#include <iostream>

#include <string>

#include <iomanip>

#include <stdexcept>

using namespace std;

class Passenger; // Forward declaration

class Seat

{

private:

int seatNo;

bool isbooked;

Passenger\* passenger; // A pointer to the passenger who booked this seat

public:

Seat() : seatNo(0), isbooked(false), passenger(nullptr) {}

Seat(int sno) : seatNo(sno), isbooked(false), passenger(nullptr) {}

bool isSeatBooked() {

return isbooked;

}

void bookSeat(Passenger\* p) {

isbooked = true;

passenger = p;

}

Passenger\* getPassenger() {

return passenger;

}

int getSeatNumber() {

return seatNo;

}

void cancelSeatBooking() {

if (isbooked) {

isbooked = false;

passenger = nullptr;

cout << "Seat booking canceled successfully." << endl;

}

else {

cout << "Seat is not booked, so cannot cancel." << endl;

}

}

};

class schedule

{

public:

string from;

string to;

string timing;

schedule() {}

schedule(string From, string To, string Time)

{

from = From;

to = To;

timing = Time;

}

};

class Train;

class Passenger

{

private:

string name;

string phoneno;

string gender;

int age;

int ticketNumber;

public:

Passenger() {}

Passenger(string n, string ph, string g, int Age)

{

name = n;

phoneno = ph;

gender = g;

age = Age;

}

void setTicketNumber(int ticketNo) {

ticketNumber = ticketNo;

}

int getTicketNumber() const {

return ticketNumber;

}

string getName() const { return name; }

string getGender() const { return gender; }

string getContact() const { return phoneno; }

int getAge() const { return age; }

void bookSeat(Train\* train, Seat\* seat); // Declaration, defined later

};

class Train

{

private:

int TrainNo;

string type;

int no\_of\_seats;

float ticketprice;

Seat\* seats[14]; // Array of seats in the Train

public:

schedule TrainSchedule;

Train() {}

Train(int bno, string Type, int nseats, float price, schedule s) : TrainSchedule(s)

{

TrainNo = bno;

type = Type;

no\_of\_seats = nseats;

ticketprice = price;

for (int i = 0; i < 14; ++i) {

seats[i] = new Seat(i + 1);

}

}

~Train() {

for (int i = 0; i < 14; ++i) {

delete seats[i];

}

}

int getTrainNumber()

{

return TrainNo;

}

string getFrom()

{

return TrainSchedule.from;

}

string getTo()

{

return TrainSchedule.to;

}

string getTime()

{

return TrainSchedule.timing;

}

float getTicketPrice()

{

return ticketprice;

}

string getType()

{

return type;

}

Seat\* getSeat(int seatNumber) {

if (seatNumber >= 1 && seatNumber <= 14) {

return seats[seatNumber - 1];

}

return nullptr;

}

int getAvailableSeats() {

int count = 0;

for (int i = 0; i < 14; i++) {

if (!seats[i]->isSeatBooked()) {

++count;

}

}

return count;

}

void viewPassengerList() {

cout << "Passenger List for Train " << TrainNo << ":\n";

for (int i = 0; i < 14; i++) {

Seat\* seat = seats[i];

if (seat->isSeatBooked()) {

cout << "Seat Number: " << seat->getSeatNumber() << ", Booked by Passenger " << seat->getPassenger()->getName() << endl;

}

}

cout << endl;

}

};

// Definition of the Passenger::bookSeat() method (defined after Train class)

void Passenger::bookSeat(Train\* train, Seat\* seat) {

seat->bookSeat(this);

cout << "Seat booked successfully for Passenger: " << name << endl;

}

class ticket

{

private:

int ticketNo;

Train\* train;

Seat\* seat;

public:

ticket() {}

ticket(int ticketNO, Train\* train, Seat\* Seat)

: ticketNo(ticketNO), train(train), seat(Seat) {}

int getTicketNumber()

{

return ticketNo;

}

Train\* getTrain()

{

return train;

}

Seat\* getSeat()

{

return seat;

}

};

class payment {

public:

payment() {}

virtual void processPayment(double amount) = 0;

virtual ~payment() {}

};

class UPIPayment : public payment {

private:

string UPId;

public:

UPIPayment() {

UPId = "";

}

void processPayment(double amount) {

cout << "Enter UPI ID: ";

cin >> UPId;

cout << "Processing UPI Id " << UPId << " with payment of Rs." << amount << endl;

}

};

class CardPayment : public payment {

private:

int cardNo;

public:

CardPayment() {

cardNo = 0;

}

void processPayment(double amount) {

cout << "Enter Card Number: ";

cin >> cardNo;

cout << "Processing Card payment of Rs." << amount << endl;

}

};

class TrainSystem {

Train\* Traines[10];

int TrainCount;

int ticketCounter;

TrainSystem() : TrainCount(0), ticketCounter(1) {}

public:

static TrainSystem& getInstance() {

static TrainSystem instance;

return instance;

}

void addTrain(Train\* train) {

if (TrainCount < 10) {

Traines[TrainCount++] = train;

}

}

Train\* getTrain(int TrainNumber) {

if (TrainNumber < 1 || TrainNumber > TrainCount) {

return nullptr;

}

return Traines[TrainNumber - 1];

}

bool findTrainesByRoute(const string& from, const string& to) {

bool found = false;

cout << "---------------------------------------------------------------------------------------" << endl;

cout << "Train Number | From | To | Time | Available Seats | Type | Ticket Price" << endl;

cout << "---------------------------------------------------------------------------------------" << endl;

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

Train\* train = Traines[i];

if (train->getFrom() == from && train->getTo() == to) {

found = true;

cout << setw(10) << train->getTrainNumber() << " | ";

cout << setw(12) << train->getFrom() << " | ";

cout << setw(12) << train->getTo() << " | ";

cout << setw(10) << train->getTime() << " | ";

cout << setw(15) << train->getAvailableSeats() << " | ";

cout << setw(10) << train->getType() << " | ";

cout << setw(25) << train->getTicketPrice() << endl;

}

}

cout << "---------------------------------------------------------------------------------------" << endl;

if (!found) {

cout << "No Trains found for the specified route." << endl;

return false;

}

return true;

}

int findTrainNumber(int TrainNumber) {

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

if (Traines[i]->getTrainNumber() == TrainNumber) {

return i;

}

}

return -1; // Train number not found

}

void printTicketDetails(ticket\* t) {

if (t) {

cout << "===========================" << endl;

cout << "Ticket Number: " << t->getTicketNumber() << endl;

cout << "Train Number: " << t->getTrain()->getTrainNumber() << endl;

cout << "From: " << t->getTrain()->getFrom() << ", To: " << t->getTrain()->getTo() << endl;

cout << "Time: " << t->getTrain()->getTime() << endl;

cout << "Seat Number: " << t->getSeat()->getSeatNumber() << endl;

cout << "Passenger Name: " << t->getSeat()->getPassenger()->getName() << endl;

cout << "Contact No.: " << t->getSeat()->getPassenger()->getContact() << endl;

cout << "Age.: " << t->getSeat()->getPassenger()->getAge() << endl;

cout << "Gender.: " << t->getSeat()->getPassenger()->getGender() << endl;

cout << "===========================" << endl;

}

else {

cout << "Ticket not found!" << endl;

}

}

ticket\* bookSeatForTicket(int TrainNumber, int seatNumber, const string& name, const string& contact, const string& gender, int age)

{

int TrainIndex = findTrainNumber(TrainNumber);

if (TrainIndex == -1) {

throw invalid\_argument("Invalid Train number!");

}

if (seatNumber < 1 || seatNumber > 14) {

throw invalid\_argument("Invalid seat number! Seat number should be between 1 and 14.");

}

Train\* train = Traines[TrainIndex];

Seat\* seat = train->getSeat(seatNumber);

if (!seat->isSeatBooked()) {

Passenger\* passenger = new Passenger(name, contact, gender, age);

// Assign ticket number (for demonstration purposes, you can use any method to generate the ticket number)

int ticketNumber = ticketCounter++;

passenger->setTicketNumber(ticketNumber);

seat->bookSeat(passenger);

// Select the payment mode

int paymentChoice;

do {

cout << "Select Payment Mode:\n";

cout << "1. UPI Payment\n";

cout << "2. Card Payment\n";

cin >> paymentChoice;

} while (paymentChoice != 1 && paymentChoice != 2);

payment\* paymentMode;

if (paymentChoice == 1) {

paymentMode = new UPIPayment();

}

else {

paymentMode = new CardPayment();

}

// Process payment

double ticketPrice = train->getTicketPrice();

paymentMode->processPayment(ticketPrice);

delete paymentMode; // Free the allocated memory

//passenger.bookSeat(Train, seat);

cout << "Seat booked successfully for Passenger: " << name << endl;

cout << endl;

return new ticket(ticketNumber, train, seat);

}

else {

cout << "Ticket not found." << endl;

return nullptr;

}

}

void cancelBookingByTicketNumber(int ticketNumber) {

char cancelChoice;

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

Train\* train = Traines[i];

for (int j = 0; j < 14; ++j) {

Seat\* seat = train->getSeat(j + 1);

Passenger\* passenger = seat->getPassenger();

if (passenger && passenger->getTicketNumber() == ticketNumber) {

cout << "Do you want to cancel the booking? (Y/N): ";

cin >> cancelChoice;

if (cancelChoice == 'Y' || cancelChoice == 'y') {

seat->cancelSeatBooking();

cout << "Ticket " << ticketNumber << " canceled successfully for Passenger: " << passenger->getName() << " and 50% refund initiated " << endl;

}

else {

cout << "Booking for Ticket " << ticketNumber << " is not canceled.\n";

}

return;

}

}

}

throw invalid\_argument("Ticket not found or already canceled.");

}

void viewTrainDetails()

{

cout << "===============================================================" << endl;

cout << "Train Details and Availability" << endl;

cout << "===========================" << endl;

cout << left << setw(10) << "Train No" << setw(15) << "From" << setw(15) << "To" << setw(11) << "Available Seats" << setw(30) << " Timing " << endl;

cout << "-----------------------------------------------------------" << endl;

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

Train\* train = Traines[i];

cout << left << setw(10) << train->getTrainNumber() << setw(15) << train->getFrom() << setw(15) << train->getTo() << setw(15) << train->getAvailableSeats()

<< setw(30) << train->getTime() << endl;

}

cout << "================================================================" << endl;

}

void selectSeat(int TrainNumber) {

int TrainIndex = findTrainNumber(TrainNumber);

if (TrainIndex == -1) {

throw invalid\_argument("Invalid Train number!");

return;

}

Train\* train = Traines[TrainIndex];

cout << "Train Number: " << train->getTrainNumber() << endl;

cout << "Available Seats: " << train->getAvailableSeats() << " out of 14" << endl;

cout << "Seats: ";

for (int seatNumber = 1; seatNumber <= 14; ++seatNumber) {

Seat\* seat = train->getSeat(seatNumber);

char availability = seat->isSeatBooked() ? 'B' : 'N';

cout << availability << " ";

if (seatNumber % 2 == 0) {

cout << endl;

cout << " ";

}

}

cout << endl << endl;

}

ticket\* getTicketByNumber(int ticketNumber) {

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

Train\* train = Traines[i];

for (int j = 0; j < 14; ++j) {

Seat\* seat = train->getSeat(j + 1);

if (seat->isSeatBooked() && seat->getPassenger()->getTicketNumber() == ticketNumber) {

return new ticket(ticketNumber, train, seat);

}

}

}

throw invalid\_argument("Ticket not found!");

}

int viewPassengerListByTrain(int TrainNumber)

{

int TrainIndex = findTrainNumber(TrainNumber);

if (TrainIndex == -1) {

cout << "Invalid Train number!" << endl;

return -1;

}

Train\* train = Traines[TrainIndex];

cout << "Passenger List for Train " << train->getTrainNumber() << ":\n";

for (int i = 0; i < 14; i++) {

Seat\* seat = train->getSeat(i + 1);

if (seat->isSeatBooked()) {

Passenger\* passenger = seat->getPassenger();

cout << "Ticket Number: " << passenger->getContact() << endl;

cout << "Passenger Name: " << passenger->getName() << endl;

cout << "Seat Number: " << seat->getSeatNumber() << endl;

cout << "-----------------------------" << endl;

}

}

return 0;

}

};

int main()

{

TrainSystem& trainSystem = TrainSystem::getInstance();

int TrainNumber, seatNumber, age, flag;

string name, contact, gender, from, to;

//Train\* Train;

ticket\* t;

schedule s1("Mumbai", "Pune", "10:15");

schedule s2("Mumbai", "Bangalore", "17:30");

schedule s3("Mumbai", "Bangalore", "19:30");

schedule s4("Mumbai", "Pune", "19:30");

schedule s5("Bangalore", "Pune", "20:10");

schedule s6("Pune", "Bangalore", "17:30");

schedule s7("Pune", "Bangalore", "21:14");

schedule s8("Bangalore", "Pune", "16:30");

schedule s9("Bangalore", "Mumbai", "14:50");

schedule s10("Bangalore", "Mumbai", "13:30");

Train trainsToAdd[] = {

Train(5066, "A/C sleeper", 14, 1800, s1),

Train(3755, "A/C seater", 14, 1600, s2),

Train(1001, "A/C sleeper", 14, 2000, s3),

Train(2344, "Non-A/C sleeper", 14, 1700, s4),

Train(6767, "A/C sleeper", 14, 1800, s5),

Train(9090, "A/C seater", 14, 1400, s6),

Train(1991, "A/C semi-sleeper", 14, 1600, s7),

Train(7776, "Non-A/C sleeper", 14, 1200, s8),

Train(3066, "A/C semi-sleeper", 14, 1400, s9),

Train(6960, "Non A/C sleeper", 14, 1700, s10),

};

int numTrainsToAdd = sizeof(trainsToAdd) / sizeof(trainsToAdd[0]);

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

trainSystem.addTrain(&trainsToAdd[i]);

}

cout << "Welcome to VRL Train booking\n" << endl;

int choice;

try {

do

{

cout << "\nChoose\n";

cout << "1-See availability of all the trains in the app\n";

cout << "2-View availability of seats for a specific train\n";

cout << "3-Book a seat\n";

cout << "4-View ticket status(Confirmation)\n";

cout << "5-Cancel booking(ticket)\n";

cout << "6-View train details along with the list of passengers\n";

cout << "7-Exit from app\n";

cin >> choice;

switch (choice)

{

case 1: // See availability of trains

trainSystem.viewTrainDetails();

break;

case 2:

cout << "Enter from city\n";

cin >> from;

cout << "Enter to city\n";

cin >> to;

cout << "Trains available for this route are\n";

trainSystem.findTrainesByRoute(from, to);

break;

case 3: // Book a seat

cout << "Enter from city\n";

cin >> from;

cout << "Enter to city\n";

cin >> to;

cout << "Trains available for this route are\n";

if (trainSystem.findTrainesByRoute(from, to))

{

cout << "Enter train number: ";

cin >> TrainNumber;

trainSystem.selectSeat(TrainNumber);

cout << "Enter seat number: ";

cin >> seatNumber;

cout << "Enter passenger name: ";

cin.ignore();

getline(cin, name);

cout << "Enter contact number: ";

cin >> contact;

cout << "Enter gender: ";

cin >> gender;

cout << "Enter age: ";

cin >> age;

t = trainSystem.bookSeatForTicket(TrainNumber, seatNumber, name, contact, gender, age);

cout << "Ticket Details:\n";

trainSystem.printTicketDetails(t);

}

break;

case 4:

int ticketNumber;

cout << "Enter ticket number: ";

cin >> ticketNumber;

t = trainSystem.getTicketByNumber(ticketNumber);

trainSystem.printTicketDetails(t);

break;

case 5:

cout << "Enter the ticket number: ";

cin >> ticketNumber;

trainSystem.cancelBookingByTicketNumber(ticketNumber);

break;

case 6: // View train details along with the list of passengers

cout << "Enter train number: ";

cin >> TrainNumber;

flag = trainSystem.viewPassengerListByTrain(TrainNumber);

if (flag == 1)

{

cout << "Train not found!" << endl;

}

break;

case 7: // Exit from the app

cout << "Exiting the app. Thank you!\n";

break;

default:

cout << "Invalid choice. Please try again.\n";

}

} while (choice != 7);

}

catch (const invalid\_argument& e)

{

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

}

catch (const exception& e) {

cout << "An error occurred: " << e.what() << endl;

}

return 0;

}