Лабораторна робота №5

**Тема:** Ієрархія об'єктів і групи. Агрегація.Ітератори.

**Мета:** Одержати практичні навички створення об'єктів-груп (агрегація, композиція) і використання методів-ітераторів.

**Порядок виконання роботи.**

1. Доповнити ієрархію класів лабораторної роботи №3 класами “група”.

Наприклад, для предметної області ВІДДІЛЕННЯ можна запропонувати класи “ВІДДІЛЕННЯ”, “студентська група”, “кафедра”. Рекомендується створити абстрактний клас - “підрозділ”, що буде предком усіх груп і абстрактний клас TObject, що знаходиться на чолі всієї ієрархії.

2.Написати для класу-групи метод - ітератор.

3.Написати процедуру функції, що виконується для всіх об'єктів, що входять у групу (дивися приклади в додатку).

4.Написати демонстраційну програму, у якій створюються, показуються і руйнуються об'єкти-групи, а також демонструється використання ітератора.

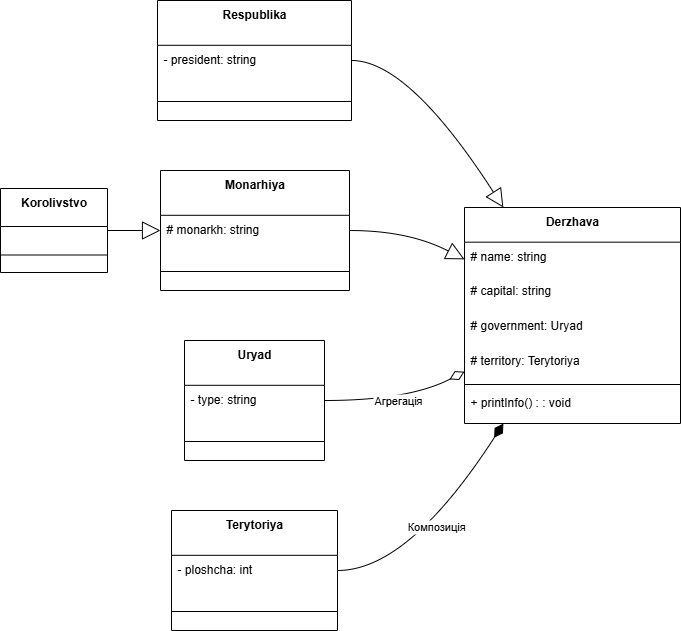
**Завдання.**

Створити агрегативну залежність між класами створеними на попередніх лабораторних роботах.

**Задача №1**

Дано перелік класів. Побудуйте об’єктну модель предметної області, враховуючи, види залежностей між класами. Реалізуйте проект за ООП. **А також додати агрегацію та композицію.**

13. республіка, монархія, королівство, держава

****

**Код програми:**

**Main.cpp (Головна програма)**

#include <iostream>

#include <string>

#include <iomanip>

#include "Derzhava.h"

#include "Respublika.h"

#include "Monarhiya.h"

#include "Korolivstvo.h"

#include "Uryad.h"

#include "Terytoriya.h"

int main() {

setlocale(0, "ukr");

// Об'єкти для агрегації

Uryad pres\_parl("Президентсько-парламентська");

Uryad const\_mon("Конституційна монархія");

Uryad parl\_mon("Парламентська монархія");

Derzhava\* catalog[100];

int count = 0;

// Ініціалізуємо дані, додаючи елементи в масив

catalog[count++] = new Respublika("Україна", "Київ", 43, &pres\_parl, Terytoriya(603628), "В. Зеленський");

catalog[count++] = new Monarhiya("Велика Британія", "Лондон", 67, &const\_mon, Terytoriya(243610), "Чарльз III");

catalog[count++] = new Korolivstvo("Іспанія", "Мадрид", 47, &parl\_mon, Terytoriya(505990), "Філіп VI");

// Виводимо шапку таблиці

printHeader();

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

catalog[i]->printInfo();

}

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

delete catalog[i];

}

return 0;

}

// Функція для виводу шапки таблиці

void printHeader() {

cout << left << setw(30) << "Базові поля" << setw(30) << "Поля нащадків" << setw(25) << "Композиція" << setw(30) << "Агрегація" << endl;

cout << string(115, '-') << endl;

}

**Derzhava.h**

#pragma once

#include <iostream>

#include <string>

#include <iomanip>

#include "Uryad.h"

#include "Terytoriya.h"

using namespace std;

class Derzhava {

protected:

string name;

string capital;

int population;

Uryad\* government; // Агрегація (вказівник)

Terytoriya territory; // Композиція (вбудований об'єкт)

public:

Derzhava(string n, string c, int p, Uryad\* g, Terytoriya t);

virtual ~Derzhava() {}

virtual void printInfo() const;

};

**Derzhava.cpp**

#include "Derzhava.h"

Derzhava::Derzhava(string n, string c, int p, Uryad\* g, Terytoriya t)

: name(n), capital(c), population(p), government(g), territory(t) {

}

void Derzhava::printInfo() const {

cout << left<< setw(30) << (name + ", " + capital)<< setw(30) << ("Населення: " + to\_string(population) + " млн")<< setw(25) << territory.getInfo()<< setw(30) << government->getType() << endl;

}

**Respublika.h**

#pragma once

#include "Derzhava.h"

class Respublika : public Derzhava {

string president;

public:

Respublika(string n, string c, int p, Uryad\* g, Terytoriya t, string pres);

void printInfo() const override;

};

**Respublika.cpp**

#include "Respublika.h"

Respublika::Respublika(string n, string c, int p, Uryad\* g, Terytoriya t, string pres)

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

}

void Respublika::printInfo() const {

cout << left << setw(30) << (name + ", " + capital)<< setw(30) << ("Президент: " + president)<< setw(25) << territory.getInfo()<< setw(30) << government->getType() << endl;

}

**Monarhiya.h**

#pragma once

#include "Derzhava.h"

class Monarhiya : public Derzhava {

protected:

string monarkh;

public:

Monarhiya(string n, string c, int p, Uryad\* g, Terytoriya t, string m);

void printInfo() const override;

};

**Monarhiya.cpp**

#include "Monarhiya.h"

Monarhiya::Monarhiya(string n, string c, int p, Uryad\* g, Terytoriya t, string m)

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

}

void Monarhiya::printInfo() const {

cout << left<< setw(30) << (name + ", " + capital)<< setw(30) << ("Монарх: " + monarkh)<< setw(25) << territory.getInfo()<< setw(30) << government->getType() << endl;

}

**Korolivstvo.h**

#pragma once

#include "Monarhiya.h"

class Korolivstvo : public Monarhiya {

public:

Korolivstvo(string n, string c, int p, Uryad\* g, Terytoriya t, string k);

void printInfo() const override;

};

**Korolivstvo.cpp**

#include "Korolivstvo.h"

Korolivstvo::Korolivstvo(string n, string c, int p, Uryad\* g, Terytoriya t, string k)

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

}

void Korolivstvo::printInfo() const {

cout << left<< setw(30) << (name + ", " + capital)<< setw(30) << ("Король: " + monarkh)<< setw(25) << territory.getInfo()<< setw(30) << government->getType() << endl;

}

**Uryad.h (Агрегація)**

#pragma once

#include <string>

class Uryad {

string type;

public:

Uryad(string t = "Невідомий");

string getType() const;

};

**Uryad.cpp (Агрегація)**

#include "Uryad.h"

Uryad::Uryad(string t) : type(t) {}

string Uryad::getType() const {

return type;

}

**Terytoriya.h**

#pragma once

#include <string>

using namespace std;

class Terytoriya {

int ploshcha;

public:

Terytoriya(int p = 0);

string getInfo() const;

};

**Terytoriya.cpp**

#include "Terytoriya.h"

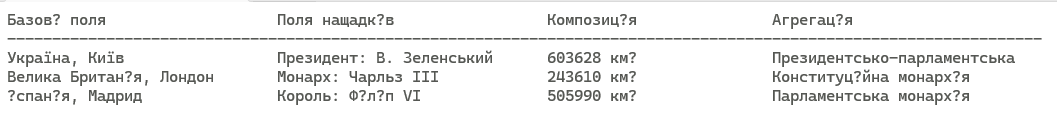
Terytoriya::Terytoriya(int p) : ploshcha(p) {}

string Terytoriya::getInfo() const {

return to\_string(ploshcha) + " км²";

}

**Результат програми:**

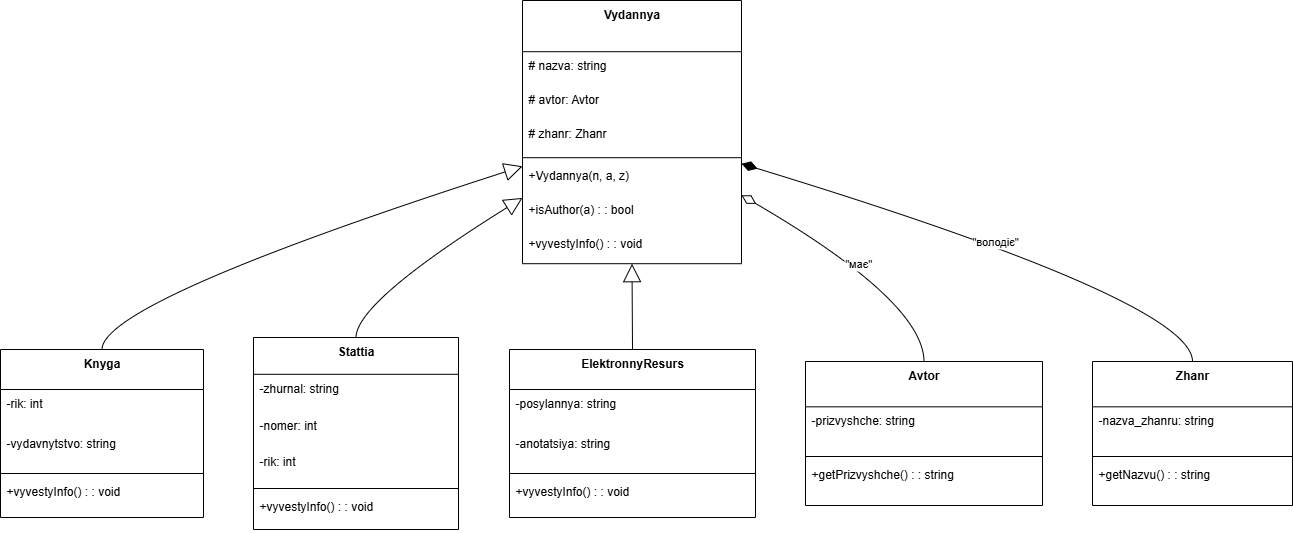
****

**Задача №2**

**Варіант 3 (Вдосконалити на агрегацію та композицію)**

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.

Створити каталог (масив) з n видань, вивести повну інформацію з каталога, а також організувати пошук видань по прізвищу автора.

****

**Код програми**

**Main.cpp (Головна програма)**

#include "Книга.h"

#include "Стаття.h"

#include "ЕлектроннийРесурс.h"

void printHeader() {

cout << left<< setw(20) << "Тип видання"<< setw(30) << "Базові поля (Назва)"<< setw(35) << "Поля нащадків"<< setw(20) << "Композиція (Жанр)"<< setw(20) << "Агрегація (Автор)" << endl;

cout << string(125, '-') << endl;

}

int main() {

setlocale(0, "ukr");

// === Create objects for aggregation (they exist independently) ===

Avtor pavlenko("Pavlenko");

Avtor ivanenko("Ivanenko");

Avtor cormen("Cormen");

Avtor guido("Guido van Rossum");

const int n = 4;

Vydannya\* catalog[n] = {

new Knyga("Philosophy of Code", &pavlenko, Zhanr("IT"), 2021, "Lviv Polytechnic"),

new Stattia("Machine Learning", &ivanenko, Zhanr("Scientific"), "Tech Today", 5, 2022),

new Knyga("Discrete Mathematics", &cormen, Zhanr("Educational"), 2017, "MIT Press"),

new ElektronnyResurs("Python Documentation", &guido, Zhanr("Technical"), "docs.python.org", "Official Docs")

};

int choice;

do {

cout << "\nMENU\n";

cout << "1. Переглянути усі видання\n";

cout << "2. Пошук за автором\n";

cout << "3. Вихід\n";

cout << "Введіть вибір: ";

cin >> choice;

cin.ignore(); // clear buffer after cin

switch (choice) {

case 1: {

cout << "\nFull Catalog\n";

printHeader();

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

catalog[i]->vyvestyInfo();

}

break;

}

case 2: {

string author\_lastname;

cout << "Введіть прізвище автора: ";

getline(cin, author\_lastname);

cout << "\nРезультати пошуку за прізвищем '" << author\_lastname << "'\n";

printHeader();

bool found = false;

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

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

catalog[i]->vyvestyInfo();

found = true;

}

}

if (!found) cout << "Не знайдено нічого.\n";

break;

}

case 3:

cout << "Вихід...\n";

break;

default:

cout << "Неправильний вибір, повторіть спробу.\n";

}

} while (choice != 3);

// Freeing memory

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

delete catalog[i];

}

return 0;

}

**Видання.h**

#pragma once

#include <iostream>

#include <string>

#include <iomanip>

#include "Avtor.h"

#include "Zhanr.h"

class Vydannya {

protected:

string nazva;

Avtor\* avtor; // Агрегація

Zhanr zhanr; // Композиція

public:

Vydannya(string n, Avtor\* a, Zhanr z);

virtual ~Vydannya() {}

virtual void vyvestyInfo() const = 0;

virtual bool isAuthor(const string& a) const;

};

**Видання.cpp**

#include "Видання.h"

Vydannya::Vydannya(string n, Avtor\* a, Zhanr z)

: nazva(n), avtor(a), zhanr(z) {

}

// Реалізація методу isAuthor у базовому класі

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

return avtor->getPrizvyshche() == a;

}

**Електроний ресурс.h**

#pragma once

#include "Видання.h"

class ElektronnyResurs : public Vydannya {

string posylannya;

string anotatsiya;

public:

ElektronnyResurs(string n, Avtor\* a, Zhanr z, string p, string an);

void vyvestyInfo() const override;

};

**Електронний ресурс.cpp**

#include "ЕлектроннийРесурс.h"

ElektronnyResurs::ElektronnyResurs(string n, Avtor\* a, Zhanr z, string p, string an)

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

}

void ElektronnyResurs::vyvestyInfo() const {

cout << left<< setw(20) << "Електронний ресурс"<< setw(30) << nazva<< setw(35) << posylannya<< setw(20) << zhanr.getNazvu()<< setw(20) << avtor->getPrizvyshche() << endl;

}

**Книга.h**

#pragma once

#include "Видання.h"

class Knyga : public Vydannya {

int rik;

string vydavnytstvo;

public:

Knyga(string n, Avtor\* a, Zhanr z, int r, string v);

void vyvestyInfo() const override;

};

**Книга.cpp**

#include "Книга.h"

Knyga::Knyga(string n, Avtor\* a, Zhanr z, int r, string v)

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

}

void Knyga::vyvestyInfo() const {

cout << left<< setw(20) << "Книга"<< setw(30) << nazva<< setw(35) << (to\_string(rik) + ", " + vydavnytstvo)<< setw(20) << zhanr.getNazvu()<< setw(20) << avtor->getPrizvyshche() << endl;

}

**Стаття.h**

#pragma once

#include "Видання.h"

class Stattia : public Vydannya {

string zhurnal;

int nomer;

int rik;

public:

Stattia(string n, Avtor\* a, Zhanr z, string zh, int num, int r);

void vyvestyInfo() const override;

};

**Стаття.cpp**

#include "Стаття.h"

Stattia::Stattia(string n, Avtor\* a, Zhanr z, string zh, int num, int r)

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

}

void Stattia::vyvestyInfo() const {

cout << left<< setw(20) << "Стаття"<< setw(30) << nazva<< setw(35) << (zhurnal + ", №" + to\_string(nomer) + ", " + to\_string(rik))<< setw(20) << zhanr.getNazvu()<< setw(20) << avtor->getPrizvyshche() << endl;

}

**Avtor.h (Агрегація)**

#pragma once

#include <string>

using namespace std;

class Avtor {

string prizvyshche;

public:

Avtor(string p) : prizvyshche(p) {}

string getPrizvyshche() const {

return prizvyshche;

}

};

**Zhanr.h (Композиція)**

#pragma once

#include <string>

using namespace std;

class Zhanr {

string nazva\_zhanru;

public:

Zhanr(string n) : nazva\_zhanru(n) {}

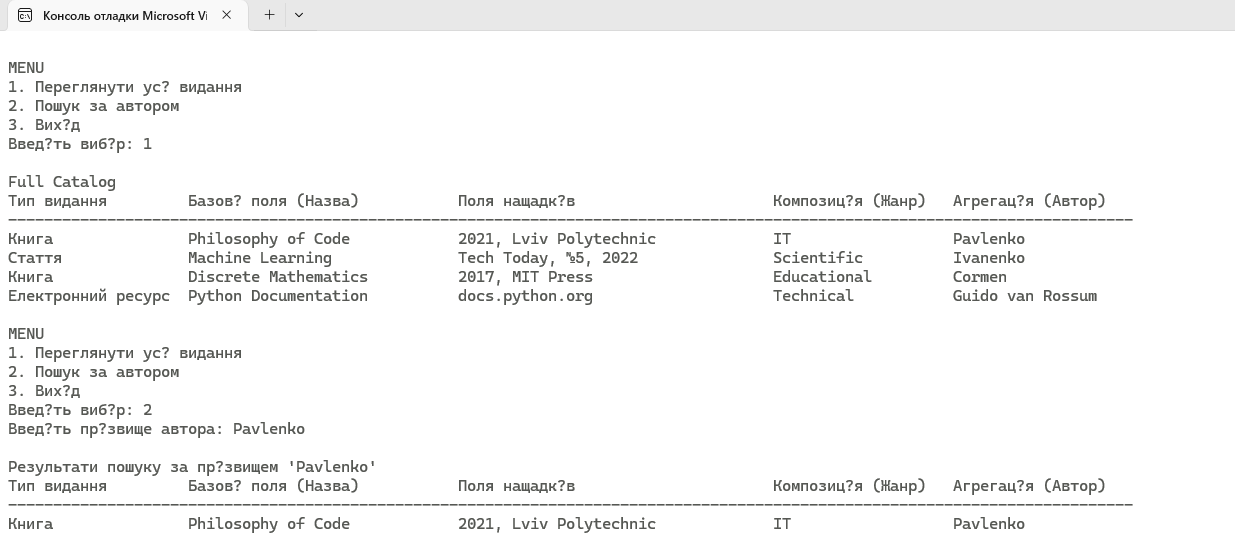
string getNazvu() const {

return nazva\_zhanru;

}

};

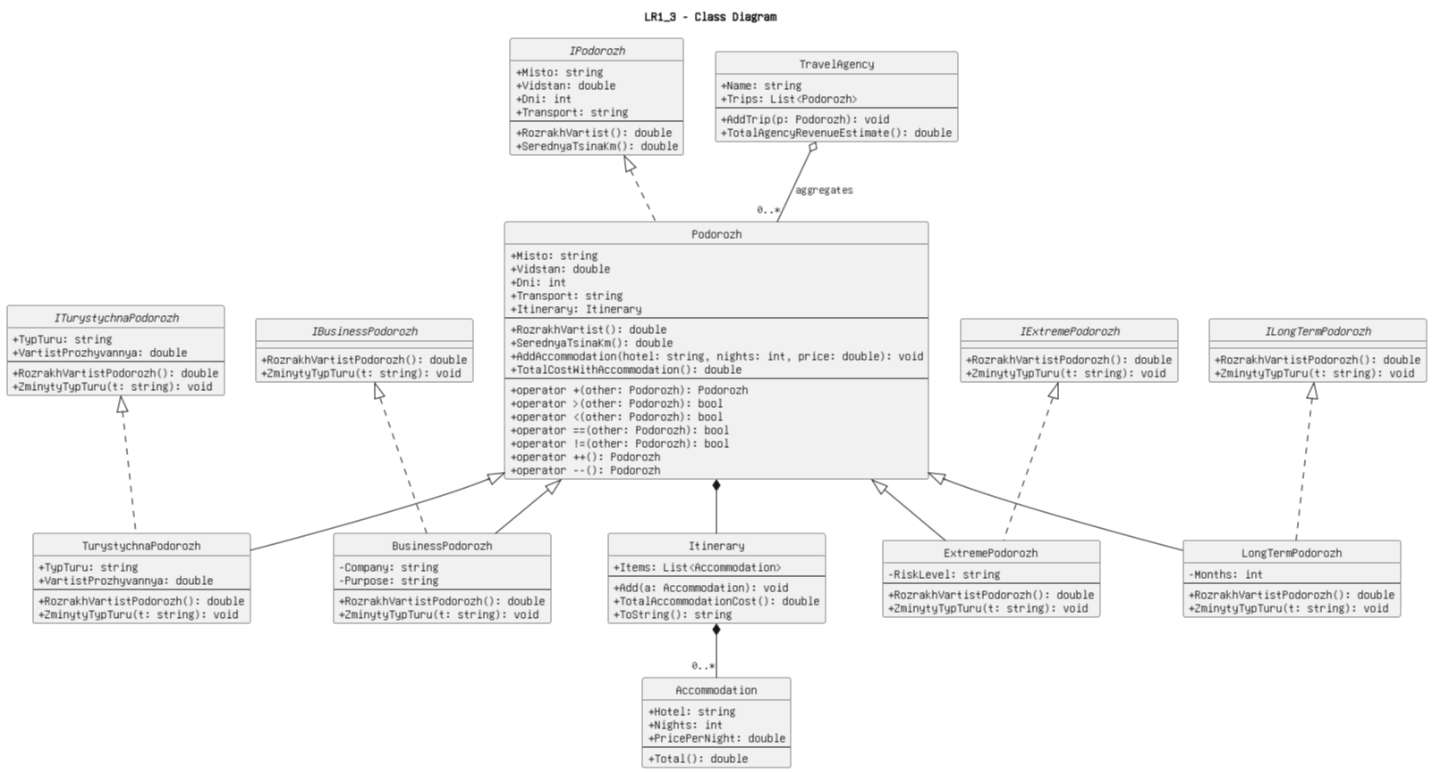
**Результат програми:**

****

**Задача №3 (Windows Forms)**

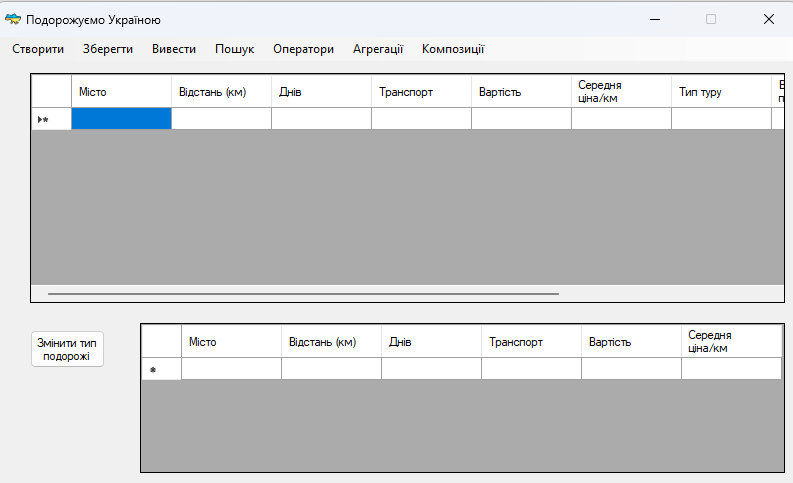
Додати агрегації та композиції до проєкту.

**Ієрархія діаграма класу**

****

**Макет форми**

На формі були текст бокси, які видалені і додані в вкладку Створити. Також додані вкладки Агрегації та Композиції.



**Код програми**

**Класи**

**Podorozh.cs**

//Podorozh.cs

using LR1\_3;

using System;

public class Podorozh : IPodorozh

{

public string Misto { get; set; }

public double Vidstan { get; set; }

public int Dni { get; set; }

public string Transport { get; set; }

private readonly Itinerary itinerary = new Itinerary();

public Itinerary Itinerary => itinerary;

// Помічник для додавання житла

public void AddAccommodation(string hotelName, int nights, double pricePerNight)

{

var acc = new Accommodation(hotelName, nights, pricePerNight);

itinerary.AddAccommodation(acc);

}

// Оцінка загальної вартості подорожі з урахуванням проживання (best-effort)

public double TotalCostWithAccommodation()

{

double baseCost = 0;

try

{

var m = this.GetType().GetMethod("RozrakhVartistPodorozh") ?? this.GetType().GetMethod("RozrakhVartist");

if (m != null)

baseCost = Convert.ToDouble(m.Invoke(this, null));

}

catch { /\* ignore \*/ }

double accom = itinerary.TotalAccommodationCost();

return baseCost + accom;

}

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;

}

public virtual string Info()

{

return $"Місто: {Misto}, Відстань: {Vidstan} км, Дні: {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();

}

}

**BusinessPodorozh.cs**

using LR1\_3;

public class BusinessPodorozh : Podorozh, IBusinessPodorozh

{

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

}

}

**TurystychnaPodorozh.cs**

using LR1\_3;

public class TurystychnaPodorozh : Podorozh, ITurystychnaPodorozh

{

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;

}

}

**ExtremePodorozh.cs**

//ExtremePodorozh.cs

using LR1\_3;

public class ExtremePodorozh : Podorozh, IExtremePodorozh

{

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

}

}

**LongTermPodorozh.cs**

//LongTermPodorozh.cs

using LR1\_3;

public class LongTermPodorozh : Podorozh, ILongTermPodorozh

{

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)

{

}

}

**Інтерфейси**

**IPodorozh.cs**

namespace LR1\_3

{

public interface IPodorozh

{

string Misto { get; set; }

double Vidstan { get; set; }

int Dni { get; set; }

string Transport { get; set; }

string Info(); //Метод для отримання інформації про подорож

double RozrakhVartist(); //Метод для розрахунку вартості подорожі

double SerednyaTsinaKm(); //Метод для розрахунку середньої ціни за км

}

}

**IBusinessPodorozh.cs**

namespace LR1\_3

{

public interface IBusinessPodorozh : IPodorozh

{

// Унікальні властивості

string CompanyName { get; set; }

string PurposeOfTrip { get; set; }

// Унікальні методи

double RozrakhVartistPodorozh();

void ZminytyTypTuru(string newType);

}

}

**IExtremePodorozh.cs**

namespace LR1\_3

{

public interface IExtremePodorozh : IPodorozh

{

// Унікальна властивість

string DangerLevel { get; set; }

// Унікальні методи

double RozrakhVartistPodorozh();

void ZminytyTypTuru(string newType);

}

}

**IlongTermPodorozh.cs**

namespace LR1\_3

{

public interface ILongTermPodorozh : IPodorozh

{

// Унікальна властивість

int Months { get; set; }

// Унікальні методи

double RozrakhVartistPodorozh();

void ZminytyTypTuru(string newType);

}

}

**Композиція**

**Accomindation.cs**

using System;

namespace LR1\_3

{

public class Accommodation

{

public string Name { get; private set; }

public int Nights { get; private set; }

public double PricePerNight { get; private set; }

public Accommodation(string name, int nights, double pricePerNight)

{

if (string.IsNullOrWhiteSpace(name)) throw new ArgumentException("Name required");

if (nights < 1) throw new ArgumentOutOfRangeException(nameof(nights));

if (pricePerNight < 0) throw new ArgumentOutOfRangeException(nameof(pricePerNight));

Name = name;

Nights = nights;

PricePerNight = pricePerNight;

}

public double TotalPrice() => Nights \* PricePerNight;

public override string ToString()

{

return $"{Name} — {Nights} night(s), {TotalPrice():0.##} грн ({PricePerNight:0.##}/night)";

}

}

}

**Itinerary.cs**

using System;

using System.Collections.Generic;

using System.Linq;

namespace LR1\_3

{

// Це об'єкт, який належить конкретній подорожі.

public class Itinerary

{

private readonly List<Accommodation> accommodations = new List<Accommodation>();

public IReadOnlyList<Accommodation> Accommodations => accommodations.AsReadOnly();

public void AddAccommodation(Accommodation acc)

{

if (acc == null) throw new ArgumentNullException(nameof(acc));

accommodations.Add(acc);

}

public bool RemoveAccommodation(Accommodation acc) => accommodations.Remove(acc);

public double TotalAccommodationCost() => accommodations.Sum(a => a.TotalPrice());

public override string ToString()

{

if (!accommodations.Any()) return "No accommodations.";

return string.Join(Environment.NewLine, accommodations.Select(a => a.ToString()));

}

}

}

**Агрегація**

**TravelAgency.cs**

using System;

using System.Collections.Generic;

using System.Linq;

namespace LR1\_3

{

public class TravelAgency

{

private readonly List<Podorozh> trips = new List<Podorozh>();

public string Name { get; set; }

public TravelAgency(string name)

{

Name = string.IsNullOrWhiteSpace(name) ? "Unnamed Agency" : name;

}

public IReadOnlyList<Podorozh> Trips => trips.AsReadOnly();

public void AddTrip(Podorozh trip)

{

if (trip == null) throw new ArgumentNullException(nameof(trip));

if (!trips.Contains(trip))

trips.Add(trip);

}

public bool RemoveTrip(Podorozh trip) => trips.Remove(trip);

public IEnumerable<Podorozh> GetTripsByCity(string city)

{

if (string.IsNullOrWhiteSpace(city)) return Enumerable.Empty<Podorozh>();

return trips.Where(t => t.Misto?.IndexOf(city, StringComparison.OrdinalIgnoreCase) >= 0);

}

public double TotalAgencyRevenueEstimate()

{

// Оцінка: сумарна вартість усіх подорожей (якщо класи Podorozh мають метод розрахунку вартості)

double sum = 0;

foreach (var t in trips)

{

try

{

// Ми робимо best-effort — якщо у Podorozh є RozrakhVartistPodorozh або RozrakhVartist — використовуємо

var method = t.GetType().GetMethod("RozrakhVartistPodorozh");

if (method != null)

{

sum += Convert.ToDouble(method.Invoke(t, null));

continue;

}

method = t.GetType().GetMethod("RozrakhVartist");

if (method != null)

sum += Convert.ToDouble(method.Invoke(t, null));

}

catch

{

}

}

return sum;

}

}

}

**Форми**

**Main Form**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace LR1\_3

{

public partial class Form1 : Form

{

private TravelAgency agency;

private Podorozh currentTrip;

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

private readonly Dictionary<Podorozh, string> tripAgencyMap = new Dictionary<Podorozh, string>();

private readonly Dictionary<string, TravelAgency> agencies = new Dictionary<string, TravelAgency>();

public Form1()

{

InitializeComponent();

agency = new TravelAgency("Моя агенція");

currentTrip = null;

agencies["Моя агенція"] = agency;

dataGridView1.AutoGenerateColumns = false;

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

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

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

dataGridView1.Columns.Add("Transport", "Транспорт");

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");

dataGridView1.Columns.Add("AgencyName", "Агенція");

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");

dataGridView1.SelectionChanged += DataGridView1\_SelectionChanged;

}

private void DataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count == 1)

{

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

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

currentTrip = podorozhi[idx];

}

}

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

{

var inputForm = new Form

{

Text = "Створити подорож",

Width = 380,

Height = 340,

StartPosition = FormStartPosition.CenterParent,

FormBorderStyle = FormBorderStyle.FixedDialog,

MaximizeBox = false,

MinimizeBox = false

};

var lblMisto = new Label { Text = "Місто:", Left = 12, Top = 20, Width = 140 };

var lblVidstan = new Label { Text = "Відстань (км):", Left = 12, Top = 60, Width = 140 };

var lblDni = new Label { Text = "Кількість днів:", Left = 12, Top = 100, Width = 140 };

var lblTransport = new Label { Text = "Вид транспорту:", Left = 12, Top = 140, Width = 140 };

var lblTypTuru = new Label { Text = "Вид подорожі:", Left = 12, Top = 180, Width = 140 };

var txtMisto = new TextBox { Left = 160, Top = 18, Width = 190 };

var txtVidstan = new TextBox { Left = 160, Top = 58, Width = 190 };

var txtDni = new TextBox { Left = 160, Top = 98, Width = 190 };

var cbTransport = new ComboBox

{

Left = 160,

Top = 138,

Width = 190,

DropDownStyle = ComboBoxStyle.DropDownList

};

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

cbTransport.SelectedIndex = 0;

var cbTypTuru = new ComboBox

{

Left = 160,

Top = 178,

Width = 190,

DropDownStyle = ComboBoxStyle.DropDownList

};

cbTypTuru.Items.AddRange(new[] { "Туристична", "Бізнес", "Екстремальна", "Довготривала" });

cbTypTuru.SelectedIndex = 0;

var btnOK = new Button { Text = "OK", Left = 110, Top = 235, Width = 100, DialogResult = DialogResult.OK };

var btnCancel = new Button { Text = "Скасувати", Left = 220, Top = 235, Width = 130, DialogResult = DialogResult.Cancel };

// прості перевірки вводу на льоту

txtVidstan.KeyPress += (s, ke) =>

{

if (!char.IsControl(ke.KeyChar) && !char.IsDigit(ke.KeyChar) && ke.KeyChar != '.') ke.Handled = true;

if (ke.KeyChar == '.' && (s as TextBox).Text.Contains(".")) ke.Handled = true;

};

txtDni.KeyPress += (s, ke) =>

{

if (!char.IsControl(ke.KeyChar) && !char.IsDigit(ke.KeyChar)) ke.Handled = true;

};

inputForm.Controls.AddRange(new Control[]

{

lblMisto, lblVidstan, lblDni, lblTransport, lblTypTuru,

txtMisto, txtVidstan, txtDni, cbTransport, cbTypTuru,

btnOK, btnCancel

});

inputForm.AcceptButton = btnOK;

inputForm.CancelButton = btnCancel;

if (inputForm.ShowDialog() != DialogResult.OK) return;

string misto = txtMisto.Text.Trim();

if (string.IsNullOrWhiteSpace(misto))

{

MessageBox.Show("Вкажіть місто.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (!double.TryParse(txtVidstan.Text, out double vidstan))

{

MessageBox.Show("Невірний формат відстані.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (!int.TryParse(txtDni.Text, out int dni))

{

MessageBox.Show("Невірний формат кількості днів.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string transport = cbTransport.SelectedItem?.ToString() ?? "Автобус";

string typTuru = cbTypTuru.SelectedItem?.ToString() ?? "Туристична";

IPodorozh p;

switch (typTuru)

{

case "Туристична":

double vartistProzh = 1000;

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

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:

MessageBox.Show("Невідомий тип подорожі.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string vartistProzhyvannyaStr = "N/A";

string zagalnaVartistStr = "N/A";

string realTypTuru = typTuru;

if (p is ITurystychnaPodorozh t)

{

vartistProzhyvannyaStr = t.VartistProzhyvannya.ToString();

zagalnaVartistStr = t.RozrakhVartistPodorozh().ToString("F2");

realTypTuru = t.TypTuru;

}

else if (p is IBusinessPodorozh b)

{

zagalnaVartistStr = b.RozrakhVartistPodorozh().ToString("F2");

}

else if (p is IExtremePodorozh ex)

{

zagalnaVartistStr = ex.RozrakhVartistPodorozh().ToString("F2");

}

else if (p is ILongTermPodorozh l)

{

zagalnaVartistStr = l.RozrakhVartistPodorozh().ToString("F2");

}

currentTrip = (Podorozh)p;

podorozhi.Add(currentTrip);

dataGridView1.Rows.Add(

currentTrip.Misto,

currentTrip.Vidstan,

currentTrip.Dni,

currentTrip.Transport,

currentTrip.RozrakhVartist(),

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

realTypTuru,

vartistProzhyvannyaStr,

zagalnaVartistStr,

tripAgencyMap.TryGetValue(currentTrip, out var agName2) ? agName2 : "—"

);

MessageBox.Show("Подорож створено і додано в таблицю.", "OK", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

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

{

if (dataGridView1.SelectedRows.Count != 1)

{

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

return;

}

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

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

{

MessageBox.Show("Некоректний вибір рядка.");

return;

}

var selectedTrip = podorozhi[index];

// маленька форма з ComboBox для вибору типу

string newType = ShowTripTypePicker();

if (string.IsNullOrEmpty(newType)) return; // скасували

// якщо у підкласів є метод ZminytyTypTuru — викликаємо

if (selectedTrip is TurystychnaPodorozh tur)

tur.ZminytyTypTuru(newType);

else if (selectedTrip is BusinessPodorozh biz)

biz.ZminytyTypTuru(newType);

else if (selectedTrip is ExtremePodorozh ext)

ext.ZminytyTypTuru(newType);

else if (selectedTrip is LongTermPodorozh lng)

lng.ZminytyTypTuru(newType);

// відобразити у гріді

if (dataGridView1.Columns.Contains("TypTuru"))

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

MessageBox.Show("Тип туру змінено!");

}

// Підбір типу подорожі — повертає обраний рядок або null

private string ShowTripTypePicker()

{

var f = new Form

{

Text = "Вибір типу туру",

Width = 320,

Height = 170,

StartPosition = FormStartPosition.CenterParent,

FormBorderStyle = FormBorderStyle.FixedDialog,

MaximizeBox = false,

MinimizeBox = false

};

var lbl = new Label { Text = "Оберіть тип:", Left = 12, Top = 20, Width = 100 };

var cb = new ComboBox

{

Left = 120,

Top = 16,

Width = 170,

DropDownStyle = ComboBoxStyle.DropDownList

};

cb.Items.AddRange(new[] { "Туристична", "Бізнес", "Екстремальна", "Довготривала" });

cb.SelectedIndex = 0;

var ok = new Button { Text = "OK", Left = 70, Top = 70, Width = 80, DialogResult = DialogResult.OK };

var cancel = new Button { Text = "Скасувати", Left = 160, Top = 70, Width = 100, DialogResult = DialogResult.Cancel };

f.Controls.AddRange(new Control[] { lbl, cb, ok, cancel });

f.AcceptButton = ok;

f.CancelButton = cancel;

return f.ShowDialog() == DialogResult.OK ? cb.SelectedItem?.ToString() : null;

}

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)

{

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

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

Podorozh p1 = podorozhi[index1];

Podorozh p2 = podorozhi[index2];

Podorozh combined = p1 + p2;

podorozhi.Add(combined);

dataGridView1.Rows.Add(

combined.Misto,

combined.Vidstan,

combined.Dni,

combined.Transport,

combined.RozrakhVartist(),

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

"—",

"—",

combined.RozrakhVartist().ToString("F2"),

tripAgencyMap.TryGetValue(combined, out var agName3) ? agName3 : "—"

);

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];

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];

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]++;

RefreshRow(index);

}

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]--;

RefreshRow(index);

}

else

{

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

}

}

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

{

}

// Додати житло: у виділену або поточну подорож; негайне оновлення таблиці

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

{

Podorozh targetTrip = null;

if (dataGridView1.SelectedRows.Count == 1)

{

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

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

targetTrip = podorozhi[idx];

}

if (targetTrip == null) targetTrip = currentTrip;

if (targetTrip == null)

{

MessageBox.Show("Оберіть подорож у таблиці або створіть поточну подорож.", "Помилка");

return;

}

string hotel = Prompt.ShowDialog("Введіть назву житла:", "Житло");

if (hotel == null) return;

string nightsStr = Prompt.ShowDialog("Кількість ночей:", "Житло");

if (nightsStr == null) return;

string priceStr = Prompt.ShowDialog("Ціна за ніч:", "Житло");

if (priceStr == null) return;

if (!int.TryParse(nightsStr, out int nights) || !double.TryParse(priceStr, out double price))

{

MessageBox.Show("Невірний формат чисел!", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

targetTrip.AddAccommodation(hotel, nights, price);

UpdateGridTotalsForTrip(targetTrip);

MessageBox.Show("Житло додано!", "OK");

}

// Додати подорож до агенції з введенням назви; відобразити у гріді

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

{

if (currentTrip == null)

{

MessageBox.Show("Спочатку створіть подорож!");

return;

}

string agencyName = Prompt.ShowDialog("Введіть назву агенції:", "Агенція");

if (string.IsNullOrWhiteSpace(agencyName))

return;

if (!agencies.TryGetValue(agencyName, out var ag))

{

ag = new TravelAgency(agencyName);

agencies[agencyName] = ag;

}

ag.AddTrip(currentTrip);

tripAgencyMap[currentTrip] = agencyName;

int idxRow = FindRowIndexForTrip(currentTrip);

if (idxRow >= 0 && dataGridView1.Columns.Contains("AgencyName"))

dataGridView1.Rows[idxRow].Cells["AgencyName"].Value = agencyName;

MessageBox.Show("Подорож додано до агенції!");

}

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

{

var sb = new StringBuilder();

sb.AppendLine("Подорожі за агенціями:");

foreach (var kv in agencies)

{

string name = kv.Key;

var ag = kv.Value;

sb.AppendLine($"— {name}:");

foreach (var trip in ag.Trips)

{

sb.AppendLine($" {trip.Misto} — {trip.TotalCostWithAccommodation():0.##} грн");

}

}

MessageBox.Show(sb.ToString());

}

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

{

double total = 0;

foreach (var ag in agencies.Values)

total += ag.TotalAgencyRevenueEstimate();

MessageBox.Show($"Загальний орієнтовний дохід (усі агенції): {total:0.##} грн");

}

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

{

if (currentTrip == null)

{

MessageBox.Show("Спочатку створіть подорож!");

return;

}

MessageBox.Show(currentTrip.Itinerary.ToString(), "Житло подорожі");

}

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

{

if (currentTrip == null)

{

MessageBox.Show("Спочатку створіть подорож!");

return;

}

double total = currentTrip.TotalCostWithAccommodation();

MessageBox.Show($"Повна вартість подорожі (з житлом): {total:0.##} грн");

}

private int FindRowIndexForTrip(Podorozh trip)

{

return podorozhi.IndexOf(trip);

}

private void UpdateGridTotalsForTrip(Podorozh trip)

{

int idx = FindRowIndexForTrip(trip);

if (idx < 0 || idx >= dataGridView1.Rows.Count) return;

double baseCost = trip.RozrakhVartist();

double totalWithAcc = trip.TotalCostWithAccommodation();

double accOnly = Math.Max(0, totalWithAcc - baseCost);

if (dataGridView1.Columns.Contains("VartistProzhyvannya"))

dataGridView1.Rows[idx].Cells["VartistProzhyvannya"].Value = accOnly.ToString("F2");

if (dataGridView1.Columns.Contains("ZagalnaVartist"))

dataGridView1.Rows[idx].Cells["ZagalnaVartist"].Value = totalWithAcc.ToString("F2");

}

// ЗБЕРЕГТИ ВСІ ПОТОЧНІ ПОДОРОЖІ У TXT

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

{

var dlg = new SaveFileDialog

{

Filter = "Текстові файли|\*.txt",

Title = "Збереження даних"

};

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

{

using (var sw = new System.IO.StreamWriter(dlg.FileName, false, Encoding.UTF8))

{

foreach (var p in podorozhi)

{

// Формат: misto;vidstan;dni;transport;baseCost;avgPerKm

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

}

}

MessageBox.Show("Дані збережено!");

}

}

private void вивестиToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{

var dlg = new OpenFileDialog

{

Filter = "Текстові файли|\*.txt",

Title = "Відкрити файл з даними"

};

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

{

podorozhi.Clear();

dataGridView1.Rows.Clear();

using (var sr = new System.IO.StreamReader(dlg.FileName, Encoding.UTF8))

{

string line;

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

{

var parts = line.Split(';');

if (parts.Length >= 3)

{

string misto = parts[0];

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

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

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

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

podorozhi.Add(p);

// Колонки у вашому dataGridView1:

// Misto, Vidstan, Dni, Transport, Vartist, Serednya, TypTuru, VartistProzhyvannya, ZagalnaVartist, AgencyName

dataGridView1.Rows.Add(

p.Misto,

p.Vidstan,

p.Dni,

p.Transport,

p.RozrakhVartist(),

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

"—", // Тип туру з файлу не знаємо

"—", // Вартість проживання

p.RozrakhVartist().ToString("F2"), // Загальна = базова

"—" // Агенція

);

}

}

}

MessageBox.Show("Дані завантажено!");

}

}

}

}

**Search Form**

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:**

|  |  |
| --- | --- |
|  | При запуску додатку, щоб створити подорож потрібно натискати вкладку створити і відкривається форма для заповнення. |
|  | Додали дані. Тепер працюємо з ними. Додамо агрегацію до першої поїздки. |
|  | В вкладці агрегації є параметри. Нам потрібно додати подорож до агенції. Додамо. Далі, проглянемо всі подорожі агенцій. Також подивимось загальний дохід усіх агенцій. Далі, виконаємо композиції. |
|  | В вкладці композиції є також параметри, нам потрібно додати житло до поточної подорожі. Додали житло до подорожі, і натискаємо показати житло, нам видало повідомлення.  Порахуємо повну вартість. |

**Висновок:** Під час виконання лабораторної роботи, я одержав практичні навички створення об'єктів-груп.