**PROGRAMS :**

**1. Mini Project: Movie Ticket Booking**

Input:

#include <iostream>

#include <iomanip>

#include <cstring>

#include<fstream>

#include<string>

using namespace std;

const int maxSeats = 16;

double total = 0.0; // Initialize the total cost to 0.0

class Display

{

public:

virtual void display() = 0;

};

class Movie

{

public:

int movie\_no;

char name[100];

int hr, min;

Movie() {}

Movie(int a, const char b[], int c, int d) {

movie\_no = a;

strcpy\_s(name, b);

hr = c;

min = d;

}

void display() {

cout << movie\_no << "\t\t" << name << "\t" << hr << ":" << min << "\t" << endl;

}

int getMovieNumber() {

return movie\_no;

}

};

class Ticket : public Movie

{

public:

int no\_of\_persons;

Movie movies[3];

Ticket() {

movies[0] = Movie(1, "Spiderman", 11, 45);

movies[1] = Movie(2, "Lost World", 12, 35);

movies[2] = Movie(3, "Star wars", 14, 15);

}

string selectMovie() {

cout << "Enter movie number: ";

int movieNumber;

cin >> movieNumber;

bool found = false;

for (int i = 0; i < 3; i++) {

if (movieNumber == movies[i].getMovieNumber()) {

found = true;

cout << "You have selected " << movies[i].name << endl;

return movies[i].name;

break;

}

}

if (!found) {

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

try { throw (movieNumber); }

catch (int i) { selectMovie(); }

}

}

int enterNumberOfPersons() {

cout << "Enter number of persons: ";

cin >> no\_of\_persons;

if (no\_of\_persons <= 0 || no\_of\_persons > maxSeats) {

cout << "Invalid number of persons.Try again\n";

try { throw (no\_of\_persons); }

catch (int i) { enterNumberOfPersons(); }

}

return no\_of\_persons;

}

};

class Booking : public Ticket

{

public:

char seat\_map[maxSeats];

int booked\_seats[maxSeats];

int booked\_count;

Booking() : booked\_count(0) {

memset(seat\_map, 'O', maxSeats);

}

void selectSeats() {

for (int i = 0; i < no\_of\_persons; i++) {

cout << "Seat for person " << i + 1 << " (e.g., A1): ";

char row;

int col;

cin >> row >> col;

int row\_index = row - 'A';

int col\_index = col - 1;

if (row\_index < 0 || row\_index >= maxSeats / 4 || col\_index < 0 || col\_index >= 4) {

cout << "Invalid seat selection.\n";

i--;

}

else {

int seat\_number = row\_index \* 4 + col\_index;

if (seat\_map[seat\_number] == 'X') {

cout << "Seat is already booked.\n";

i--;

}

else {

seat\_map[seat\_number] = 'X';

booked\_seats[booked\_count++] = seat\_number;

}

}

}

}

void display() {

cout << "The booked seats are:\n";

for (int i = 0; i < booked\_count; i++) {

int seat\_number = booked\_seats[i];

char row = 'A' + seat\_number / 4;

int col = seat\_number % 4 + 1;

cout << "Seat for Person "<<i+1<<":(" << row<<"," << col << ")\n";

}

}

double calculateTotal() {

for (int i = 0; i < booked\_count; i++) {

int seat\_number = booked\_seats[i];

char row = 'A' + seat\_number / 4;

if (row == 'A')

total += 400.0;

else if (row == 'B' || row == 'C')

total += 350.0;

else if (row == 'D')

total += 300.0;

}

return total;

}

void print()

{

ofstream file("moviet\_tickects");

if (file.is\_open())

{

// Write the ticket information to the file

file << "Movie: " << selectMovie() << "\n";

file << "Number of Persons: " << no\_of\_persons << "\n";

file << "Booked Seats: ";

for (int i = 0; i < booked\_count; i++)

{

int seat\_number = booked\_seats[i];

char row = 'A' + seat\_number / 4;

int col = seat\_number % 4 + 1;

file << "(" << row << "," << col << ")";

if (i < booked\_count - 1)

{

file << ", ";

}

}

file << "\n";

file << "Total Cost: Rs. " << calculateTotal() << "\n";

file.close();

}

else

{

cout << "Unable to open the file for ticket printing." << endl;

}

}

};

class Eatables

{

public:

double popcorn, coke, water, chips, miranda, samosa;

double pcost, ccost, wcost, chcost, mcost, scost;

Eatables() : pcost(0), ccost(0), wcost(0), chcost(0), mcost(0), scost(0) {}

void selectEatables()

{

int choice;

while (true)

{

cout << "\nEnter choice for snacks (Press 4 to skip): ";

cin >> choice;

switch (choice) {

case 1: {

cout << "No. of popcorns: ";

cin >> popcorn;

total += 100.0 \* popcorn;

break;

}

case 2: {

cout << "No. of chips: ";

cin >> chips;

total += 50.0 \* chips;

break;

}

case 3: {

cout << "No. of samosas: ";

cin >> samosa;

total += 10.0 \* samosa;

break;

}

case 4: {

return;

}

default: {

cout << "Invalid input ";

}

}

}

}

void selectDrinks()

{

int choice;

while (true)

{

cout << "\nEnter choice for drinks (Press 4 to skip): ";

cin >> choice;

switch (choice)

{

case 1: {

cout << "No. of water bottles: ";

cin >> water;

total += 10.0 \* water;

break;

}

case 2: {

cout << "No. of coke drinks: ";

cin >> coke;

total += 30.0 \* coke;

break;

}

case 3: {

cout << "No. of Mirinda drinks: ";

cin >> miranda;

total += 30.0 \* miranda;

break;

}

case 4: {

return;

}

default: {

cout << "Invalid input ";

}

}

}

}

};

template<class T>

T bill(T n1, T n2)

{

{

if (n1 >= n2)

{

T change = n1 - n2;

cout << "Total given amount is Rs. " << setprecision(2) << setiosflags(ios::fixed) << n1 << ".Change to be given is Rs. " << setprecision(2) <<setiosflags(ios::fixed)<< change << endl;

return n1, change;

}

else

{

try { throw (n1); }

catch (double i) {

cout << "Insufficient balance. Please try again." << endl;

}

}

}

}

class Payment : public Booking, public Eatables,public Display

{

public:

int choice;

double totalm = 0, amt = 0;

Payment()

{

cout << "\nYour total bill is Rs. " << total << endl;

}

void display()

{

cout << "How do you want to pay?\n1. Cash\n2. UPI\n3. Card\nEnter your choice: ";

cin >> choice;

if (choice == 1 || choice == 2 || choice == 3)

{

cout << "Enter the amount given: ";

cin >> amt;

bill(amt, total);

}

else

{

try { throw (choice); }

catch (int i) {

cout << "Invalid input...Retry\n";

display();

}

}

}

void print() {

ofstream file("Snacks Recipt");

if (file.is\_open()) {

file << "Recipt no.:" << rand() << endl;

file << "Snacks and Drinks:\n";

file << "Popcorn: " << pcost << "\n";

file << "Chips: " << chcost << "\n";

file << "Samosa: " << scost << "\n";

file << "Water Bottles: " << wcost << "\n";

file << "Coke Drinks: " << ccost << "\n";

file << "Miranda Drinks: " << mcost << "\n";

file.close();

}

else {

cout << "Unable to open the file for printing snacks and drinks bill." << endl;

}

}

~Payment()

{

cout << "Thank you....It was good serving you";

}

};

int main() {

cout << "------------MOVIE TICKET BOOKING------------" << endl;

Movie m1(1, "Spiderman", 11, 45);

Movie m2(2, "Lost World", 12, 35);

Movie m3(3, "Star wars", 14, 15);

cout << "Movie no.\tMovie name\tTime\n";

m1.display();

m2.display();

m3.display();

Ticket ticket;

ticket.selectMovie();

int numberOfPersons = ticket.enterNumberOfPersons();

cout << "---------Seating map---------\n";

for (char row = 'A'; row < 'E'; row++)

{

for (int col = 1; col <= 4; col++)

{

cout << "(" << row << "," << col << ")\t";

}

cout << "\n";

}

cout << "----------Ticket price chart----------\nRow\tPrice(RS)\nBack/Top Row(A)\t\t400\nMiddle Row(B/C)\t\t350\nFront/Bottom Row(D)\t\t300\n";

Booking booking;

booking.no\_of\_persons = numberOfPersons;

booking.selectSeats();

booking.display();

booking.print();

double totalBill = booking.calculateTotal();

cout << "------------Snack & Drinks Menu------------\n";

cout << "Snacks\nNo.\tName\tPrice(RS)\n1.\tPopcorn\t100\n2.\tChips\t50\n3.\tSamosa\t10\n";

cout << "Drinks\nNo.\tName\tPrice(RS)\n1.\tWater(500ml)\t10\n2.\tCoke(250ml)\t30\n3.\tMiranda(250ml)\t30\n";

Eatables eatables;

int choice,a;

cout << "Do you want any snacks/drinks?\nType '1' for yes or '2' for no: ";

cin >> choice;

switch (choice)

{

case 1: {

cout << "Do you want 1. Only Snacks 2. Only drinks 3. Both?\n";

cin >> a;

switch (a)

{

case 1:

{

eatables.selectEatables();

break;

}

case 2:

{

eatables.selectDrinks();

break;

}

case 3:

{

eatables.selectEatables();

eatables.selectDrinks();

break;

}

default: {

cout << "Invalid Input\n";

}

}

}

case 2:

{

cout << "Let's proceed to payment\n";

break;

}

default:

{

cout << "Invalid Input\n";

}

}

Payment payment;

payment.display();

payment.print();

return 0;

}

Output:





