A logo of a university

Description automatically generated

|  |  |
| --- | --- |
| **Submitted By** | **Zaina Hassan SP24-BSE-126**  **Abeera Rehman SP24-BSE-005** |
| **Subject** | **Object Oriented Programming (OOP)** |
| **Assignment** | **Assignment 4**  **Project (Flight Mangement System)** |

**Submitted to :**

|  |  |
| --- | --- |
| **Moderator** | **Muhammad Shahid Bhatti** |

**Assignment 4**

**Flight Management System**

**Source Code For All Classes**

**LoginUI.java**

package com.example.sp24bse126;  
  
import javafx.application.Application;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.image.Image;  
import javafx.scene.layout.\*;  
import javafx.stage.Stage;  
  
public class LoginUI extends Application {  
  
 @Override  
 public void start(Stage primaryStage) {  
 // Labels and Fields  
 Label titleLabel = new Label("Login");  
 titleLabel.setStyle("-fx-font-size: 24px; -fx-text-fill: black; -fx-font-weight: bold;"); // Set title color to black  
 Label usernameLabel = new Label("Username:");  
 usernameLabel.setStyle("-fx-text-fill: black;");  
 TextField usernameField = new TextField();  
 usernameField.setPrefWidth(150);  
 usernameField.setMaxWidth(150);  
 Label passwordLabel = new Label("Password:");  
 passwordLabel.setStyle("-fx-text-fill: black;");  
 PasswordField passwordField = new PasswordField();  
 passwordField.setPrefWidth(150); // Set the preferred width of the password field  
 passwordField.setMaxWidth(150);  
 // Buttons  
 Button loginButton = new Button("Login");  
 loginButton.setStyle("-fx-background-color: black; -fx-text-fill: white; -fx-font-size: 14px;"); // Set login button to black background with white text  
  
 Button resetPasswordButton = new Button("Reset Password");  
 resetPasswordButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-size: 14px;");  
  
 Label messageLabel = new Label();  
 messageLabel.setStyle("-fx-text-fill: black; -fx-font-style: italic;");  
  
 // Button Actions  
 loginButton.setOnAction(e -> {  
 String username = usernameField.getText();  
 String password = passwordField.getText();  
 String role = Authentication.*validateLogin*(username, password);  
  
 if (role != null) {  
 messageLabel.setText("Login successful. Role: " + role);  
 if (role.equals("admin")) {  
 new AdminPanel().start(new Stage());  
 } else if (role.equals("passenger")) {  
 new FlightSearchUI().start(new Stage());  
 }  
 } else {  
 messageLabel.setText("Invalid username or password.");  
 }  
 });  
  
 resetPasswordButton.setOnAction(e -> openResetPasswordUI()); // Open Reset Password Window  
  
 // Layout for login form  
 VBox loginForm = new VBox(10, titleLabel, usernameLabel, usernameField, passwordLabel, passwordField, loginButton, resetPasswordButton, messageLabel);  
 loginForm.setAlignment(Pos.*CENTER*);  
 loginForm.setStyle("-fx-background-color: rgba(255, 255, 255, 0.8); -fx-padding: 20px; -fx-border-radius: 10px; -fx-background-radius: 10px;");  
  
 loginForm.setPrefWidth(300);  
 loginForm.setMaxWidth(300);  
 // Set a fixed width for the form  
 loginForm.setPrefHeight(450); // Set a fixed height for the form  
 loginForm.setMaxHeight(450);  
  
 StackPane root = new StackPane();  
 String resourcePath = getClass().getResource("/com/example/sp24bse126/background.jpg").toExternalForm();  
 root.setStyle("-fx-background-image: url('" + resourcePath + "'); -fx-background-size: cover; -fx-background-position: center;");  
  
 // Add login form to the center of the screen  
 root.getChildren().add(loginForm);  
  
 Scene scene = new Scene(root, 800, 600);  
 primaryStage.setTitle("Login");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 private void openResetPasswordUI() {  
 Stage resetStage = new Stage();  
 Label titleLabel = new Label("Reset Password");  
 titleLabel.setStyle("-fx-font-size: 20px; -fx-text-fill: black; -fx-font-weight: bold;");  
  
 Label usernameLabel = new Label("Username:");  
 TextField usernameField = new TextField();  
 usernameField.setPrefWidth(150); // Set the preferred width of the username field  
 usernameField.setMaxWidth(150);  
  
 Label oldPasswordLabel = new Label("Old Password:");  
 PasswordField oldPasswordField = new PasswordField();  
 oldPasswordField.setPrefWidth(150); // Set the preferred width of the password field  
 oldPasswordField.setMaxWidth(150);  
 Label newPasswordLabel = new Label("New Password:");  
 PasswordField newPasswordField = new PasswordField();  
 newPasswordField.setPrefWidth(150); // Set the preferred width of the password field  
 newPasswordField.setMaxWidth(150);  
 Button resetButton = new Button("Reset Password");  
 resetButton.setStyle("-fx-background-color: red; -fx-text-fill: white;");  
  
 Label resetMessageLabel = new Label();  
 resetMessageLabel.setStyle("-fx-text-fill: black;");  
  
 resetButton.setOnAction(e -> {  
 String username = usernameField.getText();  
 String oldPassword = oldPasswordField.getText();  
 String newPassword = newPasswordField.getText();  
  
 boolean success = Authentication.*resetPassword*(username, oldPassword, newPassword);  
 if (success) {  
 resetMessageLabel.setText("Password reset successfully!");  
 } else {  
 resetMessageLabel.setText("Invalid username or old password.");  
 }  
 });  
  
 VBox resetLayout = new VBox(10, titleLabel, usernameLabel, usernameField, oldPasswordLabel, oldPasswordField, newPasswordLabel, newPasswordField, resetButton, resetMessageLabel);  
 resetLayout.setAlignment(Pos.*CENTER*);  
 resetLayout.setStyle("-fx-background-color: rgba(255, 255, 255, 0.8); -fx-padding: 20px; -fx-border-radius: 10px; -fx-background-radius: 10px;");  
  
 Scene resetScene = new Scene(resetLayout, 400, 400);  
  
 resetStage.setTitle("Reset Password");  
 resetStage.setScene(resetScene);  
 resetStage.show();  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }

**Authentication.java**

package com.example.sp24bse126;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public class Authentication {  
  
 private static final Map<String, String> *users* = new HashMap<>();  
 private static final Map<String, String> *roles* = new HashMap<>();  
  
 static {  
  
 *users*.put("admin", "admin123");  
 *users*.put("passenger1", "pass123");  
  
  
 *roles*.put("admin", "admin");  
 *roles*.put("passenger1", "passenger");  
 }  
  
  
 public static String validateLogin(String username, String password) {  
 if (*users*.containsKey(username) && *users*.get(username).equals(password)) {  
 return *roles*.get(username);  
 }  
 return null;  
 }  
 public static boolean resetPassword(String username, String oldPassword, String newPassword) {  
 if (*users*.containsKey(username) && *users*.get(username).equals(oldPassword)) {  
 *users*.put(username, newPassword);  
 return true; // Password reset successfully  
 }  
 return false; // Invalid username or old password  
 }  
}

**AdminPanel.java**

package com.example.sp24bse126;  
  
import javafx.application.Application;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.layout.\*;  
import javafx.stage.Stage;  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.scene.control.TableView;  
  
public class AdminPanel extends Application {  
  
 private FlightDatabase flightDatabase;  
 private TableView<Flight> flightTable; // Declare flightTable as an instance variable  
  
 @Override  
 public void start(Stage primaryStage) {  
 flightDatabase = new FlightDatabase();  
 flightDatabase.loadFlights(getClass().getResource("/flights.txt").getPath());  
  
  
 Button addButton = new Button("Add Flight");  
 Button deleteButton = new Button("Delete Flight");  
 Button viewButton = new Button("View Flights");  
  
  
 HBox buttonBox = new HBox(10, addButton, deleteButton, viewButton);  
 buttonBox.setStyle("-fx-padding: 10; -fx-alignment: center;");  
  
 // dynamic content  
 StackPane contentArea = new StackPane();  
  
 // Initialize the flight table  
 flightTable = createFlightTable();  
 // Initialize with the loaded flights  
 ObservableList<Flight> flightObservableList = FXCollections.*observableArrayList*(flightDatabase.getFlights());  
 flightTable.setItems(flightObservableList);  
  
 // Display flight table(View Flights)  
 contentArea.getChildren().add(flightTable);  
  
 // Set actions for buttons  
 addButton.setOnAction(e -> {  
 contentArea.getChildren().clear();  
 contentArea.getChildren().add(createFlightForm());  
 });  
  
 deleteButton.setOnAction(e -> {  
 contentArea.getChildren().clear();  
 contentArea.getChildren().add(createDeleteForm());  
 });  
  
 viewButton.setOnAction(e -> {  
 contentArea.getChildren().clear();  
 contentArea.getChildren().add(flightTable); // Use the instance flightTable  
 });  
  
 // Main layout  
 VBox layout = new VBox(10, buttonBox, contentArea);  
 Scene scene = new Scene(layout, 800, 600);  
 primaryStage.setTitle("Admin Panel");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 private VBox createFlightForm() {  
 VBox formLayout = new VBox(10);  
 Label flightIDLabel = new Label("Flight ID:");  
 TextField flightIDField = new TextField();  
 Label fromLabel = new Label("From:");  
 ComboBox<String> fromComboBox = new ComboBox<>();  
 fromComboBox.getItems().addAll("Karachi", "Lahore", "Islamabad", "Peshawar");  
 Label toLabel = new Label("To:");  
 ComboBox<String> toComboBox = new ComboBox<>();  
 toComboBox.getItems().addAll("Karachi", "Lahore", "Islamabad", "Peshawar");  
 Label departureLabel = new Label("Departure Time:");  
 TextField departureField = new TextField();  
 Label arrivalLabel = new Label("Arrival Time:");  
 TextField arrivalField = new TextField();  
  
 Button addFlightButton = new Button("Add Flight");  
  
 // Add Flight action  
 addFlightButton.setOnAction(e -> {  
 try {  
 String flightID = flightIDField.getText();  
 String fromCity = fromComboBox.getValue();  
 String toCity = toComboBox.getValue();  
 String departureTime = departureField.getText();  
 String arrivalTime = arrivalField.getText();  
  
 Flight newFlight = new Flight(flightID, fromCity, toCity, departureTime, arrivalTime );  
 flightDatabase.addFlight(newFlight); // Add and save flight  
 // Refresh the table view  
 flightTable.setItems(FXCollections.*observableArrayList*(flightDatabase.getFlights()));  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*, "Flight added successfully!", ButtonType.*OK*);  
 alert.showAndWait();  
 } catch (Exception ex) {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*, "Invalid input. Please try again.", ButtonType.*OK*);  
 alert.showAndWait();  
 }  
 });  
  
 formLayout.getChildren().addAll(flightIDLabel, flightIDField, fromLabel, fromComboBox, toLabel, toComboBox,  
 departureLabel, departureField, arrivalLabel, arrivalField, addFlightButton);  
 return formLayout;  
 }  
  
 private VBox createDeleteForm() {  
 VBox deleteFormLayout = new VBox(10);  
 Label deleteFlightLabel = new Label("Enter Flight ID to delete:");  
 TextField deleteFlightIDField = new TextField();  
 Button deleteFlightButton = new Button("Delete Flight");  
  
 // Delete Flight action  
 deleteFlightButton.setOnAction(e -> {  
 String flightID = deleteFlightIDField.getText();  
 boolean success = flightDatabase.deleteFlight(flightID);  
 if (success) {  
 flightTable.setItems(FXCollections.*observableArrayList*(flightDatabase.getFlights())); // Refresh table  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("Success");  
 alert.setHeaderText("Flight Deleted");  
 alert.setContentText("Flight with ID " + flightID + " has been deleted.");  
 alert.showAndWait();  
 } else {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Error");  
 alert.setHeaderText("Flight Not Found");  
 alert.setContentText("No flight found with ID " + flightID + ".");  
 alert.showAndWait();  
 }  
 });  
  
 deleteFormLayout.getChildren().addAll(deleteFlightLabel, deleteFlightIDField, deleteFlightButton);  
 return deleteFormLayout;  
 }  
  
 private TableView<Flight> createFlightTable() {  
 TableView<Flight> table = new TableView<>();  
  
 // Define columns for the flight details  
 TableColumn<Flight, String> flightIDColumn = new TableColumn<>("Flight ID");  
 flightIDColumn.setCellValueFactory(cellData -> cellData.getValue().flightIDProperty());  
  
 TableColumn<Flight, String> fromColumn = new TableColumn<>("From");  
 fromColumn.setCellValueFactory(cellData -> cellData.getValue().fromCityProperty());  
  
 TableColumn<Flight, String> toColumn = new TableColumn<>("To");  
 toColumn.setCellValueFactory(cellData -> cellData.getValue().toCityProperty());  
  
 TableColumn<Flight, String> departureColumn = new TableColumn<>("Departure");  
 departureColumn.setCellValueFactory(cellData -> cellData.getValue().departureTimeProperty());  
  
 TableColumn<Flight, String> arrivalColumn = new TableColumn<>("Arrival");  
 arrivalColumn.setCellValueFactory(cellData -> cellData.getValue().arrivalTimeProperty());  
  
 // Add columns to the table  
 table.getColumns().addAll(flightIDColumn, fromColumn, toColumn, departureColumn, arrivalColumn);  
  
 return table;  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

**FlightDataabse.java**

package com.example.sp24bse126;  
  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
public class FlightDatabase {  
 private List<Flight> flights;  
  
 // Constructor  
 public FlightDatabase() {  
 flights = new ArrayList<>();  
 }  
  
 // Load flights from a file  
 public void loadFlights(String fileName) {  
 flights.clear();  
 try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {  
 String line;  
 while ((line = br.readLine()) != null) {  
 System.*out*.println("Loading: " + line); // Debugging  
 String[] flightData = line.split(",");  
 if (flightData.length == 5) {  
 String flightID = flightData[0].trim();  
 String fromCity = flightData[1].trim();  
 String toCity = flightData[2].trim();  
 String departureTime = flightData[3].trim();  
 String arrivalTime = flightData[4].trim();  
  
 Flight flight = new Flight(flightID, fromCity, toCity, departureTime, arrivalTime);  
 flights.add(flight);  
 } else {  
 System.*err*.println("Invalid line format: " + line);  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // Save flights to a file  
 public void saveFlights(String fileName) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName))) {  
 for (Flight flight : flights) {  
 writer.write(flight.getFlightID() + "," + flight.getFromCity() + "," + flight.getToCity() + ","  
 + flight.getDepartureTime() + "," + flight.getArrivalTime() );  
 writer.newLine();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // Add a flight  
 public void addFlight(Flight flight) {  
 flights.add(flight);  
 saveFlights(getClass().getResource("/flights.txt").getPath());  
 }  
  
 // Delete a flight by ID  
 public boolean deleteFlight(String flightID) {  
 boolean removed = flights.removeIf(flight -> flight.getFlightID().equals(flightID));  
 if (removed) {  
 saveFlights(getClass().getResource("/flights.txt").getPath()); // Save changes to the file  
 }  
 return removed;  
 }  
 // Get all flights  
 public List<Flight> getFlights() {  
 return flights;  
 }  
  
 // Search for flights based on "From" and "To" cities  
 public List<Flight> searchFlights(String fromCity, String toCity) {  
 List<Flight> availableFlights = new ArrayList<>();  
 for (Flight flight : flights) {  
 if (flight.getFromCity().equalsIgnoreCase(fromCity) && flight.getToCity().equalsIgnoreCase(toCity)) {  
 availableFlights.add(flight);  
 }  
 }  
 return availableFlights;  
 }  
}

**Flight class:**

package com.example.flightreservation;  
  
import java.time.LocalDate;  
import java.time.Duration;  
import java.util.List;  
  
public class Flight {  
 private String flightID;  
 private static int *counter*=0;  
 private String source;  
 private String destination;  
 private LocalDate departureTime;  
 private LocalDate arrivalTime;  
 private int totalSeats;  
 private int availableSeats;  
 private double price;  
 private String status;  
  
 public Flight() {  
 this.flightID= String.*format*("FL%03d",++*counter*);  
 }  
  
 public void displayFlightInfo() {  
 System.*out*.println("Flight ID: " + flightID);  
 System.*out*.println("Source: " + source + " -> Destination: " + destination);  
 System.*out*.println("Departure: " + departureTime + " | Arrival: " + arrivalTime);  
 System.*out*.println("Available Seats: " + availableSeats);  
 }  
  
 public String getFlightId() {  
 return flightID;  
 }  
  
 public void setFlightId(String flightId) {  
 this.flightID = flightId;  
 }  
  
 public String getSource() {  
 return source;  
 }  
  
 public void setSource(String source) {  
 this.source = source;  
 }  
  
 public String getDestination() {  
 return destination;  
 }  
  
 public void setDestination(String destination) {  
 this.destination = destination;  
 }  
  
 public int getTotalSeats() {  
 return totalSeats;  
 }  
  
 public void setTotalSeats(int totalSeats) {  
 this.totalSeats = totalSeats;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
 public String getStatus() {  
 return status;  
 }  
  
 public void setStatus(String status) {  
 this.status = status;  
 }  
  
 public LocalDate getDepartureTime() {  
 return departureTime;  
 }  
  
 public void setDepartureTime(LocalDate departureTime) {  
 this.departureTime = departureTime;  
 }  
  
 public LocalDate getArrivalTime() {  
 return arrivalTime;  
 }  
  
 public void setArrivalTime(LocalDate arrivalTime) {  
 this.arrivalTime = arrivalTime;  
 }  
  
 public int getAvailableSeats() {  
 return availableSeats;  
 }  
  
 public void setAvailableSeats(int availableSeats) {  
 this.availableSeats = availableSeats;  
 }  
  
 public Duration getFlightDuration() {  
 return Duration.*between*(departureTime, arrivalTime);  
 }  
}

**Flightmanager Class:**

package com.example.flightreservation;  
  
import java.io.\*;  
import java.time.LocalDate;  
import java.time.LocalDateTime;  
import java.util.ArrayList;  
import java.util.List;  
import java.time.format.DateTimeFormatter;  
  
  
  
public class FlightManager {  
 private List<Flight> flights;  
  
 public FlightManager() {  
 flights = new ArrayList<>();  
 }  
  
 public void loadFlightsFromFile(String filePath) {  
 File file = new File(filePath);  
 if (!file.exists()) return;  
  
 DateTimeFormatter dateFormatter = DateTimeFormatter.*ofPattern*("dd/MM/yyyy");  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 if (!line.isBlank()) {  
 String[] parts = line.split(",");  
 Flight flight = new Flight();  
 flight.setFlightId(parts[0]);  
 flight.setSource(parts[1]);  
 flight.setDestination(parts[2]);  
  
 flight.setDepartureTime(LocalDate.*parse*(parts[3], dateFormatter));  
 flight.setArrivalTime(LocalDate.*parse*(parts[4], dateFormatter));  
  
 flight.setTotalSeats(Integer.*parseInt*(parts[5]));  
 flight.setAvailableSeats(Integer.*parseInt*(parts[6]));  
 flight.setPrice(Double.*parseDouble*(parts[7]));  
 flight.setStatus(parts[8]);  
  
 flights.add(flight);  
 System.*out*.println("Loaded flight: " + flight.getFlightId());  
 }  
 }  
 } catch (IOException e) {  
 System.*out*.println("Error loading flights: " + e.getMessage());  
 }  
 }  
  
  
 public List<Flight> searchFlights(String source, String destination, LocalDate departureDate, LocalDate arrivalDate) {  
 DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("dd/MM/yyyy");  
  
 String formattedDepartureDate = departureDate.format(formatter);  
 String formattedArrivalDate = arrivalDate.format(formatter);  
  
 List<Flight> result = new ArrayList<>();  
  
 for (Flight flight : flights) {  
 String flightDepartureDate = flight.getDepartureTime().format(formatter);  
 String flightArrivalDate = flight.getArrivalTime().format(formatter);  
  
 if (flight.getSource().equalsIgnoreCase(source)  
 && flight.getDestination().equalsIgnoreCase(destination)  
 && flightDepartureDate.equals(formattedDepartureDate)  
 && flightArrivalDate.equals(formattedArrivalDate)) {  
 result.add(flight);  
 }  
 }  
  
 return result;  
 }  
}

**FlightDetailsGUI class:**

package com.example.flightreservation;  
  
import javafx.application.Application;  
import javafx.beans.property.SimpleStringProperty;  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.layout.VBox;  
import javafx.stage.Stage;  
  
import java.time.LocalDate;  
import java.time.format.DateTimeFormatter;  
import java.util.List;  
  
public class FlightDetailsGUI extends Application {  
 private static FlightManager flightManager;  
 private Flight selectedFlight;  
 private ObservableList<Flight> searchResults;  
  
 public FlightDetailsGUI() {  
 searchResults = FXCollections.observableArrayList();  
 }  
  
 public static void setFlightManager(FlightManager manager) {  
 flightManager = manager;  
 }  
  
 @Override  
 public void start(Stage primaryStage) {  
 Label sourceLabel = new Label("Source:");  
 TextField sourceField = new TextField();  
  
 Label destinationLabel = new Label("Destination:");  
 TextField destinationField = new TextField();  
  
 Label depdateLabel = new Label("Select Departure Date:");  
 DatePicker depDatePicker = new DatePicker();  
  
 Label arrdateLabel = new Label("Select Arrival Date:");  
 DatePicker arrDatePicker = new DatePicker();  
  
 Button searchButton = new Button("Search Flights");  
 TableView<Flight> flightTable = createFlightTable();  
  
 Button nextButton = new Button("Next");  
 nextButton.setDisable(true); // Initially disable the Next button  
  
 searchButton.setOnAction(e -> { //e actionevent object  
 String source = sourceField.getText();  
 String destination = destinationField.getText();  
 LocalDate departureDate = depDatePicker.getValue();  
 LocalDate arrivalDate = arrDatePicker.getValue();  
  
 System.out.println("Searching for flights ");  
  
 if (source.isBlank() || destination.isBlank() || departureDate == null || arrivalDate == null) {  
 showErrorMessage("Please fill all fields.");  
 } else {  
 List<Flight> results = flightManager.searchFlights(source, destination, departureDate, arrivalDate);  
 if (results.isEmpty()) {  
 showErrorMessage("No flights found for the given criteria.");  
 } else {  
 searchResults.setAll(results);  
 nextButton.setDisable(false); // Enable Next button when flights are found  
 }  
 }  
 });  
  
 nextButton.setOnAction(e -> {  
 if (selectedFlight != null) {  
 PassengerGUI passengerGUI = new PassengerGUI(selectedFlight);  
 passengerGUI.start(new Stage());  
 primaryStage.close();  
 } else {  
 showErrorMessage("Please select a flight.");  
 }  
 });  
  
  
 VBox layout = new VBox(10,  
 sourceLabel, sourceField,  
 destinationLabel, destinationField,  
 depdateLabel, depDatePicker,  
 arrdateLabel, arrDatePicker,  
 searchButton, flightTable, nextButton);  
  
 Scene scene = new Scene(layout, 800, 600); //window k pixel  
 primaryStage.setTitle("Flight Search");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
  
 private TableView<Flight> createFlightTable() {  
 TableView<Flight> table = new TableView<>();  
 table.setItems(searchResults);  
  
 TableColumn<Flight, String> flightIdColumn = new TableColumn<>("Flight ID"); //gets string data from flight objects...row: flight  
 flightIdColumn.setCellValueFactory(e -> new SimpleStringProperty(e.getValue().getFlightId())); //  
  
 TableColumn<Flight, String> sourceColumn = new TableColumn<>("Source");  
 sourceColumn.setCellValueFactory(e -> new SimpleStringProperty(e.getValue().getSource()));  
  
 TableColumn<Flight, String> destinationColumn = new TableColumn<>("Destination");  
 destinationColumn.setCellValueFactory(e -> new SimpleStringProperty(e.getValue().getDestination())); //row k andr flight ka object  
  
 TableColumn<Flight, String> departureColumn = new TableColumn<>("Departure");  
 departureColumn.setCellValueFactory(e -> new SimpleStringProperty(e.getValue().getDepartureTime().format(DateTimeFormatter.ofPattern("dd/MM/yyyy"))));  
  
 TableColumn<Flight, String> arrivalColumn = new TableColumn<>("Arrival");  
 arrivalColumn.setCellValueFactory(e -> new SimpleStringProperty(e.getValue().getArrivalTime().format(DateTimeFormatter.ofPattern("dd/MM/yyyy"))));  
  
 table.getColumns().addAll(flightIdColumn, sourceColumn, destinationColumn, departureColumn, arrivalColumn);  
  
 table.setRowFactory(e -> { //cutomise row behavior  
 TableRow<Flight> row = new TableRow<>();  
 row.setOnMouseClicked(event -> {  
 if (!row.isEmpty() && event.getClickCount() == 2) { // event listener: Double-click row to select flight  
 selectedFlight = row.getItem();  
 System.out.println("Selected flight: " + selectedFlight.getFlightId());  
 }  
 });  
 return row;  
 });  
  
 return table;  
 }  
  
  
 private void showErrorMessage(String message) {  
 Alert alert = new Alert(Alert.AlertType.ERROR);  
 alert.setTitle("Error");  
 alert.setContentText(message);  
 }  
 public static void main(String[] args) {  
 FlightManager manager = new FlightManager();  
 manager.loadFlightsFromFile("flight.txt");  
 setFlightManager(manager);  
 launch(args);  
 }  
}

**Passenger Class:**

package com.example.flightreservation;  
  
public class Passenger {  
 private String firstName;  
 private String email;  
 private String phoneNumber;  
 private String passportNumber;  
 private String nationality;  
 private String address;  
  
 public Passenger(String firstName, String email, String phoneNumber, String passportNumber, String nationality, String address) {  
 this.firstName = firstName;  
 this.email = email;  
 this.phoneNumber = phoneNumber;  
 this.passportNumber = passportNumber;  
 this.nationality = nationality;  
 this.address = address;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public String getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 public String getPassportNumber() {  
 return passportNumber;  
 }  
  
 public String getNationality() {  
 return nationality;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
  
 @Override  
 public String toString() {  
 return "Name: " + firstName + ", Email: " + email + ", Phone: " + phoneNumber +  
 ", Passport: " + passportNumber + ", Nationality: " + nationality + ", Address: " + address;  
 }  
}

**PassengerGUI class:**

package com.example.flightreservation;  
  
import javafx.application.Application;  
import javafx.collections.FXCollections;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.layout.GridPane;  
import javafx.stage.Stage;  
import java.util.ArrayList;  
import java.util.List;  
  
public class PassengerGUI extends Application {  
  
 private int numberOfPassengers;  
 private List<Passenger> passengers;  
 private Flight selectedFlight;  
 private GridPane grid;  
  
 public PassengerGUI(Flight selectedFlight) {  
 this.selectedFlight = selectedFlight;  
 this.passengers = new ArrayList<>();  
 }  
  
 @Override  
 public void start(Stage primaryStage) {  
 grid = new GridPane();  
 grid.setVgap(10);  
 grid.setHgap(10);  
  
 Label passengerCountLabel = new Label("Select number of passengers:");  
 ComboBox<Integer> passengerComboBox = new ComboBox<>();  
  
 passengerComboBox.setItems(FXCollections.*observableArrayList*(1, 2, 3, 4, 5));  
 passengerComboBox.setValue(1);  
  
 grid.add(passengerCountLabel, 0, 0); //row and colum index  
 grid.add(passengerComboBox, 1, 0);  
  
 Button showFormButton = new Button("Show Form");  
 showFormButton.setOnAction(e -> {  
 numberOfPassengers = passengerComboBox.getValue();  
 displayPassengerForm(primaryStage);  
 });  
  
 grid.add(showFormButton, 1, 1);  
  
 Scene scene = new Scene(grid, 600, 500);  
 primaryStage.setTitle("Passenger Details");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 private void displayPassengerForm(Stage primaryStage) {  
 grid.getChildren().clear();  
 List<TextField[]> fieldsList = new ArrayList<>(); //overall object  
  
 for (int i = 0; i < numberOfPassengers; i++) {  
 int colOffset = i \* 2; //controls column position side by side display hojaye  
  
 Label label = new Label("Details of person " + (i + 1));  
 grid.add(label, colOffset, 0); //row 0  
  
 TextField nameField = new TextField();  
 TextField emailField = new TextField();  
 TextField phoneField = new TextField();  
 TextField passportField = new TextField();  
 TextField nationalityField = new TextField();  
 TextField addressField = new TextField();  
  
 fieldsList.add(new TextField[]{nameField, emailField, phoneField, passportField, nationalityField, addressField});  
  
 // Add labels and input fields to the grid  
 grid.add(new Label("Name:"), colOffset, 1);  
 grid.add(nameField, colOffset + 1, 1);  
 grid.add(new Label("Email:"), colOffset, 2);  
 grid.add(emailField, colOffset + 1, 2);  
 grid.add(new Label("Phone:"), colOffset, 3);  
 grid.add(phoneField, colOffset + 1, 3);  
 grid.add(new Label("Passport Number:"), colOffset, 4);  
 grid.add(passportField, colOffset + 1, 4);  
 grid.add(new Label("Nationality:"), colOffset, 5);  
 grid.add(nationalityField, colOffset + 1, 5);  
 grid.add(new Label("Address:"), colOffset, 6);  
 grid.add(addressField, colOffset + 1, 6);  
 }  
  
 Button nextButton = new Button("Next");  
 nextButton.setOnAction(e -> { //event handler  
 if (areAllFieldsFilled(fieldsList)) {  
 passengers.clear();  
 for (TextField[] fields : fieldsList) {  
 Passenger passenger = new Passenger(  
 fields[0].getText(),  
 fields[1].getText(),  
 fields[2].getText(),  
 fields[3].getText(),  
 fields[4].getText(),  
 fields[5].getText()  
 );  
 passengers.add(passenger);  
 }  
  
 SeatMapGUI seatMapGUI = new SeatMapGUI(selectedFlight, passengers);  
 primaryStage.close();  
 seatMapGUI.start(new Stage());  
  
 } else {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*, "Please fill all fields before proceeding.", ButtonType.*OK*);  
 alert.showAndWait();  
 }  
 });  
  
 grid.add(nextButton, 1, numberOfPassengers \* 7 + 1); // Add the Next button below the form  
 }  
  
 private boolean areAllFieldsFilled(List<TextField[]> fieldsList) {  
 for (TextField[] fields : fieldsList) {  
 for (TextField field : fields) {  
 if (field.getText().isEmpty()) {  
 return false;  
 }  
 }  
 }  
 return true;  
 }  
  
}**Seat Class:**

package com.example.flightreservation;  
enum SeatClass {  
 BUSINESS,  
 PREMIUM\_ECONOMY,  
 ECONOMY  
}  
public class Seat {  
  
 private final String seatID;  
 private final SeatClass seatClass;  
 private boolean isBooked;  
 private boolean isSelected;  
 private int price;  
  
 public Seat(String seatID, SeatClass seatClass) {  
 this.seatID = seatID;  
 this.seatClass = seatClass;  
 this.isBooked = false;  
 this.isSelected = false;  
 }  
  
 public String getSeatID() {  
 return seatID;  
 }  
  
 public SeatClass getSeatClass() {  
 return seatClass;  
 }  
  
 public boolean isBooked() {  
 return isBooked;  
 }  
  
 public void setBooked(boolean booked) {  
 isBooked = booked;  
 }  
  
 public boolean isSelected() {  
 return isSelected;  
 }  
  
 public void setSelected(boolean selected) {  
 isSelected = selected;  
 }  
  
 public int getPrice() {  
 return price;  
 }  
  
 public void setPrice(int price) {  
 this.price = price;  
 }  
  
 @Override  
 public String toString() {  
 return "Seat{" +  
 "seatID='" + seatID + '\'' +  
 ", seatClass=" + seatClass +  
 ", price=" + price +  
 '}';  
 }  
}

**SeatSelectionHandler Class:**

package com.example.flightreservation;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class SeatSelectionHandler {  
 private List<Seat> selectedSeats = new ArrayList<>();  
  
 public void seatSelection(Seat seat) {  
 if (seat.isBooked()) {  
 System.out.println("Seat already booked!");  
 return;  
 }  
 if (seat.isSelected()) {  
 seat.setSelected(false);  
 selectedSeats.remove(seat);  
 } else {  
 seat.setSelected(true);  
 selectedSeats.add(seat);  
 }  
 }  
  
 public List<Seat> getSelectedSeats() {  
 return selectedSeats;  
 }  
  
}

**SeatMapGUI class:**

package com.example.flightreservation;  
  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.scene.Scene;  
import javafx.scene.control.Alert;  
import javafx.scene.control.Button;  
import javafx.scene.control.ButtonType;  
import javafx.scene.layout.GridPane;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.VBox;  
import javafx.scene.text.Text;  
import javafx.stage.Stage;  
  
import java.util.List;  
  
public class SeatMapGUI extends Application {  
 private static final int ROWS = 18;  
 private static final int COLS = 6;  
  
 private int maxSeats;  
 private final Seat[][] seats = new Seat[ROWS][COLS];  
 private SeatSelectionHandler seatSelectionHandler = new SeatSelectionHandler();  
 private Flight selectedFlight;  
 private List<Passenger> passengers;  
  
 private GridPane seatGrid;  
 private Text selectionStatus; //jo seat select krenge  
 private Text totalPrice;  
  
 public SeatMapGUI(Flight selectedFlight, List<Passenger> passengers) {  
 this.selectedFlight = selectedFlight;  
 this.passengers = passengers;  
 this.maxSeats = passengers.size();  
 }  
  
 @Override  
 public void start(Stage primaryStage) {  
 seatGrid = createSeatGrid();  
 VBox bookingSection = createBookingSection(primaryStage);  
 VBox seatInfoMap = seatMap();  
  
 HBox mainLayout = new HBox(20);  
 mainLayout.setPadding(new Insets(10));  
 mainLayout.getChildren().addAll(seatGrid, bookingSection, seatInfoMap);  
  
 Scene scene = new Scene(mainLayout, 900, 600);  
 primaryStage.setTitle("Seat Selection and Price Calculation");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 private GridPane createSeatGrid() {  
 GridPane grid = new GridPane();  
 grid.setPadding(new Insets(10, 10, 10, 10));  
 grid.setHgap(10); //10 pixel gap between columns  
 grid.setVgap(10);  
  
 for (int row = 0; row < ROWS; row++) {  
 for (int col = 0; col < COLS; col++) {  
 String seatID = (row + 1) + "" + (char) ('A' + col); //concatenation  
 SeatClass seatClass;  
  
 if (row < 3) {  
 seatClass = SeatClass.BUSINESS;  
 } else if (row < 8) {  
 seatClass = SeatClass.PREMIUM\_ECONOMY;  
 } else {  
 seatClass = SeatClass.ECONOMY;  
 }  
  
 Seat seat = new Seat(seatID, seatClass);  
 seats[row][col] = seat;  
  
 Button seatButton = createSeatButton(seat);  
 if (col == 2) {  
 GridPane.setMargin(seatButton, new Insets(0, 20, 0, 0));  
 }  
 grid.add(seatButton, col, row);  
 }  
 }  
  
 return grid;  
 }  
  
 private Button createSeatButton(Seat seat) {  
 Button seatButton = new Button(seat.getSeatID());  
 updateSeatButtonStyle(seat, seatButton);  
  
 seatButton.setOnAction(e -> {  
 if (seat.isBooked()) {  
 Alert alert = new Alert(Alert.AlertType.WARNING, "This seat is already booked.", ButtonType.OK);  
 alert.showAndWait();  
 } else {  
 if (!seat.isSelected() && seatSelectionHandler.getSelectedSeats().size() >= maxSeats) {  
 Alert alert = new Alert(Alert.AlertType.WARNING, "You can select only " + maxSeats + " seats.", ButtonType.OK);  
 alert.showAndWait();  
 } else {  
 seatSelectionHandler.seatSelection(seat);  
 updateSeatButtonStyle(seat, seatButton);  
 updatestatus();  
 }  
 }  
 });  
  
 return seatButton;  
 }  
  
 private VBox createBookingSection(Stage primaryStage) {  
 VBox bookingSection = new VBox(20);  
 bookingSection.setPadding(new Insets(20));  
  
 selectionStatus = new Text("Select your seats");  
 totalPrice = new Text("Total Price: $0.0");  
 Button bookButton = new Button("Complete Reservation");  
 bookButton.setOnAction(e -> {  
 if (!seatSelectionHandler.getSelectedSeats().isEmpty()) {  
 ReservationGUI reservationGUI = new ReservationGUI(selectedFlight, passengers, seatSelectionHandler.getSelectedSeats());  
 Stage reservationStage = new Stage();  
 reservationGUI.start(reservationStage);  
 primaryStage.close();  
  
 updateSeatButtons();  
 } else {  
 Alert alert = new Alert(Alert.AlertType.WARNING, "Please select at least one seat.", ButtonType.OK);  
 alert.showAndWait();  
 }  
 });  
  
 bookingSection.getChildren().addAll(selectionStatus, totalPrice, bookButton);  
  
 return bookingSection;  
 }  
 private void updatestatus() {  
 StringBuilder status= new StringBuilder("Selected Seats: ");  
 for (Seat selectedSeat : seatSelectionHandler.getSelectedSeats()) {  
 status.append(selectedSeat.getSeatID()).append(" ");  
 }  
 selectionStatus.setText(status.toString().trim());  
 totalPrice.setText("Total Price: $" + calculateTotalPrice());  
 }  
  
  
 private void updateSeatButtonStyle(Seat seat, Button seatButton) {  
 if (seat.isBooked()) {  
 seatButton.setStyle("-fx-background-color: gray; -fx-font-size: 14px;");  
 } else if (seat.isSelected()) {  
 seatButton.setStyle("-fx-background-color: pink; -fx-font-size: 14px;");  
 } else {  
 switch (seat.getSeatClass()) {  
 case BUSINESS:  
 seatButton.setStyle("-fx-background-color: gold; -fx-font-size: 14px;");  
 break;  
 case PREMIUM\_ECONOMY:  
 seatButton.setStyle("-fx-background-color: lightblue; -fx-font-size: 14px;");  
 break;  
 case ECONOMY:  
 seatButton.setStyle("-fx-background-color: lightgreen; -fx-font-size: 14px;");  
 break;  
 }  
 }  
 }  
  
 private void updateSeatButtons() {  
 seatGrid.getChildren().forEach(e -> {  
 if (e instanceof Button seatButton) {  
 String seatId = seatButton.getText();  
 Seat seat = findSeatById(seatId);  
 if (seat != null) {  
 updateSeatButtonStyle(seat, seatButton);  
 }  
 }  
 });  
 }  
  
 private Seat findSeatById(String seatId) {  
 for (int row = 0; row < ROWS; row++) {  
 for (int col = 0; col < COLS; col++) {  
 if (seats[row][col].getSeatID().equals(seatId)) {  
 return seats[row][col];  
 }  
 }  
 }  
 return null;  
 }  
  
 private double calculateTotalPrice() {  
 double businessPrice = 1000.0;  
 double premiumEconomyPrice = 500.0;  
 double economyPrice = 350.0;  
 double total = 0.0;  
  
 for (Seat seat : seatSelectionHandler.getSelectedSeats()) {  
 switch (seat.getSeatClass()) {  
 case BUSINESS:  
 total += businessPrice;  
 break;  
 case PREMIUM\_ECONOMY:  
 total += premiumEconomyPrice;  
 break;  
 case ECONOMY:  
 total += economyPrice;  
 break;  
 }  
 }  
  
 return total;  
 }  
  
 private VBox seatMap() {  
 VBox infoMap = new VBox(20);  
 infoMap.setPadding(new Insets(20));  
  
 Text infoTitle = new Text("Seat Class Information");  
 infoMap.getChildren().add(infoTitle);  
  
 infoMap.getChildren().add(createInfoBox("Business Class", "gold"));  
 infoMap.getChildren().add(createInfoBox("Premium Economy Class", "lightblue"));  
 infoMap.getChildren().add(createInfoBox("Economy Class", "lightgreen"));  
  
 return infoMap;  
 }  
  
 private HBox createInfoBox(String label, String color) {  
 HBox infoBox = new HBox(10);  
 infoBox.setPadding(new Insets(5));  
  
 Button colorBox = new Button();  
 colorBox.setStyle("-fx-background-color: " + color + "; -fx-min-width: 30px; -fx-min-height: 30px;");  
  
 Text classLabel = new Text(label);  
 infoBox.getChildren().addAll(colorBox, classLabel);  
  
 return infoBox;  
 }  
  
 public static void main(String[] args) {  
 launch(args);  
 }  
}

**Reservation class:**

package com.example.flightreservation;  
  
import java.util.List;  
  
public class Reservation {  
 private static int *counter* = 1;  
 private int reservationNumber;  
 private Flight flight;  
 private List<Seat> seats;  
 private List<Passenger> passengers; // The selected flight  
  
 // Constructor for creating a reservation  
 public Reservation(Flight flight, List<Seat> seats, List<Passenger> passengers) {  
 this.reservationNumber = *counter*++;  
 this.flight = flight;  
 this.passengers = passengers;  
 this.seats = seats;  
 }  
  
 // Getters for the details  
 public int getReservationNumber() {  
 return reservationNumber;  
 }  
  
 public List<Passenger> getPassengers() {  
 return passengers;  
 }  
  
 public List<Seat> getSeats() {  
 return seats;  
 }  
  
 public Flight getFlight() {  
 return flight;  
 }  
  
 // Optionally, you can implement toString() to easily print reservation details  
 @Override  
 public String toString() {  
 return "Reservation Number: " + reservationNumber + "\n" +  
 "Passengers: " + passengers + "\n" +  
 "Seats: " + seats + "\n" +  
 "Flight: " + flight;  
 }  
}

**ReservationManager class:**

package com.example.flightreservation;  
  
import java.io.BufferedWriter;  
import java.io.File;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.List;  
import java.util.ArrayList;  
  
public class ReservationManager {  
 private List<Reservation> reservations;  
  
 public ReservationManager() {  
 this.reservations = new ArrayList<>();  
 }  
  
 public void makeReservation(Flight flight, List<Seat> seats, List<Passenger> passengers) {  
  
 Reservation reservation = new Reservation(flight, seats, passengers);  
 reservations.add(reservation);  
 System.out.println("Reservation successful");  
 reservationdetails(flight, seats, passengers);  
 }  
  
 public void reservationdetails(Flight flight, List<Seat> selectedSeats, List<Passenger> passengers) {  
  
 StringBuilder details = new StringBuilder();  
  
 details.append("Flight ID: ").append(flight.getFlightId()).append("\n")  
 .append("Source: ").append(flight.getSource()).append("\n")  
 .append("Destination: ").append(flight.getDestination()).append("\n")  
 .append("Departure: ").append(flight.getDepartureTime()).append("\n")  
 .append("Arrival: ").append(flight.getArrivalTime()).append("\n\n");  
  
 details.append("Passengers:\n");  
 for (Passenger passenger : passengers) {  
 details.append(passenger.getFirstName()).append("\n");  
 }  
  
 details.append("\nSelected Seats:\n");  
 for (Seat seat : selectedSeats) {  
 details.append("Seat ID: ").append(seat.getSeatID()).append(" - ")  
 .append(seat.getSeatClass()).append("\n");  
 }  
 details.append("\n--- End of Reservation ---\n\n");  
 writeToFile(details.toString());  
 }  
  
 private void writeToFile(String details) {  
 File file = new File("AllReservations.txt");  
  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(file, true))) {  
 writer.write(details);  
 } catch (IOException e) {  
 System.err.println("Error writing to file: " + e.getMessage());  
 }  
 }  
  
 public List<Reservation> getReservations() {  
 return reservations;  
 }  
}

**ReservationGUI class:**

package com.example.flightreservation;  
  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.scene.Scene;  
import javafx.scene.control.Alert;  
import javafx.scene.control.Button;  
import javafx.scene.layout.VBox;  
import javafx.stage.Stage;  
  
import java.util.List;  
  
public class ReservationGUI extends Application {  
 private Flight flight;  
 private List<Passenger> passengers;  
 private List<Seat> selectedSeats;  
  
 public ReservationGUI(Flight flight, List<Passenger> passengers, List<Seat> selectedSeats) {  
 this.flight = flight;  
 this.passengers = passengers;  
 this.selectedSeats = selectedSeats;  
 }  
  
 @Override  
 public void start(Stage primaryStage) {  
 VBox layout = new VBox(10);  
 layout.setPadding(new Insets(20));  
  
 String flightDetails = "Flight ID: " + flight.getFlightId() +  
 "\nSource: " + flight.getSource() +  
 "\nDestination: " + flight.getDestination() +  
 "\nDeparture: " + flight.getDepartureTime() +  
 "\nArrival: " + flight.getArrivalTime();  
  
 layout.getChildren().add(new javafx.scene.text.Text("Flight Details:\n" + flightDetails));  
  
 StringBuilder passengerDetails = new StringBuilder("Passengers:\n");  
 for (Passenger passenger : passengers) {  
 passengerDetails.append(passenger.getFirstName()).append("\n");  
 }  
 layout.getChildren().add(new javafx.scene.text.Text(passengerDetails.toString()));  
  
 StringBuilder seatDetails = new StringBuilder("Selected Seats:\n");  
 for (Seat seat : selectedSeats) {  
 seatDetails.append(seat.getSeatID()).append(" - ").append(seat.getSeatClass()).append("\n");  
 }  
 layout.getChildren().add(new javafx.scene.text.Text(seatDetails.toString()));  
  
 Button completeReservationButton = new Button("Complete Reservation");  
 completeReservationButton.setOnAction(e -> makeReservation());  
  
 layout.getChildren().add(completeReservationButton);  
  
 Scene scene = new Scene(layout, 400, 400);  
 primaryStage.setTitle("Reservation Details");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 private void makeReservation() {  
 try {  
 ReservationManager reservationManager = new ReservationManager();  
 reservationManager.makeReservation(flight, selectedSeats, passengers);  
 showConfirmationAlert();  
 } catch (IllegalArgumentException e) {  
 showErrorMessage("Failed to make reservation: " + e.getMessage());  
 }  
 }  
  
 private void showConfirmationAlert() {  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("Reservation Success");  
 alert.setHeaderText(null);  
 alert.setContentText("Reservation has been made successfully.");  
 alert.showAndWait();  
 }  
  
 private void showErrorMessage(String message) {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Error");  
 alert.setHeaderText(null);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}