Міністерство освіти і науки

Національний університет “Львівська політехніка”

**Кафедра ЕОМ**



**Звіт**

з лабораторної роботи № 6

# з дисципліни: “ Програмування, частина 2 (Об'єктно-орієнтоване програмування”

на тему: “ СПАДКУВАННЯ”

Виконав: ст. гр. КІ-15 Гвоздь Ю. А.

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. Похідний клас наслідує характеристики базового через специфікатор доступу (acces specifier) : "довкрапка". За допомогою специфікатора доступу можна визначити, яким чином елементи базового класу будуть успадковуватися похідним класом. При відкритому спадкуванні (використанні специфікатора publіc) у похідному класі члени базового класу мають ті ж специфікатори доступу, що й у базовому класі. При захищеному спадкуванні (використанні специфікатора protected) у похідному класі відкриті члени базового класу стають захищеними, а інші зберігають своє початкове значення специфікатора доступу. Нарешті, при закритому спадкуванні (використанні специфікатора prіvate) у похідному класі всі члени базового класу стають закритими. Кожного разу, коли з батьківського класу створюється похідний клас, то похідний клас успадковує окремі або усі риси батьківського, додаючи свої або розширюючи батьківські. Усі спадковані характеристики похідний клас використовує як свої власні. Кінцевий користувач класу не розрізняє успадкованих характеристик і маніпулює ними за допомогою оператора (.) як звичайними членами об'єкту. Об'єкт похідного класу оперує з власними та наслідуваними характеристиками через оператор (::) без жодних обмежень, згідно з правами доступу до них. Якщо є потреба явно вказати клас, якому належить даний член, то використовують оператор (::) з іменем класу. Не успадковуються: 1. Конструктор 2. Деструктор 3. Оператор присвоєння 4. Дружні конструкції 5. Конструкції створені за допомогою оператора new.

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.



Виконання завдання:

Код програми:

***//Source.cpp***

#include "Laptop.h"

int main()

{

CLaptop laptop("ASUS", "Vivobook 14");

laptop.Boot();

laptop.OpenLid();

laptop.Boot();

laptop.Execute("ls -l");

laptop.SetLatitude(14.0024);

laptop.GetLatitude();

laptop.SetLongitude(24.3250);

laptop.GetLongitude();

laptop.Shutdown();

laptop.CloseLid();

}

***//Computer.h***

#include <string.h>

#include <string>

using namespace std;

class CComputer

{

protected:

string m\_pszBrand;

string m\_pszModel;

bool m\_bIsBooted;

public:

CComputer(string brand, string model);

virtual ~CComputer();

virtual bool Boot() = 0;

virtual bool Shutdown() = 0;

virtual bool Execute(string command) = 0;

};

***//Laptop.h***

#include "Computer.h"

#include <iostream>

class CLaptop

: public CComputer

{

private:

bool m\_bLidIsOpen;

double m\_dblLatitude;

double m\_dblLongitude;

public:

CLaptop(string brand, string model);

virtual ~CLaptop();

virtual bool Boot();

virtual bool Shutdown();

virtual bool OpenLid();

virtual bool CloseLid();

virtual bool Execute(string command);

virtual bool SetLatitude(double latitude);

virtual double GetLatitude();

virtual bool SetLongitude(double longitude);

virtual double GetLongitude();

};

***//Computer.cpp***

#include "Computer.h"

CComputer::CComputer(string brand, string model)

{

this->m\_pszBrand = brand;

this->m\_pszModel = model;

this->m\_bIsBooted = false;

}

CComputer::~CComputer()

{

}

***//Laptop.cpp***

#include "Laptop.h"

CLaptop::CLaptop(string brand, string model)

: CComputer(brand, model)

{

this->m\_bLidIsOpen = 0;

this->m\_dblLatitude = 0.0;

this->m\_dblLongitude = 0.0;

}

CLaptop::~CLaptop()

{

}

bool CLaptop::Boot()

{

if (m\_bIsBooted)

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Already booted" << endl;

return false;

}

else if (!m\_bLidIsOpen)

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Lid is not openned" << endl;

return false;

}

else

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Booting...";

this->m\_bIsBooted = true;

cout << "done." << endl;

return true;

}

}

bool CLaptop::Shutdown()

{

if (!m\_bIsBooted)

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Alredy shutdowned.";

return false;

}

else

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Shutdowning...";

this->m\_bIsBooted = false;

cout << "done" << endl;

return true;

}

}

bool CLaptop::OpenLid()

{

if (m\_bLidIsOpen)

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Already opened." << endl;

return false;

}

else

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Opening...";

this->m\_bLidIsOpen = true;

cout << "done" << endl;

return true;

}

}

bool CLaptop::CloseLid()

{

if (!m\_bLidIsOpen)

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Already closed." << endl;

return false;

}

else

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Closing...";

this->m\_bLidIsOpen = false;

cout << "done." << endl;

return true;

}

}

bool CLaptop::Execute(string command)

{

if (m\_bLidIsOpen && m\_bIsBooted)

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Executing [" << command << "]...";

cout << "done." << endl;

return true;

}

else

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Error. Not possible to execute the command." << endl;

return false;

}

}

bool CLaptop::SetLatitude(double latitude)

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Setting latitude...";

this->m\_dblLatitude = latitude;

cout << "done." << endl;

return true;

}

double CLaptop::GetLatitude()

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Latitude: " << m\_dblLatitude << endl;

return true;

}

bool CLaptop::SetLongitude(double longitude)

{

cout << m\_pszBrand << "@" << m\_pszModel << ": ";

cout << "Setting longitude...";

this->m\_dblLongitude = longitude;

cout << "done" << endl;

return true;

}

double CLaptop::GetLongitude()

{

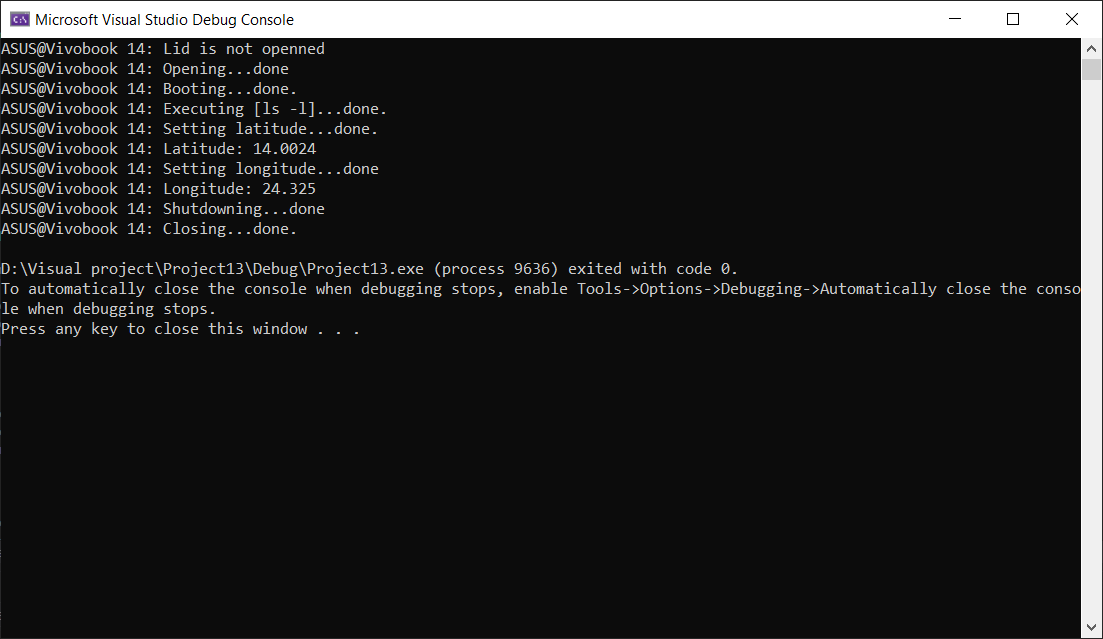
cout << m\_pszBrand << "@" << m\_pszModel <<": " ;

cout << "Longitude: " << m\_dblLongitude << endl;

return true;

}

Результат роботи програми:



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