Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

З дисципліні «Об’єктно -орієнтоване програмування»

ДОВІДНИК ГЕОГРАФА

Керівник, Ст. викл. Черепанова Ю.Ю.

Студент гр. ПЗПІ-21-8 Хмелик О. В.

Комісія:

Проф. Бондарєв В.М.,

Доц. Побіженко І. О.,

Ст. викл. Черепанова Ю.Ю.

Харків 2022

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра *програмної інженерії*

Рівень вищої освіти *перший (бакалаврський)*

Дисципліна *Об’єктно-орієнтоване програмування*

Спеціальність *121 Інженерія програмного забезпечення*

Освітня програма: *Програмна інженерія*

Курс 1 . Група *ПЗПІ-21-8*. Семестр *2 .* .

***ЗАВДАННЯ***

***на курсовий проект студента***

\_\_\_\_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_**Хмелика Олега Володимировича**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1 Тема проекту:Довідник Географа**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

2 Термін здачі студентом закінченого проекту: ***“1” - липня - 2022 р***.

3 Вихідні дані до проекту:

*Специфікація програми, методичні вказівки до виконання курсової роботи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

4 Зміст розрахунково-пояснювальної записки:

*Вступ, специфікація програми, проектна специфікація, інструкція користувача, висновки, додатки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5 Перелік графічного матеріалу:

\_\_\_*Схема об’єктної моделі, алгоритми, приклади екранних форм\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН:**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 16.02.2021 - 14.03.2022 р. |
| 2 | Формулювання вимог до програми | 14.03.2022 –.2022 р. |
| 3 | Розробка підсистеми зберігання та пошуку даних. | 14.03.2022 –.2022 р. |
| 4 | Розробка функцій ….. | 14.03.2022 –.2022 р. |
| 5 | Розробка функцій зберігання та завантаження даних | .2022 –.2022 р. |
| 6 | Тестування і доопрацювання розробленої програмної системи. | .2022 –.2022 р. |
| 7 | Оформлення пояснювальної записки, додатків, графічного матеріалу | .2022 –.2022 р. |
| 8 | Захист | 01.07.2022 – 27.07.2022 р. |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Черепанова Ю.Ю

« 26 »\_лютого\_\_\_\_\_\_\_\_\_\_ 2022 р.

**РЕФЕРАТ**

Пояснювальна записка до курсової роботи: 55 с., 13 рис., 3 табл., 1 додаток, джерел.

ГЕОГРАФІЯ, КЛАС, КРАЇНА, МАПА, МІСТО, МОВА ПРОГРАМУВАННЯ C#, ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, ПЛАТФОРМА .NET, ФОРМА.

Метою роботи є розробка програми “Довідник Географа” на засадах об'єктно-орієнтованого програмування.

Методи розробки базуються на використанні середи розробки Microsoft Visual Studio 2022, Windows Forms, платформи .NET Framework 6.0, мови програмування C#.

В результаті отримана програма під назвою “Довідник Географа”, яка дозволяє продивлятися списки міст, регіонів, країн та континентів. Можливість редагувати існуючі міста, регіони, континенти, виконувати пошук по існуючим критеріям, зберігати його результати до файлу, відображати положення міст на мапі, додавати нові міста, регіони та країни.

**ЗМІСТ**

Вступ .....................................................................................................................................6

1. Специфікація програми....................................................................................................8  
1.1 Постановка задачі................................................................................................8  
 1.2 Очікуваний інтерфейс проєкту...........................................................................8  
 1.3 Вхідні та вихідні дані..........................................................................................14

2.Проектування програми.................................................................................................16  
 2.1 Розробка ієрархії класів......................................................................................16  
 2.2 Таблиця даних.....................................................................................................17  
 2.3 Таблиця методів..................................................................................................18  
 2.4 Аномалії...……....................................................................................................35  
3. Інструкція користувача...................................................................................................36

Висновки .............................................................................................................................40  
Перелік джерел посилання.................................................................................................41

Додаток А Код програми ...................................................................................................42

**ВСТУП**

Приведений курсовий проект є закріпленням знань, набутих при вивченні дисципліни „Об’єктно-орієнтоване програмування”[3], а саме:

Знання загальних принципів об’єктно-орієнтованого програмування, реалізація об’єктної моделі мови програмування, архітектури програми з графічним інтерфейсом користувача.

Вміння створювати програми в об’єктно-орієнтованій парадигмі, доцільно використовувати в програмах можливості мови програмування C#[4]; використовування бібліотечних класів для розробки графічних застосувань з загальними засадами об’єктно-орієнтованого проектування

Використання при створенні програми таких основних принципів об’єктно-орієнтованого програмування, як успадкування, поліморфізм, інкапсуляція та абстракція.

Програма «Довідник Географа» відповідає усім загальним вимогам щодо написання програми, а саме:

Стійкість програми. Програма працює при будь-яких непередбачених діях користувача, також уся введена користувачем інформація перевіряється програмно;

Функціональна повнота. Реалізовані усі функції, зазначені в специфікації програми, навіть додавання та редагування даних;

Терміни та інтерфейс. На сторінках програми використовуються тільки терміни, зрозумілі користувачеві. У повідомленнях користувачеві програма дотримується норм ввічливості;

Використання клавіатури. На будь-якому етапі натискання будь-якої клавіші ігнорується або викликає передбачені дії.;

Дана робота не втратила свою актуальність, бо структуризація даних щодо географічних об’єктів актуальна і зараз, бо міста, регіони та країни доволі часто змінюють свої назви, з часом росте численність населення, тому потрібно мати зручну програму з актуальними даними щодо міст, країн, регіонів та можливістю їх редагувати чи добавляти.

Мова інтерфейсу програми – англійська.

Мета курсової роботи – створення програми «Довідник Географа», яка містить списки міст, країн, регіонів, континентів та можливість їх редагувати або створювати нові.

**1** **СПЕЦИФІКАЦІЯ ПРОГРАМИ**

* 1. Постановка задачі

Наша програма повинна відповідати таким критеріям та мати такий функціонал:

Проект «Довідник Географа» зобов’язую наявність таких класів, як місто, регіон, країна, континент, та класу, який буде містити данні щодо географічних об’єктів(Database).

Середовище розробки майбутнього проекту “Windows Forms”.

Має існувати можливість передивлятися, редагувати та добавляти нові географічні об’єкти, такі як міста, регіони та країни.

Має існувати можливість сортувати а також шукати за певними критеріями географічні об’єкти, такі як міста, регіони, країни, та зберігати результати пошуку в файл.

Має існувати можливість відображати міста на мапі.

Має існувати можливість повертатися на попередню сторінку, або виходити з програми при натисканні Escape.

* 1. Очікуваний інтерфейс проекту

Запускаючи програму, користувач бачить головну сторінку. Імовірний інтерфейс приведено на рисунку 1.1.

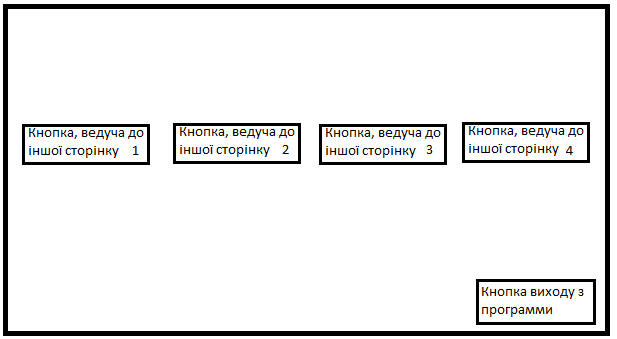


Рисунок 1.1 – Головна сторінка

При натисканні на першу кнопку головної сторінки, користувач потрапляє на сторінку з містами. Імовірний інтерфейс приведено на рисунку 1.2.



Рисунок 1.2 – Сторінка з містами

При натисканні на кнопку додавання нового елементу на сторінці з містами або кнопки редагування елемента в таблиці на сторінці з містами, користувач потрапляє на сторінку додавання або редагування міст. Імовірний інтерфейс приведено на рисунку 1.3.

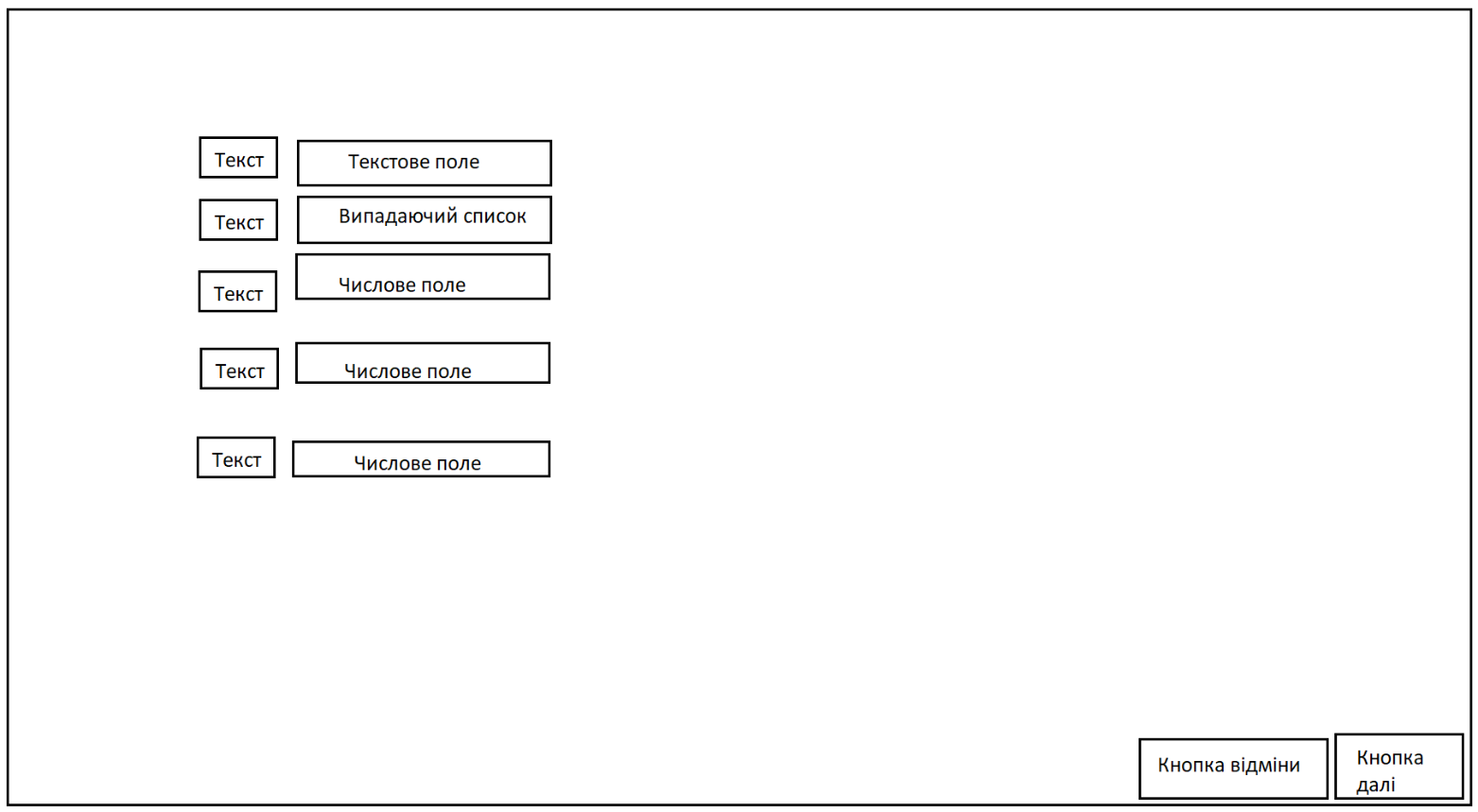


Рисунок 1.3 – Сторінка додавання або редагування міст.

При натисканні на кнопку мапи в таблиці на сторінці з містами, користувач потрапляє на сторінку мапи. Імовірний інтерфейс приведено на рисунку 1.4.



Рисунок 1.4 – Сторінка мапи

При натисканні на другу кнопку головної сторінки, користувач потрапляє на сторінку з регіонами. Імовірний інтерфейс приведено на рисунку 1.5.



Рисунок 1.5 – Сторінка з регіонами

При натисканні на кнопку додавання або редагування регіонів в таблиці, на сторінці з регіонами, користувач потрапляє на сторінку додавання або редагування регіонів. Імовірний інтерфейс приведено на рисунку 1.6.

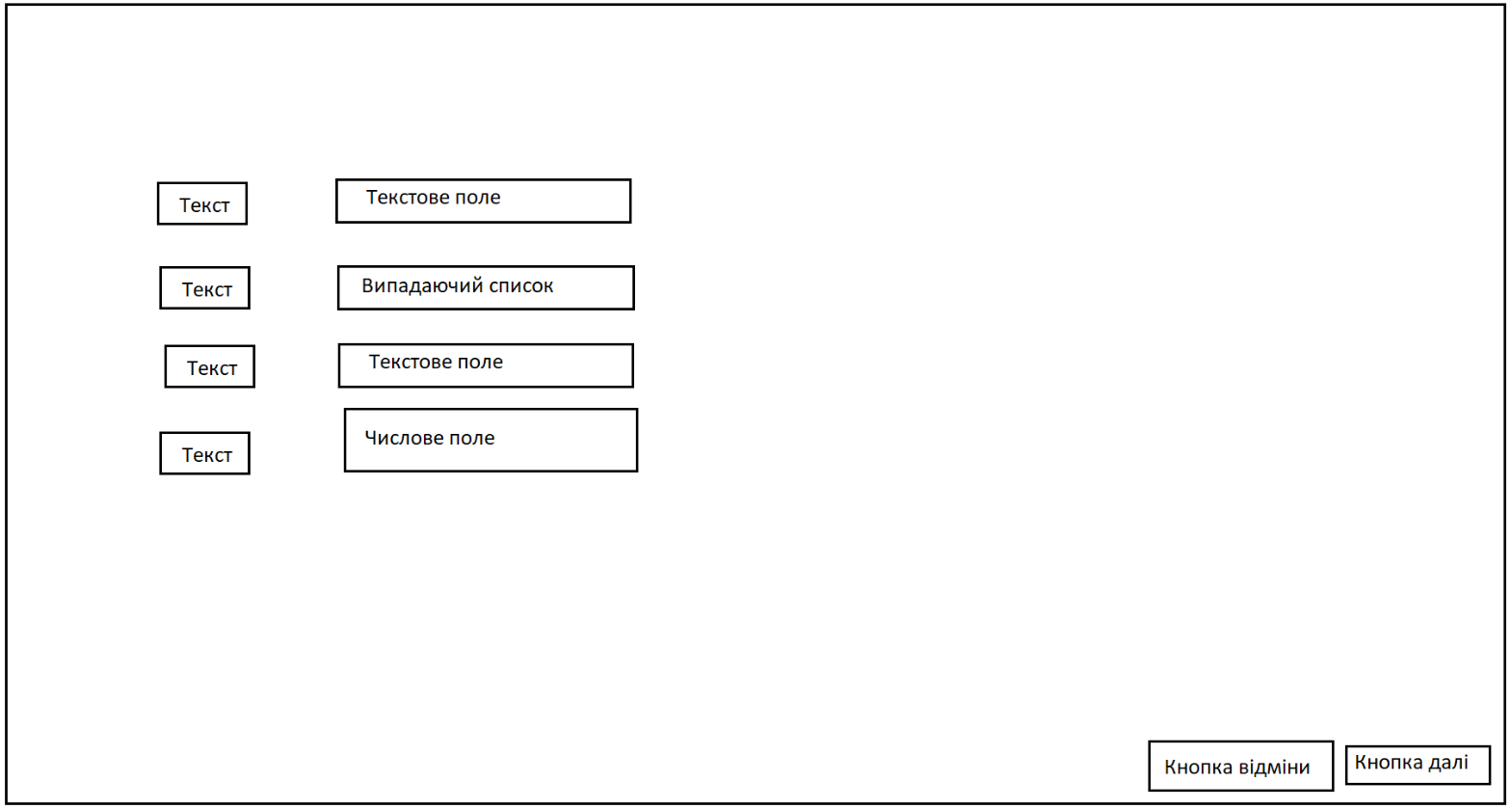


Рисунок 1.6 – Сторінка додавання або редагування регіонів

При натисканні на третю кнопку головної сторінки, користувач потрапляє на сторінку з країнами. Імовірний інтерфейс приведено на рисунку 1.7.



Рисунок 1.7 – Сторінка з країнами

При натисканні на кнопку додавання або редагування країн в таблиці, на сторінці з країнами, користувач потрапляє на сторінку додавання або редагування країн. Імовірний інтерфейс приведено на рисунку 1.8.

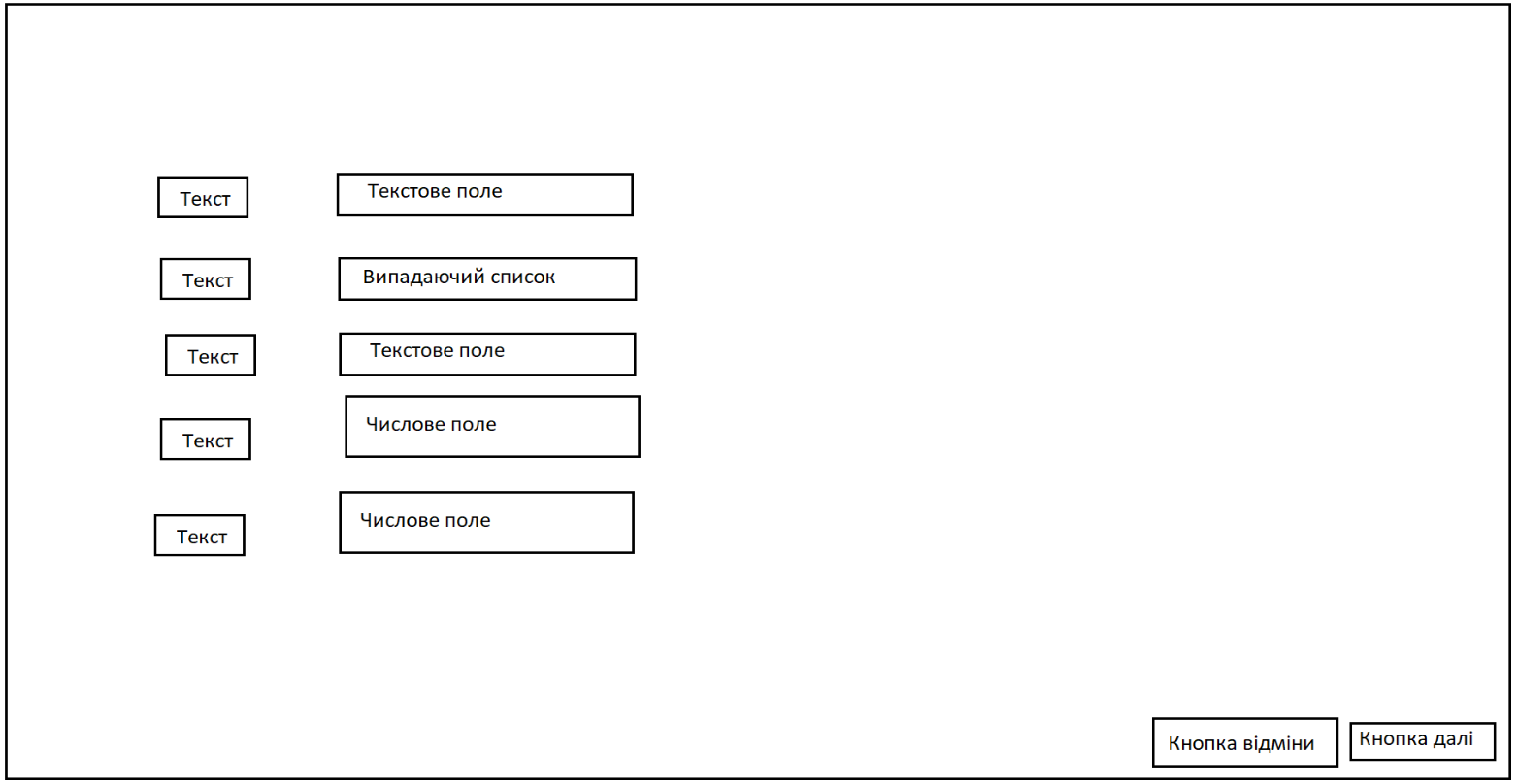


Рисунок 1.8 – Сторінка додавання або редагування країн

При натисканні на четверту кнопку головної сторінки, користувач потрапляє на сторінку з континентами. Імовірний інтерфейс приведено на рисунку 1.9.

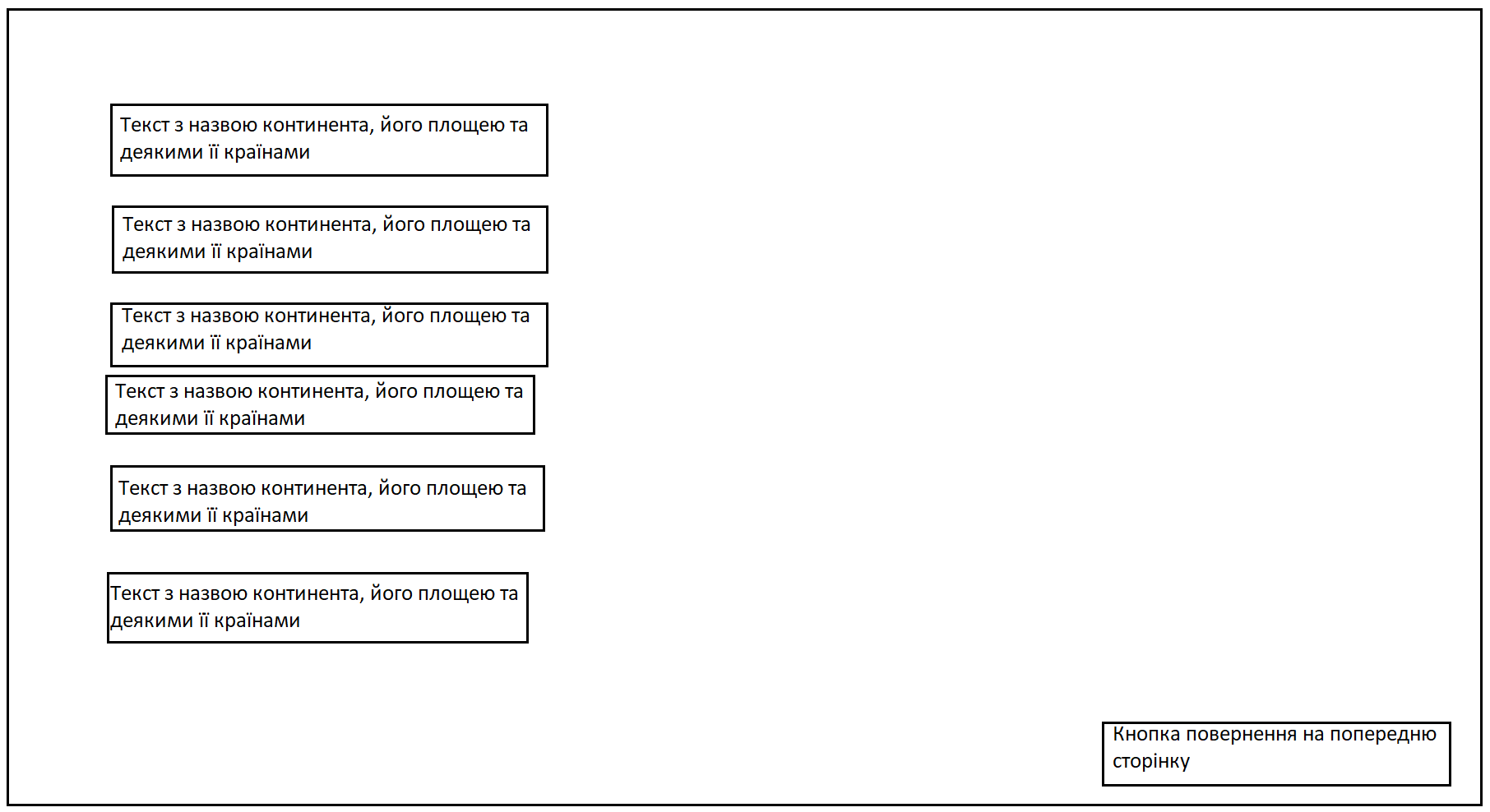


Рисунок 1.9 – Сторінка з континентами

1.3 Вхідні та Вихідні дані

Програма «Довідник Географа» отримає дані з текстових файлів, які потім зберігаються у відповідному класі, а саме:

З файлу “cities.txt” програма отримує список інформації, що містить унікальний ідентифікаційний номер міста, назву міста, назву країни, до якої належить це місто, довготу та широту цього міста, численність населення міста.

З файлу “regions.txt” програма отримує список інформації, що містить унікальний ідентифікаційний номер регіону, назву регіону, тип регіону, унікальний ідентифікаційний номер країни, до якої належить цей регіон, численність населення регіону.

З файлу “countries.txt” програма отримує список інформації, що містить унікальний ідентифікаційний номер країни, назву країни, площу, яку займає ця країна, численність населення країни, вид правління на території цієї країни, унікальний ідентифікаційний номер столиці цієї країни.

З файлу “continents.txt” програма отримує список інформації, що містить унікальний ідентифікаційний номер континенту, назву континенту, численність населення континенту, кількість країн, які належать цьому континенту, перелічення унікальних ідентифікаційний номерів країн, які належать цьому континенту.

Програма «Довідник Географа» показує розташування міста на мапі, зберігає дані до текстових файлів, при редагуванні або додаванні міст, регіонів, країн або збереженні результатів пошуку, а саме:

У файл “cities.txt” програма перезаписує дані при зміні або додаванні нового міста.

У файл “regions.txt” програма перезаписує дані при зміні або додаванні нового регіону.

У файл “cities.txt” програма перезаписує дані при зміні або додаванні нової країни.

У файл “ search\_result.txt” програма зберігає дані пошуку виконаного на сторінці міст, регіонів, країн.

На сторінці мапи користувач може бачити приблизне розташування міста на політичній мапі.

**2 ПРОЕКТУВАННЯ ПРОГРАМИ**

2.1 Розробка ієрархії класів

Ієрархії усіх класів, що наведені в проекті, відповідає таблиця 2.1.

Таблиця 2.1 – Таблиця класів.

|  |  |
| --- | --- |
| Клас | Інформація |
| GeographyUnit | Абстрактний клас, предок класів Coin, City, Region, Country. |
| City | Клас, який містить інформацію щодо міста. |
| Region | Клас, який містить інформацію щодо регіону. |
| Country | Клас, який містить інформацію щодо країни. |
| Continent | Клас, який містить інформацію щодо континенту. |
| Database | Клас, який містить усі вхідні дані та відповідає за їх зміну. |
| MainPage | Клас, який є головною сторінкою програми. |
| CityPage | Клас, який є сторінкою з містами програми. |
| RegionPage | Клас, який є сторінкою з регіонами програми. |
| CountryPage | Клас, який є сторінкою з країнами програми. |
| ContinentPage | Клас, який є сторінкою з континентами програми. |
| MapPage | Клас, який є сторінкою, на якій зображено обране місто на карті |
| AddOrChangeCityPage | Клас, який є сторінкою, на якій можливо редагувати або додавати нові міста. |
| AddOrChangeRegionPage | Клас, який є сторінкою, на якій можливо редагувати або додавати нові регіони. |
| AddOrChangeCountryPage | Клас, який є сторінкою, на якій можливо редагувати або додавати нові країни. |
| Program | Клас, який запускає головну сторінку програми. |

2.2 Таблиця даних

Далі приведена таблиця 2.2, яка містить інформацію щодо всіх полів та властивостей програми.

Таблиця 2.2 – Таблиця даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Клас | Рівень доступу | Тип даних | Назва | Інформація |
| GeographyUnit | public | Guid | Uuid | Унікальний ідентифікаційний номер географічного об’єкту. |
| public | String | Name | Назва географічного об’єкту. |
| public | int | Population | Численність населення географічного населення. |
| City | public | string | CountryName | Назва міста. |
| public | double | Latitude | Широта міста. |
| public | double | Longitude | Довгота міста. |
| Region | public | string | Type | Тип регіону. |
| public | Country | Country | Країна, в якій знаходиться регіон. |
| public | City | Capital | Столиця країни. |
| Country | public | int | Area | Площа країни. |
| public | string | GovernmentType | Тип правління на території країни. |
| public | City | Capital | Столиця країни. |
| Continent | public | List<Country> | Countries | Список країн, що є на території континенту. |
| Database | private | string | citiesPath | Шлях до файлу з усіма містами. |
| private | string | countriesPath | Шлях до файлу з усіма країнами. |
| private | string | regionsPath | Шлях до файлу з усіма регіонами. |
| private | string | continentsPath | Шлях до файлу з усіма континентами. |
| public | List<City> | Cities | Список усіх міст. |
| public | List<Region> | Regions | Список усіх регіонів. |
| public | List<Country> | Countries | Список усіх країн. |
| public | List<Continent> | Continents | Список усіх континентів. |
| MainPage | private | Database | database | База даних. |
| CityPage | private | int | curFirstCity | Число, яке є індексом першого міста, який відображається на екрані. |
| private | Database | database | База даних. |
| private | List<City> | filteredCities | Список міст, які відображаються на сторінці. |
| RegionPage | private | int | curFirstRegion | Число, яке є індексом першого регіона, який відображається на екрані. |
| private | Database | database | База даних. |
| private | List<Region> | filteredRegions | Список регіонів, які відображаються на сторінці. |
| CountryPage | private | curFirstCountry | curFirstCountry | Число, яке є індексом першої країни, яка відображається на екрані. |
| private | Database | database | База даних. |
| private | List<Country> | filtredCountries | Список країн, які відображаються на сторінці. |
| ContinentPage | private | Database | database | База даних. |
| private | List<Continent> | continents | Список континентів, які відображаються на сторінці. |
| MapPage | private | CityPage | parent | Сторінка, з якої була відкрита сторінка мапи. |
| AddOrChangeCityPage | private | Database | database | База даних. |
| private | CityPage | parent | Сторінка, з якої була відкрита сторінка зміни міста або додавання нового. |
| private | City | city | Місто, яке користувач редагує. |
| AddOrChangeRegionPage | private | Database | database | База даних. |
| private | RegionPage | parent | Сторінка, з якої була відкрита сторінка зміни регіону або додавання нового. |
| private | Region | region | Регіон, який користувач редагує. |
| AddOrChangeCountryPage | private | Database | database | База даних. |
| private | CountryPage | parent | Сторінка, з якої була відкрита сторінка зміни країни або додавання нової. |
| private | Country | country | Країна, яку користувач редагує. |

2.3 Таблиця методів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Клас | Рівень доступу | Вхідні дані | Вихідні дані | Назва | Інформація |
| City | public | Guid uuid, string name, string countryName, double latitude, double longitude, int population |  | City | Конструктор класу. Створює елемент класу. |
| public | string line |  | City | Конструктор класу. Створює елемент класу. |
| public |  | string | ToString | Перезапис метода ToString. |
| Region | public | Guid uuid, string name, string type, Country country, int population |  | Region | Конструктор класу. Створює елемент класу. |
| public |  |  | ToString | Перезапис метода ToString. |
| Country | public | Guid uuid, string name, int area, int population, string governmentType, City capital |  | Country | Конструктор класу. Створює елемент класу. |
| public |  | string | ToString | Перезапис метода ToString. |
| Continent | public | Guid uuid, string name, List<Country> countries |  | Continent | Конструктор класу. Створює елемент класу. |
| public |  | string | ToString | Перезапис метода ToString. |
| Database | public |  |  | Database | Конструктор класу. Створює елемент класу та зчитує дані з файлів. |
| public | string searchField, string searchValue, string orderByField | List<City> | GetCities | Повертає список міст в залежності від параметрів пошуку та сортування. |
| public | string searchField, string searchValue, string orderByField | List<Region> | GetRegions | Повертає список регіонів в залежності від параметрів пошуку та сортування. |
| public | string searchField, string searchValue, string orderByField | List<Country> | GetCountries | Повертає список країн в залежності від параметрів пошуку та сортування. |
| public | City newCity |  | AddCity | Додає нове місто. |
| public | City updated |  | UpdateCity | Змінює інформацію щодо існуючого міста. |
| public | Country newCountry |  | AddCountry | Додає нову країну. |
| public | Country updated |  | UpdateCountry | Змінює інформацію щодо існуючої країни. |
| public | Region newRegion |  | AddRegion | Додає новий регіон. |
| public | Region updated |  | UpdateRegion | Змінює інформацію щодо існуючого регіону. |
| public | List<City> updatedCities, string path |  | SaveCitiesToFile | Зберігає обраний список міст у встановленому місці. |
| public | List<Country> updatedCountries, string path |  | SaveCountriesToFile | Зберігає обраний список країн у встановленому місці. |
| public | List<Region> updatedRegions , string path |  | SaveRegionsToFile | Зберігає обраний список регіонів у встановленому місці. |
| public | List<Continent> updatedContinents , string path |  | SaveContinentsToFile | Зберігає обраний список континентів у встановленому місці. |
| private |  |  | GetCitiesFromFile | Зчитує інформацію з файлу до бази даних. |
| private |  |  | GetRegionsFromFile | Зчитує інформацію з файлу до бази даних. |
| private |  |  | GetCountriesFromFile | Зчитує інформацію з файлу до бази даних. |
| private |  |  | GetContinentsFromFile | Зчитує інформацію з файлу до бази даних. |
| Program | private |  |  | Main | Головний метод класу. |
| MainPage | public |  |  | MainPage | Конструктор класу. |
| private | object sender, EventArgs e |  | CityButton\_Click | Метод, який відкриває сторінку з містами. |
| private | object sender, EventArgs e |  | RegionButton\_Click | Метод, який відкриває сторінку з містами. |
| private | object sender, EventArgs e |  | CountryButton\_Click | Метод, який відкриває сторінку з містами. |
| private | object sender, EventArgs e |  | ContinentButton\_Click | Метод, який відкриває сторінку з містами. |
| private | object sender, EventArgs e |  | MainPage\_KeyDown | Метод, який відповідає за натискання клавіш. |
| private | object sender, EventArgs e |  | ExitButton\_Click | Метод, який завершує програму. |
| CityPage | public | Database databaseRe |  | CityPage | Конструктор класу. |
| public | bool changePosition |  | UpdateFilteredCities | Метод, який оновлює список міст на сторінці. |
| private |  |  | UpdatePageLabel | Метод, який оновлює номер сторінки таблиці з містами. |
| private |  |  | OnBack | Метод, який закриває цю сторінку та відкриває попередню. |
| private | object sender, KeyEventArgs e |  | CityPage\_KeyDown | Метод, який відповідає за натискання клавіш. |
| private | object sender, EventArgs e |  | SearchButton\_Click | Метод, який викликається після натискання кнопки пошуку. |
| private | object sender, EventArgs e |  | LeftButton\_Click | Метод, який відкриває попередню сторінку таблиці міст. |
| private | object sender, EventArgs e |  | RightButton\_Click | Метод, який відкриває наступну сторінку таблиці міст. |
| private | object sender, EventArgs e |  | AddCityButton\_Click | Метод, який додає нове місто |
| private | object sender, EventArgs e |  | SortParametr\_SelectedValueChanged | Метод, який викликається після зміни параметру сортування. |
| private | City city |  | AddOrChangeCity | Метод який додає або змінює місто в залежності від вхідних параметрів. |
| private | object sender , DataGridViewCellEventArgs e |  | CitiesGrid\_CellClick | Метод, який викликається при нажаті на клітинку таблиці. |
| private | object sender, EventArgs e |  | SaveResultbutton\_Click | Метод, який зберігає знайдені міста у файл. |
| private | object sender, EventArgs e |  | BackButton\_Click | Метод, який виконується при нажаті кнопки назад. |
| RegionPage | public | Database databaseRe |  | RegionPage | Конструктор класу. |
| public | bool isChangePosition |  | UpdateFiltredRegions | Метод, який оновлює список регіонів на сторінці. |
| private |  |  | UpdatePageLabel | Метод, який оновлює номер сторінки таблиці з регіонами. |
| private | object sender, KeyEventArgs e |  | OnBackButtonClick | Метод, який закриває цю сторінку та відкриває попередню при натисканні клавіши Escape. |
| private |  |  | OnBack | Метод, який закриває цю сторінку та відкриває попередню. |
| private | object sender, KeyEventArgs e |  | RegionPage\_KeyDown | Метод, який відповідає за натискання клавіш. |
| private | object sender, EventArgs e |  | LeftButton\_Click | Метод, який відкриває попередню сторінку таблиці міст. |
| private | object sender, EventArgs e |  | RightButton\_Click | Метод, який відкриває наступну сторінку таблиці міст. |
| private | object sender, EventArgs e |  | SortTypeSelector\_SelectedValueChanged | Метод, який викликається після зміни параметру сортування. |
| private | object sender, EventArgs e |  | SearchButton\_Click | Метод, який викликається після натискання кнопки пошуку. |
| private | object sender, EventArgs e |  | AddButton\_Click | Метод, який додає новий регіон. |
| private | object sender, EventArgs e |  | SaveResultButton\_Click | Метод, який зберігає знайдені регіони у файл. |
| private | object sender, DataGridViewCellEventArgs e |  | CountryGrid\_CellClick | Метод, який викликається при нажаті на клітинку таблиці. |
| private | Region region |  | AddOrChangeRegion | Метод який додає або змінює регіон в залежності від вхідних параметрів. |
| private | object sender, EventArgs e |  | BackButton\_Click | Метод, який виконується при нажаті кнопки назад. |
| CountryPage | public | Database databaseRe |  | CountryPage | Конструктор класу. |
| public | bool isChangePosition |  | UpdateFiltredCountries | Метод, який оновлює список країн на сторінці. |
| private |  |  | UpdatePageLabel | Метод, який оновлює номер сторінки таблиці з країнами. |
| private |  |  | OnBack | Метод, який закриває цю сторінку та відкриває попередню. |
| private | object sender, KeyEventArgs e |  | OnBackButtonClick | Метод, який закриває цю сторінку та відкриває попередню при натисканні клавіши Escape. |
| private | object sender, KeyEventArgs e |  | CountryPage\_KeyDown | Метод, який відповідає за натискання клавіш. |
| private | object sender, EventArgs e |  | LeftButton\_Click | Метод, який відкриває попередню сторінку таблиці. |
| private | object sender, EventArgs e |  | RightButton\_Click | Метод, який відкриває наступну сторінку таблиці. |
| private | object sender, EventArgs e |  | SortTypeSelector\_SelectedValueChanged | Метод, який викликається після зміни параметру сортування. |
| private | object sender, EventArgs e |  | SearchButton\_Click | Метод, який викликається після натискання кнопки пошуку. |
| private | object sender , DataGridViewCellEventArgs e |  | CountryGrid\_CellClick | Метод, який викликається при нажаті на клітинку таблиці. |
| private | Country country |  | AddOrChangeCountry | Метод який додає або змінює країну в залежності від вхідних параметрів. |
| private | object sender, EventArgs e |  | AddButton\_Click | Метод, який додає нову країну. |
| private | object sender, EventArgs e |  | SaveResultButton\_Click | Метод, який зберігає знайдені країни у файл. |
| private | object sender, EventArgs e |  | BackButton\_Click | Метод, який виконується при нажаті кнопки назад. |
| ContinentPage | public | Database databaseRe |  | ContinentPage | Конструктор класу. |
| private |  |  | OnBack | Метод, який закриває цю сторінку та відкриває попередню. |
| private | object sender, KeyEventArgs e |  | OnBackButtonClick | Метод, який закриває цю сторінку та відкриває попередню при натисканні клавіши Escape. |
| private | object sender, KeyEventArgs e |  | ContinentPage\_KeyDown | Метод, який відповідає за натискання клавіш. |
| private | object sender, EventArgs e |  | BackButton\_Click | Метод, який виконується при нажаті кнопки назад. |
| MapPage | public | CityPage parent, classes.City city |  | MapPage | Конструктор класу. |
| private | KeyEventArgs e |  | OnBackButtonClick | Метод, який закриває цю сторінку та відкриває попередню при натисканні клавіши Escape. |
| private | object sender, KeyEventArgs e |  | MapPage\_KeyDown | Метод, який відповідає за натискання клавіш. |
| AddOrChangeCityPage | public | CityPage CityPage, Database databaseRe, City cityRe |  | AddOrChangeCityPage | Конструктор класу. |
| private |  |  | CloseThisPage | Метод, який закриває цю сторінку та відкриває попередню. |
| private | object sender, EventArgs e |  | BackButton\_Click | Метод, який виконується при нажаті кнопки назад. |
| private | object sender, EventArgs e |  | ChangeButton\_Click | Метод, який змінює дані списку міст. |
| AddOrChangeRegionPage | public | RegionPage regionPage, Database databaseRe, Region regionRe |  | AddOrChangeRegionPage | Конструктор класу. |
| private |  |  | CloseThisPage | Метод, який закриває цю сторінку та відкриває попередню. |
| private | object sender, EventArgs e |  | BackButton\_Click | Метод, який виконується при нажаті кнопки назад. |
| private | object sender, EventArgs e |  | ChangeButton\_Click | Метод, який змінює дані списку регіонів. |
| AddOrChangeCountryPage | public | CountryPage countryPage, Database databaseRe, Country countryRe |  | AddOrChangeCountryPage | Конструктор класу. |
| private |  |  | CloseThisPage | Метод, який закриває цю сторінку та відкриває попередню. |
| private | object sender, EventArgs e |  | BackButton\_Click | Метод, який виконується при нажаті кнопки назад. |
| private | object sender, EventArgs e |  | ChangeButton\_Click | Метод, який змінює дані списку регіонів. |

2.4 Аномалії

Аномальними можна вважати натискання клавіш та введення неправильних даних на сторінках, а саме:

При натисканні на будь-якій сторінці клавіши на клавіатурі, яка не є загальноприйнятою, нічого не буде коїтись.

При спробі введення чисел, більших за модулем 180 в полі вводу широти чи довготи, буде з’являтися помилка, яка попереджає про некоректне введення даних;

При спробі ввести текст в полі з численністю населення, площею, широтою, довготою, буде з’являтися помилка, яка попереджає про некоректне введення даних;

При спробі в будь-якому випадаючому списку вибрати елемент не з нього, буде з’являтися помилка, яка попереджає про некоректне введення даних;

**3 ІНСТРУКЦІЯ КОРИСТУВАЧА**

Після запуску програми користувач потрапляє на головну сторінку, на якій розташовані 5 кнопок (рисунок 3.1).

При натисканні кнопки Exit користувач завершує користування програмою.

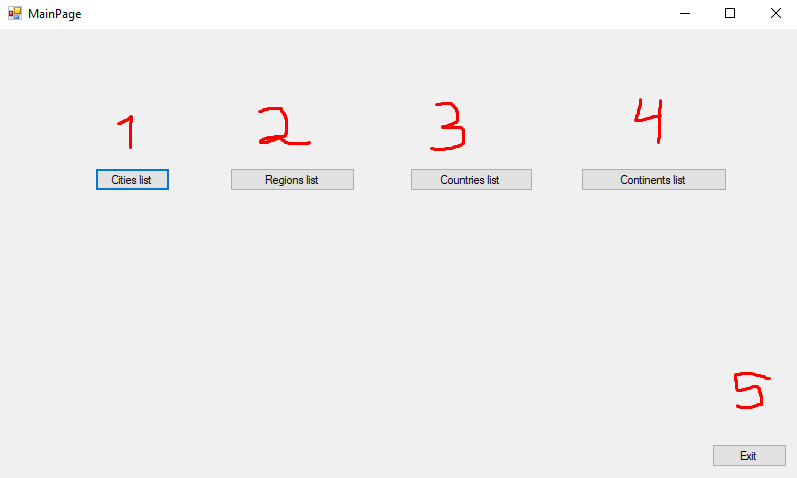


Рисунок 3.1 – Головна сторінка

При натисканні на першу кнопку, користувач потрапляє на сторінку з містами (рисунок 3.2).

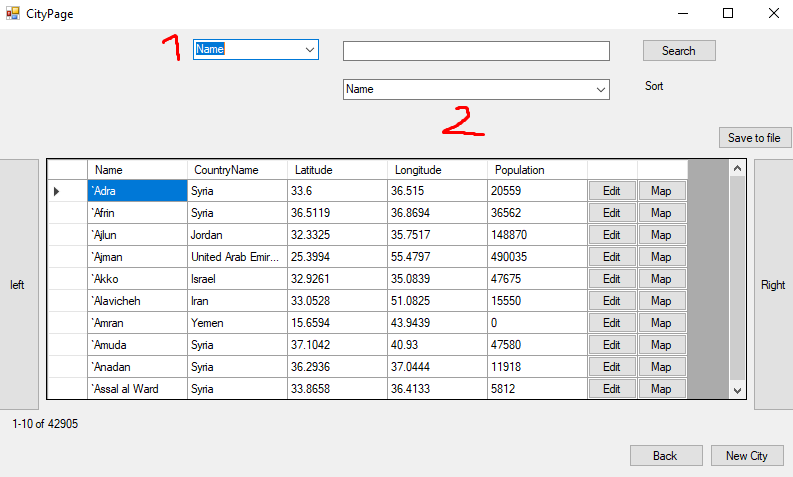


Рисунок 3.2 – Сторінка з містами

При натисканні на перший випадаючий список, можна вибрати параметр пошуку. Якщо ввести текст в полі справа від першого випадаючого списку та натиснути кнопку Search, міста зміняться на ті, які користувач шукає.

При зміні даних у другому випадаючому списку зміниться також і параметр сортування міст, які зображені на сторінці.

При натисканні Save to file, знайдені міста будуть збережені в файл.

При натисканні будь-якої кнопки Map користувач побаче розташування обраного міста на мапі (рисунок 3.3).



Рисунок 3.3 – Мапа

При натисканні кнопки left користувач побаче попередню сторінку з містами.

При натисканні кнопки right користувач побаче наступну сторінку з містами.

При натисканні кнопки New city або Edit користувач відкриє сторінку, на якій зможе додати або відредагувати обране місто (рисунок 3.4).

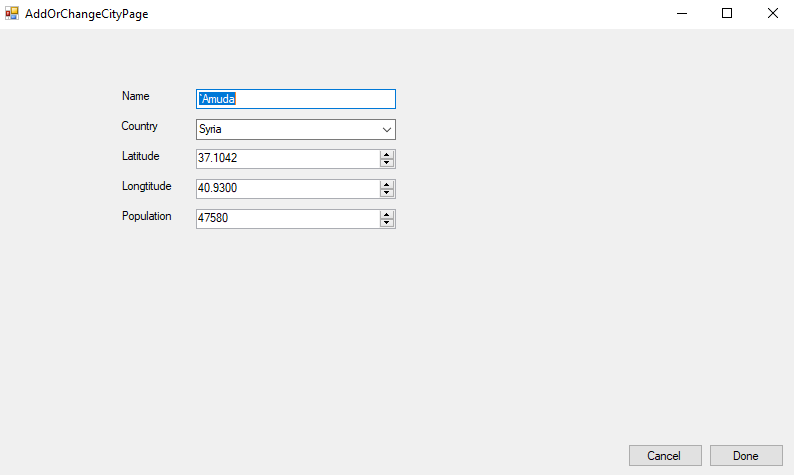


Рисунок 3.4 – Сторінка зміни або додавання нового міста

Після вибору даних в полях та випадаючому списку користувач може нажати Done та зберігти зміни та повернутися на попередню сторінку, або cancel та відмінити зміни і повернутися на попередню сторінку.

При натисканні кнопки back на сторінці міст (рисунок 3.2) користувач повернеться на попередню сторінку (рисунок 3.1).

При натисканні кнопок Region list або Country list користувач потрапляє на сторінки з аналогічним щодо сторінки з містами (рисунок 3.2) інтерфейсом, виключенням є відсутність кнопок Map.

При натискані клавіши Escape на будь-якій сторінці, відкрита сторінка закриється та буде або відкрита попередня сторінка, або завершена програма, якщо відкрита сторінка була головною сторінкою.

**ВИСНОВКИ**

Під час роботи над програмою «Довідник Географа» ми на практиці освоїли принципи об’єктно-орієнтованого програмування, більш детально вивчили мову програмування C#, середу розробки програм Windows Forms[2].

Було виконано та усі поставлені завдання та модифіковані деякі з них, такі як:

Створено програму на мові програмування C#[5] у середовищі розробки Windows Forms;

Створено «Довідник Географа», який містить міста (географічні координати, чисельність населення), регіони (вид, приналежність країні, столиця країни, чисельність населення), країни (площа, чисельність населення, форма державного правління, столиця), материки. Пошук за певними критеріями, показ на карті розташування, населеність материків та інше[1];

Програму виконано за допомогою принципів об’єктно-орієнтованого програмування успадкування, поліморфізм, інкапсуляція, абстракція;

Програма «Довідник Географа» відповідає усім загальним вимогам щодо написання програми таким, як стійкість програми, функціональна повнота, терміни та інтерфейс, використання клавіатури;

Виконуючи програму ми вивчили основні принципи об’єктно-орієнтованого програмування, та оволоділи навичками розробки об’єктних програм. Ми зосередились на найбільш важливих рисах програмування, які не залежать від таких обставин, як тип процесора або операційна система. Виконання курсової роботи дуже слушно надало нам таку можливість.

Працюючи над курсовою роботою, ми вивчили окремі фази розробки програмного забезпечення і навчилися поєднувати їх в одне ціле – в свій проект.

**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. Методичні вказівки до виконання курсової роботи з дисципліни «Об’єктно-орієнтоване програмування» для студентів першого (бакалаврського) рівня вищої освіти усіх форм навчання спеціальності 121 «Інженерія програмного забезпечення», освітньо-професійна програма «Програмна інженерія» / Упоряд.: В.М. Бондарєв, Ю.Ю. Черепанова – Харків: ХНУРЕ, 2022. – 43 с.
2. Полное руководство по языку программирования С# 10 и платформе .NET 6. URL: https://metanit.com/sharp/tutorial (дата звернення 01.06.2022).
3. Прайс М. C# 9 и .NET 5. Разработка и оптимизация. (+ Додаток) - Санкт-Петербург: Пітер, 2022 - 832 с.
4. Бондарев, В.М. Объектно-ориентированное программирование на С# – Х. : Компанія СМИТ, 2009. – 224 с.
5. Документация по .NET. URL: https://docs.microsoft.com/ru-ru/dotnet/ (дата звернення 01.06.2022)

ДОДАТОК А

Код програми

У додатку наведено лише частина коду, бо увесь код займає більш ніж 10 сторінок:

Файл GeographyUnit.cs

using System;

namespace GeografyNotebook.models.classes

{

public abstract class GeographyUnit

{

public Guid Uuid { protected set; get; }

public String Name { protected set; get; }

public int Population { protected set; get; }

}

}

Файл City.cs

using System;

namespace GeografyNotebook.models.classes

{

public class City : GeographyUnit

{

public City(Guid uuid, string name, string countryName,

double latitude, double longitude, int population)

{

Uuid = uuid;

Name = name;

CountryName = countryName;

Latitude = latitude;

Longitude = longitude;

Population = population;

}

public City(string line)

{

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

Uuid = Guid.Parse(words[0]);

Name = words[1];

CountryName = words[2];

Latitude = Double.Parse(words[3]);

Longitude = Double.Parse(words[4]);

Population = Int32.Parse(words[5]);

}

public string CountryName { protected set; get; }

public double Latitude { protected set; get; }

public double Longitude { protected set; get; }

public override string ToString()

{

return $"{Uuid};{Name};{CountryName};" +

$"{Latitude};{Longitude};{Population}";

}

}

}

Файл Region.cs

using System;

namespace GeografyNotebook.models.classes

{

public class Region : GeographyUnit

{

public Region(Guid uuid, string name, string type,

Country country, int population)

{

Uuid = uuid;

Name = name;

Type = type;

Country = country;

Capital = country.Capital;

Population = population;

}

public string Type { set; get; }

public Country Country { set; get; }

public City Capital { set; get; }

public override string ToString()

{

return $"{Uuid};{Name};{Type};{Country.Uuid};{Population}";

}

}

}

Файл Country.cs

using System;

namespace GeografyNotebook.models.classes

{

public class Country : GeographyUnit

{

public Country(Guid uuid, string name, int area, int population,

string governmentType, City capital)

{

Uuid = uuid;

Name = name;

Area = area;

Population = population;

GovernmentType = governmentType;

Capital = capital;

}

public int Area { set; get; }

public string GovernmentType { set; get; }

public City Capital { set; get; }

public override string ToString()

{

return $"{Uuid};{Name};{Area};{Population};" +

$"{GovernmentType};{Capital.Uuid}";

}

}

}

Файл Continent.cs

using System;

using System.Collections.Generic;

namespace GeografyNotebook.models.classes

{

public class Continent : GeographyUnit

{

public Continent(Guid uuid, string name, List<Country> countries)

{

Uuid = uuid;

Name = name;

Countries = countries;

Population = 0;

foreach(Country country in countries)

{

Population += country.Population;

}

}

public List<Country> Countries { set; get; }

public override string ToString()

{

string returnS = $"{Uuid};{Name};{Population};{Countries.Count}";

foreach(Country country in Countries)

{

returnS += ";" + country.Uuid.ToString();

}

return returnS;

}

}

}

Файл Database.cs

using System;

using System.IO;

using System.Linq;

using System.Collections.Generic;

namespace GeografyNotebook.models.classes

{

public class Database

{

private static readonly string citiesPath = @"..\..\assets\cities.txt";

private static readonly string countriesPath = @"..\..\assets\countries.txt";

private static readonly string regionsPath = @"..\..\assets\regions.txt";

private static readonly string continentsPath

= @"..\..\assets\continents.txt";

public Database()

{

GetCitiesFromFile();

GetCountriesFromFile();

GetRegionsFromFile();

GetContinentsFromFile();

}

public List<City> Cities { private set; get; }

public List<Region> Regions { private set; get; }

public List<Country> Countries { private set; get; }

public List<Continent> Continents { private set; get; }

public List<City> GetCities(

string? searchField = null,

string? searchValue = null,

string orderByField = "Name")

{

List<City> result = Cities.FindAll(city =>

searchField != null && searchValue != null

? city

.GetType()

.GetProperty(searchField)

.GetValue(city)

.ToString()

.Contains(searchValue)

: true);

return orderByField switch

{

"CountryName" => result.OrderBy(city => city.CountryName)

.ToList(),

"Population" => result.OrderBy(city => -city.Population)

.ToList(),

"Latitude" => result.OrderBy(city => city.Latitude)

.ToList(),

"Longitude" => result.OrderBy(city => city.Longitude)

.ToList(),

\_ => result.OrderBy(city => city.Name).ToList(),

};

}

public List<Region> GetRegions(

string? searchField = null,

string? searchValue = null,

string orderByField = "Name")

{

List<Region> result = searchField switch

{

"Country" => Regions.FindAll(region => region

.Country.Name.Contains(searchValue)),

"Population" => Regions.FindAll(region => region

.Population == Convert

.ToInt32(searchValue)),

"Type" => Regions.FindAll(region => region

.Type.Contains(searchValue)),

"Name" => Regions.FindAll(region => region

.Name.Contains(searchValue)),

\_ => Regions,

};

return orderByField switch

{

"CountryName" => result.OrderBy(region => region

.Country.Name).ToList(),

"Population" => result.OrderBy(region => -region

.Population).ToList(),

"Type" => result.OrderBy(region => region.Type).ToList(),

\_ => result.OrderBy(region => region.Name).ToList(),

};

}

public List<Country> GetCountries(

string? searchField = null,

string? searchValue = null,

string orderByField = "Name")

{

List<Country> result = searchField switch

{

"Area" => Countries.FindAll(region => region

.Area == Convert.ToInt32(searchValue)),

"Population" => Countries.FindAll(region => region

.Population == Convert.ToInt32(searchValue)),

"Government type" => Countries.FindAll(region => region

.GovernmentType.Contains(searchValue)),

"Name" => Countries.FindAll(region => region

.Name.Contains(searchValue)),

\_ => Countries,

};

return orderByField switch

{

"Area" => result.OrderBy(region => -region.Area).ToList(),

"Population" => result.OrderBy(region => -region

.Population).ToList(),

"Government type" => result.OrderBy(region => region

.GovernmentType).ToList(),

\_ => result.OrderBy(region => region.Name).ToList(),

};

}

public void AddCity(City newCity)

{

Cities.Add(newCity);

using StreamWriter writer

= new StreamWriter(citiesPath, append: true);

writer.WriteLine(newCity.ToString());

}

public void UpdateCity(City updated) {

int index = Cities.FindIndex(city => city.Uuid == updated.Uuid);

Cities[index] = updated;

SaveCitiesToFile(Cities, citiesPath);

}

public void AddCountry(Country newCountry)

{

Countries.Add(newCountry);

using StreamWriter writer

= new StreamWriter(countriesPath, append: true);

writer.WriteLine(newCountry.ToString());

}

public void UpdateCountry(Country updated)

{

int index = Countries.FindIndex(country => country.Uuid == updated.Uuid);

Countries[index] = updated;

SaveCountriesToFile(Countries, countriesPath);

}

public void AddRegion(Region newRegion)

{

Regions.Add(newRegion);

using StreamWriter writer

= new StreamWriter(regionsPath, append: true);

writer.WriteLine(newRegion.ToString());

}

public void UpdateRegion(Region updated)

{

int index = Regions.FindIndex(region => region.Uuid == updated.Uuid);

Regions[index] = updated;

SaveRegionsToFile(Regions, regionsPath);

}

public void SaveCitiesToFile(List<City> updatedCities, string path)

{

using StreamWriter writer = new StreamWriter(path);

foreach (City city in updatedCities)

{

writer.WriteLine(city.ToString());

}

}

public void SaveCountriesToFile(List<Country> updatedCountries,

string path)

{

using StreamWriter writer = new StreamWriter(path);

foreach (Country country in updatedCountries)

{

writer.WriteLine(country.ToString());

}

}

public void SaveRegionsToFile(List<Region> updatedRegions

, string path)

{

using StreamWriter writer = new StreamWriter(path);

foreach (Region region in updatedRegions)

{

writer.WriteLine(region.ToString());

}

}

public void SaveContinentsToFile(List<Continent> updatedContinents

, string path)

{

using StreamWriter writer = new StreamWriter(path);

foreach (Continent continent in updatedContinents)

{

writer.WriteLine(continent.ToString());

}

}

private void GetCitiesFromFile()

{

Cities = new List<City>();

using StreamReader reader = new StreamReader(citiesPath);

string line;

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

{

Cities.Add(new City(line));

}

}

private void GetRegionsFromFile()

{

Regions = new List<Region>();

using StreamReader reader = new StreamReader(regionsPath);

string line;

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

{

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

Region region = new Region(

uuid: Guid.Parse(words[0]),

name: words[1],

type: words[2],

country: Countries.Find(country => country

.Uuid.ToString() == words[3]),

population: Int32.Parse(words[4])

);

Regions.Add(region);

}

}

private void GetCountriesFromFile()

{

Countries = new List<Country>();

using StreamReader reader = new StreamReader(countriesPath);

string line;

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

{

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

Country country = new Country(

uuid: new Guid(words[0]),

name: words[1],

area: Int32.Parse(words[2]),

population: Int32.Parse(words[3]),

governmentType: words[4],

capital: Cities.Find(item => item

.Uuid == Guid.Parse(words[5]))

);

Countries.Add(country);

}

}

private void GetContinentsFromFile()

{

Continents = new List<Continent>();

using StreamReader reader = new StreamReader(continentsPath);

string line;

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

{

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

string[] countryIds = words

.Skip(4)

.Take(words.Length - 5)

.ToArray();

Continent continent = new Continent(

uuid: Guid.Parse(words[0]),

name: words[1],

countries: Countries.FindAll(country => Array

.Exists(countryIds, item => item == country

.Uuid.ToString()))

);

Continents.Add(continent);

}

}

}

}