Міністерство освіти і науки України

Національний університет “Львівська Політехніка”

Кафедра ЕОМ



**ЗВІТ**

З лабораторної роботи № 6

З дисципліни: “Програмування, частина 2 (Об'єктно-орієнтоване програмування)”

На тему: “СПАДКУВАННЯ”

Варіант 18

Виконав: ст. гр. КІ-15

Марков Л.Ю.

Прийняв:

Козак Н.Б.

Львів – 2020

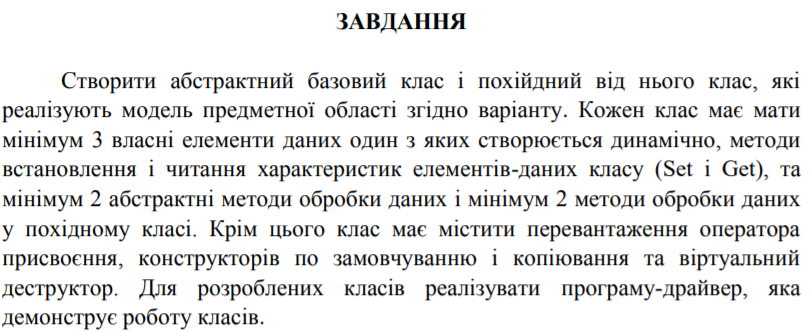
Мета: познайомитися із спадкуванням класів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Спадкування (ієрархія "іs a")

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.

Варіант 18:





Виконання:

#include <iostream>

#include <string>

using namespace std;

class CDevice {

protected:

string name;

bool power;

int\* cost = new int;

public:

CDevice() {

this->name = "Unknown";

this->power = true;

\*cost = 0;

}

CDevice(string name, bool power, int Cost) {

this->name = name;

this->power = power;

\*cost = Cost;

}

void setAll(string name, bool power, int Cost) {

this->name = name;

this->power = power;

\*cost = Cost;

}

void setPower(bool power) {

this->power = power;

}

void setName(string name) {

this->name = name;

}

void getAll() {

cout << "Device model: " << name << endl <<

"Device is ";

if (power == true)

cout << "on." << endl;

else

cout << "off" << endl;

cout << "Device cost: " << \*cost << endl;

}

virtual void print() {};

virtual void copy() {};

~CDevice() {

delete cost;

cout << "Object destoyed" << endl;

}

};

class CCopyDevice : public CDevice {

private:

int\* paperLeft = new int,

copySpeed,

printSpeed;

public:

CCopyDevice() {

this->name = "Unknown";

this->power = true;

\*cost = 0;

\*paperLeft = 0;

this->copySpeed = 0;

this->printSpeed = 0;

}

CCopyDevice(string name, bool power, int Cost, int PaperLeft, int copySpeed, int printSpeed) {

this->name = name;

this->power = power;

\*cost = Cost;

\*paperLeft = PaperLeft;

this->copySpeed = copySpeed;

this->printSpeed = printSpeed;

}

void setAll(string name, bool power, int Cost, int PaperLeft, int copySpeed, int printSpeed) {

this->name = name;

this->power = power;

\*cost = Cost;

\*paperLeft = PaperLeft;

this->copySpeed = copySpeed;

this->printSpeed = printSpeed;

}

void getAll() {

cout << "Device model: " << name << endl <<

"Device is ";

if (power == true)

cout << "on." << endl;

else

cout << "off" << endl;

cout << "Device cost: " << \*cost << endl <<

"Paper left: " << \*paperLeft << endl <<

"Copy speed: " << copySpeed << endl <<

"Print speed: " << printSpeed << endl;

}

void setCopySpeed(int copySpeed) {

this->copySpeed = copySpeed;

}

void setPrintSpeed(int printSpeed) {

this->printSpeed = printSpeed;

}

void print() override {

if (\*paperLeft != 0) {

\*paperLeft -= 1;

cout << "Printing picture..." << endl <<

"Paper Left: " << \*paperLeft << endl;

}

else

cout << "Not enough paper left." << endl;

}

void copy() override {

if (\*paperLeft != 0) {

\*paperLeft -= 1;

cout << "Copying picture..." << endl <<

"Paper Left: " << \*paperLeft << endl;

}

else

cout << "Not enough paper left." << endl;

}

CCopyDevice(const CCopyDevice& other) {

this->paperLeft = other.paperLeft;

this->copySpeed = other.copySpeed;

this->printSpeed = other.printSpeed;

}

void operator = (const CCopyDevice& other) {

if (paperLeft != 0) {

delete paperLeft;

}

this->paperLeft = other.paperLeft;

this->copySpeed = other.copySpeed;

this->printSpeed = other.printSpeed;

}

~CCopyDevice() {

delete paperLeft;

cout << "Object destroyed" << endl;

}

};

int main() {

CCopyDevice obj("Sony 2000", true, 1850, 50, 38, 30);

obj.getAll();

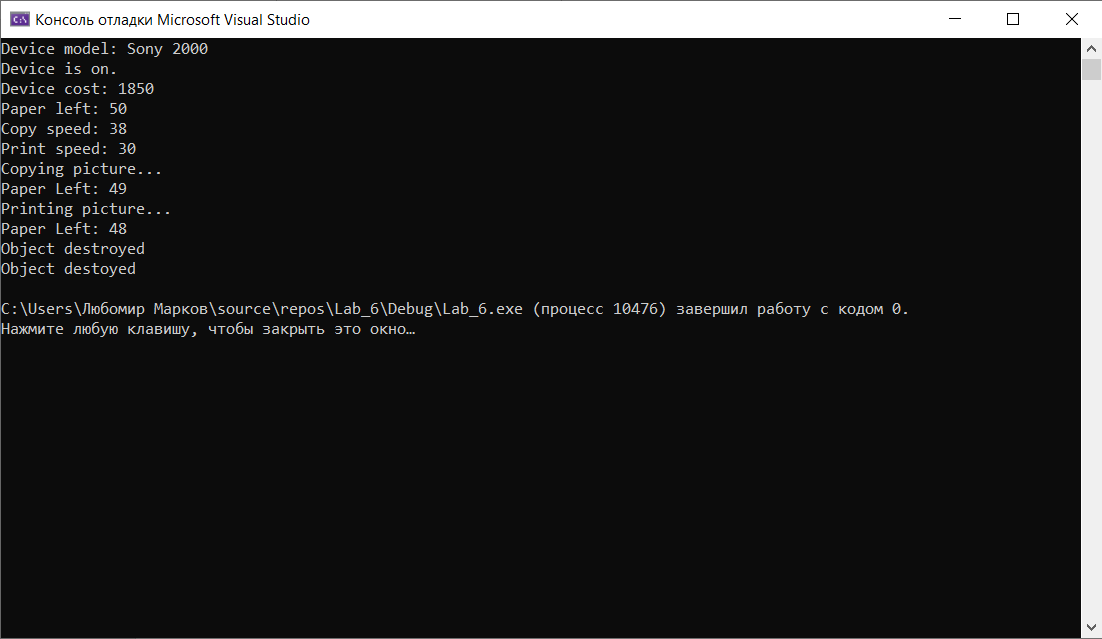
obj.copy();

obj.print();

return 0;

}

**Результат:**



Висновок: на цій лабораторній роботі я познайомився із спадкуванням класів.