

private ListView<String> patientListView = new ListView<>();

private ListView<String> appointmentListView = new ListView<>();

// Constructor to initialize the Dashboard instance

public Dashboard() {

instance = this;

}

// Get the singleton instance of Dashboard

public static Dashboard getInstance() {

return instance;

}

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("Dashboard");

VBox vbox = new VBox(10);

vbox.setPadding(new Insets(10, 10, 10, 10));

vbox.setStyle("-fx-background-color: #009688;"); // Teal background color

// Create and style buttons

Button managePatientsButton = createStyledButton("Manage Patients");

Button manageAppointmentsButton = createStyledButton("Manage Appointments");

Button logoutButton = createStyledButton("Logout");

// Set button actions

managePatientsButton.setOnAction(e -> showPatientRecordsForm(primaryStage));

manageAppointmentsButton.setOnAction(e -> showAppointmentForm(primaryStage));

logoutButton.setOnAction(e -> primaryStage.close());

// Style labels

Label patientLabel = createStyledLabel("Patient Records:");

Label appointmentLabel = createStyledLabel("Appointments:");

// Style list views

patientListView.setStyle("-fx-background-color: #E0E0E0; -fx-border-color: #004D40; -fx-border-width: 2px;");

appointmentListView.setStyle("-fx-background-color: #E0E0E0; -fx-border-color: #004D40; -fx-border-width: 2px;");

vbox.getChildren().addAll(

managePatientsButton,

manageAppointmentsButton,

logoutButton,

patientLabel,

patientListView,

appointmentLabel,

appointmentListView

);

Scene scene = new Scene(vbox, 700, 400); // Set to match the size of PatientRecordsForm

primaryStage.setScene(scene);

primaryStage.show();

updateDashboard();

}

// Create a styled button with hover effects

private Button createStyledButton(String text) {

Button button = new Button(text);

button.setFont(Font.font("Arial", FontWeight.BOLD, 14));

button.setStyle("-fx-background-color: #FF5722; -fx-text-fill: white;");

button.setOnMouseEntered(e -> button.setStyle("-fx-background-color: #E64A19; -fx-text-fill: white;"));

button.setOnMouseExited(e -> button.setStyle("-fx-background-color: #FF5722; -fx-text-fill: white;"));

return button;

}

// Create a styled label with bold and white text

private Label createStyledLabel(String text) {

Label label = new Label(text);

label.setFont(Font.font("Arial", FontWeight.BOLD, 16)); // Bold font

label.setTextFill(Color.WHITE); // White text color

return label;

}

// Show Patient Records Form

private void showPatientRecordsForm(Stage primaryStage) {

new PatientRecordsForm().start(primaryStage);

}

// Show Appointment Form

private void showAppointmentForm(Stage primaryStage) {

new AppointmentForm().start(primaryStage);

}

// Update Dashboard with patient and appointment data

public void updateDashboard() {

patientListView.getItems().clear();

appointmentListView.getItems().clear();

List<String> patients = loadDataFromTextFile("patients.txt");

patientListView.getItems().add(formatPatientHeader());

patientListView.getItems().addAll(formatPatientData(patients));

List<String> appointments = loadDataFromTextFile("appointments.txt");

appointmentListView.getItems().add(formatAppointmentHeader());

appointmentListView.getItems().addAll(formatAppointmentData(appointments));

}

// Format header for patient data

private String formatPatientHeader() {

return String.format("%-18s %-15s %-38s %-27s %-55s", "Name", "Age", "Contact", "DOB", "Medical History");

}

// Format patient data for display

private List<String> formatPatientData(List<String> data) {

List<String> formattedData = new ArrayList<>();

for (String line : data) {

String[] parts = line.split(",");

if (parts.length == 5) {

String formattedLine = String.format("%-20s %-15s %-30s %-25s %-55s", parts[0], parts[1], parts[2], parts[4], parts[3]);

formattedData.add(formattedLine);

}

}

return formattedData;

}

// Format header for appointment data

private String formatAppointmentHeader() {

return String.format("%-30s %-50s %-20s %-50s", "Name", "Date", "Time", "Reason");

}

// Format appointment data for display

private List<String> formatAppointmentData(List<String> data) {

List<String> formattedData = new ArrayList<>();

for (String line : data) {

String[] parts = line.split(",");

if (parts.length == 4) {

String formattedLine = String.format("%-30s %-45s %-20s %-50s", parts[0], parts[1], parts[2], parts[3]);

formattedData.add(formattedLine);

}

}

return formattedData;

}

// Load data from a text file

private List<String> loadDataFromTextFile(String fileName) {

List<String> data = new ArrayList<>();

File file = new File(fileName);

if (file.exists()) {

try (BufferedReader reader = new BufferedReader(new FileReader(file))) {

String line;

while ((line = reader.readLine()) != null) {

data.add(line);

}

} catch (IOException e) {

e.printStackTrace();

}

}

return data;

}

public static void main(String[] args) {

launch(args);

}

}

**PatientsRecordsForm.java**

package com.humber.project;

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.VBox;

import javafx.scene.layout.HBox;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.stage.Stage;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.List;

public class PatientRecordsForm extends Application {

private List<String> patients;

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("Patient Records");

// Layout setup

VBox vbox = new VBox(10);

vbox.setPadding(new Insets(10, 10, 10, 10));

vbox.setStyle("-fx-background-color: #009688;");

// Title Label

Label titleLabel = new Label("Patient Records Form");

titleLabel.setFont(Font.font("Arial", FontWeight.BOLD, 20));

titleLabel.setTextFill(Color.web("#FFFFFF"));

// Form layout

GridPane grid = new GridPane();

grid.setHgap(10);

grid.setVgap(10);

grid.setPadding(new Insets(10, 10, 10, 10));

// Form fields and labels

Label nameLabel = createStyledLabel("Name:");

Label ageLabel = createStyledLabel("Age:");

Label contactLabel = createStyledLabel("Contact Info:");

Label historyLabel = createStyledLabel("Medical History:");

Label dobLabel = createStyledLabel("Date of Birth:");

// Create and style form controls

TextField nameField = new TextField();

TextField ageField = new TextField();

TextField contactField = new TextField();

TextArea historyArea = new TextArea();

historyArea.setPrefRowCount(5);

historyArea.setWrapText(true);

DatePicker dobPicker = new DatePicker();

// Add controls to grid

grid.add(nameLabel, 0, 0);

grid.add(nameField, 1, 0);

grid.add(ageLabel, 0, 1);

grid.add(ageField, 1, 1);

grid.add(contactLabel, 0, 2);

grid.add(contactField, 1, 2);

grid.add(dobLabel, 0, 3);

grid.add(dobPicker, 1, 3);

grid.add(historyLabel, 0, 4);

grid.add(historyArea, 1, 4);

// Buttons

Button submitButton = new Button("Submit");

Button updateButton = new Button("Update");

Button backButton = new Button("Back to Dashboard");

// Apply styling to buttons

styleButton(submitButton);

styleButton(updateButton);

styleButton(backButton);

// Buttons layout

HBox buttonBox = new HBox(20, submitButton, updateButton);

buttonBox.setAlignment(Pos.CENTER);

buttonBox.setPadding(new Insets(10, 0, 10, 0));

// Add buttons to the grid

grid.add(buttonBox, 0, 5, 2, 1);

grid.add(backButton, 0, 6, 2, 1);

// Button actions

submitButton.setOnAction(e -> handleSubmit(nameField.getText(), ageField.getText(), contactField.getText(), historyArea.getText(), dobPicker.getValue()));

updateButton.setOnAction(e -> handleUpdate(nameField.getText(), ageField.getText(), contactField.getText(), historyArea.getText(), dobPicker.getValue()));

backButton.setOnAction(e -> showDashboard(primaryStage));

// Load patient data

patients = ReadTxtNBinary.loadPatientDataFromBinaryFile();

vbox.getChildren().addAll(titleLabel, grid);

Scene scene = new Scene(vbox, 700, 400);

primaryStage.setScene(scene);

primaryStage.show();

}

// Helper method to create and style labels

private Label createStyledLabel(String text) {

Label label = new Label(text);

label.setFont(Font.font("Arial", FontWeight.BOLD, 14));

label.setTextFill(Color.web("#FFFFFF"));

return label;

}

// Helper method to style buttons

private void styleButton(Button button) {

button.setFont(Font.font("Arial", FontWeight.BOLD, 14)); // Bold font

button.setStyle("-fx-background-color: #FF5722; -fx-text-fill: white;"); // Deep Orange background

button.setOnMouseEntered(e -> button.setStyle("-fx-background-color: #E64A19; -fx-text-fill: white;"));

button.setOnMouseExited(e -> button.setStyle("-fx-background-color: #FF5722; -fx-text-fill: white;"));

}

// Handle form submission

private void handleSubmit(String name, String age, String contact, String history, LocalDate dob) {

if (!isValidName(name)) {

showErrorDialog("Invalid Name", "Name should only contain letters.");

return;

}

if (!isValidAge(age)) {

showErrorDialog("Invalid Age", "Age should be a 1 or 2-digit integer.");

return;

}

if (!isValidContact(contact)) {

showErrorDialog("Invalid Contact Info", "Contact number should be in the format +1-nnn-nnn-nnnn.");

return;

}

if (dob == null || dob.isAfter(LocalDate.now())) {

showErrorDialog("Invalid Date", "Date of birth should be a past date.");

return;

}

String patientData = name + "," + age + "," + contact + "," + history + "," + dob;

try {

patients.add(patientData);

WriteTxtNBinary.savePatientDataToTextFile(patients);

WriteTxtNBinary.savePatientDataToBinaryFile(patients);

System.out.println("Patient data saved.");

} catch (Exception e) {

showErrorDialog("Error", "An error occurred while saving patient data.");

}

}

// Handle form update

private void handleUpdate(String name, String age, String contact, String history, LocalDate dob) {

if (!isValidName(name)) {

showErrorDialog("Invalid Name", "Name should only contain letters.");

return;

}

if (!isValidAge(age)) {

showErrorDialog("Invalid Age", "Age should be a 1 or 2-digit integer.");

return;

}

if (!isValidContact(contact)) {

showErrorDialog("Invalid Contact Info", "Contact number should be in the format +1-nnn-nnn-nnnn.");

return;

}

if (dob == null || dob.isAfter(LocalDate.now())) {

showErrorDialog("Invalid Date", "Date of birth should be a past date.");

return;

}

boolean updated = false;

List<String> updatedPatients = new ArrayList<>();

for (String patientData : patients) {

String[] parts = patientData.split(",");

if (parts[0].equals(name)) {

updatedPatients.add(name + "," + age + "," + contact + "," + history + "," + dob);

updated = true;

} else {

updatedPatients.add(patientData);

}

}

if (updated) {

try {

patients = updatedPatients;

WriteTxtNBinary.savePatientDataToTextFile(patients);

WriteTxtNBinary.savePatientDataToBinaryFile(patients);

System.out.println("Patient data updated.");

} catch (Exception e) {

showErrorDialog("Error", "An error occurred while updating patient data.");

}

} else {

System.out.println("No matching patient found to update.");

}

}

// Validate methods

private boolean isValidName(String name) {

return name != null && name.matches("[a-zA-Z]+");

}

private boolean isValidAge(String age) {

return age != null && age.matches("\\d{1,2}");

}

private boolean isValidContact(String contact) {

return contact != null && contact.matches("\\+1-\\d{3}-\\d{3}-\\d{4}");

}

// Show Dashboard

private void showDashboard(Stage primaryStage) {

new Dashboard().start(primaryStage);

}

// Show error dialog

private void showErrorDialog(String title, String message) {

Alert alert = new Alert(AlertType.ERROR);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.showAndWait();

}

public static void main(String[] args) {

launch(args);

}

}

**AppointmentForm.java**

package com.humber.project;

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.HBox;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.stage.Stage;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.List;

public class AppointmentForm extends Application {

private List<String> appointments;

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("Appointment Form");

// Layout setup

GridPane grid = new GridPane();

grid.setHgap(10);

grid.setVgap(10);

grid.setPadding(new Insets(10, 10, 10, 10));

// Title Label

Label titleLabel = new Label("Appointment Form");

titleLabel.setFont(Font.font("Arial", FontWeight.BOLD, 20)); // Bold font

titleLabel.setTextFill(Color.web("#FFFFFF")); // White color for title

// Form Labels

Label patientNameLabel = new Label("Patient Name:");

Label dateLabel = new Label("Date:");

Label timeLabel = new Label("Time:");

Label detailsLabel = new Label("Details:");

// Styling Labels

styleLabel(patientNameLabel);

styleLabel(dateLabel);

styleLabel(timeLabel);

styleLabel(detailsLabel);

// Form Controls

TextField patientNameField = new TextField();

DatePicker datePicker = new DatePicker();

TextField timeField = new TextField();

TextArea detailsArea = new TextArea();

detailsArea.setPrefRowCount(5);

detailsArea.setWrapText(true);

// Placeholder text and validation

timeField.setTextFormatter(new TextFormatter<>(change -> {

String newText = change.getControlNewText();

if (newText.matches("[0-9:]\*")) {

return change;

} else {

return null;

}

}));

datePicker.setDayCellFactory(picker -> new DateCell() {

@Override

public void updateItem(LocalDate date, boolean empty) {

super.updateItem(date, empty);

setDisable(empty || date.isBefore(LocalDate.now()));

}

});

// Adding controls to GridPane

grid.add(patientNameLabel, 0, 0);

grid.add(patientNameField, 1, 0);

grid.add(dateLabel, 0, 1);

grid.add(datePicker, 1, 1);

grid.add(timeLabel, 0, 2);

grid.add(timeField, 1, 2);

grid.add(detailsLabel, 0, 3);

grid.add(detailsArea, 1, 3);

// Buttons

Button submitButton = new Button("Submit");

Button updateButton = new Button("Update");

Button backButton = new Button("Back to Dashboard");

// Apply styling to buttons

styleButton(submitButton);

styleButton(updateButton);

styleButton(backButton);

// Buttons layout

HBox buttonBox = new HBox(20, submitButton, updateButton);

buttonBox.setAlignment(Pos.CENTER); // Center align buttons

buttonBox.setPadding(new Insets(10, 0, 10, 0)); // Padding around buttons

// Adding buttons to GridPane

grid.add(buttonBox, 0, 4, 2, 1); // Span across two columns

grid.add(backButton, 0, 5, 2, 1); // Span across two columns

// Button Actions

submitButton.setOnAction(e -> handleSubmit(patientNameField.getText(), datePicker.getValue(), timeField.getText(), detailsArea.getText()));

updateButton.setOnAction(e -> handleUpdate(patientNameField.getText(), datePicker.getValue(), timeField.getText(), detailsArea.getText()));

backButton.setOnAction(e -> showDashboard(primaryStage));

// Load existing appointment data

appointments = ReadTxtNBinary.loadAppointmentDataFromBinaryFile();

// Root Pane

GridPane rootPane = new GridPane();

rootPane.setPadding(new Insets(10));

rootPane.setStyle("-fx-background-color: #009688;"); // Teal background color

rootPane.add(titleLabel, 0, 0);

rootPane.add(grid, 0, 1);

// Scene Setup

Scene scene = new Scene(rootPane, 700, 400);

primaryStage.setScene(scene);

primaryStage.show();

}

// Styling for Labels

private void styleLabel(Label label) {

label.setFont(Font.font("Arial", FontWeight.BOLD, 14)); // Bold font

label.setTextFill(Color.web("#FFFFFF")); // White color

}

// Styling for Buttons

private void styleButton(Button button) {

button.setFont(Font.font("Arial", FontWeight.BOLD, 14)); // Bold font

button.setStyle("-fx-background-color: #FF5722; -fx-text-fill: white;"); // Deep Orange background

button.setOnMouseEntered(e -> button.setStyle("-fx-background-color: #E64A19; -fx-text-fill: white;"));

button.setOnMouseExited(e -> button.setStyle("-fx-background-color: #FF5722; -fx-text-fill: white;"));

}

// Handle form submission

private void handleSubmit(String patientName, LocalDate date, String time, String details) {

if (!isValidName(patientName)) {

showErrorDialog("Invalid Name", "Name should only contain letters.");

return;

}

if (!isValidTime(time)) {

showErrorDialog("Invalid Time", "Time should be in HH:MM 24-hour format.");

return;

}

if (date == null || !isFutureDate(date)) {

showErrorDialog("Invalid Date", "Date should be a future date.");

return;

}

String appointmentData = patientName + "," + date + "," + time + "," + details;

try {

appointments.add(appointmentData);

WriteTxtNBinary.saveAppointmentDataToTextFile(appointments);

WriteTxtNBinary.saveAppointmentDataToBinaryFile(appointments);

System.out.println("Appointment data saved.");

} catch (Exception e) {

showErrorDialog("Error", "An error occurred while saving appointment data.");

}

}

// Handle form update

private void handleUpdate(String patientName, LocalDate date, String time, String details) {

if (!isValidName(patientName)) {

showErrorDialog("Invalid Name", "Name should only contain letters.");

return;

}

if (!isValidTime(time)) {

showErrorDialog("Invalid Time", "Time should be in HH:MM 24-hour format.");

return;

}

if (date == null || !isFutureDate(date)) {

showErrorDialog("Invalid Date", "Date should be a future date.");

return;

}

boolean updated = false;

List<String> updatedAppointments = new ArrayList<>();

for (String appointmentData : appointments) {

String[] parts = appointmentData.split(",");

if (parts[0].equals(patientName) && parts[1].equals(date.toString()) && parts[2].equals(time)) {

updatedAppointments.add(patientName + "," + date + "," + time + "," + details);

updated = true;

} else {

updatedAppointments.add(appointmentData);

}

}

if (updated) {

try {

appointments = updatedAppointments;

WriteTxtNBinary.saveAppointmentDataToTextFile(appointments);

WriteTxtNBinary.saveAppointmentDataToBinaryFile(appointments);

System.out.println("Appointment data updated.");

} catch (Exception e) {

showErrorDialog("Error", "An error occurred while updating appointment data.");

}

} else {

System.out.println("No matching appointment found to update.");

}

}

// Validation Methods

private boolean isValidName(String name) {

return name != null && name.matches("[a-zA-Z]+");

}

private boolean isValidTime(String time) {

return time != null && time.matches("([01]\\d|2[0-3]):[0-5]\\d");

}

private boolean isFutureDate(LocalDate date) {

return date != null && date.isAfter(LocalDate.now());

}

// Show Dashboard

private void showDashboard(Stage primaryStage) {

new Dashboard().start(primaryStage);

}

// Show error dialog

private void showErrorDialog(String title, String message) {

Alert alert = new Alert(AlertType.ERROR);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.showAndWait();

}

public static void main(String[] args) {

launch(args);

}

}

**ReadTxtNBinary.java**

package com.humber.project;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

public class ReadTxtNBinary {

// Load patient data from a text file

public static List<String> loadPatientDataFromTextFile() {

List<String> patients = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader("patients.txt"))) {

String line;

while ((line = reader.readLine()) != null) {

patients.add(line);

}

} catch (IOException e) {

showErrorDialog("Error reading patient data", "An error occurred while reading patient data from the text file.");

}

return patients;

}

// Load patient data from a binary file

public static List<String> loadPatientDataFromBinaryFile() {

List<String> patients = new ArrayList<>();

File file = new File("patients.dat");

if (file.exists()) {

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(file))) {

patients = (List<String>) ois.readObject();

} catch (IOException | ClassNotFoundException e) {

showErrorDialog("Error reading patient data", "An error occurred while reading patient data from the binary file.");

}

}

return patients;

}

// Load appointment data from a text file

public static List<String> loadAppointmentDataFromTextFile() {

List<String> appointments = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader("appointments.txt"))) {

String line;

while ((line = reader.readLine()) != null) {

appointments.add(line);

}

} catch (IOException e) {

showErrorDialog("Error reading appointment data", "An error occurred while reading appointment data from the text file.");

}

return appointments;

}

// Load appointment data from a binary file

public static List<String> loadAppointmentDataFromBinaryFile() {

List<String> appointments = new ArrayList<>();

File file = new File("appointments.dat");

if (file.exists()) {

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(file))) {

appointments = (List<String>) ois.readObject();

} catch (IOException | ClassNotFoundException e) {

showErrorDialog("Error reading appointment data", "An error occurred while reading appointment data from the binary file.");

}

}

return appointments;

}

// Show an error dialog

private static void showErrorDialog(String title, String message) {

Alert alert = new Alert(AlertType.ERROR);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.showAndWait();

}

}

**WriteTxtNBinary.java**

package com.humber.project;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import java.io.\*;

import java.util.List;

public class WriteTxtNBinary {

// Save patient data to a text file

public static void savePatientDataToTextFile(List<String> patients) {

try (BufferedWriter writer = new BufferedWriter(new FileWriter("patients.txt"))) {

for (String patientData : patients) {

writer.write(patientData);

writer.newLine();

}

} catch (IOException e) {

showErrorDialog("Error writing patient data", "An error occurred while writing patient data to the text file.");

}

}

// Save patient data to a binary file

public static void savePatientDataToBinaryFile(List<String> patients) {

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("patients.dat"))) {

oos.writeObject(patients);

} catch (IOException e) {

showErrorDialog("Error writing patient data", "An error occurred while writing patient data to the binary file.");

}

}

// Save appointment data to a text file

public static void saveAppointmentDataToTextFile(List<String> appointments) {

try (BufferedWriter writer = new BufferedWriter(new FileWriter("appointments.txt"))) {

for (String appointmentData : appointments) {

writer.write(appointmentData);

writer.newLine();

}

} catch (IOException e) {

showErrorDialog("Error writing appointment data", "An error occurred while writing appointment data to the text file.");

}

}

// Save appointment data to a binary file

public static void saveAppointmentDataToBinaryFile(List<String> appointments) {

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("appointments.dat"))) {

oos.writeObject(appointments);

} catch (IOException e) {

showErrorDialog("Error writing appointment data", "An error occurred while writing appointment data to the binary file.");

}

}

// Show an error dialog if an exception occurs during file operations

private static void showErrorDialog(String title, String message) {

Alert alert = new Alert(AlertType.ERROR);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.showAndWait();

}

}

# Instructions on how to run the code:

* Download the MyCare.zip file and extract it.
* Open it in Eclipse IDE.
* Right click on MyCare java project and hover on Run as and then, click on Java application from the sub menu.
* To login, use username as “Priya” or “Shubham” or “Devwrat” and password as “123456”.
* After logging in, we can manage patients, manage their appointments and also logout.

# Output Screenshots:

**Login Page:**

A computer screen with a green screen

Description automatically generated

A computer screen shot of a computer screen

Description automatically generated

**Dashboard:**

**A computer screen with a green and white box

Description automatically generated**

**Patients Records Form Page:**

**A computer screen with a green and white box

Description automatically generated**

**A computer screen with a green box

Description automatically generated**

**A computer screen with a green box

Description automatically generated**

**A computer screen with a green box

Description automatically generated**

**A computer screen with a green box

Description automatically generated**

**A computer screen with a green box

Description automatically generated**

**A computer screen with a green and white box

Description automatically generated**

**A computer screen with a green box

Description automatically generatedA computer screen with a green box

Description automatically generatedA computer screen with a green and white box

Description automatically generated**

**Appointments Form Page:**

**A computer screen with a green and white box

Description automatically generated**

**A computer screen with a green and white box

Description automatically generatedA computer screen with a computer screen

Description automatically generated**

**A computer screen with a green box

Description automatically generatedA computer screen with a green box

Description automatically generatedA computer screen shot of a computer

Description automatically generatedA computer screen with a green box

Description automatically generatedA computer screen shot of a computer

Description automatically generated**

# Conclusion

In this project, I learned concepts of JavaFX, event driven programming, file handling and exception handling. By developing a user friendly medical health care application “MyCare”, I gained practical experience in managing user interactions and ensuring data persistence. The project emphasized clean code, proper documentation and best practices and providing valuable insights for future software development endeavours.