Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення
Дисципліна: Об’єктно-орієнтоване програмування

Лабораторна робота №3

Тема: «НАСЛІДУВАННЯ. СПЕЦИФІКАТОРИ ДОСТУПУ»

Виконав: ст. гр. КН-24
Куріщенко П. В.
Перевірив: асистент
Козірова Н. Л.

Кропивницький 2025

Завдання 1

1. Створіть клас Base із трьома полями з різними рівнями доступу:

```
#ifndef BASE_H
#define BASE_H

#include <iostream>

using namespace std;

class Base
{
public:
    Base();
    int publicVar;

protected:
    int protectedVar;

private:
    int privateVar;
};

#endif // BASE_H
```

2. Створіть три похідні класи, які успадковують клас Base з різними режимами доступу:

```
#ifndef PUBLICDERIVED_H
#define PUBLICDERIVED_H

#include "base.h"

class PublicDerived : public Base
{
public:
    PublicDerived();
    void changeVars();
};

#endif // PUBLICDERIVED_H
```

3. У кожному похідному класі створіть метод, який спробує змінити значення всіх трьох полів базового класу.

```
#include "publicderived.h"

PublicDerived::PublicDerived() {}

void PublicDerived::changeVars()
{
    publicVar    = 1;
```

```
    protectedVar = 1;
    privateVar   = 1;
}
```

4. У функції main() створіть об’єкти всіх трьох похідних класів та спробуйте отримати доступ до полів publicVar, protectedVar і privateVar через ці об’єкти.

```
#include "base.h"
#include "publicderived.h"
#include "protectedderived.h"
#include "privatederived.h"

int main()
{
    PublicDerived pubObj;
    pubObj.changeVars();

    cout << "Public field    = " << pubObj.publicVar << endl;
    cout << "Protected field = " << pubObj.protectedVar << endl;
    cout << "Private field   = " << pubObj.privateVar << endl;

    ProtectedDerived protObj;
    protObj.changeVars();

    cout << "Public field    = " << protObj.publicVar << endl;
    cout << "Protected field = " << protObj.protectedVar << endl;
    cout << "Private field   = " << protObj.privateVar << endl;

    PrivateDerived privObj;
    privObj.changeVars();

    cout << "Public field    = " << privObj.publicVar;
    cout << "Protected field = " << privObj.protectedVar;
    cout << "Private field   = " << privObj.privateVar;

    return 0;
}
```

5. Проаналізуйте та опишіть, які поля доступні в кожному класі та ззовні, залежно від типу успадкування.

Клас / Поле	publicVar	protectedVar	privateVar
PublicDerived (усередині)	доступне	доступне	недоступне
PublicDerived (ззовні)	доступне	недоступне	недоступне
ProtectedDerived (усередині)	доступне	доступне	недоступне
ProtectedDerived (ззовні)	недоступне	недоступне	недоступне
PrivateDerived (усередині)	доступне	доступне	недоступне
PrivateDerived (ззовні)	недоступне	недоступне	недоступне

Завдання 2

Використайте свій варіант із лабораторної роботи №2, у якому було реалізовано асоціацію між класами (наприклад, один клас містив поле об'єкта іншого класу), та перепишіть ці класи таким чином, щоб замість асоціації було використано наслідування: створіть базовий клас із загальними властивостями та методами, а також похідні класи, які наслідують базовий і розширюють його додатковими полями та функціональністю. У функції main() створіть об’єкти кожного класу, задайте їх властивості, виведіть відповідну інформацію, а також реалізуйте приклади взаємодії між об'єктами, наприклад, оновлення значень або обчислення на основі полів. Програму реалізуйте з використанням роздільної компіляції (.h + .cpp).

```
#ifndef MUSICIAN_H
#define MUSICIAN_H

#include "instrument.h"
#include "manager.h"

class Musician : public Manager
{
public:
    Musician();
    Musician(const QString& n,
             const Instrument& i,
             const QString& m)
        : name(n), instrument(i), Manager(m) {}

    QString getName();

private:
    QString name;
    Instrument instrument;
    //Manager* manager; //асоціація
};

#endif // MUSICIAN_H


#include <musicalband.h>

int main()
{
    string input;
    cout << "Enter band name: ";
    getline(cin, input);
    QString bandName = QString::fromStdString(input);

    cout << "Enter band style: ";
    getline(cin, input);
    QString bandStyle = QString::fromStdString(input);

    cout << "Enter band leader name: ";
    getline(cin, input);
```

```
    BandLeader leader(QString::fromStdString(input));

    cout << "Enter number of musicians: ";
    int numMusicians;
    cin >> numMusicians;
    cin.ignore();

    MusicalBand band(bandName, bandStyle, leader);

    for (int i = 0; i < numMusicians; ++i) {
        cout << "\n--- Musician " << (i+1) << " ---\n";

        cout << "Enter musician name: ";
        getline(cin, input);
        QString musicianName = QString::fromStdString(input);

        cout << "Enter instrument name: ";
        getline(cin, input);
        QString instrumentName = QString::fromStdString(input);

        cout << "Enter instrument production year: ";
        unsigned short year;
        cin >> year;
        cin.ignore();
        Instrument instrument(instrumentName, year);

        cout << "Enter manager name: ";
        getline(cin, input);
        //Manager* managerName = new Manager(QString::fromStdString(input));
        QString managerName = QString::fromStdString(input);

        Musician* musician = new Musician(musicianName, instrument,
managerName);
        band.addMusician(musician);
    }

    cout << "\n--- Band Info ---\n";
    cout << "Band: " << band.getName().toStdString()
         << " (" << band.getStyle().toStdString() << ")" << endl;
    cout << "Leader: " << leader.getName().toStdString() << endl;

    for (int i = 0; i < band.getLen(); ++i) {
        Musician* m = band.getList().at(i);
        cout << "Musician " << (i+1) << ": " << m->getName().toStdString() <<
                "(" << m->Manager::getName().toStdString() << ")" << endl;
    }

    for (Musician* m : band.getList()) {
        band.delMusician(m);
    }

    return 0;
}
```