НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

З В І Т

до лабораторної роботи **№6**

з курсу: «Програмування, частина 2

(Об'єктно-орієнтоване програмування)»

на тему: «Спадкування»

Варіант **№8**

Виконав: *студент групи КІ-103*
Кобзєв Роман
Прийняв: Науличний В.В

Львів – 2024

Мета роботи: познайомитися із спадкуванням класів.

Завдання

Створити абстрактний базовий клас і похійдний від нього клас, які реалізують модель предметної області згідно варіанту. Кожен клас має мати мінімум 3 власні елементи даних один з яких створюється динамічно, методи встановлення і читання характеристик елементів-даних класу (Set і Get), та мінімум 2 абстрактні методи обробки даних і мінімум 2 методи обробки даних у похідному класі. Крім цього клас має містити перевантаження оператора присвоєння, конструкторів по замовчуванню і копіювання та віртуальний деструктор. Для розроблених класів реалізувати програму-драйвер, яка демонструє роботу класів.

№	*Завдання*
16	Абстрактний клас – Годинник Похідний клас - Будильник

Текст програми

```
#include <iostream>
using namespace std;

class Clock {
protected:
    int hours;
    int minutes;
    int *seconds;

public:
    Clock() : hours(0), minutes(0), seconds(new int(0)) {}
    Clock(int h, int m, int s) : hours(h), minutes(m), seconds(new int(s)) {}
    Clock(const Clock& other) : hours(other.hours), minutes(other.minutes),
seconds(new int(*other.seconds)) {}
    virtual ~Clock() { delete seconds; }

    void setHours(int h) { hours = h; }
    void setMinutes(int m) { minutes = m; }
    void setSeconds(int s) { *seconds = s; }
    int getHours() const { return hours; }
    int getMinutes() const { return minutes; }
    int getSeconds() const { return *seconds; }

    virtual void processData1() = 0;
    virtual void processData2() = 0;

    Clock& operator=(const Clock& other) {
        if (this != &other) {
```

```
            hours = other.hours;
            minutes = other.minutes;
            *seconds = *other.seconds;
        }
        return *this;
    }
};


class AlarmClock : public Clock {
private:
    bool alarmStatus;
    string alarmTone;

public:
    AlarmClock() : Clock(), alarmStatus(false), alarmTone("Default Tone") {}
    AlarmClock(int h, int m, int s, bool status, const string& tone) : Clock(h, m,
s), alarmStatus(status), alarmTone(tone) {}

    void setAlarmStatus(bool status) { alarmStatus = status; }
    void setAlarmTone(const string& tone) { alarmTone = tone; }
    bool getAlarmStatus() const { return alarmStatus; }
    string getAlarmTone() const { return alarmTone; }


    void processData1() override {
        cout << "Processing data 1 in AlarmClock" << endl;
    }

    void processData2() override {
        cout << "Processing data 2 in AlarmClock" << endl;
    }
};

int main() {

    /*Clock clock(10, 30, 45);
    Clock clock2(clock1);
    clock2.setHours(12);
    cout << "Clock 1: " << clock1.getHours() << ":" << clock1.getMinutes() << ":"
<< clock1.getSeconds() << endl;
    cout << "Clock 2: " << clock2.getHours() << ":" << clock2.getMinutes() << ":"
<< clock2.getSeconds() << endl;
*/
    AlarmClock alarm1(6, 0, 0, true, "Beep");
    AlarmClock alarm2(alarm1);
    alarm2.setAlarmStatus(false);
    cout << "AlarmClock 1: " << alarm1.getHours() << ":" << alarm1.getMinutes() <<
":" << alarm1.getSeconds() << ", Status: " << alarm1.getAlarmStatus() << ", Tone: "
<< alarm1.getAlarmTone() << endl;
    cout << "AlarmClock 2: " << alarm2.getHours() << ":" << alarm2.getMinutes() <<
":" << alarm2.getSeconds() << ", Status: " << alarm2.getAlarmStatus() << ", Tone: "
<< alarm2.getAlarmTone() << endl;
```

```
    return 0;
}
```

Висновок

Розробка абстрактного базового класу та похідного від нього класу дозволила створити гнучку та розширювану модель предметної області. Реалізація всіх вимог, зокрема абстрактних методів, динамічних елементів даних, перевантаження операторів та конструкторів, забезпечила коректну та ефективну роботу з об'єктами класів. Написана програма-драйвер продемонструвала функціональність розроблених класів, показавши можливості як статичного, так і динамічного поліморфізму.