МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет

«Харківський політехнічний інститут»

Факультет Комп’ютерних наук та програмної інженерії

Кафедра Інформатики та інтелектуальної власності

ЗВІТ

До лабораторной роботи №3 з дисципліни

«Об’єкто-орієнтоване програмування»

Студент Бородай Д.А

Викладач Івашко А.В.

Харків 2022

**Лабораторна робота №3**

**Тема: «Спадкування класів, абстрактні класи, інтерфейси, поліморфізм»**

**Мета роботи:** навчитись реалізовувати спадкування класів, використовувати абстрактні класи та інтерфейси під час спадкування, застосовувати принцип поліморфізму після спадкування.

**Завдання:**

1. Для класу за власним варіантом з першої лабораторної створити абстрактний клас чи інтерфейс, який має стати предком, зв’язати ці класи відносинами спадкування. Наприклад для класу Book (Книга) можна створити клас Reader (Засіб читання).

2. Окрім вашого класу додати до абстрактного класу чи інтерфейсу, створеному у попередньому пункті, ще два класи спадкоємця першого рівня. Наприклад до класу Book (Книга) можна додати класи Newspaper (Газета) Magazine (Журнал) з відповідними полями та методами.

3. Для одного з доданих у попередньому завданні класів створити не менше двох спадкоємців другого рівня спадкування. Наприклад до класу Magazine (Журнал) можна додати класи PMagazine (Паперовий журнал), EMagazine (Електронний журнал), IMagazine (Інтернет журнал). У кожного нащадка має бути не менш ніж на два поля та метода більше ніж у предка.

4. Створити масив чи колекцію об’єктів усіх рівнів спадкування. Виконати відповідні функції із завдання власного варіанта.

Зміст

[1 Аналіз до предметної області 4](#_Toc119002620)

[2 Виконання роботи 6](#_Toc119002621)

[3 Довідка по роботі з програмою для користувача. 9](#_Toc119002622)

[Висновок 10](#_Toc119002623)

## 1 Аналіз до предметної області

Спадкування – це механізм отримання нового класу на основі вже існуючого. Існуючий клас може бути доповнений або змінений для створення нового класу.

Існуючі класи називаються базовими, а нові – похідними. Похідний клас успадковує опис базового класу; потім він може бути змінений додаванням нових членів, зміною існуючих функцій-членів та зміною прав доступу. Таким чином, успадкування дозволяє повторно використовувати вже розроблений код, що підвищує продуктивність програміста та зменшує ймовірність помилок. За допомогою наслідування може бути створена ієрархія класів, які спільно використовують код та інтерфейси.

Наслідувані компоненти не переміщаються у похідний клас, а залишаються у базових класах. Повідомлення, обробку якого неможливо виконати методи похідного класу, автоматично передається до базового класу. Якщо обробки повідомлення потрібні дані, відсутні у похідному класі, їх намагаються знайти автоматично і непомітно для програміста в базовому класі.

При наслідуванні деякі імена методів та даних базового класу можуть бути по-новому визначені у похідному класі. У цьому випадку відповідні компоненти базового класу стають недоступними із похідного класу. Для доступу до них використовується операція вказівки з області видимості '::'.

У ієрархії похідний об'єкт успадковує дозволені для успадкування компоненти всіх базових об'єктів (public, protected).

Абстрактний клас - це клас, у якого не реалізовано один або більше методів (деякі мови вимагають такі методи позначати спеціальними ключовими словами).

Інтерфейс - це абстрактний клас, у якого жоден метод не реалізований, всі вони публічні і немає змінних класу.

Інтерфейс потрібен зазвичай коли описується лише інтерфейс (тавтологія). Наприклад, один клас хоче дати іншому можливість доступу до деяких своїх методів, але не хоче себе розкривати. Тому він просто продає інтерфейс.

Абстрактний клас потрібен, коли потрібне сімейство класів, які мають багато спільного. Звичайно, можна застосувати інтерфейс, але тоді потрібно буде писати багато ідентичного коду.

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.

## 2 Виконання роботи

У ролі класу інтерфейсу перебуває клас Human. В собі має два поля даних ім’я, фамілію та ID-пропуск. Усі інші нащадки отримують ці властивості. Що до методів класу, усі вони віртуальні, які мають використовуватися у нащадках цього класу, а саме: вивід на екран та ініціалізація. Дані методи зазнають певних змін, що будуть описані пізніше. Приклад класу Human представлено нижче.

class Human {

protected:

string Name;

string SecondName;

string passID;

public:

virtual string getName() const { return Name; }

virtual string getpassID() const { return passID; }

virtual string getSecond() const { return SecondName; }

virtual void print() = 0;

virtual void setterAll() = 0;

};

Повний код програми наведено у додатку А.

Нащадки класу, описаного вище, є Student та Personal. Вони мають у свою чергу своїх нащадків. Усі вони мають нові поля даних та функціонально змінені методи предку. Приклад їх реалізації наведено нижче.

class Sutedent: public Human

{

protected:

string address;

string study;

};

class Extramural: public Sutedent

{

long int studyCost;

public:

void setterAll() override {

this->Name = dataNames[rand()%32];

this->SecondName = dataSecondNames[rand() % 50];

this->address = Name + SecondName + "@khpi.edu.ua";

this->study = "To pay for study";

this->passID = "ETStudent";

this->studyCost = (rand() %90000 + 1000);

}

void print() override {

cout << "|" << setw(10) << this->Name << setw(10) << this->SecondName << "|" << setw(33) << this->address;

cout << "|" << setw(12) << this->passID << setw(4) << "|" << setw(11) << "Debt: " << setw(6) << this->studyCost << setw(10) << "|" << setw(23) << this->study << setw(4) << "|" << endl;

}

};

class FullTime : public Sutedent

{

private:

string department;

string Crops[6] = { "Building 1", "Building 2", "Building 3", "Building 4", "Building 5", "Building 6" };

string Grupe[6] = { "CS-321a", "CS-321b", "CS-321c", "PE-321a", "PE-321b", "PE-321c" };

public:

void setterAll() override {

this->Name = dataNames[rand() % 32];

this->SecondName = dataSecondNames[rand() % 50];

this->department = Grupe[rand() % 6];

if(this->department == Grupe[0] or this->department == Grupe[1] or this->department == Grupe[2])

this->address = Name + SecondName + "@cs.khpi.edu.ua";

else

this->address = Name + SecondName + "@pe.khpi.edu.ua";

this->passID = "FTStudent";

this->study = Crops[rand() % 6];

}

void print() override {

cout << "|" << setw(10) << this->Name << setw(10) << this->SecondName << "|" << setw(33) << this->address;

cout << "|" << setw(12) << this->passID <<setw(4) << "|" << setw(15) << this->department << setw(12) << "|" << setw(20) << this->study << setw(7) << "|" << endl;

}

};

class Abiturient : public Sutedent {

int grade;

public:

void setterAll() override {

this->Name = dataNames[rand() % 32];

this->SecondName = dataSecondNames[rand() % 50];

this->address = Name + SecondName + "@gmail.com";

this->passID = "Abiturient";

this->grade = rand() % 81 + 120;

this->study = "To apply the documents";

}

void print() override {

cout << "|" << setw(10) << this->Name << setw(10) << this->SecondName << "|" << setw(33) << this->address;

cout << "|" << setw(13) << this->passID <<setw(3) << "|" << setw(12) << "EIE: " << grade << setw(12) << "|" <<setw(24) << this->study << setw(3) << "|" << endl;

}

};

class Staff : public Human

{

protected:

string number = "+380";

string work;

};

class Teacher : public Staff {

bool parkingPlace;

string workOps[4] = { "Paper work", "Lection work", "Practical work", "Laboratory work" };

public:

void setterAll() override {

this->Name = dataNames[rand() % 32];

this->SecondName = dataSecondNames[rand() % 50];

this->number += (to\_string(rand()%10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10));

this->work = workOps[rand()%4];

this->passID = "Teacher";

this->parkingPlace = rand() % 2;

}

void print() override {

cout << "|" << setw(10) << this->Name << setw(10) << this->SecondName << "|" << setw(33) << this->number;

cout << "|" << setw(12) << this->passID << setw(4) << "|" << setw(12) << "Parking: " << boolalpha << setw(9) << this->parkingPlace << setw(6) << "|" << setw(23) << this->work << setw(4) << "|" << endl;

}

};

class Security : public Staff {

int shift;

public:

void setterAll() override {

this->Name = dataNames[rand() % 32];

this->SecondName = dataSecondNames[rand() % 50];

this->number += (to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10));

this->shift = rand() % 6 + 1;

this->passID = "Security";

this->work = "CHECKPOINT " + to\_string(rand()%8+1);

}

void print() override {

cout << "|" << setw(10) << this->Name << setw(10) << this->SecondName << "|" << setw(33) << this->number;

cout << "|" << setw(12) << this->passID << setw(4) << "|" << setw(12) << "Shift: " << this->shift << setw(14) << "|" << setw(20) << this->work << setw(7) << "|" << endl;

}

};

Повний текст програми наведено у додатку А.

## 3 Довідка по роботі з програмою для користувача.

Програма собою представляє список вивідувачів територію навчального закладу, до списку можна додати, видалити або змінити данні. В залежності від типу даних — заповнюються відповідні поля, що продемонстровано на рисунку 3.1.

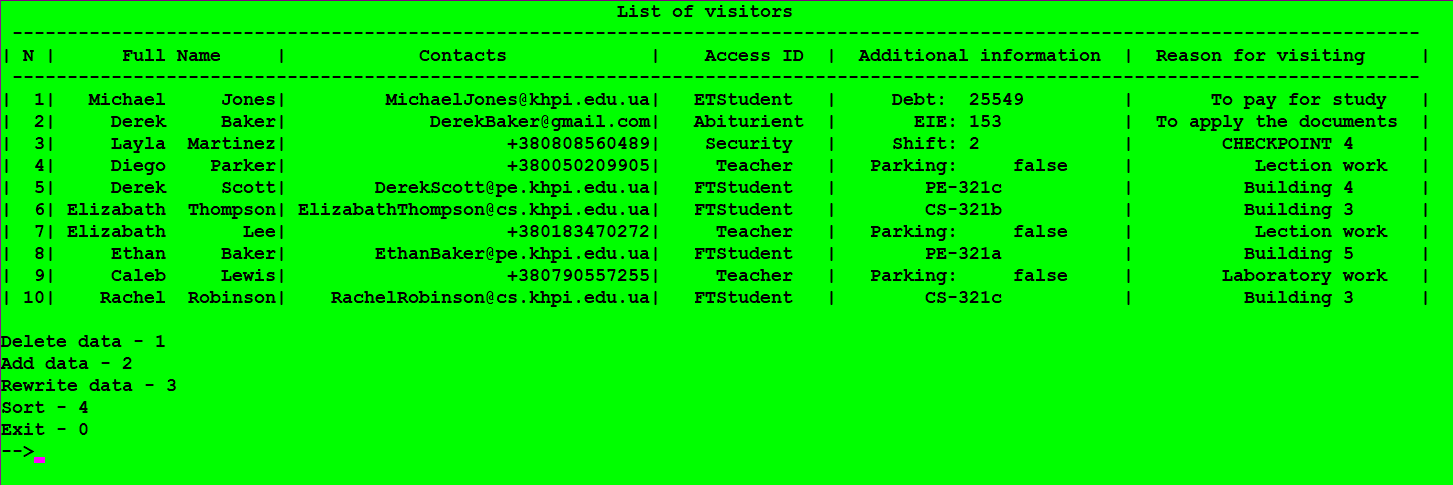


Рисунок 3.1 – Головне меню

## Висновок

Виконуючи цю лабораторну роботу, було вивчено що таке успадкування класів, здобуто навички з успадкування на мові с++ та роботою з абстрактними. Абстрактний клас - це клас, у якого не реалізовано один або більше методів (деякі мови вимагають такі методи позначати спеціальними ключовими словами).

**Додаток А**

Header.h

#pragma once

#include <iostream>

#include <string>

#include <iomanip>

#include <ctime>

#define MAXSIZE 300

using namespace std;

class Human {

protected:

string dataNames[32] = { "Ethan", "Josh", "Sam", "Tom", "Bruse", "Michael", "Diego", "Adrian", "Brian", "Carl", "Caleb", "Charles", "Daniel", "Derek", "Elliot", "Kuel",

"Sophia", "Amelia", "Ella", "Chloe", "Penelope", "Emma", "Ava", "Isabella", "Elizabath", "Ginna", "Layla", "Zoey", "Olivia", "Sarah", "Rachel", "Emily"};

string dataSecondNames[50] = { "Smith", "Johnson", "Williams", "Jones", "Brown","Davis","Miller", "Wilson","Moore","Taylor", "Anderson","Thomas","Jackson", "White","Harris","Martin", "Thompson","Garcia","Martinez", "Robinson","Clark","Rodriguez", "Lewis","Lee","Walker", "Hall","Allen","Young", "Hernandez","King","Wright", "Lopez","Hill","Scott", "Green","Adams","Baker",

"Gonzalez", "Nelson","Carter","Mitchell", "Perez","Roberts","Turner", "Phillips","Campbell","Parker", "Evans","Edwards","Collins"};

string Name;

string SecondName;

string passID;

public:

virtual string getName() const { return Name; }

virtual string getpassID() const { return passID; }

virtual string getSecond() const { return SecondName; }

virtual void print() = 0;

virtual void setterAll() = 0;

};

class Sutedent: public Human

{

protected:

string address;

string study;

};

class Extramural: public Sutedent

{

long int studyCost;

public:

void setterAll() override {

this->Name = dataNames[rand()%32];

this->SecondName = dataSecondNames[rand() % 50];

this->address = Name + SecondName + "@khpi.edu.ua";

this->study = "To pay for study";

this->passID = "ETStudent";

this->studyCost = (rand() %90000 + 1000);

}

void print() override {

cout << "|" << setw(10) << this->Name << setw(10) << this->SecondName << "|" << setw(33) << this->address;

cout << "|" << setw(12) << this->passID << setw(4) << "|" << setw(11) << "Debt: " << setw(6) << this->studyCost << setw(10) << "|" << setw(23) << this->study << setw(4) << "|" << endl;

}

};

class FullTime : public Sutedent

{

private:

string department;

string Crops[6] = { "Building 1", "Building 2", "Building 3", "Building 4", "Building 5", "Building 6" };

string Grupe[6] = { "CS-321a", "CS-321b", "CS-321c", "PE-321a", "PE-321b", "PE-321c" };

public:

void setterAll() override {

this->Name = dataNames[rand() % 32];

this->SecondName = dataSecondNames[rand() % 50];

this->department = Grupe[rand() % 6];

if(this->department == Grupe[0] or this->department == Grupe[1] or this->department == Grupe[2])

this->address = Name + SecondName + "@cs.khpi.edu.ua";

else

this->address = Name + SecondName + "@pe.khpi.edu.ua";

this->passID = "FTStudent";

this->study = Crops[rand() % 6];

}

void print() override {

cout << "|" << setw(10) << this->Name << setw(10) << this->SecondName << "|" << setw(33) << this->address;

cout << "|" << setw(12) << this->passID <<setw(4) << "|" << setw(15) << this->department << setw(12) << "|" << setw(20) << this->study << setw(7) << "|" << endl;

}

};

class Abiturient : public Sutedent {

int grade;

public:

void setterAll() override {

this->Name = dataNames[rand() % 32];

this->SecondName = dataSecondNames[rand() % 50];

this->address = Name + SecondName + "@gmail.com";

this->passID = "Abiturient";

this->grade = rand() % 81 + 120;

this->study = "To apply the documents";

}

void print() override {

cout << "|" << setw(10) << this->Name << setw(10) << this->SecondName << "|" << setw(33) << this->address;

cout << "|" << setw(13) << this->passID <<setw(3) << "|" << setw(12) << "EIE: " << grade << setw(12) << "|" <<setw(24) << this->study << setw(3) << "|" << endl;

}

};

class Staff : public Human

{

protected:

string number = "+380";

string work;

};

class Teacher : public Staff {

bool parkingPlace;

string workOps[4] = { "Paper work", "Lection work", "Practical work", "Laboratory work" };

public:

void setterAll() override {

this->Name = dataNames[rand() % 32];

this->SecondName = dataSecondNames[rand() % 50];

this->number += (to\_string(rand()%10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10));

this->work = workOps[rand()%4];

this->passID = "Teacher";

this->parkingPlace = rand() % 2;

}

void print() override {

cout << "|" << setw(10) << this->Name << setw(10) << this->SecondName << "|" << setw(33) << this->number;

cout << "|" << setw(12) << this->passID << setw(4) << "|" << setw(12) << "Parking: " << boolalpha << setw(9) << this->parkingPlace << setw(6) << "|" << setw(23) << this->work << setw(4) << "|" << endl;

}

};

class Security : public Staff {

int shift;

public:

void setterAll() override {

this->Name = dataNames[rand() % 32];

this->SecondName = dataSecondNames[rand() % 50];

this->number += (to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10) + to\_string(rand() % 10));

this->shift = rand() % 6 + 1;

this->passID = "Security";

this->work = "CHECKPOINT " + to\_string(rand()%8+1);

}

void print() override {

cout << "|" << setw(10) << this->Name << setw(10) << this->SecondName << "|" << setw(33) << this->number;

cout << "|" << setw(12) << this->passID << setw(4) << "|" << setw(12) << "Shift: " << this->shift << setw(14) << "|" << setw(20) << this->work << setw(7) << "|" << endl;

}

};

class Main {

private:

// working method

void deleteOneElement(Human\*\* List, int& size, int check);

void addOneElement(Human\*\* List, int& size, int check);

void rewriteOneElement(Human\*\* List, int& size, int check);

void Sort(Human\*\* List, int size, int button);

void create(Human\*\* List, int size);

void printList(Human\*\* List, int size);

public:

void main();

~Main() {};

};

**Додаток Б**

Source.cpp

#include "Header.h"

void Main::main() {

int size = 1000;

Human\*\* List = new Human \* [size];

cout << "How long you list could be? \n---> "; cin >> size;

create(List, size);

int button = -1;

int choose;

while (button != 0)

{

system("cls");

cout << "\t\t\t\t\t\t\tList of visitors";

printList(List, size); cout << endl;

cout << "Delete data - 1\n" <<

"Add data - 2\n" <<

"Rewrite data - 3\n" <<

"Sort - 4\n" <<

"Exit - 0\n" <<

"-->"; cin >> button;

switch (button)

{

case 1:

cout << "Choose the number of data which will bee deleted --> "; cin >> choose;

deleteOneElement(List, size, choose);

system("pause");

break;

case 2:

cout << "Choose the number of data which will bee added --> "; cin >> choose;

addOneElement(List, size, choose);

system("pause");

break;

case 3:

cout << "Choose the number of data which will bee rewrited --> "; cin >> choose;

rewriteOneElement(List, size, choose);

system("pause");

break;

case 4:

system("cls");

cout << "Sort by: \nBy First Name - 1 \nBy Second Name - 2 \nBy PassID - 3 \n-->"; cin >> button;

Sort(List, size, button);

break;

default:

break;

}

}

system("cls");

delete[] List; List = NULL;

}

void Main::create(Human\*\* List, int size)

{

for (size\_t i = 0; i < size; i++) {

int Nrand = rand() % 5 + 1;

if(Nrand == 1)

List[i] = new FullTime;

else if(Nrand == 2)

List[i] = new Extramural;

else if (Nrand == 3)

List[i] = new Abiturient;

else if (Nrand == 4)

List[i] = new Teacher;

else if (Nrand == 5)

List[i] = new Security;

}

for (size\_t i = 0; i < size; i++) List[i]->setterAll();

}

void Main::printList(Human\*\* List, int size)

{

cout << "\n --------------------------------------------------------------------------------------------------------------------------------\n";

cout << "| N " << "|" << setw(15) << "Full Name" << setw(6) << "|" << setw(20) << "Contacts" << setw(14) << "|" << setw(13) << "Access ID" << setw(4) << " | " << setw(15) << "Additional information" << setw(3) << " | " << setw(11) << "Reason for visiting " << setw(5) << " |";

cout << "\n --------------------------------------------------------------------------------------------------------------------------------\n";

for (size\_t i = 0; i < size; i++) {

cout << "|" << setw(3) << i + 1 << setw(1);

List[i]->print();

}

}

void Main::rewriteOneElement(Human\*\* List, int& size, int check)

{

if (check - 1 < 0 or check - 1 > size) {

cout << "\t\t\tError!\n"; return;

}

Human\*\* NewList = new Human \* [1000];

int Nrand = rand() % 5 + 1;

for (size\_t i = 0; i < size; i++) {

if (check - 1 != i) {

NewList[i] = List[i];

}

else

{

if (Nrand == 1)

NewList[i] = new FullTime;

else if (Nrand == 2)

NewList[i] = new Extramural;

else if (Nrand == 3)

NewList[i] = new Abiturient;

else if (Nrand == 4)

NewList[i] = new Teacher;

else if (Nrand == 5)

NewList[i] = new Security;

NewList[i]->setterAll();

}

}

for (size\_t i = 0; i < size; i++) List[i] = NewList[i];

delete[] NewList; NewList = NULL;

}

void Main::addOneElement(Human\*\* List, int& size, int check)

{

if (check - 1 < 0 or check - 1 > size) {

cout << "\t\t\tError!\n"; return;

}

Human\*\* NewList = new Human \* [1000];

int Nrand = rand() % 5 + 1;

for (size\_t i = 0, j = 0; i < size;) {

if (check - 1 != j) {

NewList[j] = List[i]; j++;

i++;

}

else

{

if (Nrand == 1)

NewList[j] = new FullTime;

else if (Nrand == 2)

NewList[j] = new Extramural;

else if (Nrand == 3)

NewList[j] = new Abiturient;

else if (Nrand == 4)

NewList[j] = new Teacher;

else if (Nrand == 5)

NewList[j] = new Security;

NewList[j]->setterAll();

j++;

}

}

size++;

for (size\_t i = 0; i < size; i++) List[i] = NewList[i];

delete[] NewList; NewList = NULL;

}

void Main::deleteOneElement(Human\*\* List, int& size, int check)

{

if (check - 1 < 0 or check - 1 > size) {

cout << "\t\t\tError!\n"; return;

}

Human\*\* NewList = new Human \* [1000];

for (size\_t i = 0, j = 0; i < size; i++)

if (check - 1 != i) {

NewList[j] = List[i]; j++;

}

size--;

for (size\_t i = 0; i < size; i++) List[i] = NewList[i];

delete[] NewList; NewList = NULL;

}

void Main::Sort(Human\*\* List, int size, int button)

{

if (button == 1) {

bool changeFlag = true;

int i = 1;

while (changeFlag) {

changeFlag = false;

for (int j = size - 1; j >= i; j--)

if (List[j]->getName() > List[j - 1]->getName()) {

swap(List[j], List[j - 1]);

changeFlag = true;

}

i++;

}

}

else if (button == 2) {

bool changeFlag = true;

int i = 1;

while (changeFlag) {

changeFlag = false;

for (int j = size - 1; j >= i; j--)

if (List[j]->getSecond() > List[j - 1]->getSecond()) {

swap(List[j], List[j - 1]);

changeFlag = true;

}

i++;

}

}

else if (button == 3) {

bool changeFlag = true;

int i = 1;

while (changeFlag) {

changeFlag = false;

for (int j = size - 1; j >= i; j--)

if (List[j]->getpassID() > List[j - 1]->getpassID()) {

swap(List[j], List[j - 1]);

changeFlag = true;

}

i++;

}

}

else

cout << "\n\t ERROR! Incorrect input! \n";

}