Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 9**

*«Принцип поліморфізму»*

**з дисципліни «***Об’єктно–орієнтоване програмування***»**

**Лектор:**

ст. викладач кафедри ПЗ

Коротєєва Т.О.

**Виконав:**

студ. групи ПЗ-11

Ільчишин О.Ю.

**Прийняла:**

асистент кафедри ПЗ

Терендій О.В.

«\_\_\_» \_\_\_\_\_\_ 2019 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2019

**Тема роботи:** Принцип поліморфізму

**Мета роботи:** Навчитись створювати списки об’єктів базового типу, що включають об’єкти похідних типів. Освоїти способи вирішення проблеми неоднозначності при множинному наслідуванні. Вивчити плюси заміщення функцій при множинному наслідуванні. Навчитись використовувати чисті віртуальні функції, знати коли варто використовувати абстрактні класи.

**Теоретичні відомості**

Поліморфізм - одна з трьох основних парадигм ООП. Якщо говорити коротко, поліморфізм – це здатність об'єкта використовувати методи похідного класу, який не існує на момент створення базового.

Уявімо ситуацію, що ми створюємо програму, в якій ми працюємо з класами тварин. Один з цих класів Bird(птахи), а другий Mammal(ссавці). Клас Bird містить функцію Fly(), а клас Mammal містить функцію Gallop() – біг галопом.

А тепер сталось так, що нам потрібно створити новий міфічний персонаж – крилатого Пегаса (Pegasus), який буде гібридом між птахом та ссавцем. В таких випадках використовують множинне наслідування.

Тепер ми зможемо пройтись по списку птахів та в кожного елемента списку викликати метод Fly() а в списку ссавців відповідно метод Gallop(). Тут ми і використовуємо принцип поліморфізму.

Для того щоб викликати функцію з класу Pegasus нам потрібно привести вказівник базового типу до похідного типу. Для цього використовуємо функцію dynamic\_cast. Якщо вказівник не вдалось привести до похідного типу функція верне нуль.

Заміщення функції вирішує дві проблеми:

1. Зникає невизначеність звертання до базових класів;
2. Функцію можна замістити таким чином, що в похідному класі при виклику цієї функції викликати функцію з базового класу.

Клас, який містить чисті віртуальні функції є абстрактним. Неможливо створити об’єкт абстрактного класу. Поміщення в клас чистої віртуальної функції означає наступне:

- неможливо створити об’єкт цього класу;

- необхідно замістити чисту віртуальну функцію в похідному класі.

Будь-який клас, наслідуваний від абстрактного класу, наслідує від нього чисту віртуальну функцію, яку необхідно замістити щоб отримати можливість створювати об’єкти цього класу.

**Індивідуальне завдання**

1. Розробити ієрархію класів відповідно до варіанту з попередньої лабораторної.

Створити базовий клас ДРУКОВАНА ПРОДУКЦІЯ (задаються назва, кількість сторінок).

Створити похідні класи КНИГА (задаються дані про автора, розмір сторінки, кількість

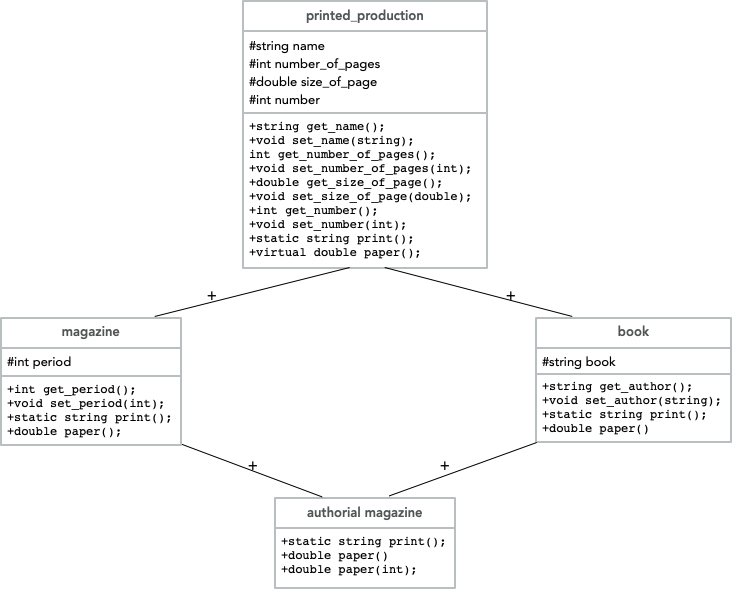
примірників) та ЖУРНАЛ (задаються розмір сторінки, кількість примірників,

періодичність виходу). Для введених даних про друковану продукцію обчислити витратипаперу (у кв. м.) – для кожного прикладу.

1. Використати множинне наслідування, продемонструвати вирішення проблеми з неоднозначністю доступу до членів базових класів за допомогою віртуального наслідування, за допомогою явного звертання до членів класу та за допомогою заміщення функцій в похідному класі.
2. Створити абстрактний клас, використати чисто віртуальну функцію, що містить реалізацію в базовому класі.
3. Для вивільнення динамічної пам’яті використовувати віртуальні деструктори.
4. Створити масив розмірністю 3 базового типу, в який помістити об’єкти похідного типу. Продемонструвати виклик функцій з об’єктів – елементів масиву. Використати оператор dynamic\_cast.
5. Сформувати звіт до лабораторної роботи. Відобразити в ньому UML-діаграму наслідування класів.

**Хід роботи**

UML діаграма класів:



Код програми:

**class.h**

#ifndef CLASS\_H

#define CLASS\_H

#include <iostream>

#include <string>

#include <utility>

*using* *namespace* std;

*class* **printed\_production** {

*protected*:

string name;

int number\_of\_pages;

double size\_of\_page;

int number;

*public*:

**printed\_production**() {};

*virtual* ~***printed\_production***() = *default*;;

**printed\_production**(string name, int number\_of\_pages, double size\_of\_pages, int number) {

*this*->name = std::move(*name*);

*this*->number\_of\_pages = number\_of\_pages;

*this*->size\_of\_page = size\_of\_pages;

*this*->number = number;

};

string **get\_name**();

void **set\_name**(string name1);

int **get\_number\_of\_pages**();

void **set\_number\_of\_pages**(int number\_of\_pages1);

double **get\_size\_of\_page**();

void **set\_size\_of\_page**(double size\_of\_page1);

int **get\_number**();

void **set\_number**(int number1);

*static* string **print**();

*virtual* double ***paper***() = 0;

};

*class* **magazine** : *virtual* *public* printed\_production {

*protected*:

int period;

*public*:

**magazine**() : printed\_production() {};

~***magazine***() *override* = *default*;;

**magazine**(string name, int number\_of\_pages, double size\_of\_page, int number, int period) :

printed\_production(std::move(*name*), number\_of\_pages, size\_of\_page, number) {

*this*->period = period;

};

int **get\_period**();

void **set\_period**(int period1);

*static* string **print**();

double ***paper***() *override*;

};

*class* **book** : *virtual* *public* printed\_production {

*protected*:

string author;

*public*:

**book**() : printed\_production() {};

~***book***() *override* = *default*;;

**book**(string name, int number\_of\_pages, double size\_of\_page, int number, string author) :

printed\_production(std::move(*name*), number\_of\_pages, size\_of\_page, number) {

*this*->author = std::move(*author*);

};

string **get\_author**();

void **set\_author**(string author);

*static* string **print**();

double ***paper***() *override*;

};

*class* **authorial\_magazine** : *public* book, *public* magazine {

*public*:

**authorial\_magazine**() : printed\_production(){};

~***authorial\_magazine***() *override* = *default*;;

**authorial\_magazine**(*const* string &name, int number\_of\_pages, double size\_of\_page, int number, string author,

int period) : printed\_production(name, number\_of\_pages, size\_of\_page, number) {

*this*->author = std::move(*author*);

*this*->period = period;

}

*static* string **print**();

double ***paper***() *override*;

double **paper**(int days);

};

#endif *//* *CLASS\_H*

**mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "class.h"

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **MainWindow**; }

QT\_END\_NAMESPACE

*class* **MainWindow** : *public* QMainWindow

{

Q\_OBJECT

*public*:

**MainWindow**(QWidget \*parent = *nullptr* );

~***MainWindow***();

*private* slots:

void **on\_book\_set\_clicked**();

void **on\_magazine\_set\_clicked**();

void **on\_am\_set\_clicked**();

void **on\_book\_paper\_clicked**();

void **on\_magazine\_paper\_clicked**();

void **on\_am\_paper\_clicked**();

void **on\_am\_paperd\_clicked**();

*private*:

Ui::MainWindow \*ui;

};

#endif *//* *MAINWINDOW\_H*

**class.cpp**

#include "class.h"

string printed\_production::**get\_name**() {

*return* *this*->name;

}

void printed\_production::**set\_name**(string name1) {

*this*->name = std::move(*name1*);

}

int printed\_production::**get\_number\_of\_pages**() {

*return* *this*->number\_of\_pages;

}

void printed\_production::**set\_number\_of\_pages**(int number\_of\_pages1) {

*this*->number\_of\_pages = number\_of\_pages1;

}

double printed\_production::**get\_size\_of\_page**() {

*return* *this*->size\_of\_page;

}

void printed\_production::**set\_size\_of\_page**(double size\_of\_page1) {

*this*->size\_of\_page = size\_of\_page1;

}

int printed\_production::**get\_number**() {

*return* *this*->number;

}

void printed\_production::**set\_number**(int number1) {

*this*->number = number1;

}

string printed\_production::**print**() {

*return* "printed\_production";

}

*//----------------------------------------------------------------------------------------------------------------------*

int magazine::**get\_period**() {

*return* period;

}

void magazine::**set\_period**(int period1) {

*this*->period = period1;

}

string magazine::**print**() {

*return* "magazine";

}

double magazine::***paper***() {

*return* number \* number\_of\_pages \* size\_of\_page;

}

*//----------------------------------------------------------------------------------------------------------------------*

string book::**get\_author**() {

*return* *this*->author;

}

void book::**set\_author**(string author1) {

*this*->author = std::move(*author1*);

}

string book::**print**() {

*return* "book";

}

double book::***paper***() {

*return* number \* number\_of\_pages \* size\_of\_page;

}

*//----------------------------------------------------------------------------------------------------------------------*

string authorial\_magazine::**print**() {

*return* "authorial\_magazine";

}

double authorial\_magazine::***paper***() {

*return* number \* number\_of\_pages \* size\_of\_page;

}

double authorial\_magazine::**paper**(int days) {

*return* (number \* number\_of\_pages \* size\_of\_page \* ((int) (days / period) + 1));

}

**main.cpp**

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(*argc*, argv);

MainWindow w;

w.show();

*return* a.exec();

}

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

book temp1;

magazine temp2;

authorial\_magazine temp3;

printed\_production \*arr[3]={&temp1, &temp2, &temp3};

*auto* ptemp1 = *dynamic\_cast*<book\*>(arr[0]);

*auto* ptemp2 = *dynamic\_cast*<magazine\*>(arr[1]);

*auto* ptemp3 = *dynamic\_cast*<authorial\_magazine\*>(arr[2]);

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(parent)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

void MainWindow::**on\_book\_set\_clicked**()

{

ptemp1->set\_name(ui->book\_name->toPlainText().toStdString());

ptemp1->set\_number\_of\_pages(ui->book\_number\_of\_pages->toPlainText().toInt());

ptemp1->set\_size\_of\_page(ui->book\_\_size\_of\_page->toPlainText().toDouble());

ptemp1->set\_number(ui->book\_number->toPlainText().toInt());

ptemp1->set\_author(ui->book\_author->toPlainText().toStdString());

}

void MainWindow::**on\_magazine\_set\_clicked**()

{

ptemp2->set\_name(ui->magazine\_name->toPlainText().toStdString());

ptemp2->set\_number\_of\_pages(ui->magazine\_number\_of\_pages->toPlainText().toInt());

ptemp2->set\_size\_of\_page(ui->magazine\_size\_of\_page->toPlainText().toDouble());

ptemp2->set\_number(ui->magazine\_number->toPlainText().toInt());

ptemp2->set\_period(ui->magazine\_period->toPlainText().toInt());

}

void MainWindow::**on\_am\_set\_clicked**()

{

ptemp3->set\_name(ui->am\_name->toPlainText().toStdString());

ptemp3->set\_number\_of\_pages(ui->am\_number\_of\_pages->toPlainText().toInt());

ptemp3->set\_size\_of\_page(ui->am\_size\_of\_page->toPlainText().toDouble());

ptemp3->set\_number(ui->am\_number->toPlainText().toInt());

ptemp3->set\_author(ui->am\_author->toPlainText().toStdString());

ptemp3->set\_period(ui->am\_period->toPlainText().toInt());

}

void MainWindow::**on\_book\_paper\_clicked**()

{

ui->book\_res->setNum(ptemp1->*paper*());

}

void MainWindow::**on\_magazine\_paper\_clicked**()

{

ui->magazine\_res->setNum(ptemp2->*paper*());

}

void MainWindow::**on\_am\_paper\_clicked**()

{

ui->am\_res1->setNum(ptemp3->*paper*());

}

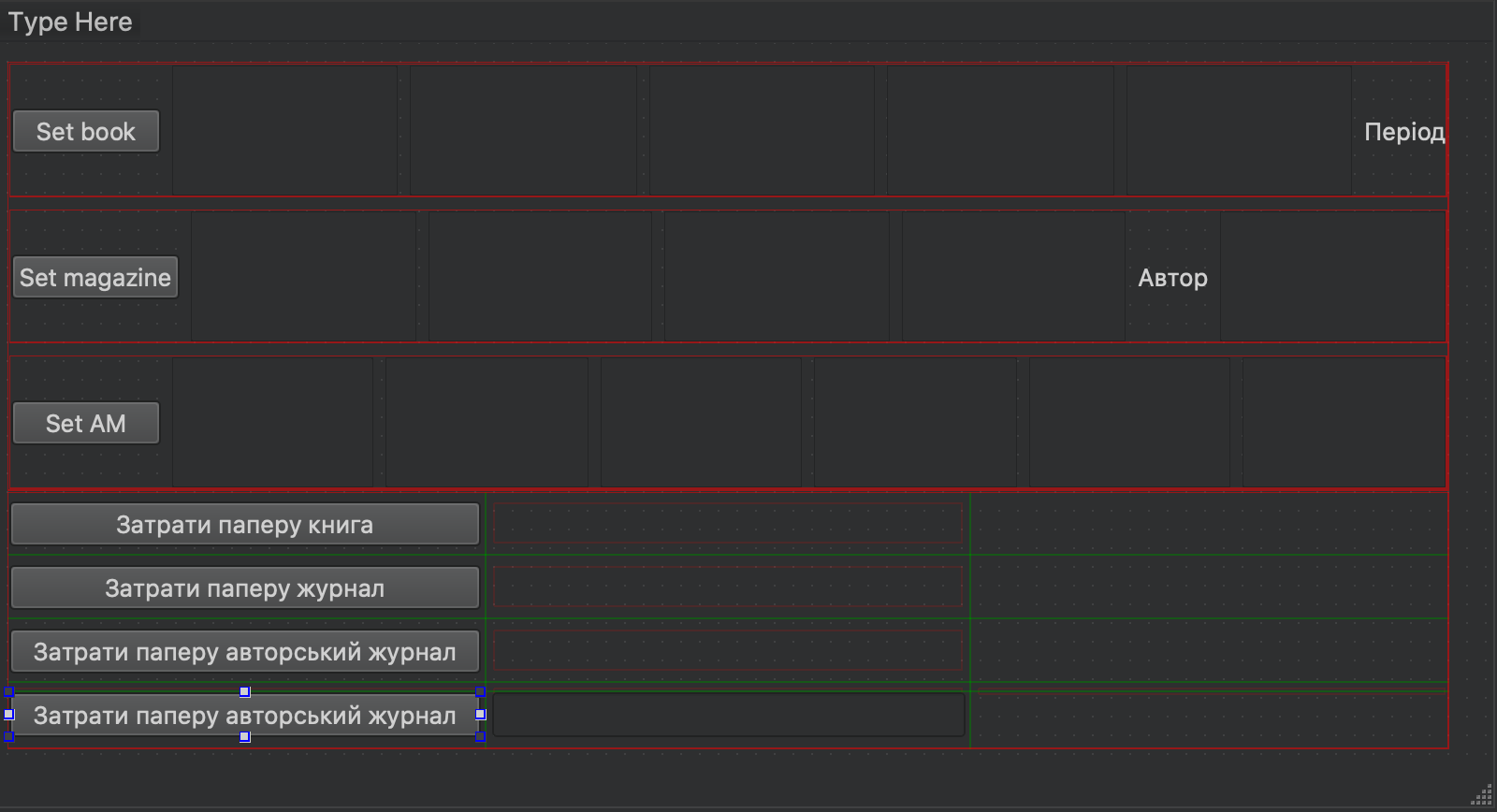
void MainWindow::**on\_am\_paperd\_clicked**()

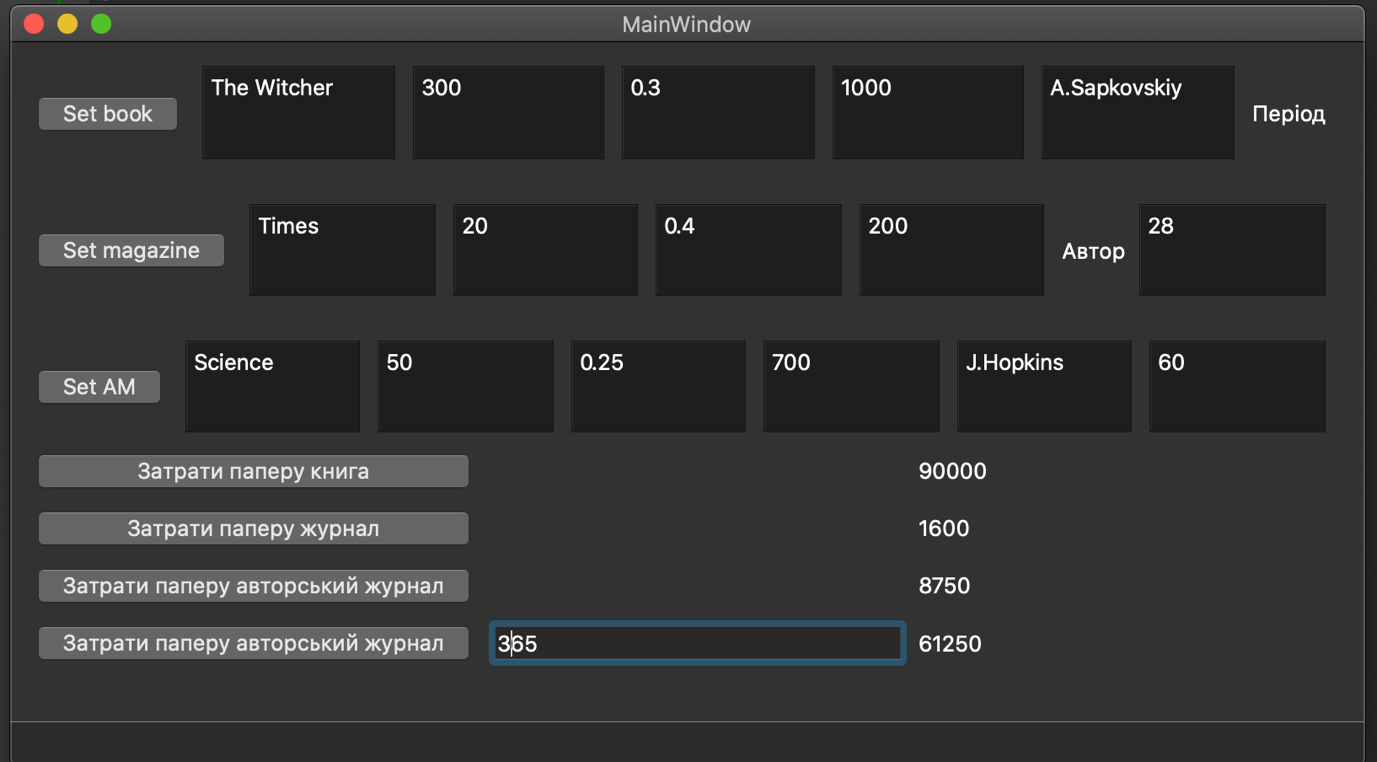
{

ui->am\_res2->setNum(ptemp3->paper(ui->days->text().toInt()));

}

**Вигляд форми та застосунку :**





**Висновки**

Під час виконання лабораторної роботи я розробив ієрархію класів відповідно варіанту(Абонемент в аквапарк). Також я дізнався про віртуальні функції, зокрема чисті, особливості їх оголошення та про інтерфейсні класи. Продемонстрував та вирішив проблеми з множинним та віртуальним наслідуванням.