PIA Airlines Reservation System in C++

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include <vector>

using namespace std;

class Person {

protected:

string name;

string CNIC;

string passportNumber;

string contactNumber;

public:

Person(string n, string cnic, string passport, string contact):name(n),CNIC(cnic),passportNumber(passport),contactNumber(contact) {

}

virtual void displayInfo() const {

cout << "Name: " << name << endl;

cout << "CNIC: " << CNIC << endl;

cout << "Passport Number: " << passportNumber << endl;

cout << "Contact Number: " << contactNumber << endl;

}

};

template <class T>

class SeatManager {

private:

T seats[20] = { 0 };

public:

void displaySeatAvailability() const {

cout << "Seat Availability:"<<endl;

for (int i = 0; i < 20; ++i) {

cout << "Seat " << i + 1 << ": " << (seats[i] ? "Booked" : "Available") << endl;

}

}

bool bookSeat(int seatNum) {

if (seatNum < 1 || seatNum > 20 || seats[seatNum - 1]) {

return false;

}

seats[seatNum - 1] = 1;

return true;

}

};

class Flight {

protected:

string flightType;

string flightName;

double costPerPassenger;

SeatManager<int> seatManager;

public:

Flight(string type, string name, double cost):flightType(type),flightName(name),costPerPassenger(cost) {

}

virtual void displayFlightInfo() const = 0;

string getFlightType() const {

return flightType;

}

string getFlightName() const {

return flightName;

}

double getCostPerPassenger() const {

return costPerPassenger;

}

friend void displayFullFlightInfo(const Flight& flight);

};

void displayFullFlightInfo(const Flight& flight) {

cout << "Flight Name: " << flight.flightName << endl;

cout << "Flight Type: " << flight.flightType << endl;

cout << "Cost Per Passenger: Rs. " << flight.costPerPassenger << endl;

flight.seatManager.displaySeatAvailability();

}

class DomesticFlight : public Flight {

public:

DomesticFlight(string name):Flight("Domestic", name, 20000.0) {

}

void displayFlightInfo() const override {

cout << "Domestic Flight: " << flightName << ", Cost: Rs. " << costPerPassenger << endl;

}

};

class InternationalFlight : public Flight {

public:

InternationalFlight(string name):Flight("International", name, 50000.0) {

}

void displayFlightInfo() const override {

cout << "International Flight: " << flightName << ", Cost: Rs. " << costPerPassenger << endl;

}

};

struct FlightNode {

Flight\* flight;

FlightNode\* next;

FlightNode(Flight\* f) : flight(f), next(nullptr) {}

};

class FlightList {

private:

FlightNode\* head;

public:

FlightList() : head(nullptr) {}

void addFlight(Flight\* flight) {

FlightNode\* newNode = new FlightNode(flight);

if (!head) {

head = newNode;

}

else {

FlightNode\* temp = head;

while (temp->next) {

temp = temp->next;

}

temp->next = newNode;

}

}

void removeLastFlight() {

if (!head) {

throw runtime\_error("No flights to cancel.");

}

if (!head->next) {

delete head;

head = nullptr;

}

else {

FlightNode\* temp = head;

while (temp->next->next) {

temp = temp->next;

}

delete temp->next;

temp->next = nullptr;

}

}

bool isEmpty() const {

return head == nullptr;

}

void displayFlights() const {

if (isEmpty()) {

cout << "No flights booked." << endl;

return;

}

FlightNode\* temp = head;

while (temp) {

cout << " - " << temp->flight->getFlightName() << " (" << temp->flight->getFlightType() << ")\n";

temp = temp->next;

}

}

void generateReceipt(ofstream& file) const {

if (isEmpty()) {

throw runtime\_error("No flights booked. Cannot generate receipt.");

}

FlightNode\* temp = head;

while (temp) {

file << "Flight: " << temp->flight->getFlightName()

<< ", Type: " << temp->flight->getFlightType()

<< ", Cost Per Passenger: Rs. " << temp->flight->getCostPerPassenger() << endl;

temp = temp->next;

}

}

~FlightList() {

while (head) {

FlightNode\* temp = head;

head = head->next;

delete temp;

}

}

};

class Passenger : public Person {

private:

FlightList bookedFlights;

int totalPassengers;

public:

Passenger(string n, string cnic, string passport, string contact):Person(n,cnic,passport,contact),totalPassengers(0) {

}

void bookFlight(Flight& flight, int passengers) {

if (passengers <= 0) {

throw runtime\_error("Invalid number of passengers.");

}

bookedFlights.addFlight(&flight);

totalPassengers += passengers;

double totalCost = passengers \* flight.getCostPerPassenger();

cout << "Flight " << flight.getFlightName() << " booked successfully for " << passengers << " passengers. Total Cost: Rs. " << totalCost << endl;

}

void cancelFlight() {

bookedFlights.removeLastFlight();

cout << "Last booked flight has been canceled.\n";

}

void generateReceipt() const {

ofstream receiptFile("receipt.txt");

receiptFile << "================ PIA RECEIPT ================\n";

receiptFile << "Name: " << name << endl;

receiptFile << "CNIC: " << CNIC << endl;

receiptFile << "Passport Number: " << passportNumber << endl;

receiptFile << "Contact Number: " << contactNumber << endl;

bookedFlights.generateReceipt(receiptFile);

receiptFile << "=============================================\n";

receiptFile.close();

cout << "Receipt generated: 'receipt.txt'.\n";

}

void displayInfo() const override {

Person::displayInfo();

cout << "Booked Flights:\n";

bookedFlights.displayFlights();

}

};

void displayFlights(vector<Flight\*>& flights) {

cout << "Available Flights:\n";

for (size\_t i = 0; i < flights.size(); ++i) {

cout << i + 1 << ". ";

flights[i]->displayFlightInfo();

}

}

void displayMenu() {

cout << "\n--- PIA Airlines Reservation System ---\n";

cout << "1. Book a Flight\n";

cout << "2. Cancel a Flight\n";

cout << "3. Generate Receipt\n";

cout << "4. Display Passenger Info\n";

cout << "5. Exit\n";

cout << "Enter your choice: ";

}

int main() {

cout << "WELCOME TO PAKISTAN INTERNATIONAL AIRLINE\n";

string name, cnic, passport, contact;

cout << "Enter Passenger Name: ";

getline(cin, name);

cout << "Enter CNIC: ";

getline(cin, cnic);

cout << "Enter Passport Number: ";

getline(cin, passport);

cout << "Enter Contact Number: ";

getline(cin, contact);

Passenger passenger(name, cnic, passport, contact);

vector<Flight\*> flights = {

new DomesticFlight("KHI to ISL"),

new DomesticFlight("ISL to LHR"),

new InternationalFlight("KHI to Dubai"),

new InternationalFlight("LHR to New York")

};

int choice;

do {

displayMenu();

cin >> choice;

cin.ignore();

try {

switch (choice) {

case 1: {

displayFlights(flights);

int flightIndex, passengers;

cout << "Enter flight number: ";

cin >> flightIndex;

cout << "Enter number of passengers: ";

cin >> passengers;

passenger.bookFlight(\*flights[flightIndex - 1], passengers);

break;

}

case 2:

passenger.cancelFlight();

break;

case 3:

passenger.generateReceipt();

break;

case 4:

passenger.displayInfo();

break;

case 5:

cout << "Exiting system. Thank you!\n";

break;

default:

cout << "Invalid choice. Try again.\n";

}

}

catch (const exception& e) {

cout << "Error: " << e.what() << endl;

}

} while (choice != 5);

for (auto flight : flights) {

delete flight;

}

return 0;

}