Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни “Об’єктно-орієнтоване програмування”

ГЕНЕАЛОГІЧНЕ ДЕРЕВО

Керівник, проф. Бондарєв В.М.

Студентка гр. ПЗПІ-22-6 Лисенко А.М.

Комісія:

проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарєв В.М.

ст.викл. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Черепанова Ю.Ю.

ст.викл. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ляпота В.М.

Харків 2023

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-22-6* . Семестр *2* .

***ЗАВДАННЯ***

***на курсовий проект студента***

Лисенко Анни Миколаївни

(Прізвище, Ім'я, По батькові)

1 Тема проекту:

Генеалогічне дерево

2 Термін здачі студентом закінченого проекту: ***“16” - червня -2023 р***.

3 Вихідні дані до проекту:

*Специфікація програми, методичні вказівки до виконання курсової роботи.*

4 Зміст розрахунково-пояснювальної записки:

*Вступ,* *опис вимог, проектування програми, інструкція користувача, висновки.*

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 13.02.2023 – 14.03.2023 р. |
| 2 | Формулювання вимог до програми | 20.03.2023 – 25.03.2023 р. |
| 3 | Розробка підсистеми зберігання та пошуку даних | 27.03.2023 – 01.04.2023 р. |
| 4 | Розробка основних функції | 01.04.2023 – 29.04.2023 р. |
| 5 | Розробка додаткових функції | 22.04.2023 – 13.05.2023 р. |
| 6 | Розробка функцій зберігання та завантаження даних | 20.04.2023 – 14.05.2023 р. |
| 7 | Тестування і доопрацювання розробленої програмної системи. | 03.05.2023 – 14.05.2023 р. |
| 8 | Оформлення пояснювальної записки, додатків, графічного матеріалу | 10.05.2023 – 09.06.2023 р. |
| 9 | Захист | 05.06.2023 – 16.06.2023 р. |

Студентка гр. ПЗПІ-22-6 Лисенко А.М.

Керівник проф. Бондарєв В.М.

« 13 » лютого 2023 р.

**РЕФЕРАТ**

Пояснювальна записка до курсової роботи: 45 с., 20 рис., 1 додаток,4 джерела.

ГЕНЕАЛОГІЧНЕ ДЕРЕВО, ПРЕДОК, НАЩАДОК, ВУЗЛИ, ПЕРСОНАЛЬНІ ДАНІ, РОДИННІ ЗВ’ЯЗКИ, ЗВІТ, ООП, .NET, МОВА C#

Метою роботи є розробка програми «Генеалогічне дерево», яка надасть можливість користувачу створити генеалогічне дерево свого роду та взаємодіяти з ним.

В результаті отримана програма, що дозволяє створювати генеалогічне дерево, пов’язуючи предка з нащадками, зберігати дані про кожного члена родини, такі як: ім’я, прізвище, ім’я по батькові, дату народження. Після побудови дерева користувач може знайти нащадків та предків для обраного вузла, редагувати дані та завантажити створене дерево у вигляді зображення.

В процесі розробки використано середовища Microsoft Visual Studio 2022, фреймворк Windows Forms, платформи .NET 7.0, мова програмування C#.

**ЗМІСТ**

[ВСТУП 6](#_Toc138375837)

[1 ОПИС ВИМОГ 7](#_Toc138375838)

[1.1 Сценарії використання програми 7](#_Toc138375839)

[1.2 Функції програми 12](#_Toc138375840)

[2 ПРОЄКТУВАННЯ ПРОГРАМИ 20](#_Toc138375841)

[2.1 Архітектура програми 20](#_Toc138375842)

[2.2 Діаграма класів 20](#_Toc138375843)

[2.3 Формат зберігання зовнішніх даних 22](#_Toc138375844)

[3 ІНСТРУКЦІЯ КОРИСТУВАЧА 23](#_Toc138375845)

[3.1 Встановлення програми 23](#_Toc138375846)

[3.2 Користування програмою 23](#_Toc138375847)

[ВИСНОВКИ 30](#_Toc138375848)

[ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 31](#_Toc138375849)

[ДОДАТОК А 32](#_Toc138375850)

# **ВСТУП**

У сучасному світі кожна людина повинна знати своє походження, історію своєї родини та предків. Багато людей зацікавлені вивченням свого родоводу, і генеалогічні дерева є популярними інструментами для представлення цієї інформації. Вони дозволяють зберігати спогади, історії і зв'язки між поколіннями для майбутніх нащадків. Попередньо зібравши всю важливу інформацію про предків від близьких та далеких родичів можна реалізувати цінний інструмент для вивчення сімейної історії.

Програма «Генеалогічне дерево» допоможе користувачеві створити просте, але зрозуміле дерево сімейних зв’язків: від найдавнішого відомого предка, до його сучасних нащадків. Програма має інтуїтивно-зрозумілий інтерфейс та функціонал. Це допоможе уникнути нерозуміння при взаємодії з програмою та головним процесом - побудовою дерева. Крім того, простота програми каже про те, що її може використовувати будь-яка людина, від школяра, який працює над шкільним проєктом, до науковця, який вивчає рід видатної історичної особистості.

Функціонал програми передбачає додавання вузлів-нащадків до обраного предка, за необхідністю, редагування цих вузлів. Пошук зв’язків для обраного вузла генеалогічного дерева визначає ланцюг його нащадків та предків. Після побудови дерева користувач може отримати зображення створених зв’язків для подальшого використання у своїх проєктах або просто для візуального збереження генеалогічного дерева в електронному форматі. Також є можливість зберегти файл у форматі json на свій комп’ютер та у подальшому додавати сімейні зв’язки або просто переглядати створене дерево безпосередньо у програмі «Генеалогічне дерево».

# **1 ОПИС ВИМОГ**

## **1.1 Сценарії використання програми**

**Сценарій 1. Побудова дерева**

*Передумова*

Користувач успішно встановив програму «Генеалогічне дерево» та відкрив її для подальшого використання.

*Основний сценарій*

1. Користувач обирає кнопку «Додати предка».
2. Програма виводить форму, в яку користувач повинен внести дані, а саме: ім’я, прізвище, ім’я по батькові, дата народження обраного предка, та натиснути кнопку «Зберегти».
3. На головній вкладці користувач побачить перший створений вузел дерева – головного предка.
4. Натиснувши на створений вузол, користувач помічає його як «виділений» та натикає на кнопку «Додати нащадка».
5. Програма виводить форму, в яку користувач повинен внести дані (ім’я, прізвище, ім’я по батькові, дата народження) та натиснути кнопку «Зберегти».
6. Пункти 4 та 5 повторювати до отримання бажаного вигляду генеалогічного дерева.

**Сценарій 2. Редагування вузлів дерева**

*Передумова*

Користувач успішно встановив програму «Генеалогічне дерево», відкрив її та побудував дерево.

*Основний сценарій*

1. Користувач обирає вузол створеного дерева, який містить помилку, вузол помічається як «обраний».
2. Далі обирається кнопка «Редагувати вузол».
3. Програма виводить форму, в якій вказані (ім’я, прізвище, ім’я по батькові, дата народження) для подальшого редагування.
4. Після внесення змін користувач натискає кнопку «Зберегти».
5. Оновлені дані вузла користувач бачить безпосередньо у дереві на головній формі.

**Сценарій 3. Видалення вузлів дерева**

*Передумова*

Користувач успішно встановив програму «Генеалогічне дерево», відкрив її та побудував дерево.

*Основний сценарій*

1. Користувач обирає вузол створеного дерева для видалення, він помічається як «обраний».
2. Користувач натискає кнопку «Видалити вузол».
3. З’являється вікно підтвердження видалення.
4. Користувач натискає «Так», підтверджуючи свій вибір.
5. Вузол видаляється з дерева.

**Сценарій 4. Пошук нащадків та предків**

*Передумова*

Користувач успішно встановив програму «Генеалогічне дерево», відкрив її та побудував дерево.

*Основний сценарій*

1. Користувач обирає вузол створеного дерева, він помічається як «обраний».
2. Далі користувач натискає на кнопку «Пошук нащадків та предків».
3. Програма виводить форму, у якій містяться дані обраного вузла та у спеціальних текстових полях - дані щодо його предків та нащадків (якщо вони відсутні, буде показано відповідний надпис).
4. Після ознайомлення з представленою інформацією користувач натискає кнопку «Закрити» та повертається на головну форму.

**Сценарій 5. Створення зображення побудованого дерева**

*Передумова*

Користувач успішно встановив програму «Генеалогічне дерево», відкрив її та побудував дерево.

*Основний сценарій*

1. У меню користувач обирає розділ «Дерево».
2. Зі списку, що з’явився, користувач обирає «Створити зображення».
3. З’являється діалогове вікно збереження файлу, де користувач вказує назву файлу, обирає місце збереження зображення та натискає «Зберегти/Save».
4. На екрані з’являється повідомлення про успішне збереження зображення дерева.
5. Після натискання «ОК» повідомлення закривається та користувач знову бачить перед собою лише головну форму.

**Сценарій 6. Повне очищення дерева**

*Передумова*

Користувач успішно встановив програму «Генеалогічне дерево», відкрив її та побудував дерево.

*Основний сценарій*

1. У меню користувач обирає розділ «Дерево».
2. Зі списку, що з’явився, користувач обирає «Очистити дерево».
3. Після натискання на кнопку з’являється вікно підтвердження очищення дерева.
4. У вікні користувач обирає «Так».
5. Створене генеалогічне дерево повністю видаляється.

**Сценарій 7. Збереження дерева**

*Передумова*

Користувач успішно встановив програму «Генеалогічне дерево», відкрив її та побудував дерево.

*Основний сценарій*

1. У меню користувач обирає розділ «Файл».
2. Зі списку, який з’явився, користувач обирає «Зберегти».
3. З’являється діалогове вікно збереження файлу, де користувач вказує назву файлу та обирає місце збереження.
4. При натисканні на «Зберегти/Save» це вікно закривається та з’являється повідомлення про збереження файлу.

*Альтернативний сценарій*

1. У меню користувач обирає розділ «Файл».
2. Зі списку, який з’явився, користувач обирає «Зберегти».
3. З’являється діалогове вікно збереження файлу, де користувач вказує назву файлу та обирає місце збереження.
4. При натисканні на «Скасувати/Cancel» (файл не зберігається), діалогове вікно закривається, користувач може знову повернутися до створення файлу.

**Сценарій 8. Створення дерева відносно обраного вузла**

*Передумова*

Користувач успішно встановив програму «Генеалогічне дерево», відкрив її та побудував дерево.

*Основний сценарій*

1. Користувач обирає вузол створеного дерева для створення нового генеалогічного дерева відносно обраної людини.
2. У меню користувач обирає розділ «Дерево».
3. Користувач натискає кнопку «Відносно обраного вузла».
4. Відкривається вікно з побудованим деревом відносно обраного вузла (тобто вузол стає предком, і дерево містить усіх його нащадків).
5. Користувач закриває вікно, натиснувши кнопку «Закрити».

*Альтернативний сценарій*

1. Користувач обирає вузол створеного дерева для створення нового генеалогічного дерева відносно обраної людини.
2. У меню користувач обирає розділ «Дерево».
3. Користувач натискає кнопку «Відносно обраного вузла».
4. Відкривається вікно з побудованим деревом відносно обраного вузла (тобто вузол стає предком, і дерево містить усіх його нащадків).
5. Користувач натискає кнопку «Зберегти».
6. З’являється діалогове вікно збереження файлу, де користувач вказує назву файлу та обирає місце збереження.
7. При натисканні на «Зберегти/Save» це вікно закривається та з’являється повідомлення про збереження файлу.

**Сценарій 9. Завантаження дерева**

*Передумова*

Користувач успішно встановив програму «Генеалогічне дерево» та відкрив її для подальшого використання.

*Основний сценарій*

1. У меню користувач обирає розділ «Дерево».
2. Зі списку, який з’явився, користувач обирає «Завантажити дерево».
3. З’являється діалогове вікно, у якому користувач обирає файл, який бажає відкрити у програмі.
4. При натисканні «Відкрити» дерево з’являється на головній формі.

*Альтернативі сценарії*

Далі можна застосовувати:

1. Сценарій 3. Видалення вузлів дерева.
2. Сценарій 4. Пошук нащадків та предків.
3. Сценарій 5. Створення зображення побудованого дерева.
4. Сценарій 6. Повне очищення дерева.
5. Сценарій 7. Збереження дерева.
6. Сценарій 8. Створення дерева відносно обраного вузла.

## **1.2 Функції програми**

**Функція 1. Додавання предка**

Побудова генеалогічного дерева починається з додавання предка. Кнопка, яка відповідає цій події знаходиться у правій верхній частині головного вікна програми (див. рис. 1.1) під назвою «Додати предка».

Далі у формі (див. рис. 1.2) користувач заповнює поля: ім’я, прізвище, ім’я по батькові, дата народження (формат запису дд.мм.рррр або можна вказати «дата невідома»). Якщо форма запису дати не буде відповідати зазначеному формату, з’явиться повідомлення про це. Жодне поле не є обов’язковим для заповнення.

Збережена інформація відображається у полі, що виділено пунктиром, та представляє собою головний вузол генеалогічного дерева.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Рисунок 1.1 – Головне вікно програми. Функція додавання предка

|  |
| --- |
|  |
|  |  |

Рисунок 1.2 – Форма для заповнення персональних даних

**Функція 2. Додавання нащадка**

Після того, як користувач додав предка, вузол (див. рис. 1.3) потрібно виділити натиснувши на нього (одразу після додавання він автоматично стає «обраним»).

**Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание**Далі потрібно обрати кнопку «Додати нащадка», яка знаходиться у верхній правій частині головного вікна (див. рис. 1.3).

Рисунок 1.3 – Головне вікно програми. Функція додавання нащадка

У формі «Персональні дані» (див. рис. 1.2) містяться поля: ім’я, прізвище, ім’я по батькові, дата народження (формат запису дд.мм.рррр або «дата невідома»). При використанні іншого формату запису програма зробить попередження. Відсутність даних в зазначених полях означає, що відповідна інформація невідома, тому заповнення усіх полів не вимагається програмою.

Збережена інформація відображається у полі, що виділено пунктиром, як вузол генеалогічного дерева, що йде від записаного предка.

**Функція 3. Пошук нащадків та предків**

Після побудови дерева можна здійснити пошук нащадків та предків обраного вузла.

Бажаний вузол помічається як «обраний» та натискається кнопка «Пошук нащадків та предків», яка знаходиться у верхній правій частині головного вікна трохи нижче інших кнопок (див. рис. 1.4).

Далі з’являється форма (див. рис. 1.5), у якій представлена інформація: дані обраного вузла (ім’я, прізвище, ім’я по батькові, дата народження) та два текстових поля (нащадки та предки), у яких буде виведено список родинних зв’язків обраного вузла генеалогічного дерева. Якщо обраний вузол не має Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описаниенащадків та/або предків, текстові поля будуть містити відповідний запис.

Рисунок 1.4 – Обраний вузол та кнопка пошуку нащадків та предків

Изображение выглядит как текст, снимок экрана, дисплей, Шрифт

Автоматически созданное описание

Рисунок 1.5 – Форма пошуку

**Функція 4. Редагування вузла**

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описаниеПід час побудови генеалогічного дерева можна редагувати створені вузли. Для цього треба обрати вузол, який містить помилку чи неправильні дані (вузол помічається як «обраний»). Після цього натиснути на кнопку «Редагувати вузол», яка знаходиться у правому верхньому кутку головного вікна програми (див. рис. 1.6), лівіше за кнопку «Видалити вузол».

Рисунок 1.6 – Редагування вузла

Далі відкривається форма (див. рис. 1.2) з введеними персональними даними, які можна зредагувати та оновити. Після збереження оновлених даних зміни вузла будуть одразу відображені у дереві.

**Функція 5. Видалення вузла**

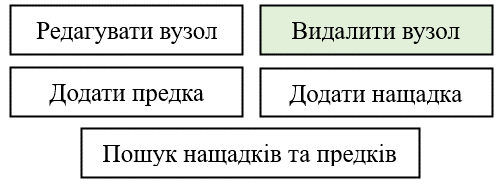
Натиснувши на бажаний вузол, він помічається як «обраний». Далі у верхній частині вікна з правого боку (див. рис. 1.7) потрібно обрати кнопку «Видалити вузол». Після підтвердження обраної дії вузол буде видалено з дерева.

Рисунок 1.7 – Видалення вузла

**Функція 6. Створення зображення**

Для збереження дерева у вигляді зображення потрібно у меню (див. рис. 1.7), що знаходиться у лівому кутку головного вікна програми, обрати розділ «Дерево». Зі списку обрати кнопку «Створити дерево» та натиснути на неї.

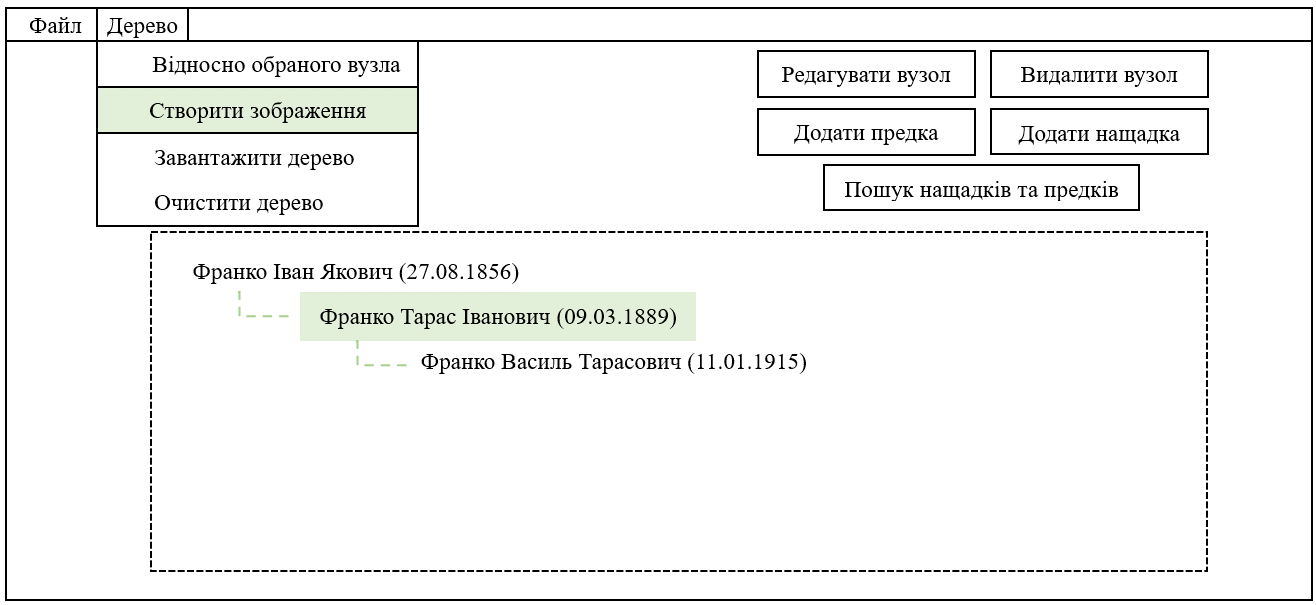
Після цього у діалоговому вікні збереження файлу, вказується назва файлу, обирається місце збереження та натискається кнопка «Зберегти/Save». На екрані з’являється повідомлення про успішне збереження зображення дерева.

Рисунок 1.8 – Зберігання зображення дерева

**Функція 7. Очищення дерева**

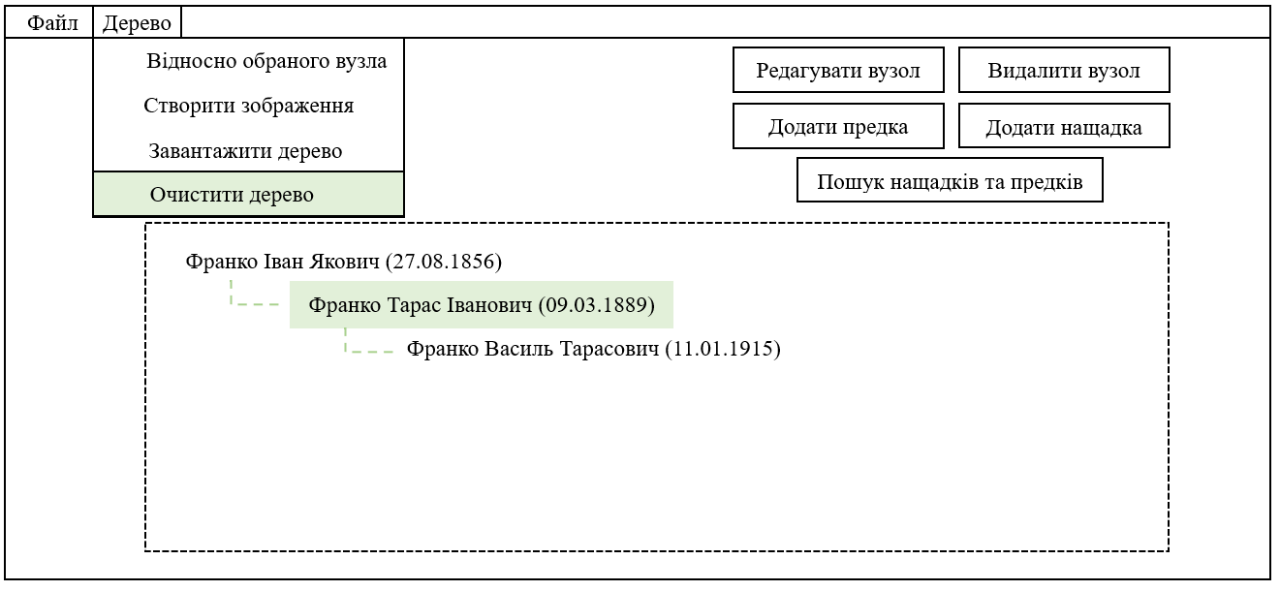
У верхньому лівому куті головної форми, де знаходиться меню, обирається розділ «Дерево». У представленому списку (див. рис. 1.9) натискається кнопка «Очистити дерево».

Рисунок 1.9 – Очищення вузлів дерева

Після підтвердження очищення дерева, з блоку, де будувалося дерево, видаляються всіх створені вузли.

**Функція 8. Завантаження дерева**

Для того, щоб завантажити раніше створене генеалогічне дерево потрібно у меню (верхній лівий кут головного вікна (рис.1.10)) обрати розділ «Дерево» → «Завантажити дерево».

Рисунок 1.10 – Завантаження дерева

Далі у діалоговому вікні завантаження файлів потрібно обрати потрібний файл (створений раніше) формату .json.

Після цього у блоці, що виділено пунктиром (рис. 1.1), у головному вікні програми буде представлено дерево, збережене в обраному файлі.

**Функція 9. Збереження файлу**

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описаниеДля збереження файлу та безпосередньо усієї інформації, яку містить генеалогічно дерево, потрібно у меню, що знаходиться у верхньому лівому куті головного вікна (рис.1.11) обрати «Файл» → «Зберегти».

Рисунок 1.11 –Збереження дерева

Далі потрібно вказати назву файлу та місце його збереження. Після збереження на екрані з’явиться повідомлення про успішно проведену операцію.

**Функція 10. Створення генеалогічного дерева відносно обраного вузла**

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описаниеЩоб створити іншу версію генеалогічного дерева відносно обраного вузла потрібно виділити потрібний вузол. Далі у меню, що знаходиться у лівому кутку вікна обрати «Дерево» → «Відносно обраного вузла» (див. рис. 1.12).

Рисунок 1.12 – Створення дерева відносно обраного вузла

Після цього з’явиться додаткове вікно програми (див. рис. 1.13), у якому буде сформоване оновлене дерево згідно обраного вузла генеалогічного дерева.

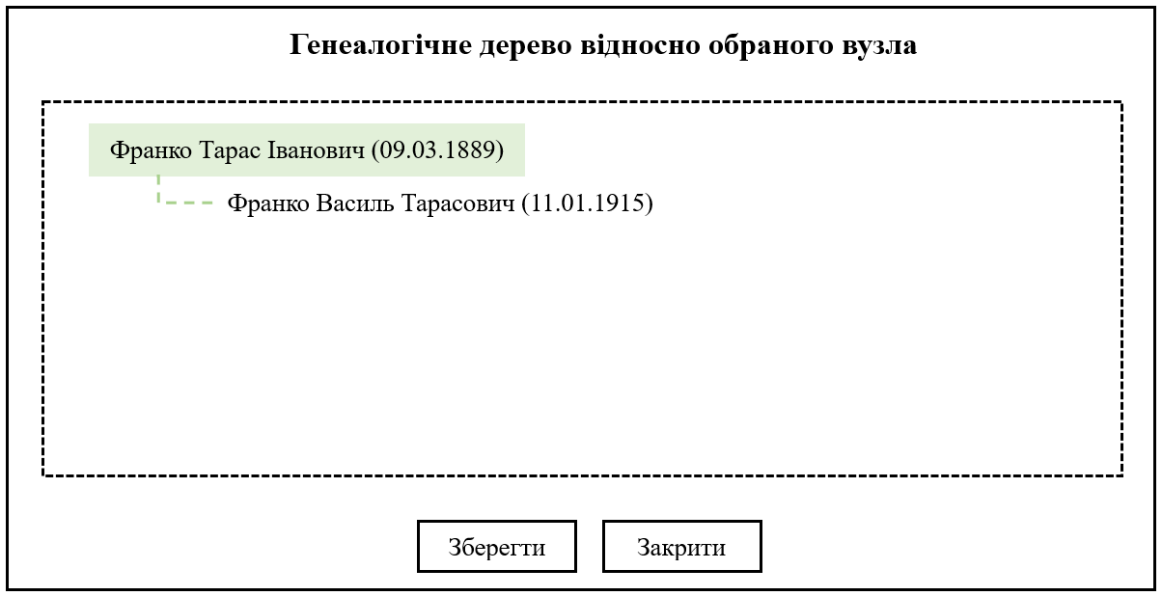


Рисунок 1.13 – Генеалогічне дерево відносно обраного вузла

# **2 ПРОЄКТУВАННЯ ПРОГРАМИ**

## **2.1 Архітектура програми**

Для організації програмної системи була обрана настільна архітектура з GUI. Для створення графічного інтерфейсу користувача на платформі Windows застосовувалась технологія Windows Forms.

GUI, як складова системи, дозволяє користувачеві взаємодіяти з програмою через графічні елементи: кнопки, поля введення тексту, списки та інші. Крім того, концепція моделі подій дозволить програмі реагувати на події, що прив’язані до відповідних елементів у GUI для виконання конкретних дій: видалення, додавання, редагування та інших. Це допоможе зробити програму інтуїтивно зрозумілою для користувача.

## **2.2 Діаграма класів**

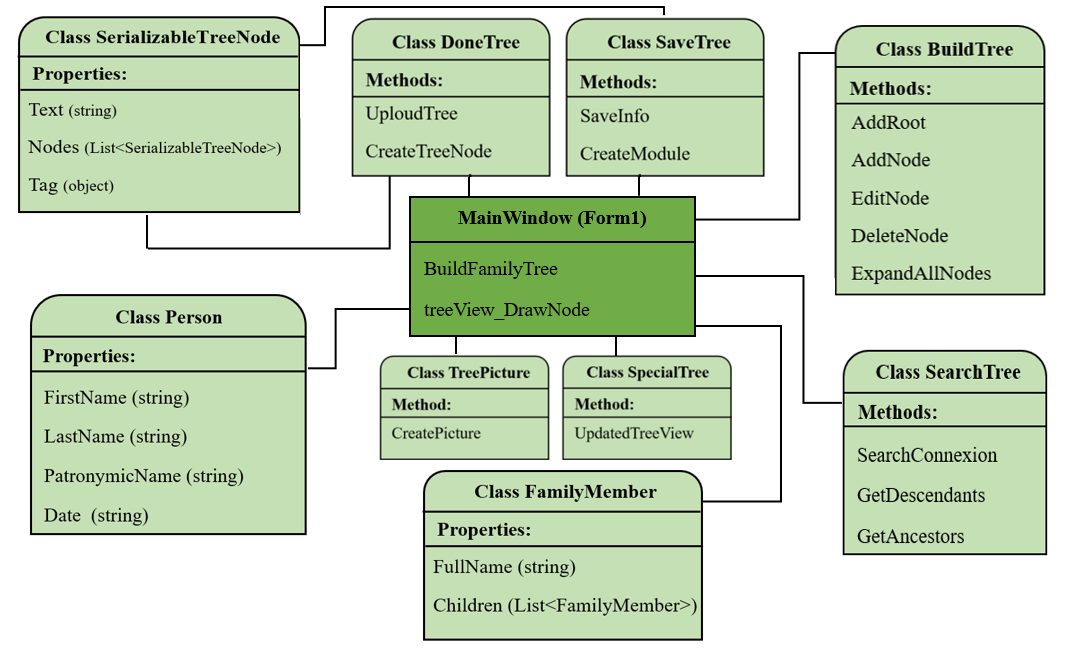
****На рисунку 2.1 зображено основні класи програми.

Рисунок 2.1 – Діаграма класів

Клас **BuildTree** відповідає за побудову генеалогічного дерева та містить такі методи як: додавання кореня, додавання, редагуванням та видалення вузла, розгортання вузлів.

Клас **SearchTree** містить методи пошуку зв’язків, предків та нащадків у генеалогічному дереві.

Клас **DoneTree** має два методи, перший – відповідає за завантаження раніше створеного файлу .json з генеалогічним деревом у treeview програми, а другий – за перетворення серіалізованих вузлів у список вузлів TreeNode.

Клас **SaveTree** містить один метод, який зберігає побудоване генеалогічне дерево (збереження інформаційної частини дерева - вузлів) та другий метод, що відповідає за серіалізацію вузлів.

Клас **TreePicture** має один метод, який відповідає за створення та збереження зображення (.png) дерева.

Клас **SpecialTree** містить метод створення генеалогічного дерева відносно обраного вузла.

Клас **Person** визначає структуру об'єкта, що представляє особу. Він має публічні властивості (FirstName, LastName, PatronymicName та Date), які використовуються для зберігання даних про особу.

Клас **FamilyMember** відповідає за збереження даних про члена сім'ї, має властивості: персональні дані та список дітей.

Клас **SerializableTreeNode** визначає структуру для представлення вузлів дерева, які можуть бути серіалізовані. Кожен вузол має текстове значення (Text), додаткову інформацію (Tag) та може мати дочірні вузли (Nodes), які також є об'єктами типу SerializableTreeNode.

## **2.3 Формат зберігання зовнішніх даних**

Для зберігання даних у програмі використовується мовно-незалежний та універсальний формат даних JSON, що базується на синтаксисі JavaScript та надає простий спосіб представлення структурованих даних у вигляді тексту.

JSON має чітку структуру, завдяки якій можна легко відлагоджувати та розуміти збережені дані.

# **3 ІНСТРУКЦІЯ КОРИСТУВАЧА**

## **3.1 Встановлення програми**

Для коректної роботи програми «Генеалогічне дерево» мають бути встановлені Visual Studio 2022 та .NET 7.0 SDK. У Visual Studio можна відкрити рішення (solution) та запустити його.

## **3.2 Користування програмою**

**Функція «Додати предка»**

*Інструкція:*

*Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание*1. На головному вікні знайти кнопку «Додати предка» (див. рис. 3.1) та натиснути на неї.

Рисунок 3.1 – Кнопки головного вікна програми

2. У вікні, що з’явилося (див. рис 3.2), заповнити потрібні та відомі поля.

3. Натиснути кнопку «Зберегти».

**Функція «Додати нащадка»**

*Інструкція:*

1. Натиснути на вузол дерева, до якого потрібно додати нащадка.

2. На головному вікні знайти кнопку «Додати нащадка» (див. рис. 3.1) та натиснути на неї.

3. У вікні, що з’явилося (див. рис 3.2), заповнити потрібні та відомі поля.

Изображение выглядит как текст, снимок экрана, Шрифт, дисплей

Автоматически созданное описание4. Натиснути кнопку «Зберегти».

Рисунок 3.2 – Заповнення персональних даних

**Функція «Видалити вузол»**

*Інструкція:*

1. Обрати вузол дерева (натиснути на нього), який треба видалити.

2. На головному вікні знайти кнопку «Видалити вузол» (див. рис. 3.1) та натиснути на неї.

3. У вікні, що з’явилося, підтвердити видалення, натиснувши «Так».

**Функція «Редагувати вузол»**

*Інструкція:*

1. Обрати вузол дерева, який потребує редагування (натиснути на нього).

2. На головному вікні знайти кнопку «Редагувати вузол» (див. рис. 3.1) та натиснути на неї.

3. У вікні (має вигляд див. рис. 3.2), що з’явилося, можна внести бажані зміни до персональних даних.

4. Натиснути кнопку «Зберегти».

**Функція «Пошук нащадків та предків»**

*Інструкція:*

1. Обрати вузол дерева, відносно якого проводитиметься пошук (натиснути на нього).

2. На головному вікні знайти кнопку «Пошук нащадків та предків» (див. рис. 3.1) та натиснути на неї.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание3. З’явиться додаткове вікно програми (див. рис. 3.3), у якому буде представлена інформація щодо нащадків та предків обраного вузла.

Рисунок 3.3 – Пошук зв’язків

4. Після ознайомлення з інформацією натиснути «Закрити».

**Функція «Очистити дерево»**

*Інструкція:*

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание1. У верхньому меню (знаходиться у лівій частині вікна) обрати розділ «Дерево» (див. рис. 3.4) та натиснути на нього.

Рисунок 3.4 – Меню. Розділ «Дерево»

2. У списку обрати «Очистити дерево».

3. У вікні, що з’явилося, підтвердити очищення всього дерева, натиснувши «Так».

**Функція «Створити зображення»**

*Інструкція:*

1. У верхньому меню (знаходиться у лівій частині вікна) обрати розділ «Дерево» (див. рис. 3.4) та натиснути на нього.

2. У списку обрати «Створити зображення».

3. У діалоговому вікні «Збереження зображення» вказати назву файлу та обрати місце його збереження.

4. Після натискання на кнопку «Зберегти» з’являється повідомлення про успішне збереження зображення генеалогічного дерева.

**Функція «Відносно обраного вузла»**

Дана функція передбачає створення генеалогічного дерева відносно обраного вузла, тобто нове побудоване дерево буде складатися тільки з цього вузла – головного предка та всіх його нащадків.

*Інструкція:*

1. Обрати вузол дерева (натиснути на нього), відносно якого проводитиметься побудова нового генеалогічного дерева.

2. У верхньому меню (знаходиться у лівій частині вікна) обрати розділ «Дерево» (див. рис. 3.4) та натиснути на нього.

3. У списку обрати «Відносно обраного вузла».

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание4. У вікні (див. рис. 3.5), що з'явилося, буде представлено дерево відносно обраного вузла.

Рисунок 3.5 – Генеалогічне дерево відносно обраного нащадка

5. Після ознайомлення з побудованим деревом натиснути «Закрити».

*Інструкція збереження дерева відносно обраного вузла:*

Після пункту 4:

5. У вікні (див. рис. 3.5), яке представляє створене генеалогічне дерево, обрати кнопку «Зберегти».

6. У діалоговому вікні «Зберегти дерево як JSON файл» вказати назву файлу та обрати місце його збереження.

7. Після натискання на кнопку «Зберегти» з’являється повідомлення про успішне збереження генеалогічного дерева.

**Функція «Завантажити дерево»**

*Інструкція:*

1. У верхньому меню (знаходиться у лівій частині вікна) обрати розділ «Дерево» (див. рис. 3.4) та натиснути на нього.

2. У списку обрати «Завантажити дерево».

3. З’являється діалогове вікно, у якому обирається файл (формату .json), який треба відкрити у програмі.

4. При натисканні «Відкрити» обране дерево з’являється у головному вікні програми.

**Функція «Зберегти»**

*Інструкція:*

1. У верхньому меню (знаходиться у лівій частині вікна) обрати розділ «Файл» (див. рис. 3.6) та натиснути на нього.

Изображение выглядит как текст, велосипед, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.6 – Меню головного вікна програми. Розділ «Файл»

2. У списку обрати «Зберегти».

3. У діалоговому вікні «Зберегти дерево як JSON файл» вказати назву файлу та обрати місце його збереження.

4. Після натискання на кнопку «Зберегти» з’являється повідомлення про успішне збереження генеалогічного дерева.

# **ВИСНОВКИ**

Під час виконання курсової роботи була розроблена програма «Генеалогічне дерево», яка дозволяє користувачам будувати генеалогічне дерево своєї родини, здійснювати пошук за родинними зв’язками нащадків та предків, зберігати створене дерево у різних форматах задля подальшого зручного використання за певним призначенням.

Щодо відповідності цілям та опису програмного завдання, можна сказати, що створений проєкт ще має багато напрямків для подальшого розвитку та покращення. Особливо для мене, це був перший досвід у створенні повноцінної програми, тому під час виконання завдання виникло досить багато труднощів. Деякі були успішно вирішені, проте залишилися й ті, котрі потребують подальшого розглядання, пошуку нових рішень та реалізацій. Наприклад, більш зрозуміліше відображання ієрархічної структури дерева допоможе візуально краще сприймати побудоване дерево сімейних зв’язків.

Безсумнівно, робота над розробкою програми «Генеалогічне дерево» допомогла краще ознайомитися з мовою програмування C#, отримати нові та поглибити наявні знання. Крім того, використання у проєкті Windows Forms, які представляють собою набір інструментів для розробки графічних інтерфейсів користувача для програм на платформі Windows, спростило процес розробки і дозволило створити функціональну програму з візуально привабливим інтерфейсом.

# **ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. C# / BestProg. Програмування: теорія та практика. URL: <https://www.bestprog.net/uk/sitemap_ua/c-3/> (дата звернення: 10.06.2023)

2. Бондарєв В.М. Електронний учбовий посібник з дисципліни «Основи програмування» URL: <http://tss.co.ua:5555/> (дата звернення: 13.06.2023)

3. Stack Overflow / Сollection of coding questions & answers. URL: <https://stackoverflow.com/> (дата звернення: 12.06.2023)

4. Офіційна документація Microsoft Docs / Microsoft Learn. URL: <https://docs.microsoft.com/> (дата звернення: 12.06.2023)

**ДОДАТОК А**

Вихідний код програми

**Form1**

using static System.Runtime.InteropServices.JavaScript.JSType;

using System.Diagnostics.Metrics;

using static System.Windows.Forms.DataFormats;

using System.Net;

using System.Text;

using System.Drawing.Imaging;

using System.Xml;

using Newtonsoft.Json;

namespace GenealogicalTree

{

public partial class Form1 : Form

{

private BuildTree buildTree;

private SearchTree searchTree;

private DoneTree doneTree;

private SaveTree saveTree;

private TreePicture treePicture;

private BuildTree expandNodes;

private SpecialTree specialTree;

public Form1()

{

InitializeComponent();

buildTree = new BuildTree(treeView);

searchTree = new SearchTree(treeView);

doneTree = new DoneTree(treeView);

saveTree = new SaveTree(treeView);

treePicture = new TreePicture(treeView);

specialTree = new SpecialTree(treeView);

}

private void BuildFamilyTree(FamilyMember member, TreeNode parentNode)

{

foreach (FamilyMember child in member.Children)

{

TreeNode childNode = new TreeNode(child.FullName);

childNode.Tag = child;

parentNode.Nodes.Add(childNode);

BuildFamilyTree(child, childNode);

}

}

private void treeView\_DrawNode(object sender, DrawTreeNodeEventArgs e)

{

bool isSelected = (e.State & TreeNodeStates.Selected) != 0;

Color foreColor = isSelected ? Color.Black : treeView.ForeColor;

Color backColor = isSelected ? Color.DarkSeaGreen : treeView.BackColor;

using (SolidBrush backgroundBrush = new SolidBrush(backColor))

{

e.Graphics.FillRectangle(backgroundBrush, e.Bounds);

}

TextRenderer.DrawText(e.Graphics, e.Node.Text, treeView.Font,

e.Bounds, foreColor, TextFormatFlags.VerticalCenter);

if (isSelected)

{

ControlPaint.DrawFocusRectangle(e.Graphics, e.Bounds);

}

}

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

{

saveTree.SaveInfo();

}

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

{

this.Close();

}

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

{

DialogResult result = MessageBox.Show("Ви впевнені, що хочете очистити все дерево?",

"Попередження про очищення", MessageBoxButtons.YesNo,

MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

treeView.Nodes.Clear();

}

}

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

{

buildTree.AddRoot();

}

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

{

buildTree.AddNode();

BuildTree.ExpandAllNodes(treeView.Nodes);

}

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

{

buildTree.EditNode();

}

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

{

searchTree.SearchConnexion();

}

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

{

buildTree.DeleteNode();

}

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

{

doneTree.UploadTree();

BuildTree.ExpandAllNodes(treeView.Nodes);

}

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

{

treePicture.CreatePicture();

}

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

{

TreeNode selectedNode = treeView.SelectedNode;

if (selectedNode != null)

{

specialTree.UpdatedTreeView(selectedNode, expandNodes);

}

else

{

MessageBox.Show("Оберіть вузол дерева.", "Помилка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

DialogResult result = MessageBox.Show("Після закриття програми " +

"дерево не буде збережено. " +

"Ви хочете зберегти дані перед виходом?", "Попередження",

MessageBoxButtons.YesNo, MessageBoxIcon.Information);

if (result == DialogResult.Yes)

{

saveTree.SaveInfo();

}

if (result == DialogResult.No)

{

Application.Exit();

}

}

}

}

**FormInfo**

using GenealogicalTree;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Runtime.InteropServices.JavaScript.JSType;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace GenealogicalTree

{

public partial class FormInfo : Form

{

public Person Data { get; private set; }

public FormInfo()

{

InitializeComponent();

}

public void GetData(Person modules)

{

firstname.Text = modules.FirstName;

lastname.Text = modules.LastName;

patronymicname.Text = modules.PatronymicName;

date.Text = modules.Date;

}

public void SaveData()

{

string firstName = firstname.Text;

string lastName = lastname.Text;

string patronymicName = patronymicname.Text;

string birthdate = date.Text;

if (string.IsNullOrEmpty(birthdate))

{

birthdate = "дата невідома";

}

else if (!ValidateBirthdate(birthdate))

{

MessageBox.Show("Невірний формат дати народження. " +

"Введіть дату у форматі 'дд.мм.рррр'. " +

"Якщо дата невідома, залиште поле порожнім.", "Помилка");

return;

}

if (ContainsDigits(firstName) || ContainsDigits(lastName) || ContainsDigits(patronymicName))

{

MessageBox.Show("Ім'я, прізвище та по батькові

не повинні містити цифри.", "Помилка");

return;

}

Person modules = new Person(firstName, lastName, patronymicName, birthdate);

Data = modules;

this.Close();

MessageBox.Show("Дані збережено успішно.", "Повідомлення");

}

private bool ValidateBirthdate(string input)

{

if (input == "дата невідома")

{

return true;

}

string pattern = @"^\d{2}\.\d{2}\.\d{4}$";

Regex regex = new Regex(pattern);

return regex.IsMatch(input);

}

private bool ContainsDigits(string input)

{

return input.Any(char.IsDigit);

}

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

{

SaveData();

}

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

{

DialogResult result = MessageBox.Show("Зміни не будуть збережені. " +

"Продовжити?", "Попередження", MessageBoxButtons.OKCancel,

MessageBoxIcon.Warning);

if (result == DialogResult.OK)

{

this.Close();

}

}

private void textBox\_TextChanged(object sender, EventArgs e)

{

System.Windows.Forms.TextBox textBox = (System.Windows.Forms.TextBox)sender;

if (!string.IsNullOrEmpty(textBox.Text))

{

int selectionStart = textBox.SelectionStart;

string firstLetter = textBox.Text.Substring(0, 1);

string remainingText = textBox.Text.Substring(1);

textBox.Text = firstLetter.ToUpper() + remainingText;

textBox.SelectionStart = selectionStart;

}

}

}

}

**SearchForm**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace GenealogicalTree

{

public partial class SearchForm : Form

{

public SearchForm()

{

InitializeComponent();

}

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

{

this.Close();

}

}

}

**ShapeForm**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.ComponentModel.Design.ObjectSelectorEditor;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace GenealogicalTree

{

public partial class ShapeForm : Form

{

private SaveTree saveTree;

public ShapeForm()

{

InitializeComponent();

}

private void treeView\_DrawNode(object sender, DrawTreeNodeEventArgs e)

{

bool isSelected = (e.State & TreeNodeStates.Selected) != 0;

Color foreColor = isSelected ? Color.Black : newTree.ForeColor;

Color backColor = isSelected ? Color.DarkSeaGreen : newTree.BackColor;

using (SolidBrush backgroundBrush = new SolidBrush(backColor))

{

e.Graphics.FillRectangle(backgroundBrush, e.Bounds);

}

TextRenderer.DrawText(e.Graphics, e.Node.Text, newTree.Font, e.Bounds, foreColor, TextFormatFlags.VerticalCenter);

if (isSelected)

{

ControlPaint.DrawFocusRectangle(e.Graphics, e.Bounds);

}

}

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

{

this.Close();

}

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

{

saveTree.SaveInfo();

}

}

}

**Class BuildTree**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GenealogicalTree

{

// Клас містить методи, які відповідать за побудову генеалогічного дерева

// (додавання предка, нащадка, редагування вузлів та їх видалення).

public class BuildTree

{

private TreeView treeView;

public BuildTree(TreeView treeView)

{

this.treeView = treeView;

}

public void AddRoot()

{

using (FormInfo newForm = new FormInfo())

{

newForm.ShowDialog();

Person modules = newForm.Data;

if (modules != null)

{

string nodeText = modules.LastName + " " + modules.FirstName +

" " + modules.PatronymicName;

string nodeDate = modules.Date;

nodeText += " " + "(" + nodeDate + ")";

TreeNode rootNode = new TreeNode(nodeText);

rootNode.Tag = modules;

treeView.Nodes.Add(rootNode);

}

}

}

public void AddNode()

{

using (FormInfo newForm = new FormInfo())

{

newForm.ShowDialog();

Person modules = newForm.Data;

if (modules != null)

{

string nodeText = modules.LastName + " " + modules.FirstName +

" " + modules.PatronymicName;

string nodeDate = modules.Date;

nodeText += " " + "(" + nodeDate + ")";

TreeNode rootNode = new TreeNode(nodeText);

rootNode.Tag = modules;

TreeNode selectedNode = treeView.SelectedNode;

if (selectedNode != null)

{

selectedNode.Nodes.Add(rootNode);

}

else

{

treeView.Nodes.Add(rootNode);

}

}

}

}

public void EditNode()

{

TreeNode selectedNode = treeView.SelectedNode;

if (selectedNode != null)

{

Person modules = selectedNode.Tag as Person;

if (modules != null)

{

using (FormInfo newForm = new FormInfo())

{

newForm.GetData(modules);

newForm.ShowDialog();

Person updatedModules = newForm.Data;

if (updatedModules != null)

{

string nodeText = updatedModules.LastName + " "

+ updatedModules.FirstName + " "

+ updatedModules.PatronymicName;

string nodeDate = updatedModules.Date;

nodeText += " " + "(" + nodeDate + ")";

selectedNode.Text = nodeText;

selectedNode.Tag = updatedModules;

}

}

}

else

{

MessageBox.Show($"Редагування даних можливе лише при " +

$"початковому створенні дерева. Завантажене дерево " +

$"можна лише переглядати, додавати нові зв'язки та " +

$"робити пошук нащадків й предків", "Попередження");

}

}

}

public void DeleteNode()

{

if (treeView.SelectedNode != null)

{

DialogResult result = MessageBox.Show("Ви впевнені, що хочете " +

"видалити вузол?", "Підтвердження видалення",

MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

treeView.SelectedNode.Remove();

}

}

}

public static void ExpandAllNodes(TreeNodeCollection nodes)

{

foreach (TreeNode node in nodes)

{

node.Expand();

ExpandAllNodes(node.Nodes);

}

}

}

}

**Class DoneTree**

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static GenealogicalTree.SaveTree;

using static System.Runtime.InteropServices.JavaScript.JSType;

namespace GenealogicalTree

{

// Клас відповідає за завантаження збереженого дерева .json

// у treeview програми (перший метод) та перетворення

// серіалізованих вузлів у список вузлів TreeNode (другий метод).

public class DoneTree

{

private TreeView treeView;

public DoneTree(TreeView treeView)

{

this.treeView = treeView;

}

public void UploadTree()

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "JSON files (\*.json)|\*.json";

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

{

string jsonFilePath = openFileDialog.FileName;

string jsonContent = File.ReadAllText(jsonFilePath);

List<SerializableTreeNode> nodes =

JsonConvert.DeserializeObject<List<SerializableTreeNode>>(jsonContent);

treeView.Nodes.Clear();

treeView.Nodes.AddRange(ConvertToTreeViewNodes(nodes).ToArray());

}

}

private List<TreeNode> ConvertToTreeViewNodes(List<SerializableTreeNode> serializableNodes)

{

List<TreeNode> treeNodes = new List<TreeNode>();

foreach (SerializableTreeNode serializableNode in serializableNodes)

{

TreeNode treeNode = new TreeNode

{

Text = serializableNode.Text,

Tag = serializableNode.Tag,

};

if (serializableNode.Nodes != null)

{

treeNode.Nodes.AddRange(ConvertToTreeViewNodes(serializableNode.Nodes).ToArray());

}

treeNodes.Add(treeNode);

}

return treeNodes;

}

}

}

**Class SaveTree**

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static GenealogicalTree.DoneTree;

namespace GenealogicalTree

{

// Клас,що містить метод збереження інформаційної частини дерева (вузлів)

// та метод серіалізації вузлів.

public class SaveTree

{

private TreeView treeView;

public SaveTree(TreeView treeView)

{

this.treeView = treeView;

}

public void SaveInfo()

{

SaveFileDialog saveDialog = new SaveFileDialog();

saveDialog.Filter = "JSON файл (\*.json)|\*.json";

saveDialog.Title = "Зберегти дерево вузлів";

DialogResult result = saveDialog.ShowDialog();

if (result == DialogResult.OK)

{

string filePath = saveDialog.FileName;

var nodes = ConvertToSerializableNodes(treeView.Nodes);

string json = JsonConvert.SerializeObject

(nodes, Newtonsoft.Json.Formatting.Indented);

File.WriteAllText(filePath, json);

MessageBox.Show("Дані з дерева були успішно збережені.");

}

}

private List<SerializableTreeNode> ConvertToSerializableNodes(TreeNodeCollection nodes)

{

List<SerializableTreeNode> serializableNodes = new List<SerializableTreeNode>();

foreach (TreeNode node in nodes)

{

SerializableTreeNode serializableNode = new SerializableTreeNode

{

Text = node.Text,

Tag = node.Tag,

Nodes = ConvertToSerializableNodes(node.Nodes)

};

serializableNodes.Add(serializableNode);

}

return serializableNodes;

}

}

}

**Class SearchTree**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GenealogicalTree

{

// Клас "Пошук у дереві" має методи пошуку зв'язків між вузлами

// (отримання предків та нащадків обраного вузла).

public class SearchTree

{

private TreeView treeView;

public SearchTree(TreeView treeView)

{

this.treeView = treeView;

}

public void SearchConnexion()

{

SearchForm searchForm = new SearchForm();

TreeNode selectedNode = treeView.SelectedNode;

if (selectedNode != null)

{

BuildTree.ExpandAllNodes(treeView.Nodes);

searchForm.node.Text = selectedNode.Text;

List<string> descendants = GetDescendants(selectedNode);

if (descendants.Count > 0)

{

searchForm.posterity.Text = string.Join(Environment.NewLine, descendants);

}

else

{

searchForm.posterity.Text = "Нащадків не знайдено.";

}

List<string> ancestors = GetAncestors(selectedNode);

if (ancestors.Count > 0)

{

searchForm.ancestry.Text = string.Join(Environment.NewLine, ancestors);

}

else

{

searchForm.ancestry.Text = "Предків не знайдено.";

}

searchForm.ShowDialog();

}

}

private List<string> GetDescendants(TreeNode node)

{

List<string> descendants = new List<string>();

foreach (TreeNode childNode in node.Nodes)

{

descendants.Add(childNode.Text);

descendants.AddRange(GetDescendants(childNode));

}

return descendants;

}

private List<string> GetAncestors(TreeNode node)

{

List<string> ancestors = new List<string>();

TreeNode parent = node.Parent;

while (parent != null)

{

ancestors.Add(parent.Text);

parent = parent.Parent;

}

return ancestors;

}

}

}

**Class SpecialTree**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GenealogicalTree

{

// Клас, що містить метод, який відповідає за створення

// генеалогічного дерева відносно обраного вузла.

public class SpecialTree

{

private TreeView treeView;

public SpecialTree(TreeView treeView)

{

this.treeView = treeView;

}

public void UpdatedTreeView(TreeNode selectedNode, BuildTree expandNodes)

{

ShapeForm updatedForm = new ShapeForm();

TreeView updatedTreeView = updatedForm.newTree;

TreeNode rootNode = new TreeNode(selectedNode.Text);

updatedTreeView.Nodes.Add(rootNode);

foreach (TreeNode childNode in selectedNode.Nodes)

{

rootNode.Nodes.Add((TreeNode)childNode.Clone());

}

BuildTree.ExpandAllNodes(updatedTreeView.Nodes);

updatedForm.ShowDialog();

}

}

}

**Class TreePicture**

using System;

using System.Collections.Generic;

using System.Drawing.Imaging;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GenealogicalTree

{

// Клас збереження дерева у вигляді зображення.

public class TreePicture

{

private TreeView treeView;

public TreePicture(TreeView treeView)

{

this.treeView = treeView;

}

public void CreatePicture()

{

using (SaveFileDialog saveDialog = new SaveFileDialog())

{

saveDialog.Filter = "PNG зображення (\*.png)|\*.png";

saveDialog.Title = "Збереження зображення";

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

{

string filePath = saveDialog.FileName;

Bitmap image = new Bitmap(treeView.Width, treeView.Height);

treeView.DrawToBitmap(image, new Rectangle

(0, 0, treeView.Width, treeView.Height));

image.Save(filePath, ImageFormat.Png);

MessageBox.Show($"Дерево було успішно збережено " +

$"у вигляді зображення.", "Збереження дерева",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

}

**Class Person**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GenealogicalTree

{

public class Person

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string PatronymicName { get; set; }

public string Date { get; set; }

public Person(string firstName, string lastName, string patronymicName, string date)

{

FirstName = firstName;

LastName = lastName;

PatronymicName = patronymicName;

Date = date;

}

}

}

**Class FamilyMember**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GenealogicalTree

{

public class FamilyMember

{

public string FullName { get; set; }

public List<FamilyMember> Children { get; set; } = new List<FamilyMember>();

public FamilyMember(string fullName)

{

FullName = fullName;

Children = new List<FamilyMember>();

}

}

}

**Class SerializableTreeNode**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GenealogicalTree

{

// Клас, що визначає структуру для представлення дерева, яке може бути серіалізовано.

public class SerializableTreeNode

{

public string Text { get; set; }

public object Tag { get; set; }

public List<SerializableTreeNode> Nodes { get; set; }

}

}