Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



**Звіт**

З лабораторної роботи № 6

З дисципліни «Об’єктно-орієнтоване програмування»

**На тему**: «Спадкування»

**Виконав:** ст. гр. КІ-12

Кузьма І.В.

**Прийняла:** доцент

Пастернак І.І.

Львів 2020

**Мета:** познайомитися із спадкуванням класів.

**Теоретичні відомості**

*Спадкування* – це механізм, за допомогою якого один клас може одержувати атрибути та функціональність іншого. Спадкування дозволяє створювати ієрархію класів.

При створенні нового класу, що повністю дублює існуючий клас і дещо розширяє його новими властивостями і функціональністю програміст може не дублювати існуючий клас і дописувати в нього розширену функціональність, а вказати, що новий клас є спадкоємцем елементів попередньо визначеного класу і визначити у ньому лише необхідну нову функціональність. В цьому випадку існуючий клас, функціональність якого розширюється у новому класі, називається *базовим класом* (base class). Новостворений клас називається *похідним класом* (derived class), або *спадкоємцем*. Кожен похідний клас може бути використаним у ролі базового класу для майбутніх похідних класів створюючи при цьому *дерево спадкування*, яке ще називають *ієрархією спадкування класів* (class hierarchy). Спадкування прийнято відображати у вигляді графу (дерева) у напрямку зверху-вниз. При цьому клас, що є у самому верху є самим першим базовим класом і називається *кореневим класом* або *коренем дерева спадкування класів*. Похідний клас, через проміжний, може наслідувати характеристики базового класу. У цьому випадку говорять, що *базовий клас є непрямим базовим класом (indirect base class) для похідного*. Зокрема, корінь дерева наслідувань є непрямим базовим класом для усіх класів, які знаходяться нижче першого рівня ієрархії. Клас, який При *одиночному спадкуванні* (single inheritance) клас породжується одним базовим класом. При *множинному спадкуванні* (multiple inheritance) похідний клас успадковує властивості декількох базових класів, причому можлива ситуація коли один базовий клас буде успадкований кілька разів по кількох гілках. При створенні об'єкта похідного класу в пам'яті зберігаються копії усіх класів, які *становлять вітку, що породила даний клас*.

Не успадковуються:

1. Конструктор

2. Деструктор

3. Оператор присвоєння

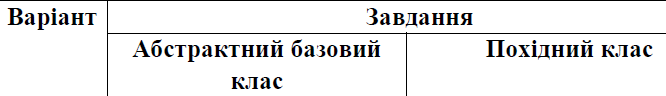
4. Дружні конструкції

5. Конструкції створені за допомогою оператора *new*.

При створенні об'єкту класу *автоматично викликається його конструктор*. Якщо об'єкт породжується від похідного класу, то, очевидно, при створенні об‘єкту похідного класу повинен викликатись і конструктор базового класу. Деструктори викликаються подібно до конструкторів, а виконуються у зворотному порядку.

**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.



|  |  |  |
| --- | --- | --- |
| 14 | Пристрій | Телевізор |

**Код програми:**

**Main**

#include <iostream>

#include"Header.h"

#include"Header1.h"

using namespace std;

int main()

{

Tv tv("L43",15);

tv.Info();

cout << endl << endl;

tv.Tv\_Info();

cout << endl << endl;

tv.Set(25);

tv.Tv\_Info();

cout << endl;

tv.Chek\_\_Frequnce();

Tv tv1("Z43", 45);

Tv tv2=tv;

cout << endl;

tv2.Tv\_Info();

cout << endl;

system("pause");

return 0;

}

**Header.h**

#pragma once

#include <iostream>

#include <time.h>

using namespace std;

class Device

{

private:

string name;

int Discount, Rozmir;

int\* Device\_Frequency;

public:

Device();

~Device();

virtual void Set(int disk)=0;

virtual int Get()=0;

virtual bool Chek\_\_Frequnce() = 0;

virtual int Ret\_\_Frequnce(int i);

virtual void Frequnce();

virtual void Info();

};

**Header.ccp**

#include "Header.h"

Device::Device()

{

srand(time(0));

name = "Device";

Discount = 10;

Rozmir = 10;

Device\_Frequency = new int[Rozmir];

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

{

Device\_Frequency[i] = rand() % 100;

}

}

Device::~Device()

{

if (Device\_Frequency != 0)

{

delete[]Device\_Frequency;

}

}

int Device::Ret\_\_Frequnce(int i)

{

return Device\_Frequency[i];

}

void Device::Frequnce()

{

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

{

cout << Device\_Frequency[i] << " ";

}

}

void Device::Info()

{

cout << "Name: " << name << endl << "Discount: " << Discount << endl<<"Frequence: ";

Frequnce();

}

**Header1.h**

#pragma once

#include <iostream>

#include <time.h>

#include <iomanip>

#include "Header.h"

using namespace std;

class Tv : public Device

{

private:

string Model;

int Discount, Rozmir;

int\* TV\_Frequencys;

friend Tv operator+(const Tv& tv, const Tv& tv2);

public:

Tv(string st, int st1)

{

Model = st;

Discount = st1;

Rozmir = 10;

TV\_Frequencys = new int[Rozmir];

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

{

TV\_Frequencys[i] = rand() % 100;

}

}

~Tv()

{

if (TV\_Frequencys != 0)

{

delete[]TV\_Frequencys;

}

}

void Set(int disk) override

{

Discount = disk;

}

int Get() override

{

return Discount;

}

bool Chek\_\_Frequnce() override

{

bool chek;

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

{

if (TV\_Frequencys[i] != Ret\_\_Frequnce(i))chek = true;

else

{

chek = false;

goto Eror;

}

}

cout << "Frequences OK" << endl;

return false;

return true;

Eror:

cout << "Frequences have a problem" << endl;

return false;

}

void Tv\_Info()

{

cout << "Model: " << Model << endl << "Discount: " << Discount << endl << "Tv Frequence: ";

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

{

cout << TV\_Frequencys[i] << " ";

}

}

Tv operator+(Tv tv)

{

tv.Discount += Discount;

tv.Model += Model;

for (int i = 0; i < tv.Rozmir; i++)

{

tv.TV\_Frequencys[i] += TV\_Frequencys[i];

}

return tv;

}

void operator=(const Tv& tv1)

{

Discount = tv1.Discount;

Model = tv1.Model;

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

{

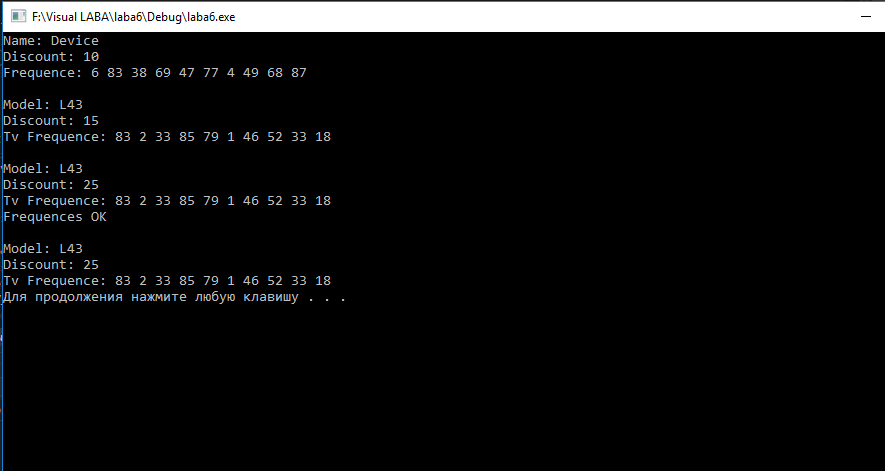
TV\_Frequencys[i] = tv1.TV\_Frequencys[i];

}

}

};

**Результат програми:**



**Висновок:** Під час виконання цієї лабораторної роботи я познайомився із спадкуванням класів.