**Лабораторна робота №3**

**Тема.** Успадкування і віртуальні функції

**Мета.** Одержати практичні навички створення ієрархії класів і використання статичних компонентів класу.

**Основний зміст роботи.**

Написати програму, в якій створюється ієрархія класів. Включити поліморфні об'єкти в зв'язаний список, використовуючи статичні компоненти класу. Показати використання віртуальних функцій.

**Порядок виконання роботи.**

1.Визначити ієрархію класів (відповідно до варіанта).

2.Визначити в класі статичний компонент - покажчик на початок зв'язаного списку об'єктів і статичну функцію для перегляду списку.

3.Реалізувати класи.

4.Написати демонстраційну програму, у якій створюються об'єкти різних класів і поміщаються в список, після чого список проглядається.

5.Зробити відповідні методи не віртуальними і подивитися, що буде.

Завдання 3.1

1. Створити абстрактний клас **Видання** з методами дозволяючими вивести на екран інформацію про видання, а також визначити чи є дане видання шуканим.
2. Створити похідні класи: **Книга** (назва, прізвище автора, рік видання, видавництво), **Стаття** (назва, прізвище автора, назва журналу, його номер і рік видання), **Електронний ресурс** (назва, прізвище автора, посилання, анотація) з своїми методами виведення інформації на екран.
3. Створити каталог (масив) з n видань, вивести повну інформацію з каталога, а також організувати пошук видань по прізвищу автора.

Постановка задачі: Реалізовано абстрактний клас Видання (Edition), який містить поля типу(type), назви(title) та автора(authorSurname), цей клас містить параметризований конструктор: Edition(string Title, string AuthorSurname) та консруктор за замовчуванням: Edition(), клас містисть чисто віртуальний метод Publish(), який показує те, як об’єкт видається також має два віртуальні методи Print() та Init(), має сатичні методи для виведенн шапки Shap() та для виведення лінійї ShowLine(), також має IsFind() для перевірки відповідності до певного імені та відртуальний деструктор Edition(), для правильного видалення.

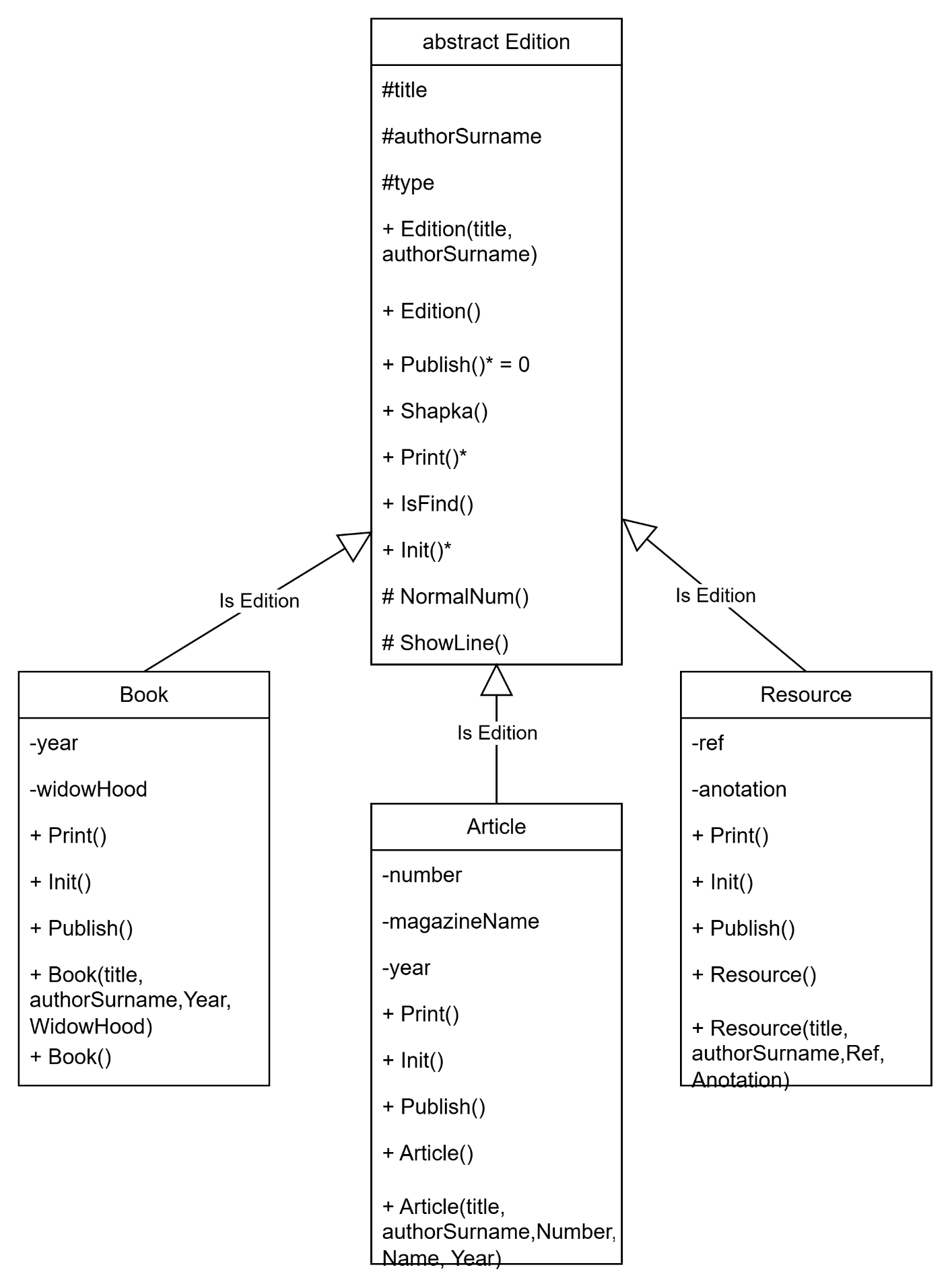
Від абстрактного класу Edition наслідуються три класи Book, Article, Reference.

Клас Book має: рік випуску(year) та видавництво(widowHood) та поля уснаслідовані від класу Edition, цей клас містить параметризований конструктор: Book(string title, string authorSurname, int Year, string WidowHood), який викликає та передає поля в Edition(title,authorSurname) та консруктор за замовчуванням: Book(), клас перевизначає віртуальний метод Publish(), який показує те, як об’єкт перевизначає два віртуальні методи Print() та Init(), має також всі поля із класу Edition.

Клас Article має: рік випуску(year) та номер(number) та назву(magazineName) та поля уснаслідовані від класу Edition, цей клас містить параметризований конструктор: Article(string title, string authorSurname,int Number, string Name, int Year), який викликає та передає поля в Edition(title,authorSurname) та консруктор за замовчуванням: Article(), клас перевизначає віртуальний метод Publish(), який показує те, як об’єкт перевизначає два віртуальні методи Print() та Init(), має також всі поля із класу Edition.

Клас Resource має: рік посилання(ref) та анотацію(anotation) та поля уснаслідовані від класу Edition, цей клас містить параметризований конструктор: Resource(string title, string authorSurname, string Ref, string Anotation), який викликає та передає поля в Edition(title,authorSurname) та консруктор за замовчуванням: Resource(), клас перевизначає віртуальний метод Publish(), який показує те, як об’єкт перевизначає два віртуальні методи Print() та Init(), має також всі поля із класу Edition.

Об’єктна модель:



Лістинг:

Edition.h:

#pragma once

#include <iostream>

#include <string>

#include <iomanip>

using namespace std;

class Edition {

protected:

string title;

string authorSurname;

string type;

public:

Edition(string Title, string AuthorSurname) : title(Title), authorSurname(AuthorSurname) { }

Edition() : title("None"), authorSurname("None") {}

virtual void Print() = 0;

virtual void Init() {

cout << "Введiть назву: "; getline(cin, title);

cout << "Введiть автора: "; getline(cin, authorSurname);

}

virtual void Shap() {

cout << setw(20) << "Title" << setw(20) << "AuthorSurname" << setw(20) << "Type";

}

bool IsFind(string Surname) { return authorSurname == Surname; }

void GetEdition() { cout << setw(20) << title << setw(20) << authorSurname << setw(20) << type;

}

virtual ~Edition() {};

protected:

int NormalNum(int num) {

while (num <= 0) {

cout << "Невiрне числове значення: " << num << endl;

cin >> num;

}

return num;

}

};

Book.h:

#pragma once

#include "Edition.h"

class Book : public Edition {

private:

int year;

string widowHood;

public:

Book(string title, string authorSurname, int Year, string WidowHood) : Edition(title, authorSurname) {

year = NormalNum(Year);

widowHood = WidowHood;

type = "Book";

}

Book() : year(0), widowHood("None") { type = "Book"; }

void Print() override {

GetEdition();

cout << setw(10) << year << setw(20) << widowHood << endl << endl;

}

void Init() override {

Edition::Init();

cout << "Введiть рiк випуску книги: "; cin >> year; year = NormalNum(year); cin.ignore();

cout << "Введiть видавництво: "; getline(cin, widowHood);

}

void Shap() override {

Edition::Shap();

cout << setw(10) << "Year" << setw(20) << "Widowhood" << endl;

}

virtual ~Book(){}

};

Article.h:

#pragma once

#include "Edition.h"

class Article: public Edition {

private:

int number;

string magazineName;

int year;

public:

Article(string title, string authorSurname,int Number, string Name, int Year) : Edition(title, authorSurname) {

number = NormalNum(Number);

magazineName = Name;

year = NormalNum(Year);

type = "Article";

}

Article() : year(0), number(0),magazineName("None") { type = "Article"; }

void Print() override {

GetEdition();

cout << setw(10) << number << setw(20) << magazineName << setw(10) <<year << endl << endl;

}

void Init() override {

Edition::Init();

cout << "Введiть номер статтi: "; cin >> number; number = NormalNum(number);

cout << "Введiть рiк випуску статті: "; cin >> year; year = NormalNum(year);

cin.ignore();

cout << "Введiть назву статтi: "; getline(cin, magazineName);

}

void Shap() override {

Edition::Shap(); cout << setw(10) << "Number" << setw(20) << "MagazineName" << setw(10) << "Year" << endl;

}

virtual ~Article() {}

};

Resource.h:

#pragma once

#include "Edition.h"

class Resource : public Edition {

private:

string ref;

string anotation;

public:

Resource(string title, string authorSurname, string Ref, string Anotation) : Edition(title, authorSurname) {

ref = Ref;

anotation = Anotation;

type = "Resource";

}

Resource() : ref("None"), anotation("None") { type = "Resource"; }

void Print() override {

GetEdition();

cout << setw(20) << ref << setw(20) << anotation << endl << endl;

}

void Init() override {

Edition::Init();

cout << "Введiть назву посилання: "; getline(cin, ref);

cout << "Введiть назву анотацію: "; getline(cin, anotation);

}

void Shap() override {

Edition::Shap();

cout << setw(20) << "Ref" << setw(20) << "Anotation"<< endl;

}

virtual ~Resource() {}

};

List.h:

#pragma once

#include "Edition.h"

class Node {

public:

Edition\* e;

Node\* next;

Node(Edition\* ed) : e(ed), next(nullptr) {}

};

class List {

private:

static Node\* head;

public:

static void Add(Edition\* e);

static void Delete();

static bool HasSurname(string surname);

static void Find(string surname);

static void Show();

static void Clear();

};

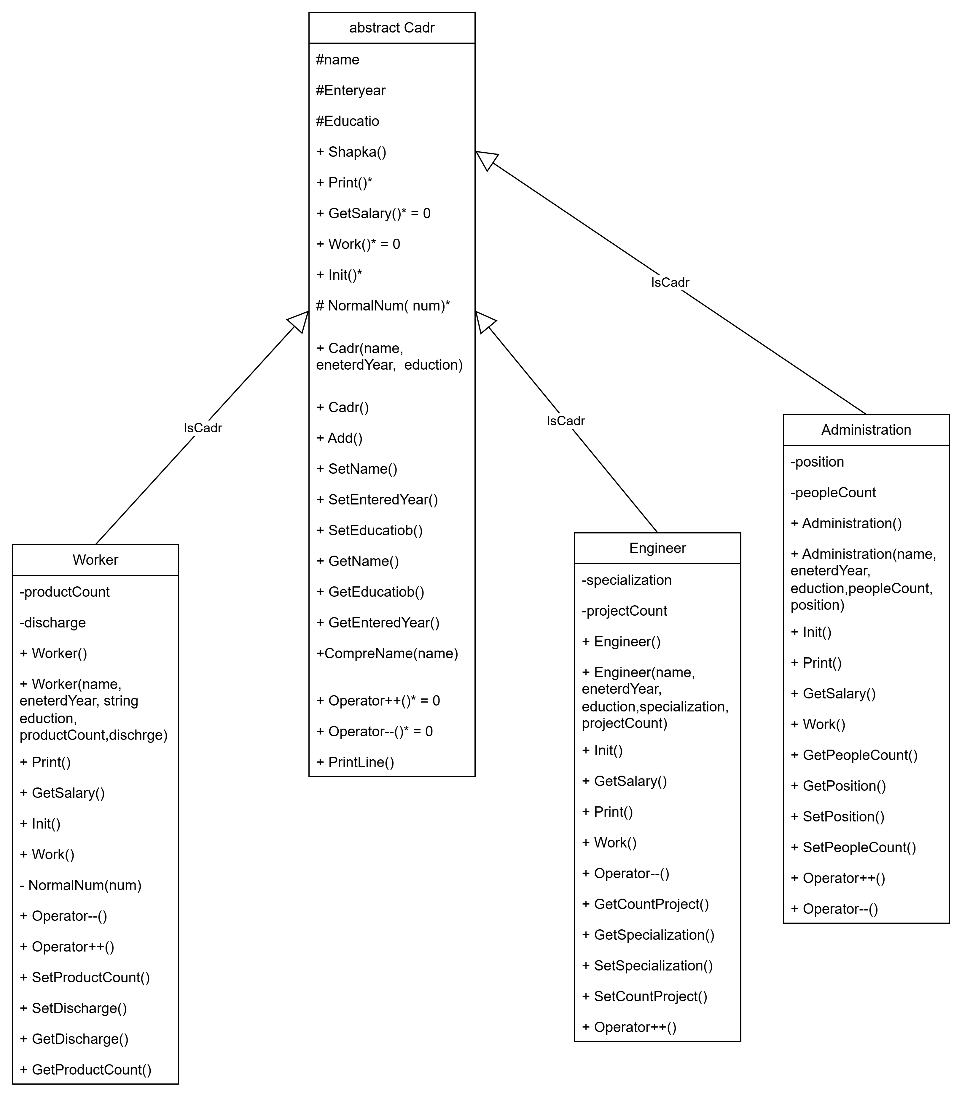
List.cpp:

Завдання 2:

*Дано перелік класів. Побудуйте обєктну модель предметної області, враховуючи, види залежностей між класами. Реалізуйте проект за ООП.*

**3. робітник, кадр, інженер, адміністрація**

Об’єктна модель:



Лістинг:

Cadr.h:

#pragma once

#include <iostream>

#include <string>

#include <iomanip>

using namespace std;

class List;

class Cadr {

protected:

string name;

int enteredYear;

string education;

public:

Cadr(string name, int enteredYear, string education);

Cadr();

void Add();

static void Shapka();

virtual void Print() {

cout << setw(13) << GetName() << " |" << setw(8) << GetEnteredYear() << " |" << setw(8) << GetEducation() << " |";

}

virtual double GetSalary() = 0;

virtual void Work() = 0;

virtual void Init();

void SetName();

void SetEnteredYear();

void SetEducation();

string GetName();

int GetEnteredYear();

string GetEducation();

bool CompareName(string name);

virtual Cadr& operator++() = 0;

virtual Cadr& operator--() = 0;

protected:

virtual void NormalNum(int& num);

static void PrintLine();

};

Cadr.cpp:

#pragma once

#include "List.h"

Cadr::Cadr(string name, int enteredYear, string education) {

this->name = name;

this->enteredYear = enteredYear;

NormalNum(this->enteredYear);

this->education = education;

Add();

}

Cadr::Cadr() : name("None"), enteredYear(0), education("None") {}

void Cadr::Add() {

List::Add(this);

}

void Cadr::Shapka() {

cout << setw(15) << "Iм'я |" << setw(10) << "Вступ |" << setw(10) << "Освiта |";

cout << setw(15) << "Продукцiї |" << setw(10) << "Розряд |";

cout << setw(15) << "Спецiалiзацiя |" << setw(10) << "Проєктiв |";

cout << setw(15) << "Посада |" << setw(10) << "Пiдданих" << endl;

PrintLine();

}

void Cadr::Init() {

SetName();

SetEducation();

SetEnteredYear();

cin.ignore();

}

void Cadr::SetName()

{

cout << "Введiть iм'я: "; getline(cin, name);

}

void Cadr::SetEnteredYear()

{

cout << "Введiть рiк вступу в компанiю: ";  cin >> enteredYear; NormalNum(enteredYear);

}

void Cadr::SetEducation()

{

cout << "Введiть, яка освiта: "; getline(cin, education);

}

string Cadr::GetName()

{

return name;

}

int Cadr::GetEnteredYear()

{

return enteredYear;

}

string Cadr::GetEducation()

{

return education;

}

bool Cadr::CompareName(string name)

{

return this->name == name;

}

void Cadr::NormalNum(int& num) {

while (num < 0) {

cin >> num;

if (num < 0)

cout << "Некоректно введенне значення." << endl;

}

}

void Cadr::PrintLine()

{

for (int i = 0; i < 110; i++)

cout << "-";

cout << endl;

}

Worker.h:

#pragma once

#include "Cadr.h"

class Worker: public Cadr {

private:

int productCount;

int discharge;

public:

Worker(string name, int enteredYear, string education, int productCount, int discharge): Cadr(name,enteredYear,education){

Cadr::NormalNum(productCount);

this->productCount = productCount;

NormalNum(discharge);

this->discharge = discharge;

}

Worker() : productCount(0), discharge(0) {}

void Print() override {

Cadr::Print();

cout << setw(13) << GetProductCount() << " |" << setw(8) << GetDischarge() << " |";

cout << setw(15) << "|" << setw(10) << "|";

cout << setw(15) << "|" <<endl;

Cadr::PrintLine();

}

double GetSalary() override{

switch (discharge)

{

case 1:

return 12000; break;

case 2:

return 15000; break;

case 3:

return 17000; break;

default:

return 20000; break;

}

}

void Work() override {

cout << "Виконати роботу робiтника." << endl;

}

void Init() override{

Cadr::Init();

SetDischarge();

SetProductCount();

cin.ignore();

}

void SetProductCount() {

cout << "Введiть кiлькiсть виготовлений продуктiв.";

cin >> productCount;

Cadr::NormalNum(productCount);

}

void SetDischarge() {

cout << "Введiть розр'яд: ";

cin >> discharge; NormalNum(discharge);

}

int GetProductCount() {

return productCount;

}

int GetDischarge() {

return discharge;

}

Worker& operator++() override {

productCount++;

return \*this;

}

Worker& operator--() override {

if (productCount > 0)

productCount--;

else

cout << "Не можна пiти в мiнус" << endl;

return \*this;

}

private:

void NormalNum(int& num) override {

while (num < 0 || num > 4) {

cout << "Невiрний розряд." << endl;

cin >> num;

}

}

};

Engineer.h:

#pragma once

#include "Cadr.h"

class Engineer : public Cadr{

private:

string specialization;

int countProject;

public:

Engineer(string name, int enteredYear, string education, string specialization, int countProject) : Cadr(name, enteredYear, education) {

this->specialization = specialization;

Cadr::NormalNum(countProject);

this->countProject = countProject;

}

Engineer() : specialization("None"), countProject(0) {}

void Work() override {

cout << "Iнженер виконав роботу." << endl;

}

double GetSalary() override {

if (countProject < 10)

return 20000;

else if (countProject < 30)

return 30000;

else

return 50000;

}

void Init() override {

Cadr::Init();

SetSpecialization();

SetCountProject();

cin.ignore();

}

void Print() override {

Cadr::Print();

cout << setw(15) << "|" << setw(10) << "|";

cout << setw(13) << GetSpecialization() << " |" << setw(8) << GetCountProject() << " |";

cout << setw(15) << "|"  << endl;

Cadr::PrintLine();

}

void SetCountProject() {

cout << "Введiть кiлькiсть проeктiв: "; cin >> countProject; Cadr::NormalNum(countProject);

}

void SetSpecialization() {

cout << "Введiть спецiалiзацiю: "; getline(cin, specialization);

}

Engineer& operator++() override {

countProject++;

return \*this;

}

Engineer& operator--() override {

if (countProject > 0)

countProject--;

else

cout << "Не можна пiти в мiнус" << endl;

return \*this;

}

string GetSpecialization() {

return specialization;

}

int GetCountProject() {

return countProject;

}

};

Administrarion.h:

#pragma once

#include "Cadr.h"

class Administration : public Cadr{

private:

string position;

int peopleCount;

public:

Administration(string name, int enteredYear, string education, string position, int peopleCount) : Cadr(name, enteredYear, education) {

this->position = position;

Cadr::NormalNum(peopleCount);

this->peopleCount = peopleCount;

}

Administration() : position("None"), peopleCount(0) {};

void Work() override {

cout << "Адмiнiстрацiя працює." << endl;

}

double GetSalary() override {

if (peopleCount< 5)

return 40000;

else if (peopleCount < 25)

return 50000;

else

return 70000;

}

void Print() override {

Cadr::Print();

cout << setw(15) << "|" << setw(10) << "|";

cout << setw(15) << "|" << setw(10) << "|";

cout << setw(13) << GetPosition() << " |" << setw(10) << GetPeopleCount() << endl;

Cadr::PrintLine();

}

void Init() override {

Cadr::Init();

SetPosition();

SetPeopleCount();

cin.ignore();

}

void SetPosition() {

cout << "Введiть посаду: "; getline(cin, position);

}

void SetPeopleCount() {

cout << "Введiть кiлькiсть пiдданих: "; cin >> peopleCount; Cadr::NormalNum(peopleCount);

}

string GetPosition() {

return position;

}

int GetPeopleCount() {

return peopleCount;

}

Administration& operator++() override {

peopleCount++;

return \*this;

}

Administration& operator--() override {

if (peopleCount > 0)

peopleCount--;

else

cout << "Не можна пiти в мiнус" << endl;

return \*this;

}

};

List.h:

#pragma once

#include "Cadr.h"

class Node {

public:

Cadr\* people;

Node\* next;

Node(Cadr\* c) : people(c), next(nullptr) {}

};

class List {

private:

static Node\* head;

static bool HasName(string name);

public:

static void Add(Cadr\* cadr);

static void Delete();

static void Show();

static void Find(string name);

static void Clear();

};

List.cpp:

#include "List.h"

Node\* List::head = nullptr;

void List::Add(Cadr\* cadr) {

Node\* newNode = new Node(cadr);

if (!head)

head = newNode;

else {

Node\* c = head;

while (c->next)

c = c->next;

c->next = newNode;

}

}

void List::Delete() {

if (!head)

return;

if (!head->next) {

delete head->people;

delete head;

head = nullptr;

}

else {

Node\* c = head, \* last = head->next;

while (last->next) {

c = last;

last = last->next;

}

delete last->people;

delete last;

c->next = nullptr;

}

}

void List::Show() {

if (!head)

return;

Cadr::Shapka();

Node\* c = head;

while (c) {

c->people->Print();

c = c->next;

}

}

bool List::HasName(string name) {

bool find = false;

Node\* c = head;

while (c) {

if (c->people->CompareName(name))

find = true;

c = c->next;

}

return find;

}

void List::Find(string name)

{

if (!HasName(name)) {

cout << "Вказаних елементiв не знайдено." << endl;

return;

}

Cadr::Shapka();

Node\* c = head;

while (c)

{

if (c->people->CompareName(name))

c->people->Print();

c = c->next;

}

}

void List::Clear()

{

while (head)

Delete();

}