 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Практичне заняття №3**

з дисципліни «Об’єктно-орієнтоване програмування»

на тему: «Успадкування,поліморфізм»

Варіант №11

**Виконав:**

студент гр. БС-52

Розюк П.В.

**Перевірив:**

ст. викладач каф. БМК

Радогуз А.В.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2017

**Завдання:**

1.Вивчити теоретичні теоретичні основи визначення класів, що використовують спадкування властивостей.

2. Відповідно до свого варіанту завдання розробити діаграму класів в нотації UML та написати програму.

3.Розробити програмний додаток з використанням успадкування, що реалізує ієрархій класів відповідно до свого варіанту: **Базовий клас**: лікарський засіб (назва,ціна). **Нащадки:** таблетки (кількість в упаковці), мікстура (об’єм). **Завдання**: використовуючи віртуальні функції, не знаючи з об’єктом якого класу ви працюєте, виведіть на екран назву та ціну лікарського засобу.

4.Скласти і захистити звіт. **🞏 Практичне заняття без зауважень** **🞏 Практичне заняття має зауваження:**

**🞏 несвоєчасний захист**

**🞏 присутні зауваження до UML діаграми:**

**🞏 діаграма класу не відповідає коду**

**🞏 виконані не за стандартом:**

**🞏 атрибути**

**🞏 відношення**

**🞏 багатостатність**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 присутні зауваження до коду:**

**🞏 задача завдання вирішена хибно**

**🞏 код програми не компілюється**

**🞏 хибно задані специфікатори доступу**

**🞏 помилки у визначенні конструкторів / деструкторів**

**🞏 використано глобальні змінні**

**🞏 статичні змінні при роботі з масивами**

**🞏 оформлення коду**

**🞏 присутні зайві символи «{» та «}»**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 невірні відповіді на запитання:**

**🞏 №1 🞏 №2 🞏 №3 🞏 №4 🞏 №5**

**🞏 №6 🞏 №7 🞏 №8 🞏 №9 🞏 №10**

**🞏 маються інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Лістинг програми:**

**Main:**

#include "drug.h"

#include "mixture.h"

#include "tablets.h"

#include <iostream>

#include <conio.h>

#include <string>

using namespace std;

void input(string \*name, int \*price)

{

cout << "Name: ";

cin >> \*name;

cout << "Price: ";

cin >> \*price;

}

void objects(drug \*obj[], int n)

{

string name;

int price, number, size, a;

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

{

cout << "Do you want input data of tablets-1, or mixture-2: ";

cin >> a;

switch (a)

{

case 1:

{

input(&name, &price);

cout << "Number: ";

cin >> number;

obj[i] = new tablets(number, name, price);

break;

}

case 2:

{

input(&name, &price);

cout << "Size: ";

cin >> size;

obj[i] = new mixture(size, name, price);

break;

}

}

}

}

void main(int argc, const char \* argv[])

{

const int n = 2;

drug \*obj[n];

objects(obj, n);

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

obj[i]->show();

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

delete obj[i];

\_getch();

}

drug.h:

#pragma once

#include <iostream>

using namespace std;

class drug

{

protected:

string name;

int price;

public:

drug() {};

drug(string cur\_name, int cur\_price);

virtual void show();

~drug();

};

drug.cpp:

#include "drug.h"

#include <iostream>

#include <string.h>

#include <string>

using namespace std;

drug::drug(string cur\_name, int cur\_price)

{

this->name = cur\_name;

this->price =cur\_price;

}

void drug::show()

{

cout << "\n Medical drug:"<<endl;

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

cout << "Price: " << price << endl;

}

drug::~drug()

{

}

tablets.h:

#pragma once

#include "drug.h"

class tablets :public drug

{

protected:

int number;

public:

tablets() {};

tablets(int cur\_number, string name, int price) :drug(name, price)

{

this->number = cur\_number;

};

void show();

~tablets();

};

tablets.cpp:

#include "tablets.h"

#include "drug.h"

#include <string>

#include <iostream>

using namespace std;

tablets::~tablets()

{};

void tablets::show()

{

cout << "\nMedical drug:" << endl;

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

cout << "Number: " << number << endl;

cout << "Price: " << price << endl;

}

mixture.h:

#pragma once

#include "drug.h"

class mixture:public drug

{

protected:

int size;

public:

mixture() {};

mixture(int cur\_size, string name, int price) :drug(name, price)

{

this->size = cur\_size;

}

void show();

~mixture() {};

};

mixture.cpp:

#include "mixture.h"

#include "drug.h"

#include <string>

void mixture::show()

{

cout << "\nMedical drug: " << endl;

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

cout << "Size: " << size<< endl;

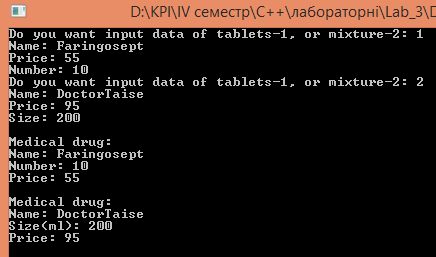
cout << "Price: " << price << endl;

}

UML- діаграма:



Тест роботи програми:



Контрольні запитання:

1. Успадкування – властивість, за допомогою якої об’єкт одного класу(нащадок), може набувати властивостей об’єкту іншого класу(базовий). За рахунок цього ми досягаємо однієї з цілей ООП – повторне використання коду. Успадкування дозволяє додавати нові можливості, додавти данні, які представляє клас, змінювати поведінку метоів класу.

2. Який клас називається базовим, а який нащадком?

Батьківський-той від якого успадковується, нащадок-той що спадкує.

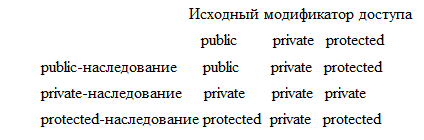
3. Скільки базових класів може мати похідний клас?

Безліч

4. Чи може похідний клас бути базовим?

Так

5. Як змінюється доступ до елементів базового класу при спадкуванні з різними специфікаторами доступу: з розділів класу, із програми, з інших класів?



6. У чому різниця між простим і множинним спадкуванням? Просте спадкування – в базового класу є тільки один нащадок. Множинне – в базового класу більше ніж один нащадок.

7. Чи можна з класу-нащадка одержати доступ до private частини базового класу, якщо специфікатор доступу при спадкуванні private? Ні, не можна.

8. Чи успадковуються конструктори, деструктори? Так, класи нащадки викликають спочатку конструктор базового класу, потім власні. З деструкторами навпаки, спочатку власний, потім базовий.

9. Що таке віртуальна функція? Віртуальна функція – це метод, який ми маємо перевизначити в класах нащадках. При посиланні на об’єкт нащадках за допомогою вказінвика, що маж тип базового класу ми можемо викликати віртуальну функцію і виконати версію цієї функції для нащадка

10. Чи можна віртуальну функцію визначити як static, відповідь пояснити? Ні, тоді реалізація буде однаковою, як для базового, так і для нащадків.