**Лабораторна робота № 4**

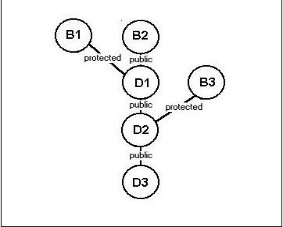
**Тема:** Побудова множинного наслідування класів. ( С++ ) Інтерфейси- С#.

**Мета:** Навчитися організовувати класи та похідні класи з своїми даними та властивостями: визначати структуру класу, типи даних та методів, організовувати необхідні конструктори та деструктори, використовувати специфікатори доступу; будувати ієрархію класів, використовуючи множинне наслідування; визначати та оперувати об’єктами цих класів; отримати практичні вміння та навички проектування та побудови ієрархії класів. Отримати практичні навички використання інтерфейсів.

**Задача №1**

Необхідно побудувати ієрархію класів що відповідає схемі наслідування, наведеній у варіанті завдання. Кожен клас повинен містити конструктор-ініціалізатор, і функцію show() для виведення значень.

**Варіант №1.**



**Постановка задачі:**

**B1, B2, B3** — базові класи з методами **show()**.

**D1** успадковує **B1** (protected) і **B2** (public).

**D2** успадковує **B1** (protected) і **B3** (protected).

**D3** успадковує **D1** (private) і **D2** (public).

**Код програми:**

**B1.h**

#pragma once

#include <iostream>

#include <string>

using namespace std;

class B1 {

protected:

string nameB1;

B1();

~B1();

public:

void input();

void show();

};

**B1.cpp**

#include "B1.h"

#include <iostream>

using namespace std;

B1::B1() {

cout << "Class B1, predkiv nema\n";

}

B1::~B1() {

cout << "Destructor B1\n";

}

void B1::input() {

cout << "Enter name for B1: ";

cin >> nameB1;

}

void B1::show() {

cout << "Class B1, name = " << nameB1 << endl;

}

**B2.h**

#pragma once

#include <iostream>

#include <string>

using namespace std;

class B2 {

protected:

string nameB2;

public:

B2();

~B2();

void input();

void show();

};

**B2.cpp**

#include "B2.h"

#include <iostream>

using namespace std;

B2::B2() {

cout << "Class B2, predkiv nema\n";

}

B2::~B2() {

cout << "Destructor B2\n";

}

void B2::input() {

cout << "Enter name for B2: ";

cin >> nameB2;

}

void B2::show() {

cout << "Class B2, name = " << nameB2 << endl;

}

**B3.h**

#pragma once

#include <iostream>

#include <string>

using namespace std;

class B3 {

protected:

string nameB3;

B3();

~B3();

public:

void input();

void show();

};

**B3.cpp**

#include "B3.h"

#include <iostream>

using namespace std;

B3::B3() {

cout << "Class B3, predkiv nema\n";

}

B3::~B3() {

cout << "Destructor B3\n";

}

void B3::input() {

cout << "Enter name for B3: ";

cin >> nameB3;

}

void B3::show() {

cout << "Class B3, name = " << nameB3 << endl;

}

**D1.h**

#pragma once

#include "B1.h"

#include "B2.h"

class D1 : public B2, protected B1 {

protected:

string nameD1;

public:

D1();

~D1();

void input();

void show();

};

**D1.cpp**

#include "D1.h"

#include <iostream>

using namespace std;

D1::D1() {

cout << "Class D1, predok public B2, protected B1\n";

}

D1::~D1() {

cout << "Destructor D1\n";

}

void D1::input() {

B1::input();

B2::input();

cout << "Enter name for D1: ";

cin >> nameD1;

}

void D1::show() {

B1::show();

B2::show();

cout << "Class D1, name = " << nameD1 << endl;

}

**D2.h**

#pragma once

#include "D1.h"

#include "B3.h"

class D2 : public D1, protected B3 {

protected:

string nameD2;

public:

D2();

~D2();

void input();

void show();

};

**D2.cpp**

#include "D2.h"

#include <iostream>

using namespace std;

D2::D2() {

cout << "Class D2, predok public D1, protected B3\n";

}

D2::~D2() {

cout << "Destructor D2\n";

}

void D2::input() {

cout << "Enter name for D2: ";

cin >> nameD2;

D1::input();

B3::input();

}

void D2::show() {

D1::show();

B3::show();

cout << "Class D2, name = " << nameD2 << endl;

}

**D3.h**

#pragma once

#include "D2.h"

class D3 : public D2 {

protected:

string nameD3;

public:

D3();

~D3();

void input();

void show();

};

**D3.cpp**

#include "D3.h"

#include <iostream>

using namespace std;

D3::D3() {

cout << "Class D3, predok public D2\n";

}

D3::~D3() {

cout << «Destructor D3\n»;

}

void D3::input() {

cout << «Enter name for D3: «;

cin >> nameD3;

D2::input();

}

void D3::show() {

D2::show();

cout << «Class D3, name = « << nameD3 << endl;

}

**Main.cpp**

#include <iostream>

#include «D3.h»

using namespace std;

int main() {

setlocale(0, «ukr»);

D3 d3;

d3.input();

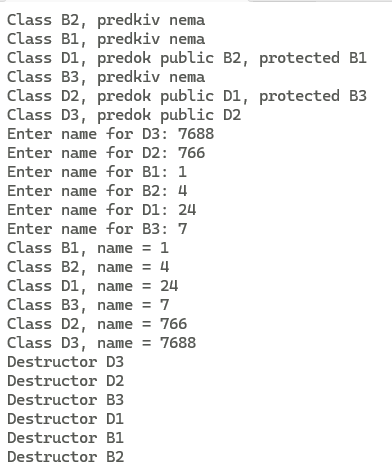
d3.show();

return 0;

}

**Результат програми:**

Вводимо дані наглядно по рівню. Тобто D3 – 4-х значне число, D2 – 3-х значне число, B1,B2,B3 – одинарне число.



**Завдання №2**

**Умова задачі:**До С# форми, додати 4-5 інтерфейсів.

**Код програми:**

**Class1.cs**

using LR1\_3;

public class Podorozh : IPodorozh

{

public string Misto { get; set; }

public double Vidstan { get; set; }

public int Dni { get; set; }

public string Transport { get; set; }

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();

}

}

**Class2.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;

}

}

**Class3.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

}

}

**Class4.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

}

}

**Class5.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)

{

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

}

}

**Interface1.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

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

}

}

**Interface2.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LR1\_3

{

public interface ITurystychnaPodorozh : IPodorozh

{

// Унікальні властивості туристичної подорожі

string TypTuru { get; set; }

double VartistProzhyvannya { get; set; }

// Унікальні методи

double RozrakhVartistPodorozh();

void ZminytyTypTuru(string newType);

}

}

**Interface3.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

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);

}

}

**Interface4.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LR1\_3

{

public interface IExtremePodorozh : IPodorozh

{

// Унікальна властивість

string DangerLevel { get; set; }

// Унікальні методи

double RozrakhVartistPodorozh();

void ZminytyTypTuru(string newType);

}

}

**Interface5.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LR1\_3

{

public interface ILongTermPodorozh : IPodorozh

{

int Months { get; set; }

// Унікальні методи

double RozrakhVartistPodorozh();

void ZminytyTypTuru(string newType);

}

}

**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

{

// 1. Зчитуємо дані з полів вводу

string misto = textBox1.Text;

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

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

string transport = comboBox1.SelectedItem.ToString();

string typTuru = comboBox2.SelectedItem.ToString();

// Створюємо змінну базового інтерфейсу для зберігання будь-якого типу подорожі

IPodorozh p;

// 2. Використовуємо switch для створення об'єкта відповідного класу

switch (typTuru)

{

case "Туристична":

// Для туристичної подорожі задаємо вартість проживання

double vartistProzhyvannya = 1000;

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");

}

// Додаємо створений об'єкт до загального списку

podorozhi.Add((Podorozh)p);

// 3. Готуємо дані для колонок, що залежать від типу подорожі

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");

}

// 4. Додаємо новий рядок у таблицю з усіма даними

dataGridView1.Rows.Add(

p.Misto,

p.Vidstan,

p.Dni,

p.Transport,

p.RozrakhVartist(), // Базова вартість

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

realTypTuru,

vartistProzhyvannyaStr, // Вартість проживання (тільки для туристичних)

zagalnaVartistStr // Повна вартість (для всіх спеціальних типів)

);

// 5. Очищуємо поля для наступного вводу

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

}

catch (Exception ex)

{

// Якщо дані введено некоректно, показуємо повідомлення про помилку

MessageBox.Show("Помилка: " + ex.Message + "\nБудь 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;

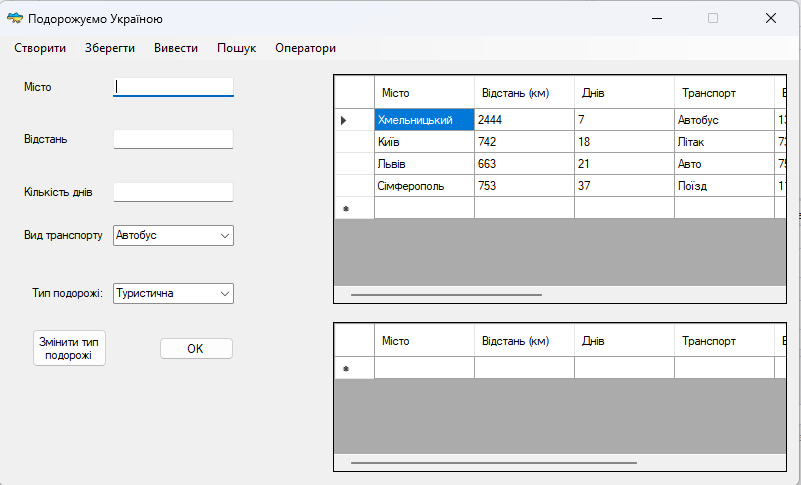
this.Close();

}

}

}

**Результат програми:**

****

В зовнішньому вигляді не змінилось абсолютно нічого, тільки додались інтерфейси для покращення роботи програми.

**Висновок:** Під час виконання лабораторної роботи я навчився організовувати класи та похідні класи з своїми даними та властивостями: визначати структуру класу, типи даних та методів, організовувати необхідні конструктори та деструктори, використовувати специфікатори доступу.