**OOP Assignment 3**

**24k-0880**

**Muhammad Umer Karim**

Question 1:

#include<iostream>

#include<fstream>

**using** **namespace** std;

**class** user{

**public**:

string id;

string Name;

**bool** feesPaid;

**virtual** string getType() = 0;

**virtual** ~user() {}

**void** save(string fname){

}

**void** load(string fname){

}

};

**class** student : **public** user {

**public**:

string getType() **override** {

**return** "student";

}

};

**class** faculty : **public** user {

**public**:

string getType() **override** {

**return** "faculty";

}

};

**class** driver{

**public**:

string name;

**bool** LiscenceValid;

driver() : name(""), LiscenceValid(**false**) {}

driver(string n, **bool** valid) : name(n), LiscenceValid(valid){}

};

**class** route{

**public**:

string Start;

string End;

**int** distanceCovered;

route(){

}

route(string start, string end, **int** distance) : Start(start), End(end), distanceCovered(distance) {}

};

**class** seats{

**public**:

user\* occupant;

string seatType;

**void** assignStudent(student& s) {

**if**(seatType == "faculty"){

**throw** ("Seat is reserved for faculty.\n");

}

occupant = &s;

}

**void** assignFaculty(faculty& f) {

occupant = &f;

**if**(seatType == "faculty"){

**throw** ("Seat is reserved for students.\n");

}

}

};

**class** vehicle{

**public**:

seats\* Seats;

**int** currentSeat;

**int** capacity;

driver Driver;

route\* Route;

**bool** AC;

vehicle(){}

vehicle(**int** cap, **bool** ac) {

Seats = **new** seats[cap];

capacity = cap;

currentSeat = 0;

AC = ac;

Route = **nullptr**;

}

**void** bookSeat(student& s) {

**if** (!s.feesPaid)

**throw** "Fees not paid.";

**if** (currentSeat >= capacity)

**throw** "Seats are full.";

**if** (AC)

cout << "Additional 2000RS are required for this AC Vehicle." << endl;

Seats[currentSeat].assignStudent(s);

currentSeat++;

cout << "Seat Successfully Booked." << endl;

}

**void** bookSeat(faculty& f) {

**if** (!f.feesPaid)

**throw** "Fees not paid.";

**if** (currentSeat >= capacity)

**throw** "Seats are full.";

**if** (AC)

cout << "Additional 2000RS are required for this AC Vehicle." << endl;

Seats[currentSeat].assignFaculty(f);

currentSeat++;

cout << "Seat Successfully Booked." << endl;

}

**void** assignDriver(**const** driver& d) {

Driver = d;

}

**void** assignRoute(route\* r) {

Route = r;

}

**virtual** ~vehicle() {

**delete**[] Seats;

}

**virtual** string getType() {

**return** "vehicle";

}

};

**class** coaster : **public** vehicle {

**public**:

coaster(**bool** ac = **false**) : vehicle(32, ac) {

}

};

**class** bus : **public** vehicle {

**public**:

bus(**bool** ac = **false**) : vehicle(52, ac) {

}

};

**class** transporter {

**public**:

string name;

vehicle\*\* points;

**int** totalPoints;

**int** MaxPoints;

driver\* drivers;

**int** totalDrivers;

**int** MaxDrivers;

route\* routes;

**int** totalRoutes;

**int** MaxRoutes;

transporter(**int** p, **int** d, **int** r)

: MaxPoints(p), MaxDrivers(d), MaxRoutes(r), totalPoints(0), totalDrivers(0), totalRoutes(0) {

points = **new** vehicle\*[MaxPoints];

drivers = **new** driver[MaxDrivers];

routes = **new** route[MaxRoutes];

}

**void** addVehicle(vehicle\* v) {

**if** (totalPoints >= MaxPoints) **throw** "Maximum vehicle points reached.";

points[totalPoints++] = v;

}

**void** addDriver(driver d) {

**if** (totalDrivers >= MaxDrivers) **throw** "Maximum drivers reached.";

drivers[totalDrivers++] = d;

}

**void** addRoute(route r) {

**if** (totalRoutes >= MaxRoutes) **throw** "Maximum routes reached.";

routes[totalRoutes++] = r;

}

~transporter() {

**for** (**int** i = 0; i < totalPoints; ++i) {

**delete** points[i];

}

**delete**[] points;

**delete**[] drivers;

**delete**[] routes;

}

};

**int** main() {

cout<<"24k-0880 - Muhammad Umer Karim - Assignemtn 3 - Question 1"<<endl<<endl;

**int** maxPoints, maxDrivers, maxRoutes;

cout << "Enter max number of vehicle points: ";

cin >> maxPoints;

cout << "Enter max number of drivers: ";

cin >> maxDrivers;

cout << "Enter max number of routes: ";

cin >> maxRoutes;

transporter t(maxPoints, maxDrivers, maxRoutes);

**try** {

t.addRoute(route("A", "B", 15));

t.addRoute(route("B", "C", 25));

t.addRoute(route("C", "D", 35));

t.addDriver(driver("Ali", **true**));

t.addDriver(driver("Ahmed", **true**));

t.addDriver(driver("Sara", **false**));

t.addVehicle(**new** coaster(**true**));

t.addVehicle(**new** coaster(**false**));

t.addVehicle(**new** bus(**true**));

t.addVehicle(**new** bus(**false**));

t.addVehicle(**new** coaster(**true**));

t.points[0]->assignDriver(t.drivers[0]);

t.points[0]->assignRoute(&t.routes[0]);

t.points[1]->assignDriver(t.drivers[1]);

t.points[1]->assignRoute(&t.routes[1]);

t.points[2]->assignDriver(t.drivers[2]);

t.points[2]->assignRoute(&t.routes[2]);

}

**catch** (**const** **char**\* msg) {

cout << "Initialization Error: " << msg << endl;

}

**int** choice;

**do** {

cout << "\n--- Transport System Menu ---\n";

cout << "1. Add Driver\n";

cout << "2. Add Route\n";

cout << "3. Add Vehicle\n";

cout << "4. Assign Driver and Route to Vehicle\n";

cout << "5. Book Seat\n";

cout << "6. Save All Data\n";

cout << "0. Exit\n";

cout << "Choose an option: ";

cin >> choice;

**try** {

**if** (choice == 1) {

string name;

**bool** lic;

cout << "Enter driver name: ";

cin >> name;

cout << "Is license valid (1/0)? ";

cin >> lic;

t.addDriver(driver(name, lic));

}

**else** **if** (choice == 2) {

string start, end;

**int** dist;

cout << "Enter start: ";

cin >> start;

cout << "Enter end: ";

cin >> end;

cout << "Enter distance: ";

cin >> dist;

t.addRoute(route(start, end, dist));

}

**else** **if** (choice == 3) {

**int** type;

**bool** ac;

cout << "Enter type (1 for Bus, 0 for Coaster): ";

cin >> type;

cout << "Is AC available (1/0)? ";

cin >> ac;

**if** (type == 1)

t.addVehicle(**new** bus(ac));

**else**

t.addVehicle(**new** coaster(ac));

}

**else** **if** (choice == 4) {

**int** vid, did, rid;

cout << "Enter vehicle index: ";

cin >> vid;

cout << "Enter driver index: ";

cin >> did;

cout << "Enter route index: ";

cin >> rid;

**if** (vid >= t.totalPoints || did >= t.totalDrivers || rid >= t.totalRoutes)

**throw** "Invalid index for assignment.";

t.points[vid]->assignDriver(t.drivers[did]);

t.points[vid]->assignRoute(&t.routes[rid]);

}

**else** **if** (choice == 5) {

**int** vid, userType;

cout << "Enter vehicle index: ";

cin >> vid;

**if** (vid >= t.totalPoints) **throw** "Invalid vehicle index.";

cout << "Enter user type (1 = Student, 2 = Faculty): ";

cin >> userType;

**if** (userType == 1) {

student s;

cout << "Enter ID: "; cin >> s.id;

cout << "Enter Name: "; cin >> s.Name;

cout << "Fees Paid (1/0): "; cin >> s.feesPaid;

t.points[vid]->bookSeat(s);

}

**else** **if** (userType == 2) {

faculty f;

cout << "Enter ID: "; cin >> f.id;

cout << "Enter Name: "; cin >> f.Name;

cout << "Fees Paid (1/0): "; cin >> f.feesPaid;

t.points[vid]->bookSeat(f);

}

**else** **throw** "Invalid user type.";

}

**else** **if** (choice == 6) {

ofstream ufile("users.txt");

ofstream rfile("routes.txt");

ofstream dfile("drivers.txt");

ofstream vfile("vehicles.txt");

**for** (**int** i = 0; i < t.totalRoutes; i++) {

rfile << t.routes[i].Start << " " << t.routes[i].End << " " << t.routes[i].distanceCovered << endl;

}

**for** (**int** i = 0; i < t.totalDrivers; i++) {

dfile << t.drivers[i].name << " " << t.drivers[i].LiscenceValid << endl;

}

**for** (**int** i = 0; i < t.totalPoints; i++) {

vfile << (**dynamic\_cast**<bus\*>(t.points[i]) ? "Bus" : "Coaster") << " "

<< t.points[i]->capacity << " " << t.points[i]->AC << endl;

}

ufile.close();

rfile.close();

dfile.close();

vfile.close();

cout << "Data saved successfully.\n";

}

**else** **if** (choice != 0) {

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

}

}

**catch** (**const** **char**\* msg) {

cout << "Error: " << msg << endl;

}

} **while** (choice != 0);

**return** 0;

}

Output:

