Лабораторна робота №6

Тема: Композиція об’єктів в ООП
Мета: ознайомитись із способами та механізмами об’єктної композиції в ООП

Завдання 1. Розробіть клас Student (в окремих файлах student.h і student.cpp) із атрибутами: прізвище, ім’я, по батькові, номер залікової книжки, державник/платник (тип bool).
Визначте для даного класу конструктор по замовчуванню, який буде запитувати у користувача дані для заповнення атрибутів об’єкта; параметризований конструктор; операцію виводу у потік.
У головній функції виконайте перевірку функціонування методів класу створивши три об’єкти різними способами і вивівши їх на екран за допомогою оператора виводу у потік.

Завдання 2. Розробіть клас Grupa, який міститиме як атрибут назву групи (тип char * або std::string), спеціальність і список студентів групи, студенти описуються за допомогою класу Student, який визначений у попередньому завданні.
Визначте для даного класу всі можливі конструктори, деструктор, операції виводу в потік. Тип відношення між класами Grupa і Student – агрегація із кардинальністю 0..01 – 1..*

Завдання 3. Розробіть клас Facultet, який міститиме наступні атрибути: назву факультету (тип char * або std::string) і список груп, групи описуються за допомогою класу Grupa, який визначений у попередньому завданні.
Визначте для даного класу всі можливі конструктори, деструктор, операції виводу в потік. Тип відношення між класами Facultet і Grupa – композиція із кардинальністю 1 – 1..*.

Student.h:

```
#pragma once
#include <string>
class Student {
private:
    std::string surname;
    std::string name;
    std::string fname;
    int gradebook_num;
    bool SFE;           // state form of education
public:
    Student();
    void setSurname(std::string surn);
    Student(std::string, std::string, std::string, int, bool sfe = false);
    friend std::ostream& operator << (std::ostream& os, const Student&);
    friend std::istream& operator >> (std::istream is, Student&);
};
```

Student.cpp:

```cpp
#include "Student.h"
#include <iostream>
#include <string>

using namespace std;

Student::Student() {
    cout << "Surname: ";
    cin >> surname;
    cout << "Name: ";
    cin >> name;
    cout << "Father's name: ";
    cin >> fname;
    cout << "Enter No. of grade book: ";
    cin >> gradebook_num;
    cout << "Is form of education state? Y or N";
    char guess;
    cin >> guess;
    switch (guess) {
    case 'Y':
    case 'y':
        SFE = true;
        break;
    case 'N':
    case 'n':
    default:
        SFE = false;
    }

}
void Student::setSurname(string surn) {
    this->surname = surn;
}
Student::Student(string surname, string name, string fname = "", int grade_num = 0,
bool sfe)
    : surname(surname), name(name), fname(fname), gradebook_num(grade_num), SFE(sfe)
{}

istream& operator >> (istream is, Student& st) {
    cout << "Surname: ";
    is >> st.surname;
    cout << "Name: ";
    is >> st.name;
    cout << "Father's name: ";
    is >> st.fname;
    cout << "Enter No. of grade book: ";
    is >> st.gradebook_num;
    cout << "Is form of education state? Y or N";
    char guess;
    cin >> guess;
    switch (guess) {
    case 'Y':
    case 'y':
        st.SFE = true;
        break;
    case 'N':
    case 'n':
    default:
```

```
        st.SFE = false;
    }
    return is;
}

ostream& operator << (ostream& os, const Student& st) {
    os << st.surname << ", " << st.name << ", " << st.fname << ", "
        << st.gradebook_num << ", ";
    if (st.SFE) {
        os << "YES";
    }
    else {
        os << "NO";
    }
    os << endl;
    return os;
}
```

Grupa.h:

```
#pragma once
#include "Student.h"
#include <iostream>

class Grupa {
private:
    std::string groupName;
    std::string specialty;
    int studCopacity;
    Student* stud;
public:
    Grupa();
    Grupa(std::string, std::string, int, Student*);
    //Grupa(const Grupa&);
    void showStudent(int id);
    ~Grupa();

    friend std::ostream& operator << (std::ostream& os, const Grupa& gr);
    friend std::istream& operator >> (std::istream is, Grupa&);
};
```

Grupa.cpp:

```
#include "Grupa.h"
#include <iostream>
using namespace std;

Grupa::Grupa() : groupName(""), specialty(""), studCopacity(0), stud(nullptr) {}
Grupa::Grupa(string grName, string specialty, int capacity, Student* st)
    : groupName(grName), specialty(specialty), studCopacity(capacity), stud(st) {}
Grupa::~Grupa() {
}

//Grupa::Grupa(const Grupa& gr) {
//    this->groupName = gr.groupName;
//    this->specialty = gr.specialty;
//    this->studCopacity = this->studCopacity;
//    if (gr.stud) {
//        this->stud = new Student()
//    }
```

```
//}

void Grupa::showStudent(int id) {
    cout << this->stud[id];
}

std::ostream& operator << (std::ostream& os, const Grupa& gr) {
    os << "Group name: " << gr.groupName << endl
        << "Group specialty: " << gr.specialty << endl
        << "Students in group: " << gr.studCopacity << endl;
    return os;
}
```

Facultet.h:

```
#pragma once
#include "Grupa.h"
#include <string>

class Facultet {
private:
    std::string name;
    Grupa* groupList;
    int groupAmount;
public:
    //Facultet();
    Facultet(std::string, int);
    Facultet(std::string, int, Grupa*);
    void setName(std::string );
    void showGroups() const;
    ~Facultet();

    friend std::ostream& operator << (std::ostream& os, const Facultet&);
    friend std::istream& operator >> (std::istream is, Facultet&);
};
```

Facultet.cpp:

```
#include "Facultet.h"
#include "Grupa.h"
#include <iostream>
using namespace std;

//Facultet::Facultet() : name(""), groupAmount(0), groupList(nullptr) {}
Facultet::Facultet(string name, int groupAmount)
    : name(name), groupAmount(groupAmount) {
    this->groupList = new Grupa[groupAmount];
}
Facultet::Facultet(string name, int groupAmount, Grupa* gr)
    : name(name), groupAmount(groupAmount), groupList(gr) {}
void Facultet::setName(string name) {
    this->name = name;
}

Facultet::~Facultet() {
    if (groupList) {
        delete[] groupList;
        cout << "groups deleted" << endl;
    }
```

```
}


void Facultet::showGroups() const {
    for (int i = 0; i < this->groupAmount; i++)
        cout << this->groupList[i] << endl;
}

istream& operator >> (istream is, Facultet& f) {
    cout << "Name of faculty: ";
    is >> f.name;
    cout << "Amount of groups: ";
    is >> f.name;
    return is;
}

ostream& operator << (ostream& os, const Facultet& f) {
    os << f.name << ", " << f.groupAmount << endl;
    os << endl;
    return os;
}
```

lab6.cpp:

```
#include <iostream>
#include <string>
#include <Windows.h>
#include "Student.h"
#include "Grupa.h"
#include "Facultet.h"
#define STUDENTS 3
using namespace std;


int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);


    /**************** task 1 /****************/
    Student studs[STUDENTS] = {Student(), Student("Іваненко", "Іван",  "Іванович",
65334, 1), Student(studs[1]) };

    studs[1].setSurname("Ivanenko");
    for (int i = 0; i < STUDENTS; ++i) {
        cout << studs[i];
    }

    /**************** task 1 /****************/


    /**************** task 2 /****************/
    Student *st = new Student[3]{
        Student("Іваненко", "Іван", "Іванович", 65334, 1),
        Student("Петренко", "Петро", "Федорович", 65334),
        Student("Мінімальний", "Максим", "Середньович", 23464, 0)};

    Grupa* gr = new Grupa("cs", "comp sci", 3, st);
    cout << *gr;
    gr->showStudent(1);
    delete gr;

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

```
        cout << st[i];
    }

    /*************** task 2 /***************/


    /*************** task 3 /***************/
    Grupa* group = new Grupa[2]{
        Grupa("se", "softwave engineer", 3, st),
        Grupa("cs", "computer science", 3, st)
    };
    Facultet *facul = new Facultet("electronic machines", 2, group);
    cout << *facul;
    facul->showGroups();
    delete facul;

    /*************** task 3 /***************/

}
```

Результат:

```
Surname: Тараш
Name: Іван
Father's name: Нікнеймович
Enter No. of grade book: 64806
Is form of education state? Y or N n
Тараш, Іван, Нікнеймович, 64806, NO
Ivanenko, Іван, Іванович, 65334, YES
Іваненко, Іван, Іванович, 65334, YES
Group name: cs
Group specialty: comp sci
Students in group: 3
Петренко, Петро, Федорович, 65334, NO
Іваненко, Іван, Іванович, 65334, YES
Петренко, Петро, Федорович, 65334, NO
Мінімальний, Максим, Середньович, 23464, NO
electronic machines, 2

Group name: se
Group specialty: softwave engineer
Students in group: 3

Group name: cs
Group specialty: computer science
Students in group: 3

groups deleted
```

Висновок: ознайомився із способами та механізмами об’єктної композиції в ООП.