OOP-ASSIGNMENT 2

Task1:

#include <iostream>

using namespace std;

class User

{

public:

    string name;

    int id;

    User(string *n*, int *i*) : name(*n*), id(*i*) {}

    virtual void payment() = 0;

    virtual ~User() {}

};

class Student : public User

{

public:

    double fee;

    Student(string *n*, int *i*, double *f*) : User(*n*, *i*), fee(*f*) {}

    void payment() override

    {

        cout << name << " pays " << fee << " per semester for transport." << endl;

    }

};

class Teacher : public User

{

public:

    double fee;

    Teacher(string *n*, int *i*, double *f*) : User(*n*, *i*), fee(*f*) {}

    void payment() override

    {

        cout << name << " pays " << fee << " per month for transport." << endl;

    }

};

class Staff : public User

{

public:

    double fee;

    Staff(string *n*, int *i*, double *f*) : User(*n*, *i*), fee(*f*) {}

    void payment() override

    {

        cout << name << " pays " << fee << " per month for transport." << endl;

    }

};

class Route

{

public:

    string route\_name;

    int route\_num;

    string stops[10];

    Route(string *n*, int *num*, string *stop*[10]) : route\_name(*n*), route\_num(*num*)

    {

        for (int i = 0; i < 10; i++)

            stops[i] = *stop*[i];

    }

    bool operator == (const Route &*other*)

    {

        return route\_num == *other*.route\_num;

    }

    void display()

    {

        cout << "( " << route\_name  << " ) -" << " route number: " << route\_num << endl;

    }

};

class Bus

{

public:

    int bus\_id;

    Route\* route\_number;

    Bus(int *id*, Route\* *route*) : bus\_id(*id*), route\_number(*route*) {}

    void displayer\_for\_bus()

    {

        cout << endl << "bus id: " << bus\_id << " assigned to route: "; route\_number->display();

    }

};

class TransportSystem

{

public:

    User\* user[100];

    Bus\* point[50];

    Route\* route[20];

    int u\_counter, p\_counter, r\_counter;

    TransportSystem() : u\_counter(0), p\_counter(0), r\_counter(0) {}

    void add\_user(User\* *u*)

    {

        user[u\_counter++] = *u*;

    }

    void add\_route(Route\* *r*)

    {

        route[r\_counter++] = *r*;

    }

    void add\_bus(Bus\* *b*)

    {

        point[p\_counter++] = *b*;

    }

    void process\_payment()

    {

        for (int i = 0; i < u\_counter; i++)

            user[i]->payment();

    }

};

int main()

{

    TransportSystem system;

    string stops1[10] = {"Fast", "Landhi", "Quaidabad", "Malir", "Malir Halt", "Drigh Road", "Millennium", "Johar Mor", "Jauhar Chowrangi", "Mosmiyat"};

    Route r1("Main Campus - City Center", 1, stops1);

    system.add\_route(&r1);

    Student s("Baqar", 240006, 38000);

    Teacher t("Prof: Ahmed", 156, 10000);

    Staff st("Mr Anwer", 323, 8000);

    system.add\_user(&s);

    system.add\_user(&t);

    system.add\_user(&st);

    Bus b1(1, &r1);

    system.add\_bus(&b1);

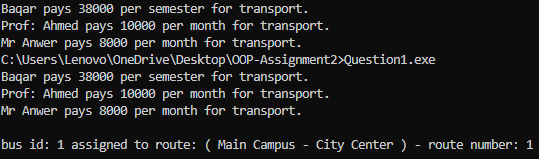
    system.process\_payment();

    b1.displayer\_for\_bus();

    return 0;

}

Output:



Task2:

#include <iostream>

#include <string>

using namespace std;

class Ghost

{

public:

    string name;

    int lvl;

    Ghost(string *n*) : name(*n*)

    {

        lvl = (name.length() \* 7 + 4) % 10 + 1;

    }

    virtual void haunt() = 0;

    friend ostream& operator<<(ostream& *o*, const Ghost& *g*)

    {

*o* << *g*.name << " (level: " << *g*.lvl << ")";

        return *o*;

    }

    virtual ~Ghost() {}

};

class Poltergeist : public virtual Ghost

{

public:

    Poltergeist(string *n*) : Ghost(*n*) {}

    void haunt() override

    {

        cout << name << " moves objects violently" << endl;

    }

};

class Banshee : public virtual Ghost

{

public:

    Banshee(string *n*) : Ghost(*n*) {}

    void haunt() override

    {

        cout << name << " screams loudly" << endl;

    }

};

class ShadowGhost : public virtual Ghost

{

public:

    ShadowGhost(string *n*) : Ghost(*n*) {}

    void haunt() override

    {

        cout << name << " whispers creepily" << endl;

    }

};

class ShadowPoltergeist : public ShadowGhost, public Poltergeist

{

public:

    ShadowPoltergeist(string *n*) : Ghost(*n*), ShadowGhost(*n*), Poltergeist(*n*) {}

    void haunt() override

    {

        ShadowGhost::haunt();

        Poltergeist::haunt();

    }

};

class Visitor

{

public:

    string name;

    int visitor\_lvl;

    Visitor(string *n*, int *b*) : name(*n*), visitor\_lvl(*b*) {}

    void react(int *lvl*)

    {

        if (*lvl* > visitor\_lvl + 2)

        {

            cout << name << " screams and runs away" << endl;

        }

        else if (*lvl* < visitor\_lvl - 2)

        {

            cout << name << " laughs and taunts the ghost" << endl;

        }

        else

        {

            cout << name << " feels uneasy and gets a shaky voice" << endl;

        }

    }

};

class HauntedHouse

{

public:

    string name;

    Ghost\* ghosts[5];

    int counter;

    HauntedHouse(string *n*) : name(*n*), counter(0) {}

    ~HauntedHouse()

    {

        for (int i = 0; i < counter; i++)

        {

            delete ghosts[i];

        }

    }

    void add(Ghost\* *g*)

    {

        if (counter < 5)

            ghosts[counter++] = *g*;

    }

    void stimulate(Visitor& *v*)

    {

        cout << endl << name << " haunting simulation begins!" << endl;

        for (int i = 0; i < counter; i++)

        {

            ghosts[i]->haunt();

*v*.react(ghosts[i]->lvl);

        }

    }

};

void visit(HauntedHouse& *house*, Visitor *visitors*[], int *size*)

{

    cout << "Visitors are entering " << *house*.name << "..." << endl;

    for (int i = 0; i < *size*; i++)

    {

*house*.stimulate(*visitors*[i]);

    }

}

int main()

{

    HauntedHouse house1("spooky kingdom");

    house1.add(new Poltergeist("baqar"));

    house1.add(new Banshee("kashif"));

    house1.add(new ShadowGhost("omer"));

    house1.add(new ShadowPoltergeist("anwer"));

    Visitor visitors[3] =

    {

        Visitor("ali", 2),

        Visitor("amir", 6),

        Visitor("kashan", 9)

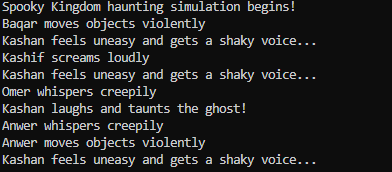
    };

    visit(house1, visitors, 3);

    return 0;

}

Output:



Task3:

#include <iostream>

#include <string>

using namespace std;

class Vehicle

{

public:

    string id;

    static int deliveries;

    Vehicle(string *i*) : id(*i*)

    {

        deliveries++;

    }

    virtual ~Vehicle()

    {

        deliveries--;

    }

    virtual void calc\_route() = 0;

    virtual void est\_time() = 0;

    virtual void deliver(string *package\_num*)

    {

        cout << "Vehicle " << id << " delivering package " << *package\_num* << endl;

    }

    virtual bool operator==(const Vehicle &*other*) = 0;

    friend void solve\_issue(Vehicle &*v1*, Vehicle &*v2*);

    int static total\_del()

    {

        return deliveries;

    }

};

int Vehicle::deliveries = 0;

class Drone : public Vehicle

{

public:

    Drone(string *id*) : Vehicle(*id*) {}

    void calc\_route() override

    {

        cout << "drone " << id << " calculating aerial route." << endl;

    }

    void est\_time() override

    {

        cout << "drone " << id << " estimating fast delivery time." << endl;

    }

    void deliver(string *package\_num*, string *urgentlvl*)

    {

        cout << "drone " << id << " delivering package " << *package\_num* << " with high-speed for " << *urgentlvl* << " urgency." << endl;

    }

    bool operator==(const Vehicle &*other*) override

    {

        return id == *other*.id;

    }

};

class Timeship : public Vehicle

{

public:

    Timeship(string *id*) : Vehicle(*id*) {}

    void calc\_route() override

    {

        cout << "timeShip " << id << " calculating time-travel path." << endl;

    }

    void est\_time() override

    {

        cout << "timeShip " << id << " estimating historical delivery accuracy." << endl;

    }

    void deliver(string *package\_num*, string *urgentlvl*)

    {

        cout << "timeship " << id << " validating historical accuracy for package " << *package\_num* << " with urgency " << *urgentlvl* << endl;

    }

    bool operator==(const Vehicle &*other*) override

    {

        return id == *other*.id;

    }

};

class Hyperpod : public Vehicle

{

public:

    Hyperpod(string *id*) : Vehicle(*id*) {}

    void calc\_route() override

    {

        cout << "hyperPod " << id << " calculating underground tunnel route." << endl;

    }

    void est\_time() override

    {

        cout << "hyperpod " << id << " estimating bulk delivery time." << endl;

    }

    bool operator==(const Vehicle &*other*) override

    {

        return id == *other*.id;

    }

};

void solve\_issue(Vehicle &*v1*, Vehicle &*v2*)

{

    cout << "resolving conflict between " << *v1*.id << " and " << *v2*.id << endl;

    cout << "Choosing the more efficient vehicle based on urgency and capacity." << endl;

}

int main()

{

    Drone d("K0006");

    Timeship s("K0022");

    Hyperpod p("K2539");

    d.deliver("PK233", "Iftar");

    s.deliver("PK234", "Historical");

    p.calc\_route();

    cout << endl;

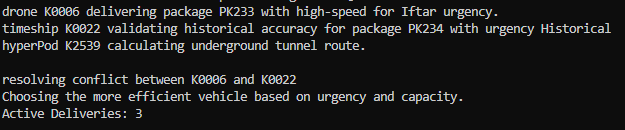
    solve\_issue(d, s);

    cout << "Active Deliveries: " << Vehicle::total\_del() << endl;

    return 0;

}

Output:



Task4:

#include <iostream>

using namespace std;

unsigned long hash\_passcode(string *password*)

{

    unsigned long hash = 5381;

    for (char c : *password*)

    {

        hash = (hash \* 33) + c;

    }

    return hash;

}

class User {

public:

    string name;

    int id;

    string email;

    unsigned long hashed\_password;

    string permissions[3];

    User(string *n*, int *id*, string *e*, string *password*, string *perm1* = "", string *perm2* = "", string *perm3* = "") : name(*n*), id(*id*), email(*e*), hashed\_password(hash\_passcode(*password*))

    {

        permissions[0] = *perm1*;

        permissions[1] = *perm2*;

        permissions[2] = *perm3*;

    }

    virtual void display()

    {

        cout << "naam: " << name << endl << "id: " << id << endl << "email: " << email << endl;

    }

    bool validate(string *p*)

    {

        return hash\_passcode(*p*) == hashed\_password;

    }

    bool permisssion(string *a*)

    {

        for (int i = 0; i < 3; i++)

        {

            if (permissions[i] == *a*)

                return true;

        }

        return false;

    }

    virtual bool access()

    {

        if (permisssion("full lab access"))

            {

                cout << name << " has full access to the lab." << endl;

                return true;

            }

        else

            {

                cout << name << " does not have lab access." << endl;

                return false;

            }

    }

};

class Student : public User

{

public:

    int assignments[5];

    Student(string *name*, int *id*, string *email*, string *password*) : User(*name*, *id*, *email*, *password*, "submit\_assignment")

    {

        for (int i = 0; i < 5; i++)

        {

            assignments[i] = 0;

        }

    }

    void submit(int *i*)

    {

        if (*i* >= 0 && *i* < 5)

        {

            assignments[*i*] = 1;

            cout << "Assignment " << *i* + 1 << " submitted by " << name << "." << endl;

        }

        else

        {

            cout << "Invalid assignment index." << endl;

        }

    }

    void display() override

    {

        User::display();

        cout << "Assignments: ";

        for (int i = 0; i < 5; i++)

        {

            cout << assignments[i] << " ";

        }

        cout << endl;

    }

};

class TA : public Student

{

public:

    string assign\_student[10];

    int counter;

    string projects[2];

    int p\_counter;

    TA(string *name*, int *id*, string *email*, string *password*) : Student(*name*, *id*, *email*, *password*)

    {

        permissions[1] = "view\_projects";

        permissions[2] = "manage\_students";

        counter = 0;

        p\_counter = 0;

    }

    void assign(string *s\_name*)

    {

        if (counter < 10)

            {

                assign\_student[counter++] = *s\_name*;

                cout << *s\_name* << " assigned to TA " << name << endl;

            }

        else

            {

                cout << "TA " << name << " cannot manage more than 10 students." << endl;

            }

    }

    void adding\_projects(string *p\_name*)

    {

        if (p\_counter < 2)

            {

                projects[p\_counter++] = *p\_name*;

                cout << "TA " << name << " started working on project: " << *p\_name* << endl;

            }

        else

            {

                cout << "TA " << name << " is already working on 2 projects." << endl;

            }

    }

    void display() override

    {

        Student::display();

        cout << "Assigned Students: ";

        for (int i = 0; i < counter; i++)

        {

            cout << i+1 << ". " << assign\_student[i] << endl;

        }

        cout << "Projects: ";

        for (int i = 0; i < p\_counter; i++)

        {

            cout << projects[i] << " ";

        }

        cout << endl;

    }

};

class Professor : public User

{

public:

    Professor(string *name*, int *id*, string *email*, string *password*) : User(*name*, *id*, *email*, *password*, "assign\_projects", "full\_lab\_access") {}

    void assign\_project(TA& *ta*, string *p\_name*)

    {

*ta*.adding\_projects(*p\_name*);

    }

    void display() override

    {

        User::display();

        cout << "Professor with full lab access." << endl;

    }

};

void validateAndPerformAction(User\* *user*, string *password*, string *action*)

{

    if (*user*->validate(*password*))

        {

            cout << "aathentication successful for " << *user*->name << "." << endl;

                if (*user*->permisssion(*action*))

                {

                    cout << "Action permitted: " << *action* << "." << endl;

                }

            else

                {

                    cout << "Action denied: " << *action* << "." << endl;

                }

        }

    else

        {

            cout << "Authentication failed for " << *user*->name << "." << endl;

        }

}

int main()

{

    Student s1("Baqar", 240006, "k240006@nu.edu.pk", "password123");

    TA ta1("Mr TA", 0000, "k200000@nu.edu.pk", "securepass");

    Professor p1("Mr Prof", 23567, "mr.prof@nu.edu.pk", "professorpass");

    s1.display();

    validateAndPerformAction(&s1, "password123", "submit assignment");

    validateAndPerformAction(&ta1, "securepass", "manage students");

    validateAndPerformAction(&p1, "professorpass", "assign projects");

    s1.submit(1);

    s1.display();

    ta1.assign("Baqar");

    ta1.adding\_projects("AI Research");

    ta1.display();

    return 0;

}

Output:

