**Лабораторна робота №3. 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.

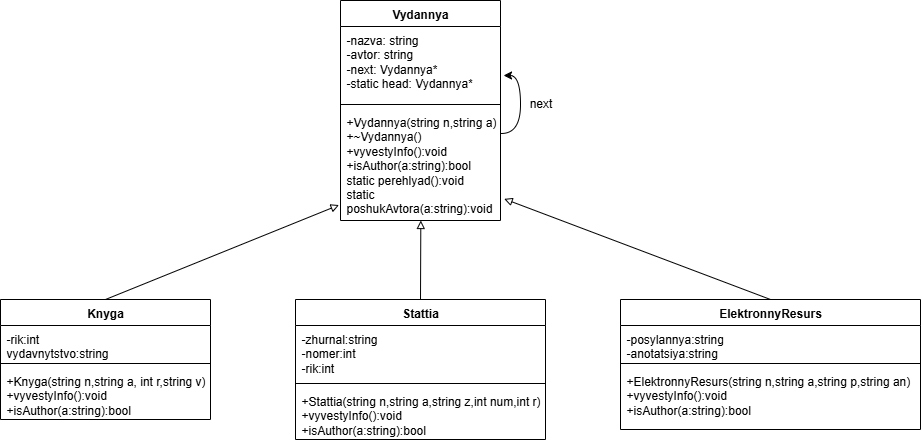
5.Зробити відповідні методи не віртуальними і подивитися, що буде.

6.Реалізувати варіант, коли об'єкт додається в список при створенні, тобто в конструкторі (дивися пункт 6 наступного розділу).

**Задача №1**

**Варіант 3**

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:**

Необхідно розробити ієрархію класів для опису видань, що можуть зберігатися у каталозі.  
Передбачається реалізація базового абстрактного класу та кількох похідних класів, а також механізм роботи зі зв’язаним списком.

**1. Базовий клас Vydannya**

Призначення: описує загальні властивості будь-якого видання.

* **Поля:**

string nazva — назва видання.

string avtor — прізвище автора.

Vydannya\* next — вказівник на наступний елемент списку.

static Vydannya\* head — статичний покажчик на початок зв’язаного списку.

* **Конструктор:**

Vydannya(string n, string a) — ініціалізує назву і автора та автоматично додає об’єкт у кінець списку.

* **Деструктор:**

virtual ~Vydannya() — віртуальний деструктор.

* **Методи (чисто віртуальні):**

virtual void vyvestyInfo() const = 0; — вивести інформацію про видання.

virtual bool isAuthor(const string& a) const = 0; — перевірити, чи належить видання вказаному автору.

* **Статичні методи:**

static void perehlyad(); — перегляд усього списку.

static void poshukAvtora(const string& a); — пошук видань за прізвищем автора.

2. **Похідні класи**

**Knyga (Книга)**

* **Додаткові поля:**

int rik — рік видання.

string vydavnytstvo — назва видавництва.

* **Конструктор:**

Knyga(string n, string a, int r, string v) — ініціалізує всі поля і додає об’єкт у список.

* **Методи:**

void vyvestyInfo() const override;

bool isAuthor(const string& a) const override;

**Stattia (Стаття)**

* **Додаткові поля:**

string zhurnal — назва журналу.

int nomer — номер журналу.

int rik — рік виходу.

* **Конструктор:**

Stattia(string n, string a, string z, int num, int r)

* **Методи:**

void vyvestyInfo() const override;

bool isAuthor(const string& a) const override;

**ElektronnyResurs (Електронний ресурс)**

* **Додаткові поля:**

string posylannya — посилання на ресурс.

string anotatsiya — анотація.

* **Конструктор:**

ElektronnyResurs(string n, string a, string p, string an)

* **Методи:**

void vyvestyInfo() const override;

bool isAuthor(const string& a) const override;

**Код програми:**

**Main.cpp(ГОЛОВНА ПРОГРАМА)**

#include "Книга.h"

#include "Стаття.h"

#include "ЕлектроннийРесурс.h"

int main() {

setlocale(0, "ukr");

// ===== Створюємо каталог (масив) =====

const int n = 4;

Vydannya\* catalog[n] = {

new Knyga("Philosophy of Code", "Pavlenko", 2021, "Lviv Polytechnic"),

new Stattia("Machine Learning", "Ivanenko", "Tech Today", 5, 2022),

new Knyga("Discrete Mathematics", "Cormen", 2017, "MIT Press"),

new ElektronnyResurs("Python Docs", "Guido", "docs.python.org", "Official Documentation")

};

int choice;

do {

cout << "\n=== МЕНЮ ===\n";

cout << "1. Перегляд всіх видань\n";

cout << "2. Пошук видань за автором\n";

cout << "3. Вихід\n";

cout << "Введіть номер дії: ";

cin >> choice;

cin.ignore(); // очищаємо буфер після cin

switch (choice) {

case 1: {

cout << "\n=== Повний каталог ===\n";

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

catalog[i]->vyvestyInfo();

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

}

break;

}

case 2: {

string author;

cout << "Введіть прізвище автора: ";

getline(cin, author);

bool found = false;

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

if (catalog[i]->isAuthor(author)) {

catalog[i]->vyvestyInfo();

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

found = true;

}

}

if (!found) cout << "Видань з автором '" << author << "' не знайдено.\n";

break;

}

case 3:

cout << "Вихід...\n";

break;

default:

cout << "Невірний вибір, спробуйте ще раз.\n";

}

} while (choice != 3);

// Звільняємо пам’ять

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

delete catalog[i];

}

return 0;

}

**Видання.h**

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Vydannya {

protected:

string nazva;

string avtor;

Vydannya\* next; // вказівник на наступний елемент

static Vydannya\* head; // початок списку

public:

Vydannya(string n, string a);

virtual ~Vydannya();

virtual void vyvestyInfo() const = 0;

virtual bool isAuthor(const string& a) const = 0;

static void perehlyad();

static void poshukAvtora(const string& a);

};

**Видання.сpp**

#include "Видання.h"

// Ініціалізація статичного поля

Vydannya\* Vydannya::head = nullptr;

Vydannya::Vydannya(string n, string a) : nazva(n), avtor(a), next(nullptr) {

if (!head) {

head = this;

}

else {

Vydannya\* temp = head;

while (temp->next) temp = temp->next;

temp->next = this;

}

}

Vydannya::~Vydannya() {}

void Vydannya::perehlyad() {

Vydannya\* temp = head;

while (temp) {

temp->vyvestyInfo();

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

temp = temp->next;

}

}

void Vydannya::poshukAvtora(const string& a) {

Vydannya\* temp = head;

bool found = false;

while (temp) {

if (temp->isAuthor(a)) {

temp->vyvestyInfo();

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

found = true;

}

temp = temp->next;

}

if (!found) cout << "Видань з автором '" << a << "' не знайдено.\n";

}

**Електронний ресурс.h**

#pragma once

#include "Видання.h"

class ElektronnyResurs : public Vydannya {

string posylannya;

string anotatsiya;

public:

ElektronnyResurs(string n, string a, string p, string an);

void vyvestyInfo() const override;

bool isAuthor(const string& a) const override;

};

**Електронний ресурс.cpp**

#include "ЕлектроннийРесурс.h"

#include <iostream>

using namespace std;

ElektronnyResurs::ElektronnyResurs(string n, string a, string p, string an)

: Vydannya(n, a), posylannya(p), anotatsiya(an) {

}

void ElektronnyResurs::vyvestyInfo() const {

cout << "Електронний ресурс\nНазва: " << nazva<< "\nАвтор: " << avtor<< "\nПосилання: " << posylannya<< "\nАнотація: " << anotatsiya << endl;

}

bool ElektronnyResurs::isAuthor(const string& a) const {

return avtor == a;

}

**Книга.h**

#pragma once

#include "Видання.h"

class Knyga : public Vydannya {

int rik;

string vydavnytstvo;

public:

Knyga(string n, string a, int r, string v);

void vyvestyInfo() const override;

bool isAuthor(const string& a) const override;

};

**Книга.cpp**

#include "Книга.h"

#include <iostream>

using namespace std;

Knyga::Knyga(string n, string a, int r, string v)

: Vydannya(n, a), rik(r), vydavnytstvo(v) {

}

void Knyga::vyvestyInfo() const {

cout << "Книга\nНазва: " << nazva<< "\nАвтор: " << avtor<< "\nРік: " << rik<< "\nВидавництво: " << vydavnytstvo << endl;

}

bool Knyga::isAuthor(const string& a) const {

return avtor == a;}

**Стаття.h**

#pragma once

#include "Видання.h"

class Stattia : public Vydannya {

string zhurnal;

int nomer;

int rik;

public:

Stattia(string n, string a, string z, int num, int r);

void vyvestyInfo() const override;

bool isAuthor(const string& a) const override;

};

**Стаття.cpp**

#include "Стаття.h"

#include <iostream>

using namespace std;

Stattia::Stattia(string n, string a, string z, int num, int r)

: Vydannya(n, a), zhurnal(z), nomer(num), rik(r) {

}

void Stattia::vyvestyInfo() const {

cout << "Стаття\nНазва: " << nazva<< "\nАвтор: " << avtor<< "\nЖурнал: " << zhurnal<< "\nНомер: " << nomer<< "\nРік: " << rik << endl;

}

bool Stattia::isAuthor(const string& a) const {

return avtor == a;

}

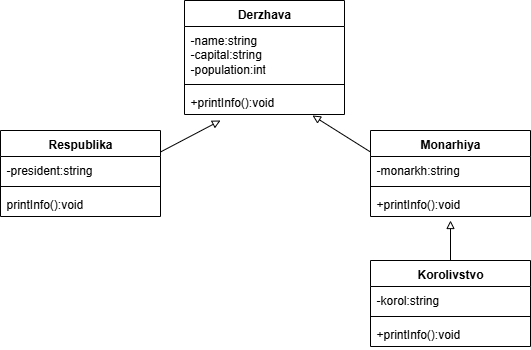
**Результат програми:**

|  |  |
| --- | --- |
|  | При запуску програми в нас є меню вибору:  1.Перегляд всіх видань  2.Пошук видань за автором  3.Вихід  Протестуємо програму. Для цього виберемо пункт 1. |
|  | Обрали варіант 1 і нам відображається повний каталог книг, статтей, електронних ресурсів. Тепер протестуємо пошук за автором. |
|  | Шукаємо за автором Cormen. Пошук виконався успішно. Тепер протестуємо іншого автора. |
|  | Шукаємо за автором Yevchuk. Програма видала повідомлення про те, що такого автора не було знайдено. Виконаємо вихід з програми. |

**Задача №2**

Дано перелік класів. Побудуйте об’єктну модель предметної області, враховуючи, види залежностей між класами. Реалізуйте проект за ООП.

**13. республіка, монархія, королівство, держава**



**Постановка задачі**

Реалізувати ієрархію класів для предметної області “Держава” з використанням об’єктно-орієнтованого програмування, забезпечити додавання об’єктів різних типів у каталог та їх перегляд через меню.

**1. Базовий клас Derzhava**

**Поля (protected):**

name — назва держави (string)

capital — столиця (string)

population — населення (int)

**Конструктор:**

Derzhava(string n, string c, int p) — ініціалізація назви, столиці та населення

**Методи:**

virtual void printInfo() const — вивід інформації про державу

**Деструктор:** virtual ~Derzhava()

**2. Похідний клас Respublika (успадковує від Derzhava)**

**Поля (private):**

president — ім’я президента (string)

**Конструктор:**

Respublika(string n, string c, int p, string pres)

**Методи:**

void printInfo() const override — вивід інформації про республіку (назва, столиця, населення, президент)

**3. Похідний клас Monarhiya (успадковує від Derzhava)**

**Поля (protected):**

monarkh — ім’я монарха (string)

**Конструктор:**

Monarhiya(string n, string c, int p, string m)

**Методи:**

void printInfo() const override — вивід інформації про монархію (назва, столиця, населення, монарх)

**4. Похідний клас Korolivstvo (успадковує від Monarhiya)**

**Поля (private):**

korol — ім’я короля або королеви (string)

**Конструктор:**

Korolivstvo(string n, string c, int p, string k)

**Методи:**

void printInfo() const override — вивід інформації про королівство (назва, столиця, населення, король/королева)

**Код програми**

**Main.cpp(ГОЛОВНА ПРОГРАМА)**

#include <iostream>

#include <string>

#include "Derzhava.h"

#include "Respublika.h"

#include "Monarhiya.h"

#include "Korolivstvo.h"

using namespace std;

int main() {

setlocale(0, "ukr");

Derzhava\* catalog[100];

int count = 0;

int choice;

do {

cout << "\n=== МЕНЮ ===\n";

cout << "1. Додати республiку\n";

cout << "2. Додати монархiю\n";

cout << "3. Додати королiвство\n";

cout << "4. Показати усi держави\n";

cout << "0. Вихiд\n";

cout << "Ваш вибір: ";

cin >> choice;

cin.ignore(); // щоб прибрати залишок символів з буфера після cin

if (choice == 1) {

string n, c, pres;

int p;

cout << "Назва: ";

getline(cin, n);

cout << "Столиця: ";

getline(cin, c);

cout << "Населення (млн): ";

cin >> p;

cin.ignore();

cout << "Президент: ";

getline(cin, pres);

catalog[count++] = new Respublika(n, c, p, pres);

}

else if (choice == 2) {

string n, c, mon;

int p;

cout << "Назва: ";

getline(cin, n);

cout << "Столиця: ";

getline(cin, c);

cout << "Населення (млн): ";

cin >> p;

cin.ignore();

cout << "Монарх: ";

getline(cin, mon);

catalog[count++] = new Monarhiya(n, c, p, mon);

}

else if (choice == 3) {

string n, c, korol;

int p;

cout << "Назва: ";

getline(cin, n);

cout << "Столиця: ";

getline(cin, c);

cout << "Населення (млн): ";

cin >> p;

cin.ignore();

cout << "Король: ";

getline(cin, korol);

catalog[count++] = new Korolivstvo(n, c, p, korol);

}

else if (choice == 4) {

for (int i = 0; i < count; i++) {

catalog[i]->printInfo();

}

}

} while (choice != 0);

for (int i = 0; i < count; i++) delete catalog[i];

return 0;

}

**Derzhava.h**

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Derzhava {

protected:

string name;

string capital;

int population;

public:

Derzhava(string n, string c, int p);

virtual ~Derzhava() {}

virtual void printInfo() const;

};

**Derzhava.cpp**

#include "Derzhava.h"

Derzhava::Derzhava(string n, string c, int p) : name(n), capital(c), population(p) {}

void Derzhava::printInfo() const {

cout << "Держава: " << name << ", столиця: " << capital

<< ", населення: " << population << " млн" << endl;

}

**Respublika.h**

#pragma once

#include "Derzhava.h"

class Respublika : public Derzhava {

string president;

public:

Respublika(string n, string c, int p, string pres);

void printInfo() const override;

};

**Respublika.cpp**

#include "Respublika.h"

Respublika::Respublika(string n, string c, int p, string pres)

: Derzhava(n, c, p), president(pres) {

}

void Respublika::printInfo() const {

cout << "Республіка: " << name << ", столиця: " << capital

<< ", населення: " << population << " млн"

<< ", президент: " << president << endl;

}

**Monarhiya.h**

#pragma once

#include "Derzhava.h"

class Monarhiya : public Derzhava {

protected:

string monarkh;

public:

Monarhiya(string n, string c, int p, string m);

void printInfo() const override;

};

**Monarhiya.cpp**

#include "Monarhiya.h"

Monarhiya::Monarhiya(string n, string c, int p, string m)

: Derzhava(n, c, p), monarkh(m) {

}

void Monarhiya::printInfo() const {

cout << "Монархія: " << name << ", столиця: " << capital

<< ", населення: " << population << " млн"

<< ", монарх: " << monarkh << endl;

}

**Korolivstvo.h**

#pragma once

#include "Monarhiya.h"

class Korolivstvo : public Monarhiya {

public:

Korolivstvo(string n, string c, int p, string k);

void printInfo() const override;

};

**Korolivstvo.cpp**

#include "Korolivstvo.h"

Korolivstvo::Korolivstvo(string n, string c, int p, string k)

: Monarhiya(n, c, p, k) {

}

void Korolivstvo::printInfo() const {

cout << «Королівство: « << name << «, столиця: « << capital<< «, населення: « << population << « млн»<< «, король: « << monarkh << endl;

}

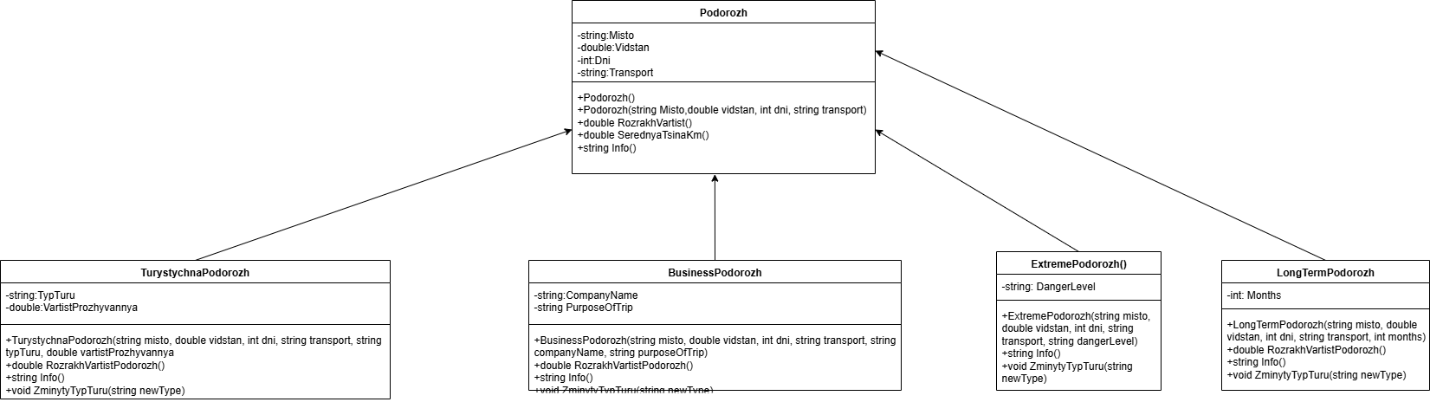
**Результат програми**

|  |  |
| --- | --- |
|  | Під час запуску програми, з’являється меню вибору. Протестуємо кожний вибір |
|  | Додамо республіку та переглянемо результат. |
|  | Додамо монархію та переглянемо результат. |
|  | Додамо королівство та переглянемо результат |

**Задача №3**

Утворити похідний клас, залучивши до нього як мінімум два додаткових поля таким чином, щоб клас набув більшої спеціалізованості. Для похідного класу використати кон­структор, щоб він містив усі аргументи, необхідні для ініціалізації об'єкту похідного класу. Створити додаткові необхідні функції, що дозволяють перевірити роботу похідних класів.

**Ієрархія**

****

**Код програми:**

**Class1.cs**

public class Podorozh

{

public string Misto { get; set; }

public double Vidstan { get; set; }

public int Dni { get; set; }

public string Transport { get; set; }

public Podorozh()

{

Misto = "Львів";

Vidstan = 150;

Dni = 1;

Transport = "Автобус";

}

public Podorozh(string misto, double vidstan, int dni, string transport)

{

Misto = misto;

Vidstan = vidstan;

Dni = dni;

Transport = transport;

}

public double RozrakhVartist()

{

double tsinaZaKm = 5;

double tsinaZaDen = 200;

return (Vidstan \* tsinaZaKm) + (Dni \* tsinaZaDen);

}

public double SerednyaTsinaKm()

{

return RozrakhVartist() / Vidstan;

}

// Тепер метод Info позначений як virtual

public virtual string Info()

{

return $"Місто: {Misto}, Відстань: {Vidstan} км, Днів: {Dni}, Транспорт: {Transport}, Вартість: {RozrakhVartist()} грн";

}

// Перевантаження операторів

public static Podorozh operator +(Podorozh a, Podorozh b)

{

return new Podorozh(

a.Misto + "-" + b.Misto, // нова назва (об’єднані міста)

a.Vidstan + b.Vidstan, // сумарна відстань

a.Dni + b.Dni, // сумарні дні

a.Transport // лишаємо транспорт від першої (можна інакше)

);

}

public static bool operator >(Podorozh a, Podorozh b)

{

return a.RozrakhVartist() > b.RozrakhVartist();

}

public static bool operator <(Podorozh a, Podorozh b)

{

return a.RozrakhVartist() < b.RozrakhVartist();

}

public static bool operator ==(Podorozh a, Podorozh b)

{

if (ReferenceEquals(a, null) && ReferenceEquals(b, null))

return true;

if (ReferenceEquals(a, null) || ReferenceEquals(b, null))

return false;

return a.Misto == b.Misto && a.Transport == b.Transport;

}

public static bool operator !=(Podorozh a, Podorozh b)

{

return !(a == b);

}

public override bool Equals(object obj)

{

if (obj is Podorozh other)

return this == other;

return false;

}

public static Podorozh operator ++(Podorozh a)

{

if (a == null) return a;

a.Dni += 1;

return a;

}

public static Podorozh operator --(Podorozh a)

{

if (a == null) return a;

a.Dni = a.Dni > 1 ? a.Dni - 1 : 1;

return a;

}

public override int GetHashCode()

{

return (Misto + Transport).GetHashCode();

}

}

**Class2.cs**

public class TurystychnaPodorozh : Podorozh

{

public string TypTuru { get; set; }

public double VartistProzhyvannya { get; set; }

public TurystychnaPodorozh(string misto, double vidstan, int dni, string transport, string typTuru, double vartistProzhyvannya)

: base(misto, vidstan, dni, transport)

{

TypTuru = typTuru;

VartistProzhyvannya = vartistProzhyvannya;

}

public double RozrakhVartistPodorozh()

{

return RozrakhVartist() + VartistProzhyvannya;

}

public override string Info()

{

return $"{base.Info()}, Тип туру: {TypTuru}, Вартість проживання: {VartistProzhyvannya} грн, Загальна вартість: {RozrakhVartistPodorozh()} грн";

}

// Метод для зміни типу туру

public void ZminytyTypTuru(string newType)

{

TypTuru = newType;

}

}

**Class3.cs**

public class BusinessPodorozh : Podorozh

{

public string CompanyName { get; set; }

public string PurposeOfTrip { get; set; }

public BusinessPodorozh(string misto, double vidstan, int dni, string transport, string companyName, string purposeOfTrip)

: base(misto, vidstan, dni, transport)

{

CompanyName = companyName;

PurposeOfTrip = purposeOfTrip;

}

public double RozrakhVartistPodorozh()

{

double businessCosts = 500; // Фіксовані витрати для бізнес-подорожей

return RozrakhVartist() + businessCosts;

}

public override string Info()

{

return $"{base.Info()}, Компанія: {CompanyName}, Мета поїздки: {PurposeOfTrip}, Загальна вартість: {RozrakhVartistPodorozh()} грн";

}

// Метод для зміни типу туру

public void ZminytyTypTuru(string newType)

{

// Додайте логіку для зміни типу туру, якщо це потрібно

}}

**Class4.cs**

public class ExtremePodorozh : Podorozh

{

public string DangerLevel { get; set; }

public ExtremePodorozh(string misto, double vidstan, int dni, string transport, string dangerLevel)

: base(misto, vidstan, dni, transport)

{

DangerLevel = dangerLevel;

}

public double RozrakhVartistPodorozh()

{

double dangerSurcharge = 1000; // Додаткові витрати на екстремальність подорожі

return RozrakhVartist() + dangerSurcharge;

}

public override string Info()

{

return $"{base.Info()}, Рівень небезпеки: {DangerLevel}, Загальна вартість: {RozrakhVartistPodorozh()} грн";

}

// Метод для зміни типу туру

public void ZminytyTypTuru(string newType)

{

// Додайте логіку для зміни типу туру, якщо це потрібно

}

}

**Class5.cs**

public class LongTermPodorozh : Podorozh

{

public int Months { get; set; }

public LongTermPodorozh(string misto, double vidstan, int dni, string transport, int months)

: base(misto, vidstan, dni, transport)

{

Months = months;

}

public double RozrakhVartistPodorozh()

{

double longTermSurcharge = 3000; // Додаткові витрати на довготривалі подорожі

return RozrakhVartist() + longTermSurcharge;

}

public override string Info()

{

return $"{base.Info()}, Тривалість: {Months} місяців, Загальна вартість: {RozrakhVartistPodorozh()} грн";

}

// Метод для зміни типу туру

public void ZminytyTypTuru(string newType)

{

// Додайте логіку для зміни типу туру, якщо це потрібно

}

}

**Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace LR1\_3

{

public partial class Form1 : Form

{

private List<Podorozh> podorozhi = new List<Podorozh>();

public Form1()

{

InitializeComponent();

dataGridView1.AutoGenerateColumns = false;

dataGridView1.Columns.Add("Misto", "Місто");

dataGridView1.Columns.Add("Vidstan", "Відстань (км)");

dataGridView1.Columns.Add("Dni", "Днів");

dataGridView1.Columns.Add("Transport", "Транспорт"); // нова колонка

dataGridView1.Columns.Add("Vartist", "Вартість");

dataGridView1.Columns.Add("Serednya", "Середня ціна/км");

dataGridView1.Columns.Add("TypTuru", "Тип туру");

dataGridView1.Columns.Add("VartistProzhyvannya", "Вартість проживання");

dataGridView1.Columns.Add("ZagalnaVartist", "Загальна вартість");

dataGridView2.AutoGenerateColumns = false;

dataGridView2.Columns.Add("Misto", "Місто");

dataGridView2.Columns.Add("Vidstan", "Відстань (км)");

dataGridView2.Columns.Add("Dni", "Днів");

dataGridView2.Columns.Add("Transport", "Транспорт");

dataGridView2.Columns.Add("Vartist", "Вартість");

dataGridView2.Columns.Add("Serednya", "Середня ціна/км");

// Додаємо елементи в comboBox1 для вибору транспорту

comboBox1.Items.AddRange(new string[] { "Автобус", "Поїзд", "Літак", "Авто" });

comboBox1.SelectedIndex = 0; // за замовчуванням "Автобус"

// Додаємо елементи в comboBox2 для вибору типу туру

comboBox2.SelectedIndex = 0; // за замовчуванням "Екскурсія"

textBox1.KeyPress += TextBox1\_KeyPress;

textBox2.KeyPress += TextBox2\_KeyPress;

textBox3.KeyPress += TextBox3\_KeyPress;

}

private void TextBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsLetter(e.KeyChar) && e.KeyChar != ' ')

{

e.Handled = true; // забороняємо введення

MessageBox.Show("Тільки літери та пробіли дозволені!");

}

}

private void TextBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && e.KeyChar != '.')

{

e.Handled = true;

MessageBox.Show("Тільки цифри та крапка дозволені!");

}

if (e.KeyChar == '.' && (sender as TextBox).Text.Contains("."))

{

e.Handled = true;

}

}

private void TextBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

MessageBox.Show("Тільки цифри дозволені!");

}

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

string misto = textBox1.Text;

double vidstan = double.Parse(textBox2.Text);

int dni = int.Parse(textBox3.Text);

string transport = comboBox1.SelectedItem.ToString(); // вибір транспорту

string typTuru = comboBox2.SelectedItem.ToString(); // вибір типу подорожі

double vartistProzhyvannya = 1000; // можна змінити на введення через текстове поле або комбобокс

// Створюємо відповідний об'єкт в залежності від типу подорожі

Podorozh p;

switch (typTuru)

{

case "Туристична":

p = new TurystychnaPodorozh(misto, vidstan, dni, transport, typTuru, vartistProzhyvannya);

break;

case "Бізнес":

p = new BusinessPodorozh(misto, vidstan, dni, transport, "ACME Corp", "Переговори");

break;

case "Екстремальна":

p = new ExtremePodorozh(misto, vidstan, dni, transport, "Високий");

break;

case "Довготривала":

p = new LongTermPodorozh(misto, vidstan, dni, transport, 6);

break;

default:

throw new Exception("Невідомий тип подорожі");

}

podorozhi.Add(p);

// Додаємо дані в DataGridView

if (p is TurystychnaPodorozh turystychnaTrip)

{

dataGridView1.Rows.Add(

turystychnaTrip.Misto,

turystychnaTrip.Vidstan,

turystychnaTrip.Dni,

turystychnaTrip.Transport,

turystychnaTrip.RozrakhVartist(),

turystychnaTrip.SerednyaTsinaKm().ToString("F2"),

turystychnaTrip.TypTuru,

turystychnaTrip.VartistProzhyvannya,

turystychnaTrip.RozrakhVartistPodorozh()

);

}

else

{

// Для інших типів подорожей, де немає методу RozrakhVartistPodorozh

dataGridView1.Rows.Add(

p.Misto,

p.Vidstan,

p.Dni,

p.Transport,

p.RozrakhVartist(),

p.SerednyaTsinaKm().ToString("F2"),

typTuru,

vartistProzhyvannya,

"N/A"

);

}

// Очищаємо текстові поля після додавання подорожі

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

}

catch

{

MessageBox.Show("Будь ласка, введіть коректні дані!");

}

}

private void створитиToolStripMenuItem\_Click(object sender, EventArgs e)

{

podorozhi.Clear();

dataGridView1.Rows.Clear();

MessageBox.Show("Створено новий список подорожей!");

}

private void зберегтиToolStripMenuItem\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Текстові файли|\*.txt";

saveFileDialog.Title = "Збереження даних";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

using (System.IO.StreamWriter sw = new System.IO.StreamWriter(saveFileDialog.FileName))

{

foreach (var p in podorozhi)

{

sw.WriteLine($"{p.Misto};{p.Vidstan};{p.Dni};{p.Transport};{p.RozrakhVartist()};{p.SerednyaTsinaKm():F2}");

}

}

MessageBox.Show("Дані збережено!");

}

}

private void вивестиToolStripMenuItem\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Текстові файли|\*.txt";

openFileDialog.Title = "Відкрити файл з даними";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

podorozhi.Clear();

dataGridView1.Rows.Clear();

using (System.IO.StreamReader sr = new System.IO.StreamReader(openFileDialog.FileName))

{

string line;

while ((line = sr.ReadLine()) != null)

{

string[] parts = line.Split(';');

// Перевіряємо, що в рядку є хоча б 3 колонки (misto, vidstan, dni)

if (parts.Length >= 3)

{

string misto = parts[0];

double vidstan = double.Parse(parts[1]);

int dni = int.Parse(parts[2]);

// Якщо transport є у файлі, беремо його, інакше дефолт

string transport = parts.Length > 3 ? parts[3] : "Не вказано";

Podorozh p = new Podorozh(misto, vidstan, dni, transport);

podorozhi.Add(p);

// Додаємо в DataGridView, не забуваємо колонку Transport

dataGridView1.Rows.Add(

p.Misto,

p.Vidstan,

p.Dni,

p.Transport,

p.RozrakhVartist(),

p.SerednyaTsinaKm().ToString("F2")

);

}

}

}

MessageBox.Show("Дані завантажено!");

}

}

private void пошукToolStripMenuItem\_Click(object sender, EventArgs e)

{

Form2 formSearch = new Form2(podorozhi);

if (formSearch.ShowDialog() == DialogResult.OK)

{

dataGridView2.Rows.Clear();

foreach (var p in formSearch.SearchResults)

{

dataGridView2.Rows.Add(

p.Misto,

p.Vidstan,

p.Dni,

p.Transport,

p.RozrakhVartist(),

p.SerednyaTsinaKm().ToString("F2")

);

}

}

}

private void RefreshRow(int index)

{

if (index < 0 || index >= podorozhi.Count) return;

Podorozh p = podorozhi[index];

dataGridView1.Rows[index].Cells[0].Value = p.Misto;

dataGridView1.Rows[index].Cells[1].Value = p.Vidstan;

dataGridView1.Rows[index].Cells[2].Value = p.Dni;

dataGridView1.Rows[index].Cells[3].Value = p.Transport;

dataGridView1.Rows[index].Cells[4].Value = p.RozrakhVartist();

dataGridView1.Rows[index].Cells[5].Value = p.SerednyaTsinaKm().ToString("F2");

}

private void обєднатиToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count == 2)

{

// Беремо 2 вибрані рядки

int index1 = dataGridView1.SelectedRows[0].Index;

int index2 = dataGridView1.SelectedRows[1].Index;

Podorozh p1 = podorozhi[index1];

Podorozh p2 = podorozhi[index2];

// Використовуємо оператор +

Podorozh combined = p1 + p2;

// Додаємо нову подорож у список і таблицю

podorozhi.Add(combined);

dataGridView1.Rows.Add(

combined.Misto,

combined.Vidstan,

combined.Dni,

combined.Transport,

combined.RozrakhVartist(),

combined.SerednyaTsinaKm().ToString("F2")

);

MessageBox.Show("Подорожі успішно об’єднано!");

}

else

{

MessageBox.Show("Виберіть рівно 2 подорожі для об’єднання!");

}

}

private void toolStripMenuItem4\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count == 2)

{

int index1 = dataGridView1.SelectedRows[0].Index;

int index2 = dataGridView1.SelectedRows[1].Index;

Podorozh p1 = podorozhi[index1];

Podorozh p2 = podorozhi[index2];

// Використовуємо оператори > і <

if (p1 > p2)

MessageBox.Show($"{p1.Misto} дорожча за {p2.Misto}");

else if (p1 < p2)

MessageBox.Show($"{p2.Misto} дорожча за {p1.Misto}");

else

MessageBox.Show("Подорожі мають однакову вартість!");

}

else

{

MessageBox.Show("Виберіть рівно 2 подорожі для порівняння!");

}

}

private void перевіритиРівністьToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count == 2)

{

int index1 = dataGridView1.SelectedRows[0].Index;

int index2 = dataGridView1.SelectedRows[1].Index;

Podorozh p1 = podorozhi[index1];

Podorozh p2 = podorozhi[index2];

// Використовуємо оператор == і !=

if (p1 == p2)

MessageBox.Show("Подорожі однакові (місто + транспорт)");

else

MessageBox.Show("Подорожі різні");

}

else

{

MessageBox.Show("Виберіть рівно 2 подорожі для перевірки!");

}

}

private void toolStripMenuItem2\_Click(object sender, EventArgs e)

{

int index = -1;

if (dataGridView1.SelectedRows.Count > 0)

index = dataGridView1.SelectedRows[0].Index;

else if (dataGridView1.SelectedCells.Count > 0)

index = dataGridView1.SelectedCells[0].RowIndex;

if (index >= 0 && index < podorozhi.Count)

{

podorozhi[index]++; // мутуємо об’єкт

RefreshRow(index); // оновлюємо таблицю

}

else

{

MessageBox.Show("Виберіть одну подорож!");

}

}

private void toolStripMenuItem3\_Click(object sender, EventArgs e)

{

int index = -1;

if (dataGridView1.SelectedRows.Count > 0)

index = dataGridView1.SelectedRows[0].Index;

else if (dataGridView1.SelectedCells.Count > 0)

index = dataGridView1.SelectedCells[0].RowIndex;

if (index >= 0 && index < podorozhi.Count)

{

podorozhi[index]--; // мутуємо об’єкт

RefreshRow(index); // оновлюємо таблицю

}

else

{

MessageBox.Show("Виберіть одну подорож!");

}

}

private void buttonChangeType\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count == 1)

{

int index = dataGridView1.SelectedRows[0].Index;

// Отримуємо поточну подорож

Podorozh selectedTrip = podorozhi[index]; // Замість TurystychnaPodorozh використовуємо Podorozh для всіх типів

if (selectedTrip != null)

{

// Перевіряємо, чи вибрано значення у comboBox2

if (comboBox2.SelectedItem != null)

{

string newType = comboBox2.SelectedItem.ToString(); // Вибране значення з комбобокса

// Зміна типу туру залежно від того, який тип подорожі

if (selectedTrip is TurystychnaPodorozh turystychnaTrip)

{

turystychnaTrip.ZminytyTypTuru(newType);

}

else if (selectedTrip is BusinessPodorozh businessTrip)

{

businessTrip.ZminytyTypTuru(newType); // Якщо є відповідний метод в класі BusinessPodorozh

}

else if (selectedTrip is ExtremePodorozh extremeTrip)

{

extremeTrip.ZminytyTypTuru(newType); // Якщо є відповідний метод в класі ExtremePodorozh

}

else if (selectedTrip is LongTermPodorozh longTermTrip)

{

longTermTrip.ZminytyTypTuru(newType); // Якщо є відповідний метод в класі LongTermPodorozh

}

// Оновлюємо таблицю після зміни типу

dataGridView1.Rows[index].Cells["TypTuru"].Value = newType;

MessageBox.Show("Тип туру змінено!");

}

else

{

MessageBox.Show("Будь ласка, виберіть тип туру!");

}

}

else

{

MessageBox.Show("Будь ласка, виберіть подорож для зміни типу!");

}

}

else

{

MessageBox.Show("Будь ласка, виберіть одну подорож для зміни типу!");

}

}

}

}

**Form2.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace LR1\_3

{

public partial class Form2 : Form

{

private List<Podorozh> \_list;

public List<Podorozh> SearchResults { get; private set; }

public Form2(List<Podorozh> list)

{

InitializeComponent();

\_list = list;

// Заповнення combobox1 критеріями пошуку

comboBox1.Items.Clear();

comboBox1.Items.Add("Місто");

comboBox1.Items.Add("Транспорт");

comboBox1.Items.Add("Дні");

comboBox1.Items.Add("Тип туру"); // Додаємо новий критерій для пошуку по типу подорожі

comboBox1.Items.Add("Місто + Транспорт + Дні"); // для комбінованого пошуку

comboBox1.SelectedIndex = 0; // за замовчуванням перший елемент

// Заповнення combobox2 транспортом

comboBox2.Items.Clear();

comboBox2.Items.Add("Будь-який"); // для всіх

foreach (var t in \_list.Select(p => p.Transport).Distinct())

{

comboBox2.Items.Add(t);

}

comboBox2.SelectedIndex = 0; // за замовчуванням “Будь-який”

}

private void buttonSearch\_Click(object sender, EventArgs e)

{

string crit = comboBox1.SelectedItem?.ToString(); // критерій пошуку

string mistoText = textBox1.Text.Trim();

string transportText = comboBox2.SelectedItem?.ToString();

string dniText = textBox2.Text.Trim();

var results = \_list.AsEnumerable();

if (!string.IsNullOrEmpty(crit))

{

switch (crit)

{

case "Місто":

if (string.IsNullOrEmpty(mistoText))

{

MessageBox.Show("Будь ласка, введіть місто для пошуку!");

return;

}

results = results.Where(p => p.Misto.IndexOf(mistoText, StringComparison.OrdinalIgnoreCase) >= 0);

break;

case "Транспорт":

if (string.IsNullOrEmpty(transportText))

{

MessageBox.Show("Будь ласка, введіть транспорт для пошуку!");

return;

}

results = results.Where(p => p.Transport.IndexOf(transportText, StringComparison.OrdinalIgnoreCase) >= 0);

break;

case "Дні":

if (string.IsNullOrEmpty(dniText))

{

MessageBox.Show("Будь ласка, введіть кількість днів для пошуку!");

return;

}

if (!int.TryParse(dniText, out int dni))

{

MessageBox.Show("Кількість днів повинна бути числом!");

return;

}

results = results.Where(p => p.Dni == dni);

break;

case "Тип туру": // Додаємо умову для пошуку по типу подорожі

if (string.IsNullOrEmpty(mistoText))

{

MessageBox.Show("Будь ласка, введіть тип туру для пошуку!");

return;

}

results = results.Where(p => (p is TurystychnaPodorozh trip) &&trip.TypTuru.IndexOf(mistoText, StringComparison.OrdinalIgnoreCase) >= 0);

break;

case "Місто + Транспорт + Дні":

if (string.IsNullOrEmpty(mistoText) || string.IsNullOrEmpty(transportText) || string.IsNullOrEmpty(dniText))

{

MessageBox.Show("Будь ласка, заповніть усі поля для комбінованого пошуку!");

return;

}

if (!int.TryParse(dniText, out int dniCombined))

{

MessageBox.Show("Кількість днів повинна бути числом!");

return;

}

results = results.Where(p =>p.Misto.IndexOf(mistoText, StringComparison.OrdinalIgnoreCase) >= 0 &&p.Transport.Equals(transportText, StringComparison.OrdinalIgnoreCase) &&p.Dni == dniCombined);

break;

}

}

var finalResults = results.ToList();

if (finalResults.Count == 0)

{

MessageBox.Show("Нічого не знайдено!");

return;

}

double totalCost = finalResults.Sum(p => p.RozrakhVartist());

double avgDays = finalResults.Average(p => p.Dni);

double minDistance = finalResults.Min(p => p.Vidstan);

double maxDistance = finalResults.Max(p => p.Vidstan);

MessageBox.Show($"Знайдено {finalResults.Count} подорожей\n" +$"Сумарна вартість: {totalCost}\n" +$"Середня кількість днів: {avgDays:F1}\n" +$"Відстань: від {minDistance} до {maxDistance} км"

);

SearchResults = finalResults;

this.DialogResult = DialogResult.OK;

this.Close();

}

}

}

**Результат програми:**

|  |  |
| --- | --- |
|  | Додамо дані в нашу програму. Протестуємо зміну типів подорожей на інші. Для цього виділимо рядок з подорожжю.  Змінимо першу подорож з бізнес на туристичну. |
|  | Програма вивела повідомлення про успішну зміну типу подорожі. Протестуємо з другою подорожжю, змінимо її на бізнес. |
|  | Програма вивела успішне повідомлення про зміну типу подорожжі. |

**Висновок:** Під час виконання лабораторної роботи я одержав практичні навички створення ієрархії класів і використання статичних компонентів класу.