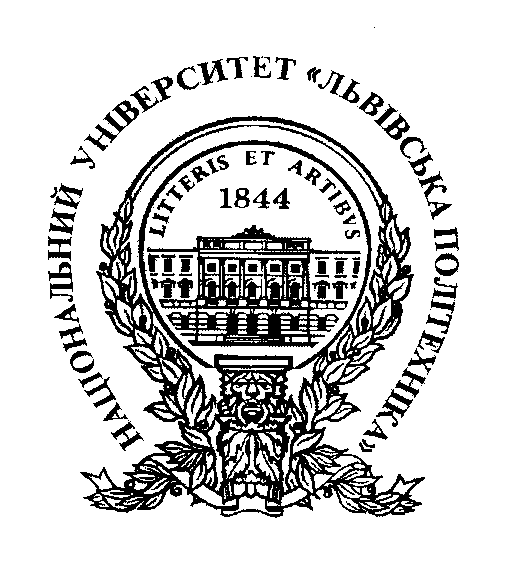
**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Національний університет “Львівська політехніка”**

**Інститут КНІ**

**Кафедра ПЗ**



**Курсова робота**

**З об’єктно-орієнтованого програмування**

**на тему: «Облік жителів»**

Студентки 2 курсу 2 групи

Спеціальності 6.050103

«Програмна інженерія»

Остапюк Зої Вікторівни

Керівник: асистент кафедри ПЗ,

Крупко Н. А.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів \_\_\_\_\_ Оцінка ECTS \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Львів 2016**

**Зміст:**

1. Завдання. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
2. Алгоритм розв’язку задачі. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
3. UML діаграма класів. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
4. UML діаграми послідовності виконання . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
5. UML діаграма прецедентів. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
6. Текст розробленої програми. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
7. Протокол роботи програми. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
8. Інструкція користувача . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32

**Завдання (1 варіант):**

Розробити програму засобами ООП на будь-якій мові, що підтримує концепцію ООП згідно вказаного варіанту.

Продемонструвати викладачу роботу розробленої програми.

Сформувати звіт з курсової роботи (не менше 20 стор.). Звіт має містити: 1) завдання; 2) алгоритм розв‘язку задачі; 3) діаграми UML класів, послідовності виконання, взаємозв‘язків; 4) текст розробленої програми з коментарями; 5) протокол роботи програми для кожного пункту завдання; 6) інструкція користувача. Інтерфейс програми українською мовою; 7) компакт-диск з записаною інформацією (звіт і програма).

Створити таблицю (№, прізвище, ім’я, номер телефону, адреса). Програма має передбачати ввід даних в двох режимах ( з клавіатури та з файлу). Передбачити виняткові ситуації. Програма має надавати також наступний функціонал:

1. за заданим прізвищем визначити адресу (врахувати наявність в таблиці однакових прізвищ);

2. за телефоном визначити імена та адреси;

3. за адресою визначити прізвище та ім’я;

4. за телефоном визначити всіх, хто мають однакові імена та проживають на одній і тій же вулиці;

5. вивести всі записи, що мають однакове поле «Адреса» (поле задається).

**Алгоритм розв’язку задачі:**



**UML діаграма класів**

*Рис. 1 «UML діаграма класів»*

**UML діаграми послідовності виконання**



*Рис.2 «UML діаграми послідовності виконання»*

**UML діаграма прецедентів**



*Рис.3 «UML діаграма прецедентів»*

**Текст розробленої програми з коментарями**

***Person.cs***

using System;

namespace PeopleAccounting

{

public class Person : IEquatable<Person>

{

private string firstName;

private string lastName;

private PhoneNumber number;

private Address address;

public Person() { }

public Person(string lname, string fname, int id, PhoneNumber num, Address addr)

{

ID = id;

LastName = lname;

FirstName = fname;

Number = num;

Address = addr;

}

// Ідентифікаційний номер людини, не може повторюватись в межах

// одного контейнера

[ColumnName("№")]

public int ID { get; set; }

[ColumnName("Прiзвіще")]

public string LastName

{

get

{

return lastName;

}

set

{

if (String.IsNullOrWhiteSpace(value))

{

throw new ArgumentNullException(nameof(lastName));

}

lastName = value;

}

}

// Ім’я людини, обов’язково має бути заповненим

[ColumnName("Ім'я")]

public string FirstName

{

get

{

return firstName;

}

set

{

if (String.IsNullOrWhiteSpace(value))

{

throw new ArgumentNullException(nameof(firstName));

}

firstName = value;

}

}

// Прізвище людини, обов’язково має бути заповненим

[ColumnName("Номер телефону")]

public PhoneNumber Number

{

get

{

return number;

}

set

{

if (value == null)

{

throw new ArgumentNullException(nameof(number));

}

number = value;

}

}

[ColumnName("Адреса")]

public Address Address

{

get

{

return address;

}

set

{

if (value == null)

{

throw new ArgumentNullException(nameof(address));

}

address = value;

}

}

// Метод з інтерфейсу для порівняння людей для функцій пошуку

public bool Equals(Person other)

{

return this.FirstName == other.FirstName &&

this.LastName == other.LastName &&

this.Address.Equals(other.Address) &&

this.Number.Equals(other.Number);

}

public override string ToString()

{

return String.Format("{0} {1} {2} {3}", LastName, FirstName, Number.ToString(), Address.ToString());

}

}

}

***Address.cs***

using System;

namespace PeopleAccounting

{

public class Address : IEquatable<Address>

{

private string country;

private string region;

private string locality;

private string street;

private int buildNumber;

private int apartNumber;

public Address() { }

public Address(string country, string region, string locality, string street, int buildNum, int apNum)

{

Country = country;

Locality = locality;

Region = region;

Street = street;

BuildingNumber = buildNum;

ApartamentNumber = apNum;

}

// Країна – обов’язкове поле

[ColumnName("Країна")]

public string Country

{

get

{

return country;

}

set

{

if (String.IsNullOrWhiteSpace(value))

{

throw new ArgumentNullException("country");

}

country = value;

}

}

// Область – обов’язкове поле

[ColumnName("Область")]

public string Region

{

get

{

return region;

}

set

{

if (String.IsNullOrWhiteSpace(value))

{

throw new ArgumentNullException("region");

}

region = value;

}

}

// Місце проживання – обов’язкове поле

[ColumnName("Місто/село/смт")]

public string Locality

{

get

{

return locality;

}

set

{

if (String.IsNullOrWhiteSpace(value))

{

throw new ArgumentNullException("locality");

}

locality = value;

}

}

// Вулиця – обов’язкове поле

[ColumnName("Вулиця")]

public string Street

{

get

{

return street;

}

set

{

if (String.IsNullOrWhiteSpace(value))

{

throw new ArgumentNullException("street");

}

street = value;

}

}

// Номер – обов’язкове поле

[ColumnName("Номер будинку")]

public int BuildingNumber

{

get

{

return buildNumber;

}

set

{

if (value <= 0)

{

throw new ArgumentOutOfRangeException(nameof(buildNumber));

}

buildNumber = value;

}

}

// Номер – обов’язкове поле

[ColumnName("Номер квартири")]

public int ApartamentNumber

{

get

{

return apartNumber;

}

set

{

if (value <= 0)

{

throw new ArgumentOutOfRangeException(nameof(apartNumber));

}

apartNumber = value;

}

}

public bool Equals(Address other)

{

return this.Country == other.Country &&

this.Region == other.Region &&

this.Locality == other.Locality &&

this.Street == other.Street &&

this.BuildingNumber == other.BuildingNumber &&

this.ApartamentNumber == other.ApartamentNumber;

}

public override string ToString()

{

return String.Format("{0}, {1} обл., {2}, вул. {3} буд.{4}, кв.{5}", Country, Region, Locality, Street, BuildingNumber, ApartamentNumber);

}

}

}

***PhoneNumber.cs***

using System;

namespace PeopleAccounting

{

public class PhoneNumber : IEquatable<PhoneNumber>

{

// Для контексту програми коректними номерами вважатимуться

// лише українські телефонні номери з відповідним кодом

// доступні будь-які оператори

public const string CountryCode = "+380";

private string number;

public PhoneNumber(string number)

{

if (!IsValid(number))

{

throw new InvalidNumberException(number);

}

Number = number;

}

public string Number

{

get

{

return CountryCode + number;

}

set

{

if (!IsValid(value))

{

throw new InvalidNumberException(value);

}

number = value.Substring(4);

}

}

// Функція перевіряє чи задана стрічка може інтерпретуватись

// як телефонний номер: починається з коду країни та містить

// лише цифри

public static bool IsValid(string number)

{

if (String.IsNullOrWhiteSpace(number))

{

return false;

}

if (number.Length != 13 || !number.StartsWith(CountryCode))

{

return false;

}

uint tryParse;

return uint.TryParse(number.Substring(4), out tryParse);

}

public bool Equals(PhoneNumber other)

{

return this.number == other.number;

}

public override string ToString()

{

return Number;

}

}

}

***ColumnNameAttribute.cs***

using System;

namespace PeopleAccounting

{

public class ColumnNameAttribute : Attribute

{

public ColumnNameAttribute(string name)

{

ColumnName = name;

}

// Назва стовпця, що відображатиметься при наповненні таблиці

// цей клас необхідний для коректної генерації таблиці,

// бо за замовчуванням назви колонок відповідають

// назві властивості з відкритим типом доступу

public string ColumnName { get; set; }

}

}

*InvalidNumberException.cs*

using System;

namespace PeopleAccounting

{

public class InvalidNumberException : Exception

{

public InvalidNumberException() : base("The number provided had invalid format or it is null") { }

public InvalidNumberException(string invNum, string message) : base(message)

{

InvalidValue = invNum;

}

public InvalidNumberException(string invNum)

{

InvalidValue = invNum;

}

public InvalidNumberException(string message, Exception inner) : base(message, inner) { }

// Стрічка, що не інтерпретується коректно як номер телефону

public string InvalidValue { get; set; }

}

}

***InvalidLinesDictionary.cs***

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

namespace PeopleAccounting

{

public class InvalidLinesDictionary : IEnumerable<KeyValuePair<int, string>>

{

private IDictionary<int, string> errorDictionary;

// Максимальна кількість записів, необхідна, бо файли можуть бути

// дуже великими, тому немає необхідності в збереженні

// кожного неправильного запису

private int maxSize;

public InvalidLinesDictionary(int max)

{

if (max < 0)

{

throw new ArgumentOutOfRangeException();

}

maxSize = max;

errorDictionary = new Dictionary<int, string>();

}

public int Count { get { return errorDictionary.Count; } }

public int MaxSize { get { return maxSize; } }

// В словник додається KeyValuePair<int, string>

public void Add(int line, string record)

{

if (errorDictionary.Count == maxSize)

{

// Видалити найдавніший запис

errorDictionary.Remove(errorDictionary.Keys.Min());

}

errorDictionary.Add(line, record);

}

public void Clear()

{

errorDictionary.Clear();

}

public IEnumerator<KeyValuePair<int, string>> GetEnumerator()

{

return errorDictionary.GetEnumerator();

}

IEnumerator IEnumerable.GetEnumerator()

{

return errorDictionary.GetEnumerator();

}

}

}

***IPeopleRepository.cs***

using System.Collections.Generic;

namespace PeopleAccounting

{

public interface IPeopleRepository : IEnumerable<Person>

{

Person this[int index] { get; }

int Count { get; }

void Add(Person person);

bool Delete(Person person);

void Update(int index, Person person);

void Clear();

// Функції фільтрування, задані для варіанту

IList<Person> FindPeopleByLastname(string lastname);

IList<Person> FindPeopleByPhoneNumber(PhoneNumber number);

IList<Person> FindPeopleByAddress(Address address);

IList<Person> FindSameNamesStreetsByPhoneNumber(PhoneNumber number);

}

}

***PeopleRepository.cs***

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

namespace PeopleAccounting

{

public class PeopleRepository : IPeopleRepository

{

private IList<Person> people;

public PeopleRepository()

{

people = new List<Person>();

}

public Person this[int index]

{

get

{

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

{

throw new ArgumentOutOfRangeException(nameof(index));

}

return people[index];

}

}

// Основні функції для будь-якого контейнера/репозиторію

public int Count { get { return people.Count; } }

public void Add(Person person)

{

var peopleWithId = people.Where(p => p.ID == person.ID).FirstOrDefault();

if (peopleWithId != null)

{

throw new ArgumentException(person.ID.ToString());

}

people.Add(person);

}

public bool Delete(Person person)

{

if (people.Contains(person))

{

people.Remove(person);

return true;

}

return false;

}

public void Update(int index, Person person)

{

if (person == null)

{

throw new ArgumentNullException(nameof(person));

}

var peopleWithId = people.Where(p => p.ID == person.ID).FirstOrDefault();

if (person.ID != people[index].ID && peopleWithId != null)

{

throw new ArgumentException(person.ID.ToString());

}

people[index] = person;

}

public void Clear()

{

people.Clear();

}

public IList<Person> FindPeopleByLastname(string lastname)

{

if (String.IsNullOrWhiteSpace(lastname))

{

throw new ArgumentNullException();

}

// Використання LINQ для фільтрування записів

return people.Where(p => p.LastName == lastname).ToList();

}

public IList<Person> FindPeopleByAddress(Address address)

{

if (address == null)

{

throw new ArgumentNullException();

}

// Використання LINQ для фільтрування записів

return people.Where(p => p.Address.Equals(address)).ToList();

}

public IList<Person> FindPeopleByPhoneNumber(PhoneNumber number)

{

if (number == null)

{

throw new ArgumentNullException();

}

// Використання LINQ для фільтрування записів

return people.Where(p => p.Number.Equals(number)).ToList();

}

public IList<Person> FindSameNamesStreetsByPhoneNumber(PhoneNumber number)

{

var sameNumber = FindPeopleByPhoneNumber(number);

if (sameNumber == null)

{

return null;

}

var sameNameGroups = sameNumber.GroupBy(p => p.FirstName);

// Знайти найбільшу групу людей з однаковими іменами

var group = sameNameGroups.First();

foreach (var item in sameNameGroups)

{

if (item.Count() > group.Count())

{

group = item;

}

}

// Знайти найбільшу групу людей з однаковими іменами і,

// що проживають на тих самих вулицях

var sameStreetGroups = group.Select(g => g).GroupBy(p => p.Address.Street);

var resultGroup = sameStreetGroups.First();

foreach (var item in sameStreetGroups)

{

if (item.Count() > resultGroup.Count())

{

resultGroup = item;

}

}

return resultGroup.Select(g => g).ToList();

}

public IEnumerator<Person> GetEnumerator()

{

return people.GetEnumerator();

}

IEnumerator IEnumerable.GetEnumerator()

{

return people.GetEnumerator();

}

}

}

***PeopleRepositoryFileHandler.cs***

using System;

using System.IO;

using System.Text;

namespace PeopleAccounting

{

public class PeopleRepositoryFileHandler

{

private const int lineEntriesCount = 10;

public PeopleRepositoryFileHandler()

{

Report = new InvalidLinesDictionary(10);

}

public bool TryParseFromString(string line, out Person person)

{

string[] entries = line.Trim(' ', '\t').Split(new char[] { '|' }, StringSplitOptions.RemoveEmptyEntries);

if (!AreLinesCorrectForParsing(entries))

{

person = null;

return false;

}

PhoneNumber number = new PhoneNumber(entries[3]);

Address address = new Address

{

Country = entries[4],

Region = entries[5],

Locality = entries[6],

Street = entries[7],

BuildingNumber = int.Parse(entries[8]),

ApartamentNumber = int.Parse(entries[9])

};

// Формування результату

person = new Person(entries[1], entries[2], int.Parse(entries[0]), number, address);

return true;

}

private bool AreLinesCorrectForParsing(string[] line)

{

if (line.Length != lineEntriesCount)

{

return false;

}

uint tryParse;

if (!uint.TryParse(line[0], out tryParse) || // Cheking ID

!uint.TryParse(line[8], out tryParse) || // Checking building number

!uint.TryParse(line[9], out tryParse) || // CHecking apartment number

!PhoneNumber.IsValid(line[3])) // Checking phone number

{

return false;

}

return true;

}

// Основна функція для видобутку списку з файлу,

// використовує інші методи класу для розпарсення

// стрічок з текстового файлу

public IPeopleRepository ReadFromTextFile(string path)

{

if (String.IsNullOrWhiteSpace(path))

{

throw new ArgumentNullException("path", "Please, provide this instance with a path to a file");

}

string[] lines = null;

try

{

lines = File.ReadAllText(path).Split('\n');

}

catch (Exception e)

{

throw new ArgumentOutOfRangeException("Invalid path or file", e);

}

Report.Clear();

return PeopleRepoFromStrings(lines);

}

public IPeopleRepository PeopleRepoFromStrings(string[] lines)

{

IPeopleRepository repo = new PeopleRepository();

for (int i = 0; i < lines.Length; i++)

{

// Приклад коректного запису

// 1|Остапюк|Зоя|+380937538109|Україна|Львівська|Львів|Гнатюка|20|9

Person result;

if (!TryParseFromString(lines[i], out result))

{

Report.Add(i, lines[i]);

continue;

}

try

{

repo.Add(result);

}

catch (ArgumentException)

{

Report.Add(i + 1, lines[i]);

continue;

}

}

return repo;

}

// Метод, необхідний для запису об’єкту репозиторію

// в текстовий файл, який можна буде потім коректно

// прочитати методом класу

public void WriteToFile(string path, IPeopleRepository repo)

{

StringBuilder lines = new StringBuilder();

for (int i = 0; i < repo.Count; i++)

{

lines.Append(String.Format("{0}|{1}|{2}|{3}|{4}|{5}|{6}|{7}|{8}|{9}", repo[i].ID, repo[i].LastName, repo[i].FirstName, repo[i].Number,

repo[i].Address.Country, repo[i].Address.Region, repo[i].Address.Locality,

repo[i].Address.Street, repo[i].Address.BuildingNumber, repo[i].Address.ApartamentNumber));

if (i < repo.Count - 1)

{

lines.AppendLine();

}

}

File.WriteAllText(path, lines.ToString());

}

public InvalidLinesDictionary Report { get; private set; }

}

}

***FullName.cs***

namespace PeopleAccounting.UI.PersonDisplayHelpers

{

// Клас, необхідний для фільтрації результату функції

// пошуку лише імені та прізвища за певним критерієм

public class FullName

{

[ColumnName("Ім'я")]

public string FirstName { get; set; }

[ColumnName("Прізвище")]

public string LastName { get; set; }

}

}

***NameAndAddress.cs***

namespace PeopleAccounting.UI.PersonDisplayHelpers

{

public class NameAndAddress

{

// Клас, необхідний для фільтрації результату функції

// пошуку лише імені та адреси за певним критерієм

[ColumnName("Ім'я")]

public string FirstName { get; set; }

[ColumnName("Адреса")]

public Address Address { get; set; }

}

}

***NameAndStreet.cs***

namespace PeopleAccounting.UI.PersonDisplayHelpers

{

public class NameAndStreet

{

// Клас, необхідний для фільтрації результату функції

// пошуку лише імені та вулиці за певним критерієм

[ColumnName("Ім'я")]

public string FirstName { get; set; }

[ColumnName("Вулиця")]

public string Street { get; set; }

}

}

***MainWindow.cs***

using Microsoft.Win32;

using PeopleAccounting.UI.PersonDisplayHelpers;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

namespace PeopleAccounting.UI

{

public partial class MainWindow : Window

{

private IPeopleRepository repo;

public MainWindow()

{

InitializeComponent();

setUpInterface();

repo = new PeopleRepository();

this.dataGridPeople.ItemsSource = repo;

}

// Функції для видобутку об’єктів, необхідних для додавання/

// пошуку/видалення записів з контейнера та таблиці

private void setUpInterface()

{

this.choosePanel.IsChecked = true;

this.toolBarVisibility.IsChecked = true;

(this.tabSystem.Items[1] as TabItem).Visibility = Visibility.Collapsed;

this.btnDeleteItem.IsEnabled = false;

this.btnEditItem.IsEnabled = false;

}

private Person getPersonFromInfoEditor()

{

Person result = null;

try

{

PhoneNumber number = new PhoneNumber(this.number.Text);

Address address = new Address

{

Country = this.country.Text,

Region = this.region.Text,

Locality = this.locality.Text,

Street = this.street.Text,

BuildingNumber = this.buildNum.Value.Value,

ApartamentNumber = this.apartNum.Value.Value

};

result = new Person

{

ID = this.idNum.Value.Value,

LastName = this.lastName.Text,

FirstName = this.firstName.Text,

Number = number,

Address = address

};

}

catch (InvalidNumberException)

{

MessageBox.Show("Будь ласка, введіть правильні дані для телефона", this.Title);

}

catch (ArgumentNullException)

{

MessageBox.Show("Будь ласка, заповніть усі поля", this.Title);

}

catch (InvalidOperationException)

{

MessageBox.Show("Будь ласка, заповнюйте поля номерів будинку/квартири/ІД, використовуючи числа", this.Title);

}

return result;

}

private Address getAddress()

{

Address result = null;

try

{

result = new Address

{

Country = this.findCountry.Text,

Region = this.findRegion.Text,

Locality = this.findLocality.Text,

Street = this.findStreet.Text,

BuildingNumber = this.findBuild.Value.Value,

ApartamentNumber = this.findApart.Value.Value

};

}

catch (ArgumentNullException)

{

MessageBox.Show("Будь ласка, заповніть усі поля", this.Title);

}

catch (InvalidOperationException)

{

MessageBox.Show("Будь ласка, введіть номер будинку і квартири правильно", this.Title);

}

return result;

}

private PhoneNumber getNumber()

{

PhoneNumber result = null;

if(!PhoneNumber.IsValid(this.findPhoneNumber.Text))

{

MessageBox.Show("Будь ласка, введіть правильні дані для телефона", this.Title);

}

else

{

result = new PhoneNumber(this.findPhoneNumber.Text);

}

return result;

}

// Регіон коду, що відповідає лише за інтерфейс користувача

// Тут записані обробники подій зміни станів якихось елементів

// інтерфейсу

#region Window Related Events

private void dataGridPeople\_AutoGeneratingColumn(object sender, DataGridAutoGeneratingColumnEventArgs e)

{

var desc = e.PropertyDescriptor as PropertyDescriptor;

var attribute = desc.Attributes[typeof(ColumnNameAttribute)] as ColumnNameAttribute;

if (attribute != null)

{

e.Column.Header = attribute.ColumnName;

}

}

private void choosePanel\_Checked(object sender, RoutedEventArgs e)

{

this.mainGrid.ColumnDefinitions[0].Width = new GridLength(0.5, GridUnitType.Star);

this.mainGrid.ColumnDefinitions[1].Width = new GridLength(1.3, GridUnitType.Star);

}

private void choosePanel\_Unchecked(object sender, RoutedEventArgs e)

{

this.mainGrid.ColumnDefinitions[0].Width = new GridLength(0, GridUnitType.Star);

this.mainGrid.ColumnDefinitions[1].Width = new GridLength(1, GridUnitType.Star);

}

private void toolBarVisibility\_Checked(object sender, RoutedEventArgs e)

{

this.mainToolBarTray.Visibility = Visibility.Visible;

}

private void toolBarVisibility\_Unchecked(object sender, RoutedEventArgs e)

{

this.mainToolBarTray.Visibility = Visibility.Collapsed;

}

private void dataGridPeople\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

var item = (sender as DataGrid).SelectedItem;

if (item != null && this.tabSystem.SelectedIndex == 0)

{

Person selectedPerson = item as Person;

this.idNum.Value = selectedPerson.ID;

this.lastName.Text = selectedPerson.LastName;

this.firstName.Text = selectedPerson.FirstName;

this.number.Text = selectedPerson.Number.ToString();

this.country.Text = selectedPerson.Address.Country;

this.region.Text = selectedPerson.Address.Region;

this.locality.Text = selectedPerson.Address.Locality;

this.street.Text = selectedPerson.Address.Street;

this.buildNum.Value = selectedPerson.Address.BuildingNumber;

this.apartNum.Value = selectedPerson.Address.ApartamentNumber;

this.btnDeleteItem.IsEnabled = true;

this.btnEditItem.IsEnabled = true;

}

else

{

this.btnDeleteItem.IsEnabled = false;

this.btnEditItem.IsEnabled = false;

}

}

#endregion

// Основні функції пошуку та фільтрування

#region Main Functions

private void btnfindAddressByLastname\_Click(object sender, RoutedEventArgs e)

{

if (String.IsNullOrWhiteSpace(this.findLastname.Text))

{

MessageBox.Show("Введіть прізвище", this.Title);

return;

}

if (this.dataGridPeople.ItemsSource != null)

{

IEnumerable<Person> result = repo.FindPeopleByLastname(this.findLastname.Text);

if (result.Count() == 0)

{

MessageBox.Show("На жаль, не було знайдено результатів", this.Title);

return;

}

(this.tabSystem.Items[1] as TabItem).Visibility = Visibility.Visible;

this.tabSystem.SelectedIndex = 1;

this.dataGridResults.ItemsSource = result.Select(p => p.Address);

this.dataGridResults.Items.Refresh();

}

}

private void btnfindNamesAndAddressByPhone\_Click(object sender, RoutedEventArgs e)

{

if (this.dataGridPeople.ItemsSource != null)

{

PhoneNumber number = getNumber();

if (number == null)

{

return;

}

IEnumerable<Person> result = repo.FindPeopleByPhoneNumber(number);

if (result.Count() == 0)

{

MessageBox.Show("На жаль, не було знайдено результатів", this.Title);

return;

}

(this.tabSystem.Items[1] as TabItem).Visibility = Visibility.Visible;

this.tabSystem.SelectedIndex = 1;

this.dataGridResults.ItemsSource = result.Select(p => new NameAndAddress

{

FirstName = p.FirstName,

Address = p.Address

});

this.dataGridResults.Items.Refresh();

}

}

private void btnfindPeopleByAddress\_Click(object sender, RoutedEventArgs e)

{

Address address = getAddress();

if (this.dataGridPeople.ItemsSource != null && address != null)

{

IEnumerable<Person> result = repo.FindPeopleByAddress(address);

if (result.Count() == 0)

{

MessageBox.Show("На жаль, не було знайдено результатів", this.Title);

return;

}

(this.tabSystem.Items[1] as TabItem).Visibility = Visibility.Visible;

this.tabSystem.SelectedIndex = 1;

this.dataGridResults.ItemsSource = result;

this.dataGridResults.Items.Refresh();

}

}

private void btnfindSameNameAddressByPhone\_Click(object sender, RoutedEventArgs e)

{

if (this.dataGridPeople.ItemsSource != null)

{

PhoneNumber number = getNumber();

if (number == null)

{

return;

}

IEnumerable<Person> result = repo.FindSameNamesStreetsByPhoneNumber(number);

if (result.Count() == 0)

{

MessageBox.Show("На жаль, не було знайдено результатів", this.Title);

return;

}

(this.tabSystem.Items[1] as TabItem).Visibility = Visibility.Visible;

this.tabSystem.SelectedIndex = 1;

this.dataGridResults.ItemsSource = result.Select(p => new NameAndStreet

{

FirstName = p.FirstName,

Street = p.Address.Street

});

this.dataGridResults.Items.Refresh();

}

}

private void btnfindFullNameByAddress\_Click(object sender, RoutedEventArgs e)

{

Address address = getAddress();

if (this.dataGridPeople.ItemsSource != null && address != null)

{

IEnumerable<Person> result = repo.FindPeopleByAddress(address);

if (result.Count() == 0)

{

MessageBox.Show("На жаль, не було знайдено результатів", this.Title);

return;

}

(this.tabSystem.Items[1] as TabItem).Visibility = Visibility.Visible;

this.tabSystem.SelectedIndex = 1;

this.dataGridResults.ItemsSource = result.Select(p => new FullName

{

FirstName = p.FirstName,

LastName = p.LastName

});

this.dataGridResults.Items.Refresh();

}

}

#endregion

// Кнопки, що розташові в головному меню та мають відповідники

// на панелі інструментів

#region Menu Buttons

private void btnNew\_Click(object sender, RoutedEventArgs e)

{

this.fileName.Text = "Немає відкритих файлів";

repo.Clear();

this.dataGridPeople.Items.Refresh();

(this.tabSystem.Items[1] as TabItem).Visibility = Visibility.Collapsed;

}

private void btnOpen\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog openDlg = new OpenFileDialog();

openDlg.DefaultExt = "txt";

openDlg.Filter = "Txt файли| \*.txt";

if (openDlg.ShowDialog() == true)

{

PeopleRepositoryFileHandler reader = new PeopleRepositoryFileHandler();

try

{

repo = reader.ReadFromTextFile(openDlg.FileName);

this.dataGridPeople.ItemsSource = repo;

this.fileName.Text = openDlg.FileName;

(this.tabSystem.Items[1] as TabItem).Visibility = Visibility.Collapsed;

if (reader.Report.Count != 0)

{

StringBuilder errors = new StringBuilder();

foreach (var error in reader.Report)

{

errors.AppendLine(String.Format("№{0} - {1}", error.Key, error.Value));

}

MessageBox.Show(errors.ToString(), "Некоректні записи");

}

}

catch (ArgumentException)

{

MessageBox.Show("Оберіть правильний .txt файл", this.Title);

}

}

}

private void btnSave\_Click(object sender, RoutedEventArgs e)

{

if (!this.fileName.Text.EndsWith(".txt"))

{

btnSaveAs\_Click(sender, e);

return;

}

try

{

if (!System.IO.File.Exists(this.fileName.Text))

{

MessageBox.Show("Сталася помилка. Перевірте, чи редагований файл створений", this.Title);

}

PeopleRepositoryFileHandler writer = new PeopleRepositoryFileHandler();

writer.WriteToFile(this.fileName.Text, repo);

}

catch

{

MessageBox.Show("Сталася помилка. Перевірте, чи редагований файл створений", this.Title);

}

}

private void btnSaveAs\_Click(object sender, RoutedEventArgs e)

{

SaveFileDialog saveDlg = new SaveFileDialog();

saveDlg.DefaultExt = "txt";

saveDlg.Filter = "Txt файли| \*.txt";

if (saveDlg.ShowDialog() == true)

{

PeopleRepositoryFileHandler writer = new PeopleRepositoryFileHandler();

try

{

writer.WriteToFile(saveDlg.FileName, repo);

this.fileName.Text = saveDlg.FileName;

}

catch

{

MessageBox.Show("Оберіть правильний .txt файл", this.Title);

}

}

}

private void btnClose\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void btnEditItem\_Click(object sender, RoutedEventArgs e)

{

int index = this.dataGridPeople.SelectedIndex;

if (-1 < index && index < repo.Count)

{

Person person = getPersonFromInfoEditor();

if (person != null)

{

try

{

repo.Update(index, person);

this.dataGridPeople.Items.Refresh();

(this.tabSystem.Items[1] as TabItem).Visibility = Visibility.Collapsed;

}

catch (ArgumentException ex)

{

MessageBox.Show(String.Format("Людина з таким ІД вже існує", ex.ParamName), this.Title);

}

}

}

}

private void btnAddItem\_Click(object sender, RoutedEventArgs e)

{

Person person = getPersonFromInfoEditor();

if (person != null)

{

try

{

repo.Add(person);

this.dataGridPeople.Items.Refresh();

(this.tabSystem.Items[1] as TabItem).Visibility = Visibility.Collapsed;

}

catch (ArgumentException ex)

{

MessageBox.Show(String.Format("Людина з таким ІД вже існує", ex.ParamName), this.Title);

}

}

}

private void btnDeleteItem\_Click(object sender, RoutedEventArgs e)

{

Person selectedPerson = this.dataGridPeople.SelectedItem as Person;

if (selectedPerson != null)

{

repo.Delete(selectedPerson);

}

this.dataGridPeople.Items.Refresh();

(this.tabSystem.Items[1] as TabItem).Visibility = Visibility.Collapsed;

}

private void btnClear\_Click(object sender, RoutedEventArgs e)

{

repo.Clear();

this.dataGridPeople.Items.Refresh();

(this.tabSystem.Items[1] as TabItem).Visibility = Visibility.Collapsed;

}

private void about\_Click(object sender, RoutedEventArgs e)

{

AboutWindow abt = new AboutWindow();

abt.ShowDialog();

}

private void help\_Click(object sender, RoutedEventArgs e)

{

HelpWindow help = new HelpWindow();

help.ShowDialog();

}

#endregion

}

}

***HelpWindow.cs***

using System;

using System.Windows;

using System.Windows.Documents;

using System.Windows.Media.Imaging;

namespace PeopleAccounting.UI

{

/// <summary>

/// Interaction logic for HelpWindow.xaml

/// Клас для відображення інструкції користувача

/// </summary>

public partial class HelpWindow : Window

{

public HelpWindow()

{

InitializeComponent();

}

private void open\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Програма відкриває текстові файли \*.txt і переносить їх у таблицю, зручну для читання: пункт головного меню Файл->Відкрити… або кнопка панелі інструментів. Файл повинен бути заповнений рядками, розділеними символом переносу рядка (кожний запис з нового рядка), дані про людину мають бути організовані у форматі:\n\n\n");

Bold form = new Bold();

form.Inlines.Add("{ІД}|{Прізвище}|{Ім’я}|{Телефону(«+380…»)}|{Країна}|{Область}|{Місто/село/смт}|{Вулиця}|{Номер будинку}|{Номер квартири}\n\n\n");

Run content1 = new Run("Опрацьовуються лише ті рядки файлу, для яких розрахований функціонал ПЗ. Якщо в файлі деякі рядки не можуть бути інтерпретовані як інформація про людину, то вони будуть занесені до журналу некоректних записів, вміст якого відобразиться одразу після завершення заповнення таблиці.");

// Додання ілюстрацій до параграфу

this.image.Source = new BitmapImage(new Uri(Environment.CurrentDirectory + "\\report.png"));

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

this.paragraph.Inlines.Add(form);

this.paragraph.Inlines.Add(content1);

}

private void save\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Для збереження таблиці у текстовий файл потрібно перейти до пункту головного меню Файл->Зберегти, або, якщо потрібно створити новий документ, Файл->Зберегти як… або за допомогою відповідних кнопок на панелі інструментів.");

this.image.Source = null;

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void create\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Щоб створити новий файл, треба перейти до пункту головного меню Файл->Новий або скористатись кнопкою на панелі інструментів.");

this.image.Source = null;

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void sort\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Сортувати список можна за ІД, прізвищем або іменем. Для цього необхідно натиснути на заголовок відповідної колонки в таблиці з записами.");

this.image.Source = null;

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void add\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Щоб додати новий запис в таблицю, заповніть коректно форму редактора інформації на оберіть пункт головного меню Редагування->Додати до списку, або відповідну кнопку панелі інструментів.");

// Додання ілюстрацій до параграфу

this.image.Source = new BitmapImage(new Uri(Environment.CurrentDirectory + "\\infoEditor.png"));

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void edit\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Редагування запису в таблиці можна здійснити, натиснувши на кнопці панелі інструментів, попередньо коректно заповнивши форму редактора інформації.");

// Додання ілюстрацій до параграфу

this.image.Source = new BitmapImage(new Uri(Environment.CurrentDirectory + "\\editRecord.png"));

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void delete\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Щоб видалити запис, треба виділити його в таблиці, натиснувши на нього і обрати пункт головного меню Редагування->Видалити запис, або натиснути кнопку на панелі інструментів.");

this.image.Source = null;

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void clear\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Щоб очистити весь список, оберіть пункт головного меню Редагування->Очистити список або відповідну кнопку на панелі інструментів.");

this.image.Source = null;

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void toolBar\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Щоб приховати панель інструментів, перейдіть у пункт головного меню Вигляд->Панель інструментів та зніміть галочку.");

// Додання ілюстрацій до параграфу

this.image.Source = new BitmapImage(new Uri(Environment.CurrentDirectory + "\\hideToolbar.png"));

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void choose\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Щоб приховати панель з формою редактора інформації та категоріями пошуку, перейдіть у пункт головного меню Вигляд->Панель вибору та зніміть галочку.");

// Додання ілюстрацій до параграфу

this.image.Source = new BitmapImage(new Uri(Environment.CurrentDirectory + "\\hideChoose.png"));

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void addressesByLastname\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Щоб знайти адреси людей за прізвищем, введіть його в форму критеріїв пошуку та натисніть на кнопку напроти тексту «Адреси за прізвищем:». Результати будуть відображені в новій вкладці поряд з вкладкою з основним списком.");

// Додання ілюстрацій до параграфу

this.image.Source = new BitmapImage(new Uri(Environment.CurrentDirectory + "\\mainFunctions.png"));

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void namesAddressesByNumber\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Щоб знайти імена та адреси людей за номером, введіть його в форму критеріїв пошуку та натисніть на кнопку напроти тексту «Імена і адреси за номером:». Результати будуть відображені в новій вкладці поряд з вкладкою з основним списком.");

// Додання ілюстрацій до параграфу

this.image.Source = new BitmapImage(new Uri(Environment.CurrentDirectory + "\\mainFunctions.png"));

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void fullnameByAddress\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Щоб знайти імена та прізвища людей за адресою, введіть її в форму критеріїв пошуку та натисніть на кнопку напроти тексту «Повне ім’я за адресою:». Результати будуть відображені в новій вкладці поряд з вкладкою з основним списком.");

// Додання ілюстрацій до параграфу

this.image.Source = new BitmapImage(new Uri(Environment.CurrentDirectory + "\\mainFunctions.png"));

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void namesStreetsByNumber\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Щоб знайти однакові імена та вулиці людей за однаковим номером, введіть його в форму критеріїв пошуку та натисніть на кнопку напроти тексту «Імена/вулиці(однакові) за номером:». Результати будуть відображені в новій вкладці поряд з вкладкою з основним списком.");

// Додання ілюстрацій до параграфу

this.image.Source = new BitmapImage(new Uri(Environment.CurrentDirectory + "\\mainFunctions.png"));

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void peopleByLastname\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Щоб знайти повну інформацію про людей за прізвищем, введіть його в форму критеріїв пошуку та натисніть на кнопку напроти тексту «Люди за прізвищем:». Результати будуть відображені в новій вкладці поряд з вкладкою з основним списком.");

// Додання ілюстрацій до параграфу

this.image.Source = new BitmapImage(new Uri(Environment.CurrentDirectory + "\\mainFunctions.png"));

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void error1\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Перевірте коректність введеного номера телефону: він повинен починатись з коду України «+380» та містити загалом 13 цифр разом з знаком плюс.");

this.image.Source = null;

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void error2\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Необхідно перевірити чи всі поля адреси чи редактора інформації заповнені.");

this.image.Source = null;

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void error3\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Перевірте, чи в поля номерів будинку, квартири чи ІД записані натуральні числа.");

this.image.Source = null;

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void error4\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Під час вибору адреси як критерію пошуку перевірте, чи в полях номерів будинку та квартири введені натуральні числа.");

this.image.Source = null;

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void error5\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Під час намагання програми знайти людей, використовуючи критерії прізвища, було визначено, що користувач не заповнив поле.");

this.image.Source = null;

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void error6\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("В головному списку не було знайдено записів, що відповідають критерію пошуку.");

this.image.Source = null;

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void error7\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Під час додавання запису, було визначено, що в списку вже є людина з таким же ідентифікатором, змініть його, або видаліть існуючий запис.");

this.image.Source = null;

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void error8\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Під час спроби зберегти зміни в редагованому файлі виявилось, що він був стертий з диску або переміщений. Оберіть функцію Файл->Зберегти як… .");

this.image.Source = null;

this.paragraph.Inlines.Clear();

this.paragraph.Inlines.Add(content);

}

private void error9\_Selected(object sender, RoutedEventArgs e)

{

Run content = new Run("Під час спроби записати список в текстовий файл сталась помилка. Оберіть інший файл.");

this.image.Source = null;

this.paragraph.Inlines.Clear();

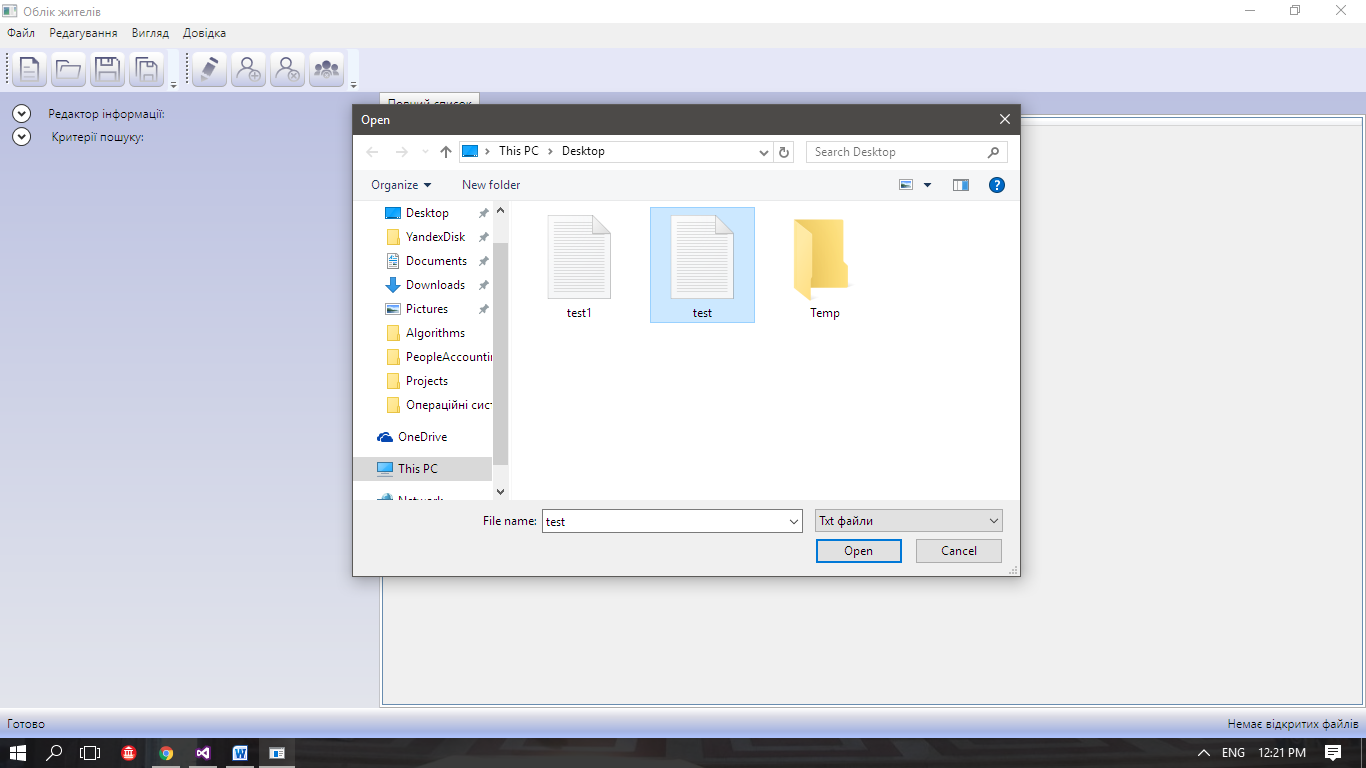
this.paragraph.Inlines.Add(content);

}

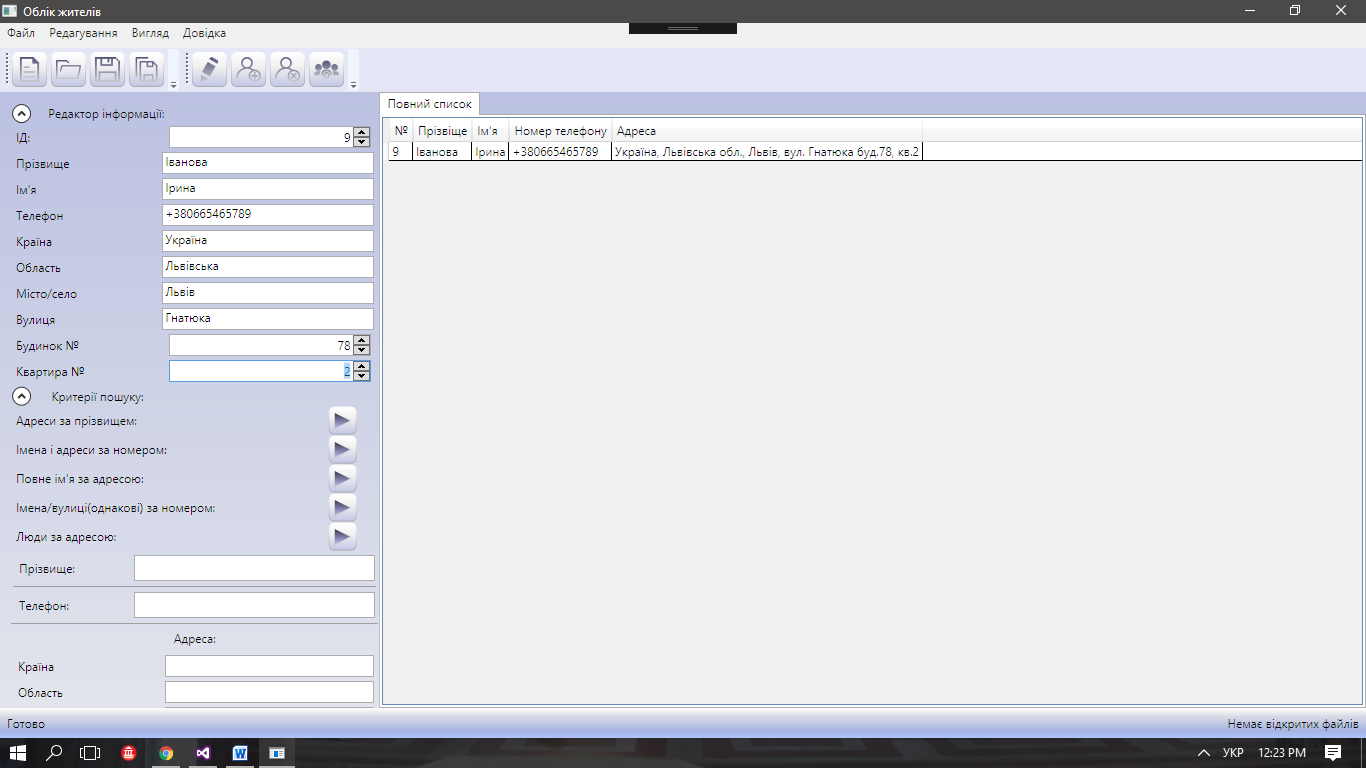
}

}

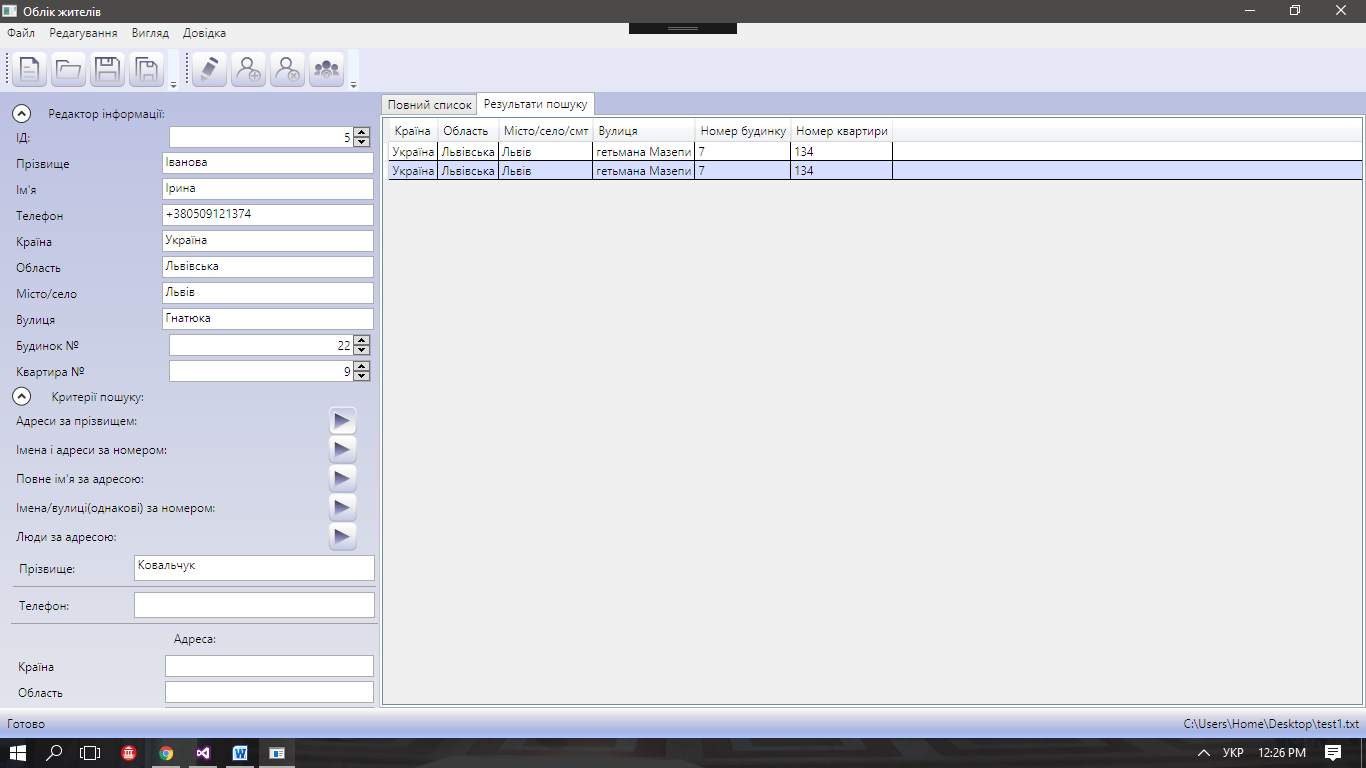
**Протокол роботи програми**



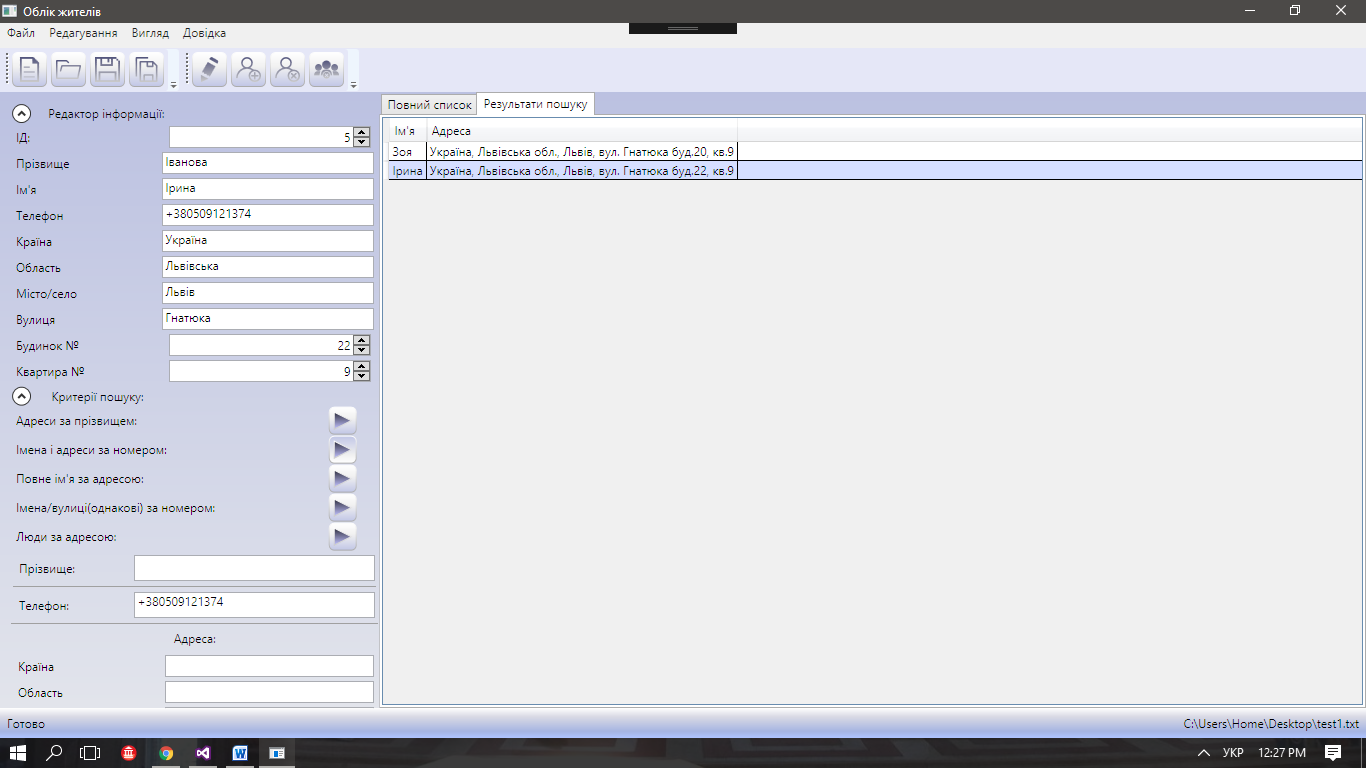
*Рис. 4 «Відкриття списку з текстового файлу»*



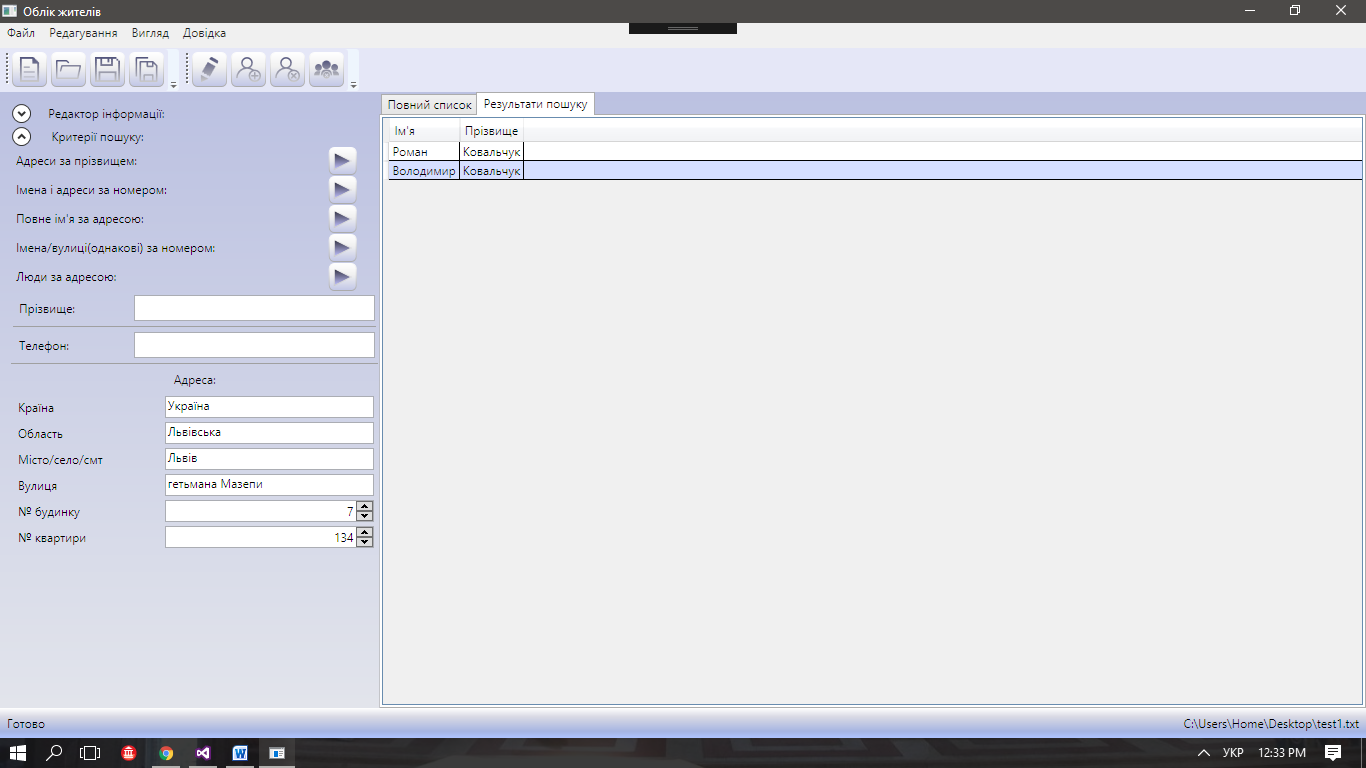
*Рис. 5 «Заповнення списку з клавіатури»*



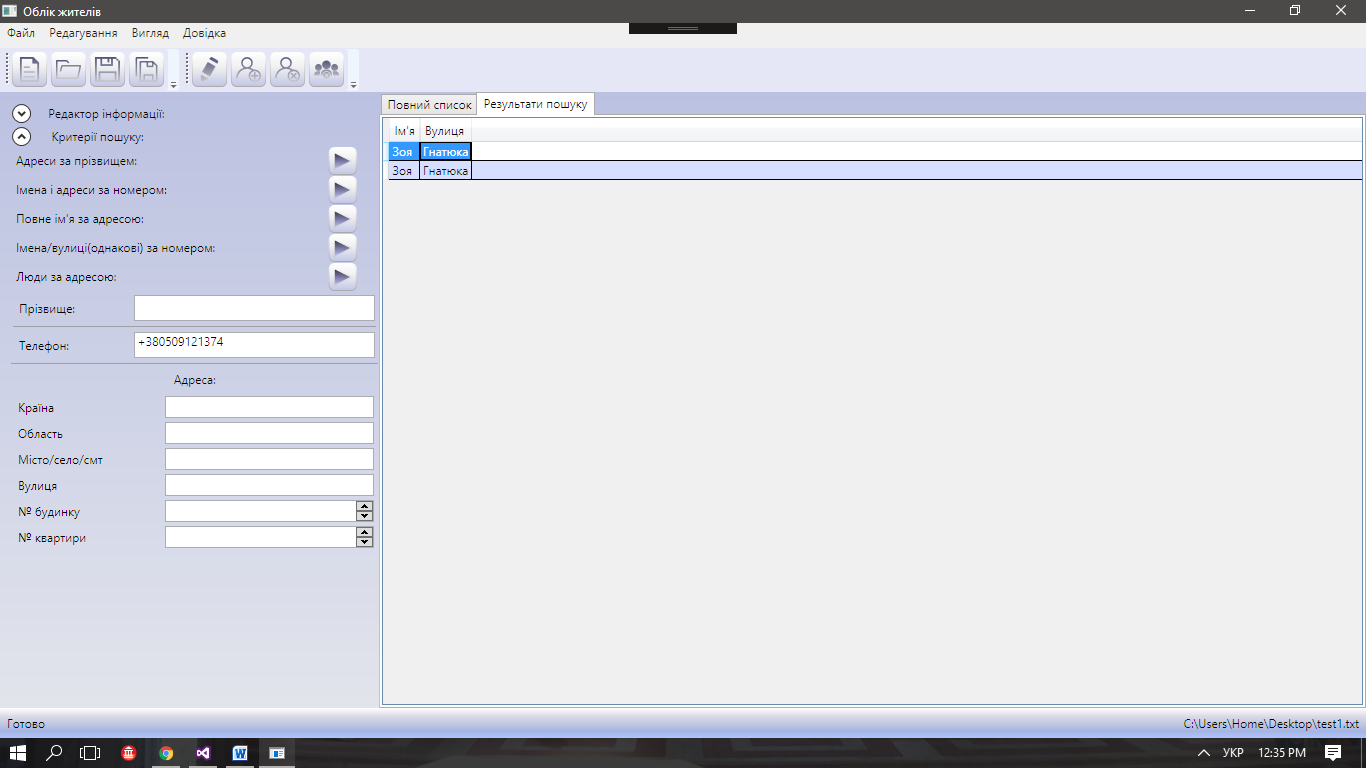
*Рис. 6 «Пошук адреси людей за заданим прізвищем»*



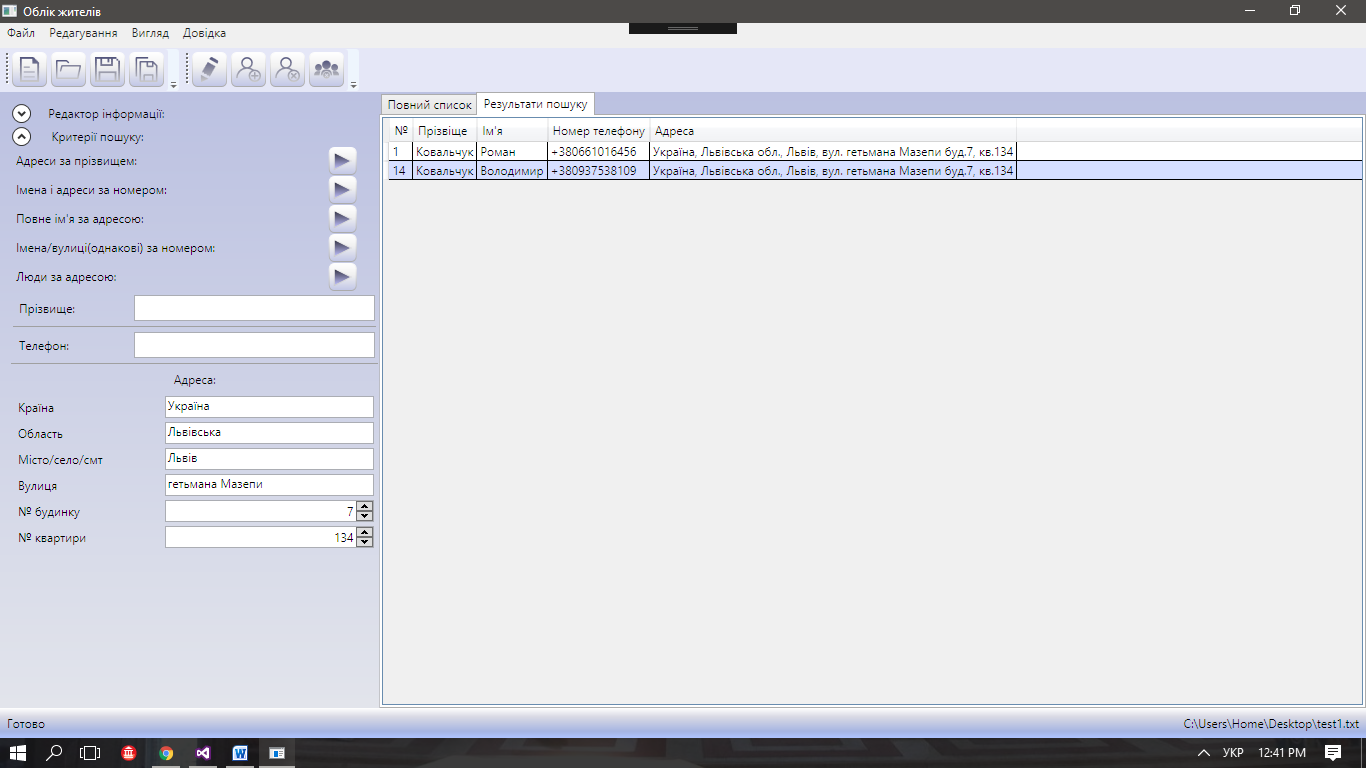
*Рис. 7 «Пошук імен та адрес за номером телефона»*



*Рис. 8 «Пошук ім’я та прізвища за адресою»*



*Рис.9 «Однакові імена та вулиці за номером»*



*Рис. 10 «Інформація про людей за адресою»*

**Інструкція користувача**

1. **Компоненти ПЗ.**

Програму розроблено на мові програмування С# у середовищі розробки Microsoft Visual Studio 2015 і вона може експлуатуватися під управлінням сімейства операційних систем Windows. Під час проектування системи брався до уваги обєктно-орієнтований підхід до програмуванння.

1. **Системні вимоги.** Для коректної роботи програми необхідна користувацька машина з процесором не менше 200 MHz, оперативною пам’яттю не менше 256 Mb.
2. **Встановлення ПЗ.**

Для встановлення та роботи з ПЗ необхідно відкрити файл PeopleAccounting.exe.

1. **Налаштування ПЗ.**

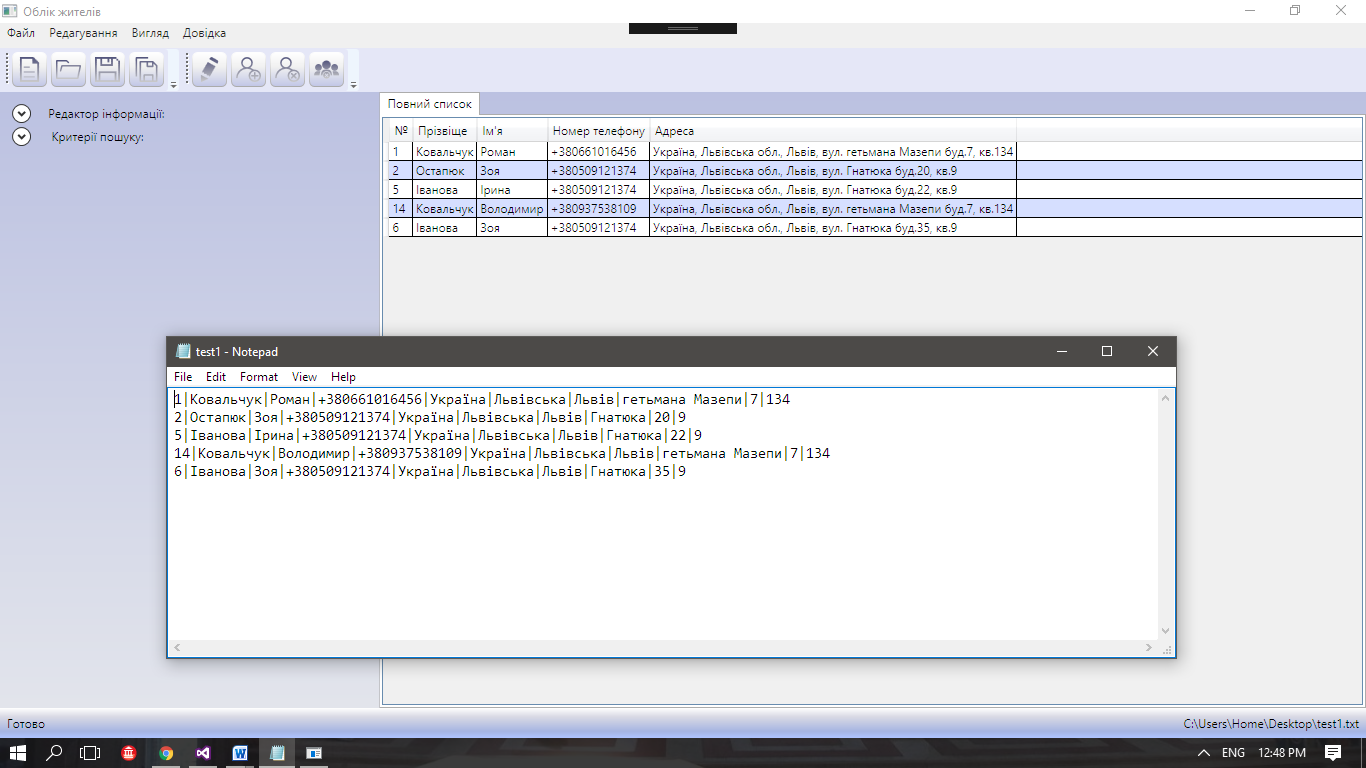
Для роботи програми необхідно не менше оперативної пам’яті, ніж 256 МБ, 5 МБ місця на диску.

1. **Базові функції.**
   1. Програма відкриває текстові файли \*.txt і переносить їх у таблицю, зручну для читання: пункт головного меню ***Файл->Відкрити***… або кнопка панелі інструментів. Файл повинен бути заповнений рядками, розділеними символом переносу рядка (кожний запис з нового рядка), дані про людину мають бути організовані у форматі:

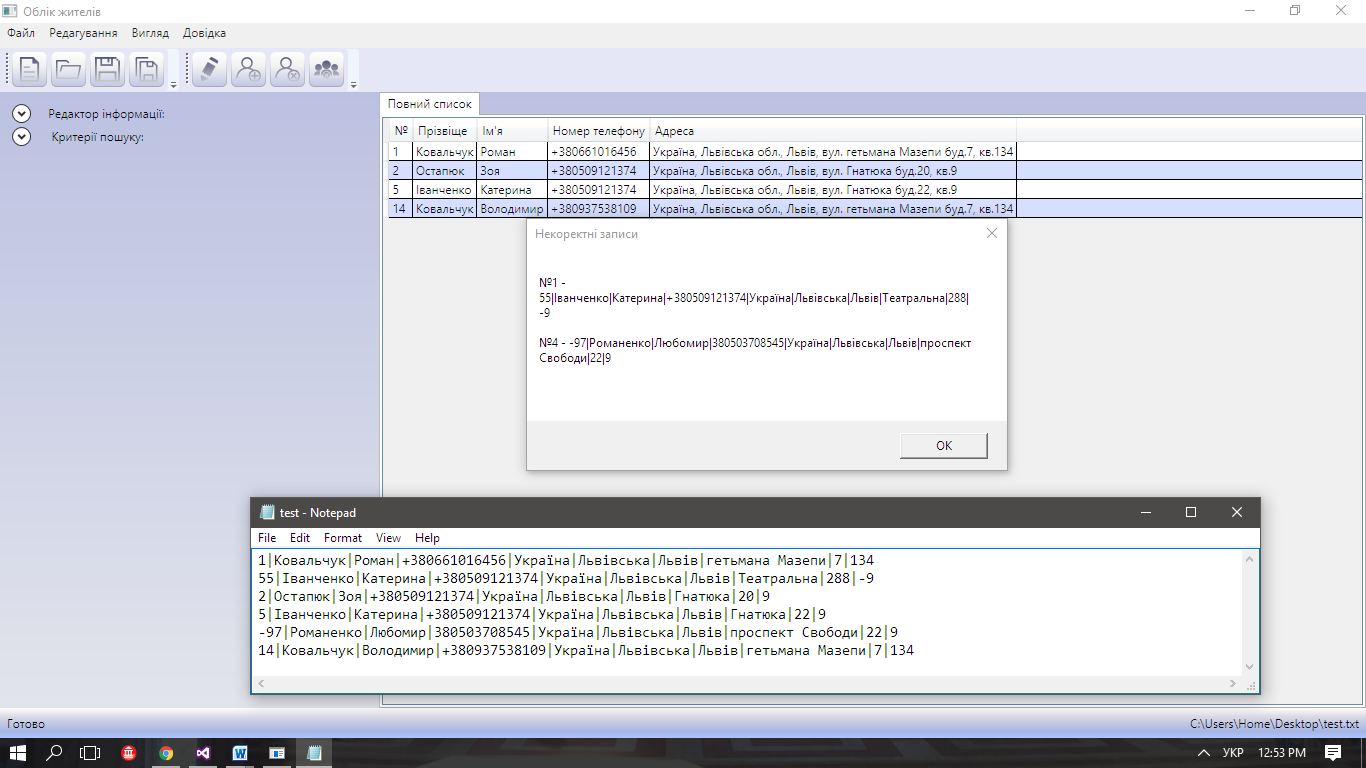
***{ІД}|{Прізвище}|{Ім’я}|{Телефону(«+380…»)}|{Країна}|{Область}|{Місто/село/смт}|***

***{Вулиця}|{Номер будинку}|{Номер квартири}***

Опрацьовуються лише ті рядки файлу, для яких розрахований функціонал ПЗ. (Рис. 11). Якщо в файлі деякі рядки не можуть бути інтерпретовані як інформація про людину, то вони будуть занесені до журналу некоректних записів, вміст якого відобразиться одразу після завершення заповнення таблиці (Рис. 12).

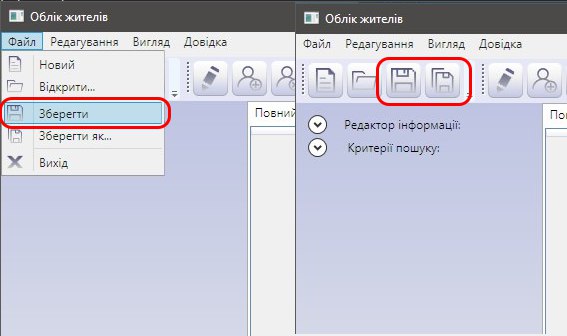


*Рис. 11 «Відкритий файл»*



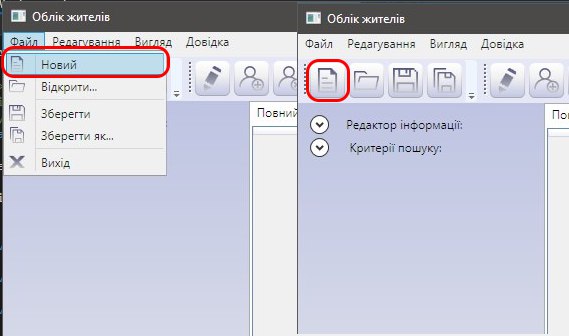
*Рис. 12 «Повідомлення про некоректні записи»*

* 1. Для збереження таблиці у текстовий файл потрібно перейти до пункту головного меню ***Файл->Зберегти***, або, якщо потрібно створити новий документ, ***Файл->Зберегти як…*** або за допомогою відповідних кнопок на панелі інструментів.

****

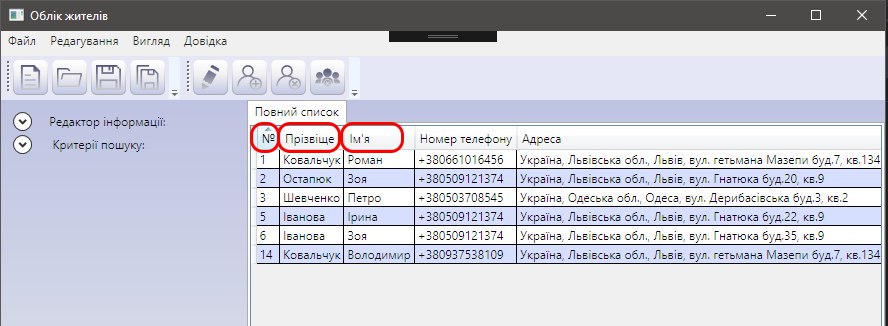
*Рис. 13 «Збереження файлу»*

* 1. Щоб створити новий файл, треба перейти до пункту головного меню ***Файл->Новий*** або скористатись кнопкою на панелі інструментів.

.****

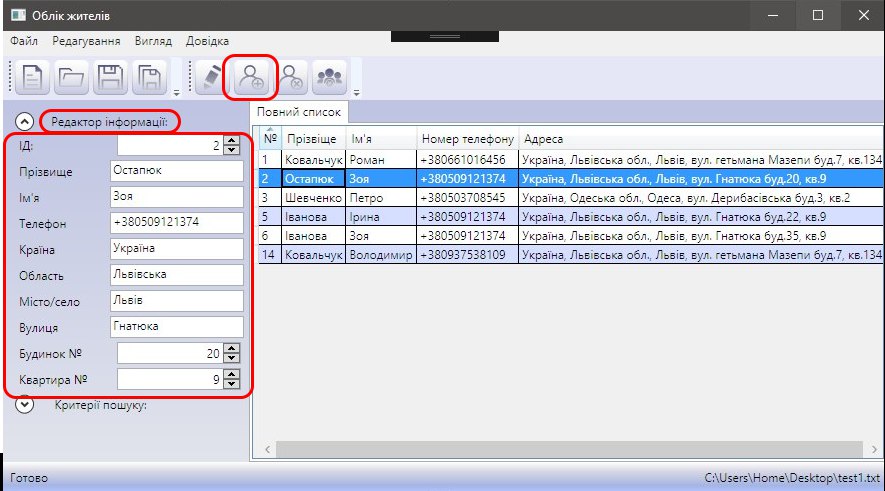
*Рис. 14 «Створення списку»*

* 1. Сортувати список можна за ІД, прізвищем або іменем. Для цього необхідно натиснути на заголовок відповідної колонки в таблиці з записами.

****

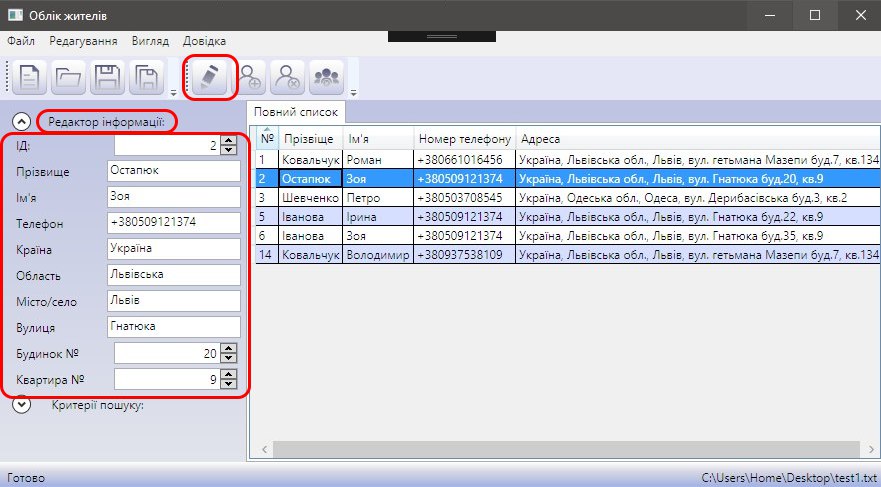
*Рис. 15 «Сортування записів»*

* 1. Щоб додати новий запис в таблицю, заповніть коректно форму редактора інформації на оберіть пункт головного меню ***Редагування->Додати до списку,*** або відповідну кнопку панелі інструментів.

****

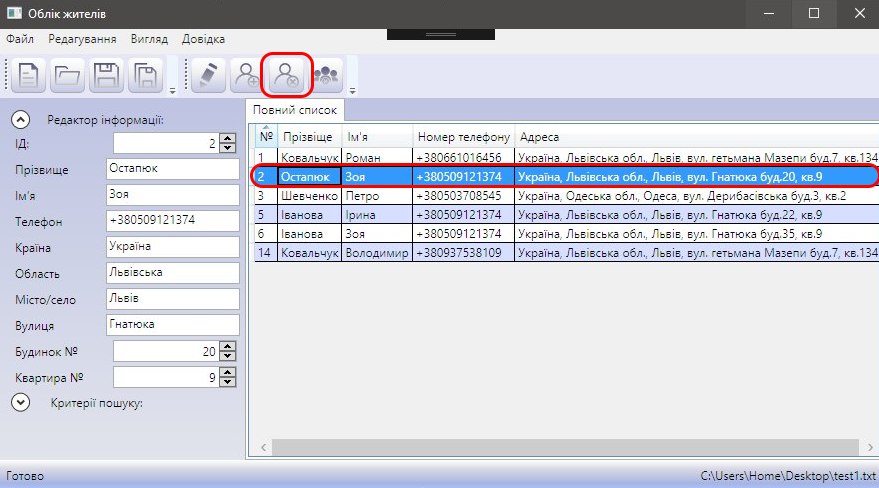
*Рис. 16 «Додавання запису»*

* 1. Редагування запису в таблиці можна здійснити, натиснувши на кнопці панелі інструментів, попередньо коректно заповнивши форму редактора інформації.

****

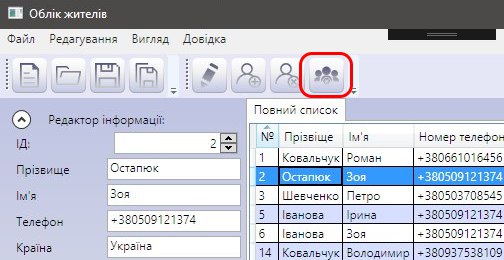
*Рис. 17 «Редагування запису»*

* 1. Щоб видалити запис, треба виділити його в таблиці, натиснувши на нього і обрати пункт головного меню ***Редагування->Видалити запис***, або натиснути кнопку на панелі інструментів.

****

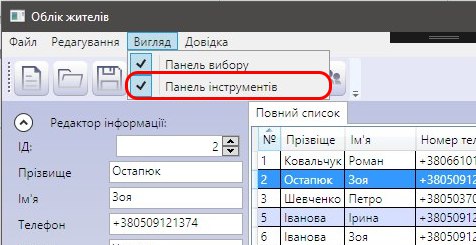
*Рис. 18 «Видалення запису»*

* 1. Щоб очистити весь список, оберіть пункт головного меню ***Редагування->Очистити список*** або відповідну кнопку на панелі інструментів.

****

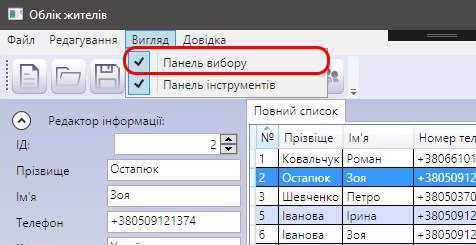
*Рис. 19 «Очищення списку»*

* 1. Щоб приховати панель інструментів, перейдіть у пункт головного меню ***Вигляд->Панель інструментів*** та зніміть галочку.

****

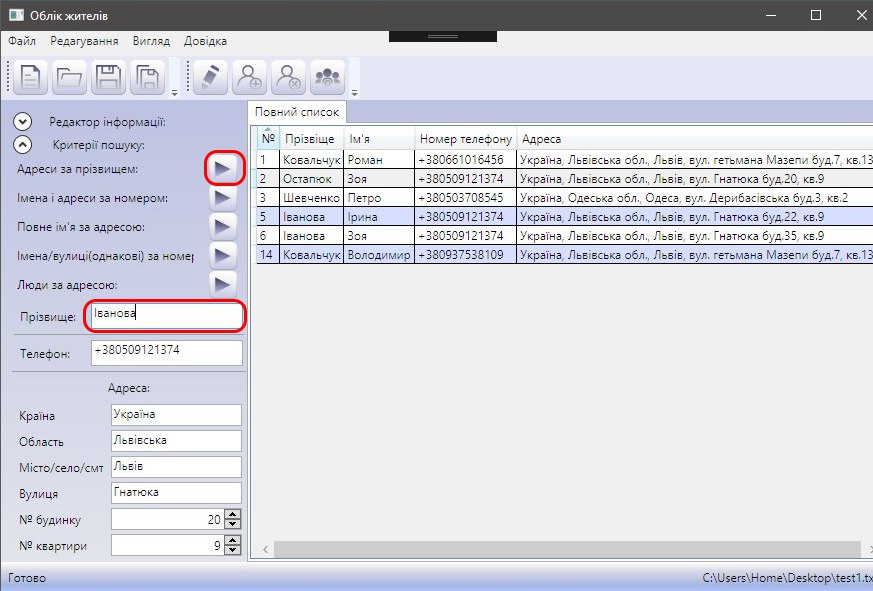
*Рис. 20 «Видимість панелі інструментів»*

* 1. Щоб приховати панель з формою редактора інформації та категоріями пошуку, перейдіть у пункт головного меню ***Вигляд->Панель вибору*** та зніміть галочку.

****

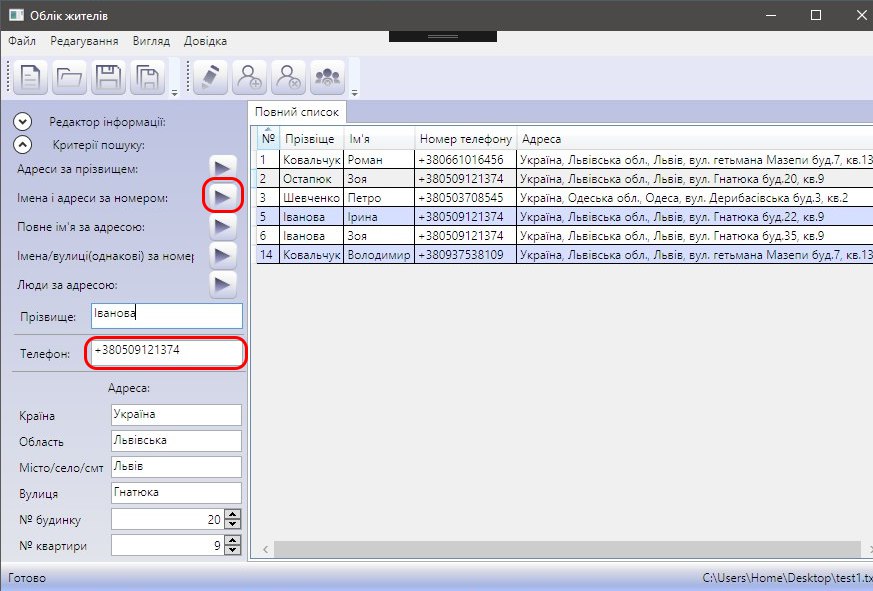
*Рис. 21 «Видимість панелі вибору»*

* 1. Щоб знайти адреси людей за прізвищем, введіть його в форму критеріїв пошуку та натисніть на кнопку напроти тексту ***«Адреси за прізвищем:».*** Результати будуть відображені в новій вкладці поряд з вкладкою з основним списком.

****

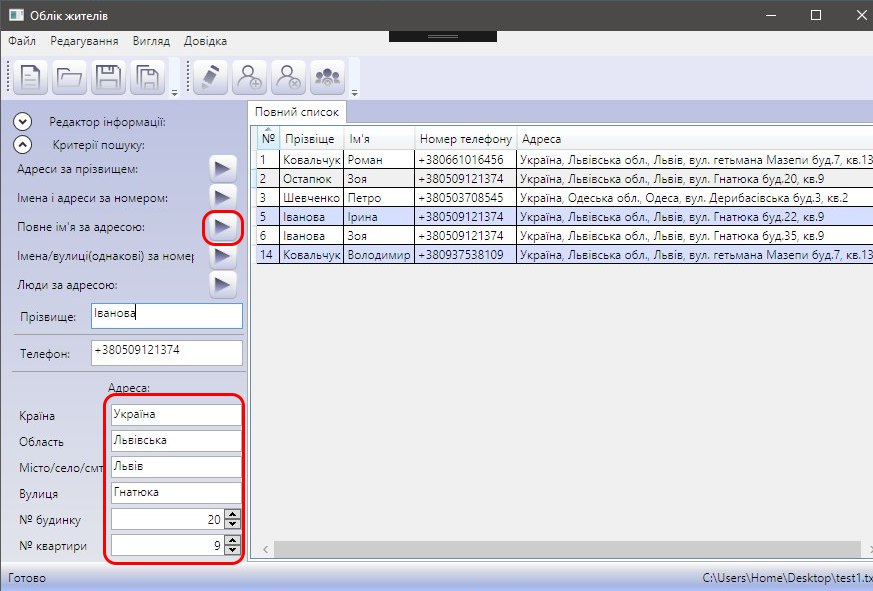
*Рис. 22 «Адреси за прізвищем»*

* 1. Щоб знайти імена та адреси людей за номером, введіть його в форму критеріїв пошуку та натисніть на кнопку напроти тексту ***«Імена і адреси за номером:».*** Результати будуть відображені в новій вкладці поряд з вкладкою з основним списком.

****

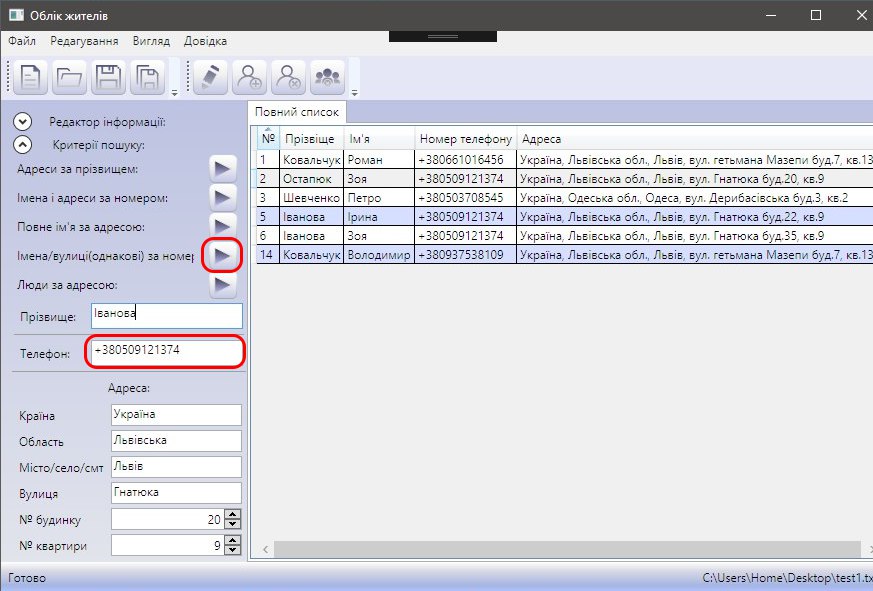
*Рис. 23 «Імена та адреси за номером»*

* 1. Щоб знайти імена та прізвища людей за адресою, введіть її в форму критеріїв пошуку та натисніть на кнопку напроти тексту ***«Повне ім’я за адресою:».*** Результати будуть відображені в новій вкладці поряд з вкладкою з основним списком.

****

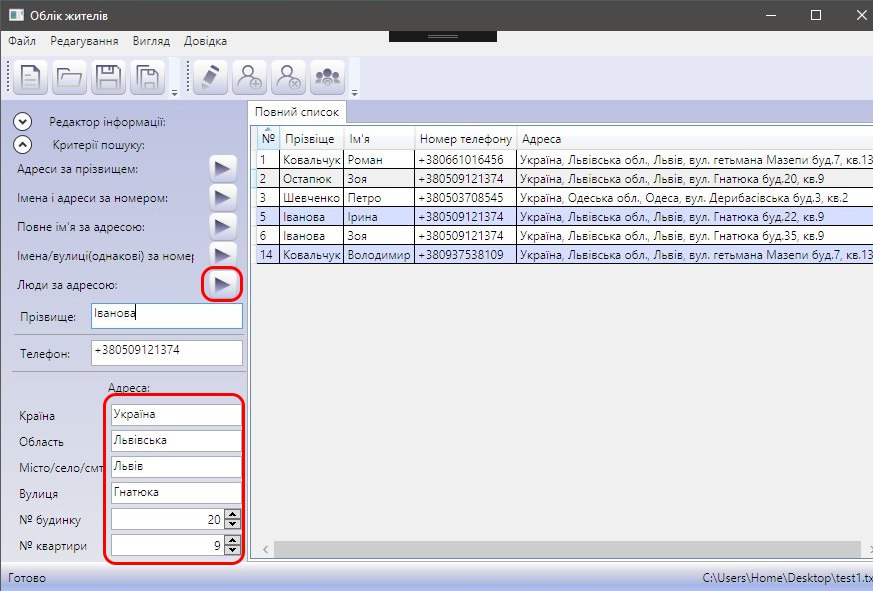
*Рис. 24 «Повне ім’я за адресою»*

* 1. Щоб знайти однакові імена та вулиці людей за однаковим номером, введіть його в форму критеріїв пошуку та натисніть на кнопку напроти тексту ***«Імена/вулиці(однакові) за номером:».*** Результати будуть відображені в новій вкладці поряд з вкладкою з основним списком.

**

*Рис. 25 «Імена/вулиці(однакові) за номером»*

* 1. Щоб знайти повну інформацію про людей за прізвищем, введіть його в форму критеріїв пошуку та натисніть на кнопку напроти тексту ***«Люди за прізвищем:».*** Результати будуть відображені в новій вкладці поряд з вкладкою з основним списком.

**

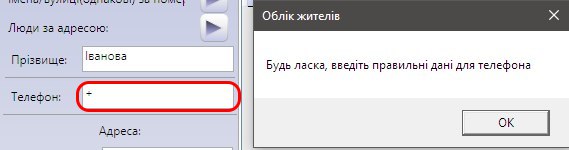
*Рис. 26 «Люди за прізвищем»*

1. **Аналіз помилок.**

Якщо файл занадто великий і програма не може здійснити операції над ним, вона аварійно завершить роботу. Файл не зміниться.

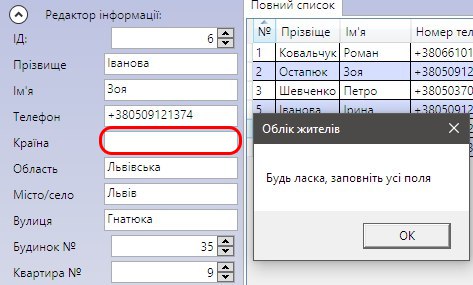
Можливі такі повідомлення про помилку:

1. **"Будь ласка, введіть правильні дані для телефона"** Перевірте коректність введеного номера телефону: він повинен починатись з коду України «+380» та містити загалом 13 цифр разом з знаком плюс.

**

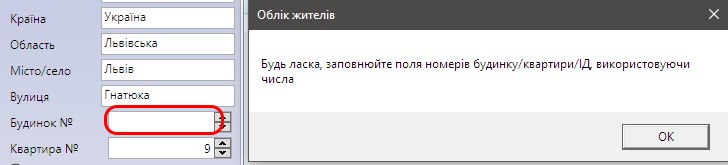
*Рис. 27 «Будь ласка, введіть правильні дані для телефона»*

1. **"Будь ласка, заповніть усі поля"** Необхідно перевірити чи всі поля адреси чи редактора інформації заповнені**.**

****

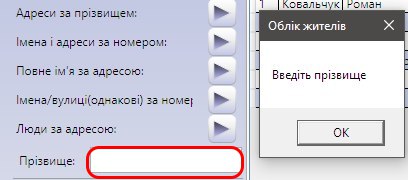
*Рис. 28 «Будь ласка, заповніть усі поля»*

1. **"Будь ласка, заповнюйте поля номерів будинку/квартири/ІД, використовуючи числа"** Перевірте, чи в поля номерів будинку, квартири чи ІД записані натуральні числа.

****

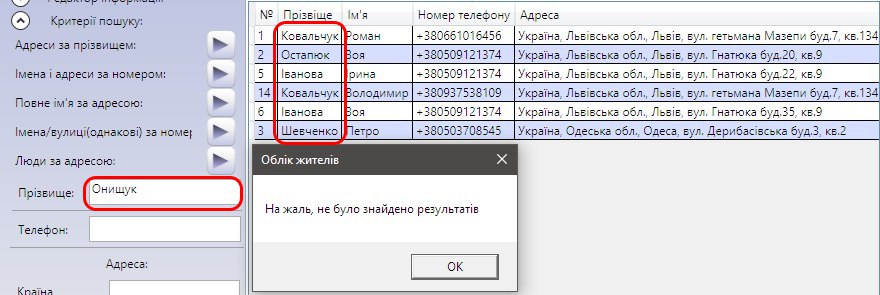
*Рис. 29 «Будь ласка, заповнюйте поля номерів будинку/квартири/ІД, використовуючи числа"*

1. **"Введіть прізвище"** Під час намагання програми знайти людей, використовуючи критерії прізвища, було визначено, що користувач не заповнив поле.

****

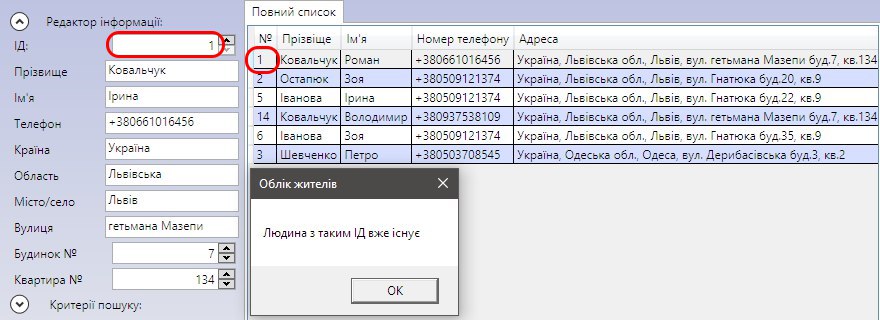
*Рис. 30 «Введіть прізвище»*

1. **"На жаль, не було знайдено результатів"** В головному списку не було знайдено записів, що відповідають критерію пошуку.

****

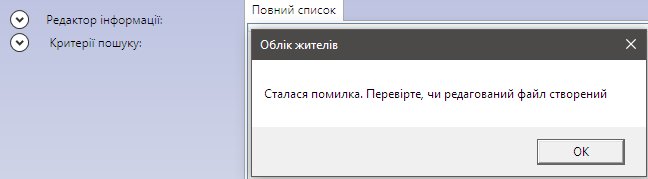
*Рис. 31 «На жаль, не було знайдено результатів»*

1. **"Людина з таким ІД вже існує"** Під час додавання запису, було визначено, що в списку вже є людина з таким же ідентифікатором, змініть його, або видаліть існуючий запис.

****

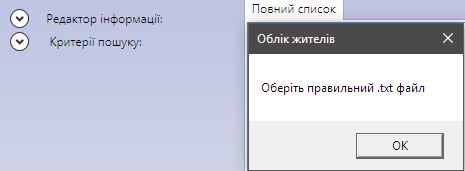
*Рис. 32 «Людина з таким ІД вже існує"*

1. **"Сталася помилка. Перевірте, чи редагований файл створений"** Під час спроби зберегти зміни в редагованому файлі виявилось, що він був стертий з диску або переміщений. Оберіть функцію ***Файл->Зберегти як…*** .



*Рис. 33 «Сталася помилка. Перевірте, чи редагований файл створений»*

1. **"Оберіть правильний .txt файл"** Під час спроби записати список в текстовий файл сталась помилка. Оберіть інший файл.



*Рис. 34 «Оберіть правильний .txt файл»*