Міністерство освіти і науки України
Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з лабораторної роботи № 1

з дисципліни «Soft skills, групова динаміка та комунікації» на тему:

«ОФОРМЛЕННЯ ДОКУМЕНТАЦІЇ ЗА ДОПОМОГОЮ ТЕКСТОВОГО РЕДАКТОРА»

Виконал(а):

студент групи КНТ-113сп І. А. Щедровський

Прийнял(а)

асистент: А. В. Бєлова (Д. А. Каврін)

2022

1 ОФОРМЛЕННЯ ДОКУМЕНТАЦІЇ ЗАДОПОМОГОЮ ТЕКСТОВОГО РЕДАКТОРА

1.1 Мета роботи

1.1.1 Вивчити основні можливості сучасних текстових редакторів.

1.1.2 Навчитися редагувати документи відповідно до діючих стандартів за допомогою засобів текстового редактору.

1.2 Завдання роботи

1.2.1 Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи дані методичні вказівки, а також рекомендовану літературу.

1.2.2 Вивчити можливості роботи з одним з сучасних текстових редакторів, зокрема дослідити пункти головного меню та навчитися: – створювати нові документи та зберігати їх у різних форматах; – встановлювати параметри сторінки та параметри абзаців; – змінювати параметри шрифту;

– працювати зі стилями;

– працювати з макросами;

– розташовувати текст у декілька колонок;

– працювати з маркованими та нумерованими списками; – створювати та редагувати таблиці, формули, графічні примітиви.

1.2.3 Особисто обрати документ для редагування (обсягом не менше ніж 30 сторінок). Обраний документ обов’язково повинен бути структурований на розділи (заголовки першого рівня), підрозділи (заголовки другого рівня), пункти (заголовки третього рівня) та підпункти (заголовки четвертого рівня). Крім того документ, що редагується, повинен містити наступні елементи:

– таблиці;

– рисунки;

– формули;

– нумеровані та марковані списки.

У випадку, якщо обраний документ не містить вказаних елементів і не є структурованим, необхідно самостійно додати ці елементи, узгодивши їх з текстом документу, та виконати розбиття документу на розділи, підрозділи, пункти та підпункти для його подальшого редагування відповідно до діючих стандартів.

1.2.4 Ознайомитися зі стандартом підприємства ДСТУ 3008:2015.

1.2.5 В обраному для редагування документі створити (або модифікувати існуючі) стилі: «Заголовок 1», «Заголовок 2», «Заголовок 3», «Заголовок 4», «Звичайний» таким чином, щоб вони відповідали вимогам ДСТУ 3008:2015 до: заголовків розділів, підрозділів, пунктів, підпунктів та основного тексту.

1.2.6 Відредагувати текст відповідно до ДСТУ 3008:2015, використовуючи засоби текстового редактору, зокрема створені стилі.

1.2.7 Додати у документ зміст, використовуючи засоби текстового редактору для автоматичного створення змісту.

1.2.8 Оформити звіт з роботи.

1.2.9 Відповісти на контрольні запитання.

1.3 Короткі теоретичні відомості

Microsoft Word (MS Word) – це текстовий процесор, що входить до офісного пакету Microsoft Office компанії Microsoft, надаючи користувачам можливості переглядати існуючі текстові документи, створювати нові, редагувати існуючі. Фактично саме цей текстовий процесор є стандартом в цій області, відповідно і формат текстового документа Microsoft Word загалом вважається стандартом.

Тому альтернативні рішення зазвичай мають можливість відкривати (імпортувати загалом), зберігати (експортувати) текстові документи в характерному для Word форматі (doc, docx). При цьому функціонально фактично на даний момент альтернативні рішення мають відповідні засоби, що є у Word.

Microsoft Word надає цілий ряд засобів, які дозволяють форматувати текст необхідним чином, а також виконувати супутні процеси, пов’язані з представленням у текстових документах таблиць, графіків (зображень, схем тощо) та формул.

1.4 Фрагмент тексту

2 Проектування програмного забезпечення системи

В даному розділі розглядається процес проектування програмного забезпечення системи моделювання структур графів. Система призначена для зручного візуального моделювання графів, що може бути використана як користувачами, так і розробниками для відображення роботи різних алгоритмів.

2.1 Постановка мети

Основною метою цієї курсової роботи є розробка програмного забезпечення для моделювання структур графів з використанням об'єктно-орієнтованого програмування (ООП). Програмний продукт повинен демонструвати високий рівень точності відображення графічних структур та забезпечувати зручний інтерфейс для користувача.

2.2 Аналіз функцій системи

Аналіз функцій системи є важливим етапом у розробці програмного забезпечення для моделювання структур графів. Він дозволяє визначити основні функціональні вимоги до системи, які є необхідними для її подальшого проектування та реалізації. Детальний розгляд функцій дозволяє забезпечити повноту та ефективність розроблюваного програмного продукту.

У рамках курсової роботи передбачається розробка програмного забезпечення для моделювання структур графів з використанням об'єктно-орієнтованого програмування. Основними функціями системи є:

– відображення графа. Граф повинен відображатися на сторінці на всю сторінку. При цьому повинні бути відображені всі його елементи, а саме: вершини, ребра та їхні властивості;

– запуск алгоритмів. Алгоритми повинні можна запускати через графічний інтерфейс. При цьому повинні бути доступні всі необхідні параметри для запуску алгоритму;

– можливість створення та видалення нод. Користувач повинен мати можливість створювати нові вершини та видаляти існуючі;

– можливість створення, зміни та видалення граней. Користувач повинен мати можливість створювати нові ребра, змінювати їхні властивості та видаляти існуючі;

– міжвіконна взаємодія. Система повинна забезпечувати взаємодію між різними вікнами. Наприклад, користувач повинен мати можливість перетягувати вершини з одного вікна в інше;

– наявність декількох режимів роботи застосунку. Система повинна підтримувати кілька режимів роботи, наприклад, напрямлений та не напрямлений граф, граф з вагами та без;

– візуальне підсвічування виконання алгоритму. При виконанні алгоритму він повинен підсвічуватись на графіку. Це дозволить користувачеві простежити за його ходом;

– можливість переміщуватись по внутрішнім координатам. Користувач повинен мати можливість переміщуватись по графу, використовуючи внутрішні координати.

Відображення графа є однією з основних функцій системи. Граф повинен відображатись на сторінці на всю сторінку. При цьому повинні бути відображені всі його елементи, а саме: вершини, ребра та їхні властивості.

Відображення вершин може здійснюватися за допомогою різних способів, наприклад, у вигляді точок, прямокутників, кіл тощо. Ребра можуть відображатися у вигляді ліній, стріл тощо. Властивості вершин і ребер можуть відображатися у вигляді текстових позначок або графічних елементів.

Алгоритми повинні можна запускати через графічний інтерфейс. При цьому повинні бути доступні всі необхідні параметри для запуску алгоритму.

Параметри алгоритму можуть бути представлені у вигляді текстових полів, списків, діалогових вікон тощо. При запуску алгоритму повинні бути перевірені всі введені параметри. Якщо параметри введені неправильно, алгоритм не повинен запускатися.

Користувач повинен мати можливість створювати нові вершини та видаляти існуючі.

Створення вершини може здійснюватися за допомогою кнопки або меню. При створенні вершини користувач повинен ввести її ім'я та інші необхідні параметри.

Видалення вершини може здійснюватися за допомогою кнопки або меню. При видаленні вершини всі ребра, які ведуть до цієї вершини, також повинні бути видалені.

Користувач повинен мати можливість створювати нові ребра, змінювати їхні властивості та видаляти існуючі.

Створення ребра може здійснюватися за допомогою кнопки або меню. При створенні ребра користувач повинен вибрати початкову та кінцеву вершини ребра, а також ввести його властивості.

Зміна властивостей ребра може здійснюватися за допомогою кнопки або меню. При зміні властивостей ребра користувач повинен внести необхідні зміни.

Видалення ребра може здійснюватися за допомогою кнопки або меню. При видаленні ребра воно буде видалене з графа.

Система повинна забезпечувати взаємодію між різними вікнами. Наприклад, користувач повинен мати можливість перетягувати вершини з одного вікна в інше.

Міжвіконна взаємодія може здійснюватися за допомогою об'єктів, які можуть переміщуватись між вікнами.

2.3 Проєктування графічного інтерфейсу

Розробка графічного інтерфейсу (GUI) є ключовим етапом у створенні програмного забезпечення для моделювання структур графів. Інтерфейс повинен бути зручним, інтуїтивно зрозумілим та забезпечувати зручність взаємодії користувача з системою.

Основні елементи інтерфейсу:

- поле для відображення графа;
- меню;
- контекстне меню;
- панель на якій буде форма для запуску алгоритмів.

Поле для відображення графа - основний візуальний елемент, що представляє собою графічне відображення нод (вузлів) та ребер графу.

Меню містить кнопки, що дозволяють користувачу виконувати різні дії, такі як відкриття панелі для запуску алгоритмів або ж запускати їх напряму.

Контекстне меню – дозволяє взаємодіяти з окремими елементами графу . Воно виводиться за правим кліком мишею. Містить опції для швидкого виклику функцій, таких як видалення старих нод або ж створення нових.

Панель – блок який відображається справа на єкрані та може в собі рендерити різні форми, які потрібні для вказання даних для запуску алгоритмів.

Меню розподілене на логічні розділи, відповідальні за різні аспекти взаємодії з графом. Кнопки та опції мають текстовий та графічний вигляд для забезпечення належного рівня інтуїтивності та доступності.

Першою функцією є відкриття іншого пресету. Всього в проєкті є 4 пресети між якими можна перемикатись та редагувати кожен окремо.

Другою функцією є відображення напрямків ребер. Включення або виключення відображення напрямку ребер, що корисно при роботі з орієнтованими графами. Це не тільки візуально, алгоритми також будуть враховувати це

Третьою функцією є відображення ваги ребер. Можливість ввімкнення або вимкнення відображення ваги ребер на графі для наглядності та аналізу. Якщо вага вимкнена – алгоритми будуть вважати, що вона дорівнює 1 для всіх ребер

Контекстне меню може взаємодіяти з окремими елементами графу. Воно динамічно змінюється відповідно до контексту та містить деякі опція для зміни елемента, на якому воно було викликано.

Якщо контекстне меню було викликано в пустому місці показується тільки один елемент - Створити нову ноду. Також в контекстних меню передбачені шорткати, за допомогою яких можна швидко обирати яку саме операцію необхідно зробити.

Якщо контекстне меню було викликано на ноді показується один пункт меню - Видалити ноду.

Відображення графів відбувається в основному вікні програми. Кожна нода представляється графічним елементом, а ребра відображаються лініями між відповідними нодами. Графічний інтерфейс повинен бути плавним, зручним у використанні та забезпечувати достатню простоту для розуміння структури графу.

Графічний інтерфейс дозволяє користувачеві динамічно змінювати вигляд графу.

Для реалізації графічного інтерфейсу буде використано HTML та CSS.

Графічний інтерфейс розробляється з урахуванням принципів зручності використання. Це включає в себе інтуїтивний дизайн елементів управління, легкий доступ до основних функцій та зручне розташування елементів інтерфейсу.

Застосунок повинен бути сумісний з різними операційними системами (Windows, Linux, MacOS) та різними браузерами (Chrome, Firefox, Opera). Це

забезпечить універсальність програмного забезпечення та розширить коло його користувачів.

2.4 Розробка структури системи

Структурна модель системи, показана на рисунку 2.1, визначає основні компоненти та їх взаємозв'язки.

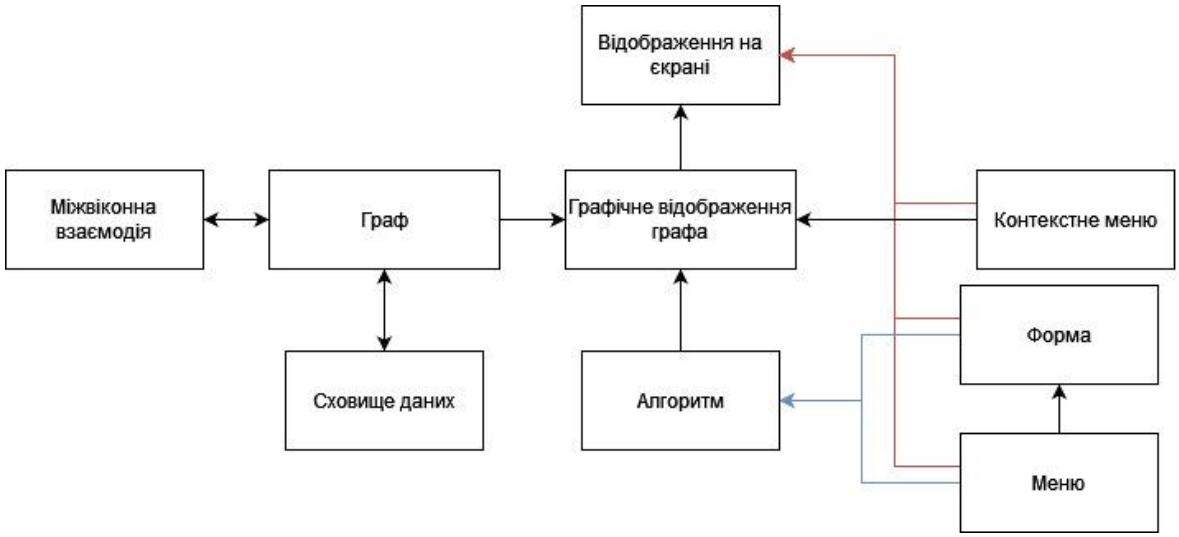

Рисунок 2.1 – Структурна модель застосунку

Система має основні компоненти, які представленні на таблиці 2.1:

Таблиця 2.1 – Опис компонентів системи

Назва компонента	Опис компонента
Відображення графа	Це основний компонент системи
Контектне меню	Потрібне для створення та видалення нод
Меню	Потрібне для відкриття форми та запуску алгоритмів. Основна точка взаємодії з користувачем
Форма для запуску алгоритму	Потрібна для запуску алгоритму з деякими вхідними даними

Продовження таблиці 2.1

Алгоритм	Алгоритм який виконується вважається окремим компонентом
Сховище даних	Потрібне для збереження стану застосунку, такого як внутрішній зсув та пресети
Міжвіконна взаємодія	Окремий складний компонент

Центральним елементом є компонент відображення графа, що виступає як основний модуль системи. Він відповідає за графічне представлення структури графа на екрані, забезпечуючи зручну взаємодію з користувачем та аналіз графічної інформації.

Контекстне меню відповідає за можливість створення та видалення вузлів графа. Цей компонент дозволяє користувачеві динамічно редагувати структуру графа, адаптуючи її під конкретні вимоги та завдання.

Меню, яке є основною точкою взаємодії з користувачем, відповідає за відкриття форми та запуск алгоритмів. Це важливий компонент, який забезпечує зручний доступ до основних функцій системи.

Форма для запуску алгоритмів використовується для введення вхідних даних та запуску алгоритму. Цей компонент дозволяє користувачеві взаємодіяти з алгоритмуми, вказуючи необхідні параметри та спостерігаючи за їхнім виконанням.

Алгоритм, що виконується, розглядається як окремий компонент системи. Цей елемент відповідає за реалізацію конкретного алгоритмічного підходу та взаємодію з іншими компонентами.

Сховище даних використовується для зберігання стану застосунку, таких як внутрішній зсув та пресети. Цей компонент забезпечує постійний доступ до поточного стану системи та можливість збереження його для подальшого використання.

Міжвіконна взаємодія розглядається як окремий складний компонент, що забезпечує гармонійну взаємодію різних вікон системи та підтримує цілісність користувацького інтерфейсу.

Система використовує об'єктно-орієнтований підхід для моделювання компонентів. Наприклад, інтерфейси та класи використовуються для представлення нод, ребер, GUI, контролера та інших об'єктів системи. Це дозволяє створювати гнучкі та розширювані об'єктні структури, сприяє повторному використанню коду та підтримці чистоти дизайну.

Компоненти системи взаємодіють динамічно, реагуючи на події та зміни, що виникають у процесі взаємодії з користувачем або виконання алгоритмів. Це забезпечує живий та відзивчивий характер системи.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Розробка UML діаграми класів

На рисунку 3.1 представлена UML діаграма класів.

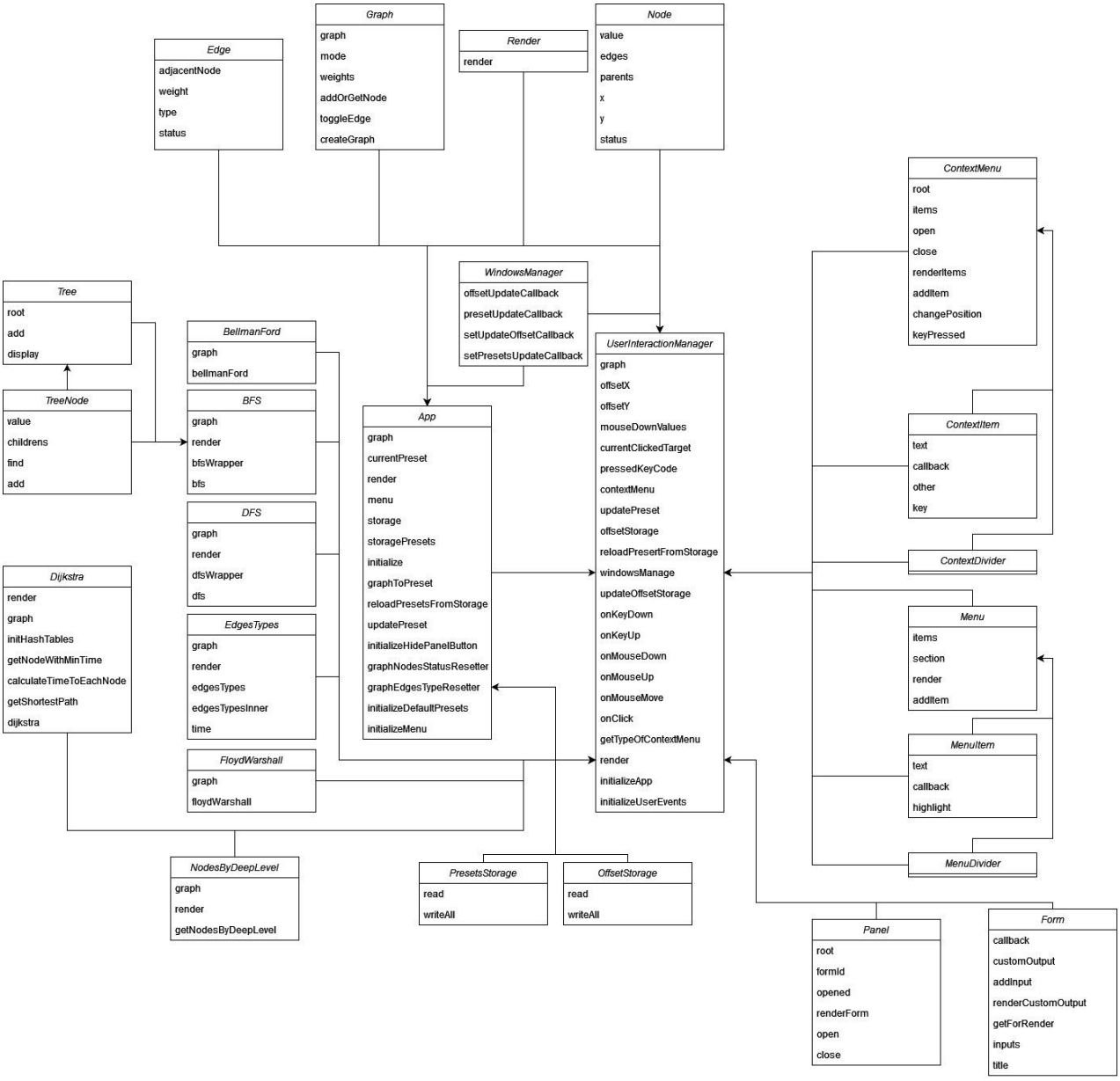

Рисунок 3.1 – UML діаграма класів

3.2 Розробка UML діаграми використання

UML діаграма використання визначає, які функції можуть бути використані користувачами. До основних входять створення та редагування графів, взаємодія з контекстним меню та основними функціями графічного інтерфейсу.

На рисунку 3.2 показана UML діаграма використання.

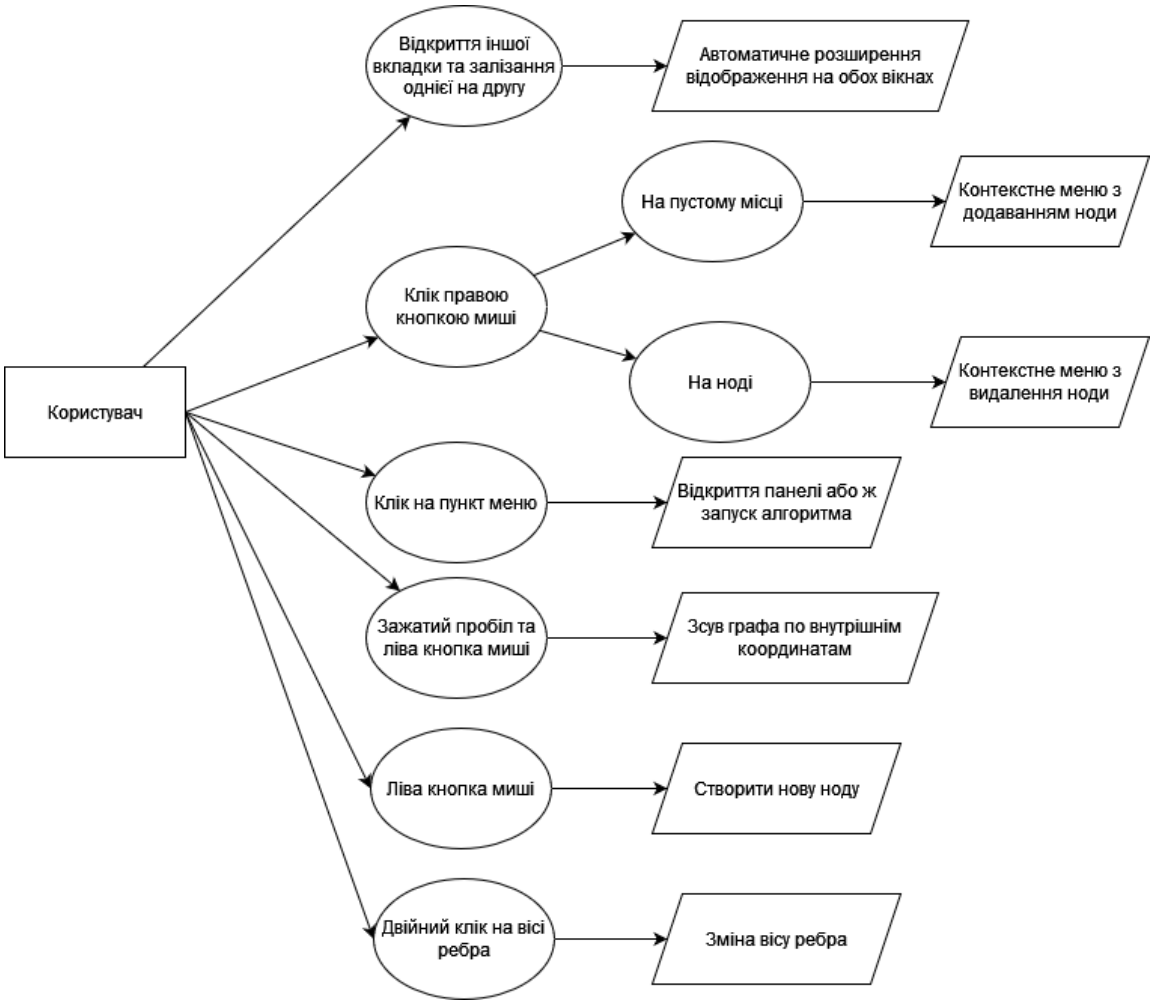

Рисунок 3.2 – UML діаграма використання

3.3 Опис класів програмного комплексу

Першим основним класом програми є точка графу – нода. Клас Node має наступні властивості:

– value – Значення ноди, яке відображається при візуалізації графа;

– edges – Множина граней які виходять з даної точки;

– parents – Map всіх батьківських нод. Батьківська нода – нода, з якої виходить грань через яку можна попасти в цю ноду;

– x – Положення на візуалізації по координаті X;

– y – Положення на візуалізації по координаті Y;

– status – Статус ноди. Потрібен для візуального відображення. Всього має 4 значення: default - біла, progress - жовта, done - зелена, passed - сіра.

Другим основним класом є грань – Edge. Грань має наступні властивості:

– adjacentNode – Нода, на яку посилається ця грань;

– weight – Вага грані. Може бути від’ємною;

– status – Статус грані. Це поле потрібно для реалізації можливості включати та виключати направлений режим. Це поле має два значення: standart – звичайна грань та no-direction – грань яка працює тільки коли виключений напрямлений режим показу;

– type – Тип грані. Потрібен для візуального виділення грані. Може мати наступні значення: default – звичайна грань, forward – рожева, cross – зелена та back – блакитна.

Третій клас на основі якого працює весь застосунок – клас Graph.

Цей клас має наступні властивості та методи:

– graph – Map, де в якості ключа число або строка, а в якості значення – нода;

– mode – Напрямлений або не напрямлений граф. Це поле потрібно для правильної роботи алгоритмів;

– weights – Чи має граф ваги, або ж ні. Це поле потрібно для правильної роботи алгоритмів;

– addOrGetNode() – Метод, який приймає данні графа, тобто Map, та значення ноди. Цей метод або ж повертає ноду, якщо вона вже існує, або ж створює її та повертає;

– toggleEdge() – Метод, який приймає три параметра: ноду, з якої буде йти грань, ноду в яку буде йти грань та вагу ребра. Цей метод створює грань, якщо між цими нодами немає зв’язку. Якщо між нодами є зв’язок, але тип цієї грані – no-direction – змінює його на standart. Якщо між нодами є зв’язок, але тип цієї грані standart – змінює на no-direction. Але якщо тип грані direction і грань яка йде з 2 ноди в першу є no-direction – обидві видаляються;

– createGraph() – Метод, який створює граф з готового пресету. Готовий пресет представляє собою масив нод та масив граней.

Розглянемо класи однієї з найважливіших функцій – класи зберігання даних

Для зберігання зсуву по внутрішнім координатам існує клас OffsetStorage. Цей клас має наступні властивості та методи:

– key – Ключ для зберігання даних в сховищі даних;

– writeAll – Метод, який приймає зсув та записує його до сховища;

– read – Метод, який читає данні зі сховища, валідує та повертає.

Для зберігання пресетів існує клас PresetStorage. Цей клас має наступні властивості та методи:

– key – Ключ для зберігання даних в сховищі даних;

– writeAll – Метод, який приймає великий об’єкт пресета та записує його до сховища даних;

– read – Метод, який читає данні зі сховища, валідує та повертає.

Однією з дуже важливих функцій програми є меню. Меню представляє собою головний клас для самої меню, а також два додаткових, один для елемента меню, другий для роздільника елементів.

Головний клас меню – Menu. Цей клас має наступні властивості та методи:

– items – Елементи меню які потрібно буде відображати;

– section – Елемент, в який будуть рендеритись;

– render() – Відрендерити меню в section;

– addItem() – Додати елемент меню до масиву елементів. Елементом меню є MenuItem або ж MenuDivider.

Для відображення напрямленого графа використовуються математичні формули. Вони використовуються саме для визначення позиції стрілки відносно ноди та її градус повороту для правильного відображення

Для цього використовуються дві математичні формули. Перша – формула знаходження кута між двома векторами, формула представлена під номером 1.

$$a = \operatorname{acos} \frac{\vec{a} * \vec{b}}{|\vec{a}| * |\vec{b}|} \quad (1)$$

Таким чином ми можемо знайти градус кута між проєкцією вектора a на вісь X. А тепер потрібно визначити позицію точки в прямокутних координатах, тобто X та Y. Для цього можна використати формулу полярних координат, яка представлена під номером 2.

$$\begin{cases} x = r\cos\varphi \\ y = r\sin\varphi \end{cases} \quad (2)$$

Клас для елемента меню – MenuItem, має наступні властивості:

- text – Текст елемента меню;
- callback() – Функція зворотнього виклику, яка буде викликана після кліку по цьому елементу;
- highlight – Властивість, яка потрібна для підсвічування активного елемента меню.

Клас для роздільника меню – MenuDivider. Цей клас не має ніяких властивостей та методів.

Оскільки деякі алгоритми потребують вхідних даних для запуску – потрібна деяка форма, куди можна записувати значення та місце, де ця форма, як і інші форми або ж кастомні елементи, можуть відображатись.

Для створення форми існує клас Form, який має наступні методи та властивості:

– inputs – Масив інпутів, які будуть відображатись;

– title – Назва форми;

– callback – Функція зворотнього виклику, яка буде викликана при submit форми;

– customOutput – Спеціальне поле нижче форми куди алгоритми зможуть виводити інформацію про результат виконання;

– addInput() – Додати новий елемент вводу;

– renderCustomOutput() – Відобразити для користувача спеціальне поле для виводу інформації алгоритму;

– getForRender() – Взяти готову форму, щоб потім відрендерити, наприклад, в Panel.

Користувачу зручно, якщо він клацає на пункт меню і в нього з'являється деяка панель справа де є форма для запуску алгоритму.

Для реалізації цієї панелі існує клас Panel, який має наступні властивості та методи:

– root – Елемент DOM дерева, куди буде відображатись форма;

– formId – Унікальний ідентифікатор форми, потрібний для правильної реалізації закриття панелі при повторному кліці на елемент меню з однаковим алгоритмом;

– renderForm() – Відобразити форму в панелі;

– open() – Показати панель користувачу на єкран;

– close() – Скрити панель з поля зору користувача;

– opened – Поле, яке необхідно для реалізації закриття/відкриття панелі.

Також в програмі реалізоване контекстне меню, яке потрібно для більш зручного створення та видалення нод – ContextMenu. Цей клас має наступні властивості та методи:

– root – Елемент DOM дерева, в який буде відображатись контекстне меню;

– items – Елементи контекстного меню, які будуть відображатись;

– open – Відкрити контекстне меню;

– close – Закрити контекстне меню;

– renderItems() – Відобразити елементи в контекстному меню;

– addItem() – Додати новий елемент;

– changePosition() – Змінити позицію контекстного меню;

– keyPressed() – Спрацьовує при натисканні клавіши на клавіатурі лише коли меню відкрите. Потрібно для роботи шорткатів.

Контестне меню має свої елементи, які реалізовані через клас ContextItem. Цей клас має наступні методи та властивості:

– text – Текст елемента меню;

– callback – Функція зворотнього виклику, яка спрацьовує при кліці на цей елемент, або після шорткату;

– other – Додаткові параметри;

– key – Клавіша для шорткату.

Також в контекстному меню можна розділяти елементи, які їх дуже багато або ж для зручності. Для цього існує клас ContextDivider який не має властивостей або ж методів.

Одним з головних класів програм є клас для рендеру графа на сторінку – Render. Цей клас має наступні методи:

– render() – Метод, який рендерить граф на сторінку. Головний метод класу;

– getNodeStatusForRender() – Приватний метод який визначає статус ноди для рендера;

– getRenderedCircles() – Приватний метод, який з нод робить кружки для рендера;

– getLinesForRender() – Приватний метод, який з граней робить відрізки та математично розраховує положення стрілок.

Також досить важливим класом є WindowsManager. Цей клас дозволяє реалізувати міжвіконну взаємодію та має наступні властивості та методи:

– offsetUpdateCallback – Приватна функція зворотнього виклику, яка викликається, якщо в сховищі даних offset щось змінюється;

– presetUpdateCallback – Приватна функція зворотнього виклику, яка викликається, якщо в сховищі даних preset щось змінюється;

– setUpdateOffsetCallback() – Встановлює функцію offsetUpdateCallback;

– setPresetsUpdateCallback() – Встановлює функцію presetUpdateCallback.

Головним класом застосунку є App. Цей клас має наступні властивості та методи:

– graph – Граф, на якому буде працювати весь застосунок;

– currentPreset – Пресет, який відображається зараз;

– render – фукнція для виклику рендера графа. Встановлюється з UserInteractionManager;

– menu – Меню, яке зараз відображається;

– storage – Сховище для пресетів;

– storagePresets – Локальні данні з сховища для пресетів;

– initialize() – Інітіалізація, в якій створюються UserInteractionManager, render, меню та кнопка для сховування меню;

– graphToPreset() – Перевести граф в пресет для зберігання в сховищі;

– reloadPresetsFromStorage() – Оновити локальні пресети зі сховища та перерендерити граф;

– updatePreset() – Оновити пресет та записати до сховища;

– initializeHidePanelButton() – Інітіалізація кнопки для сховування меню;

– graphNodesStatusResetter() – Поставити всім нодам статус в default;

– graphEdgesTypeResetter() – Поставити всім граням тип в standart;

– algorithmWrapper() – Спеціальний метод для полегшення використання елементів меню, який автоматично створює унікальний ідентифікатор кожного алгоритму, викликає graphNodesStatusResetter до виконання алгоритму та після завершення через деяку затримку;

– initializeDefaultPresets() – Коли в сховищі ще немає пресетів – їх потрібно інітіалізувати;

– initializeMenu() – Інітіалізація меню. Створення елементів на основі класів алгоритмів.

Також для взаємодії з користувачем існує окремий клас UserInteractionManager, який дуже сильно пов’язаний з App, але виконує окремі функції. Цей клас має наступні властивості та методи:

– graph – Граф, береться з App;

– offsetX – Зміщення по локальній координаті X;

– offsetY - Зміщення по локальній координаті Y;

– mouseDownValues – Властивість, необхідна для правильної роботи кліків миши. Також використовується для створення граней від однієї ноди до іншої;

– currentClickedTarget – – Властивість, необхідна для правильної роботи кліків миши;

– pressedKeyCode – Нажата клавіша;

– contextMenu – Контекстне меню;

– updatePreset() – Оновлення пресету після зміни графа – ця функція береться з класу App;

– offsetStorage – Сховище для локальних зміщень;

– reloadPresetsFromStorage() – Метод перезавантаження пресетів зі сховища. Використовується в мульти-віконній взаємодії;

– windowsManager – Властивість яка преставляє клас для мульти-віконних взаємодій;

– updateOffsetStorage() – Метод для оновлення сховища зсувів;

– onKeyDown() – Метод, який викликається при натисканні клавіши клавіатури;

– onKeyUp() – Метод, який викликається при відпусканні клавіши клавіатури;

– onMouseDown() – Метод, який викликається при натисканні на ліву кнопку миші;

– onMouseUp() – Метод, який викликається при підніманні лівої кнопки миші;

– onMouseMove() – Метод, який викликається при переміщенні миші;

– onClick() – Метод, який викликається при кліці на ліву кнопку миші, де клік – це опускання клавіши, а потім підіймання;

– getTypeOfContextMenu() – Метод, який повертає тип контекстного меню в залежності від елемента, на якому воно було викликано;

– onContextMenu() – Метод, який викликається при виклику контекстного меню;

– render() – Метод рендера графа;

– initializeApp() – Метод ініціалізації який викликається з класу App;

– initializeUserEvents() – Метод ініціалізації користувацьких ефектів, таких як клік, переміщення миші та інше.

Висновки

Під час виконання лабораторної роботи я вивчив можливості роботи з текстовим редактором. Зокрема, я дослідив пункти головного меню та навчився створювати нові документи та зберігати їх у різних форматах. Я також встановив параметри сторінки та параметри абзаців, змінював параметри шрифту та працював зі стилями. Крім того, я ознайомився з роботою з макросами, розташовував текст у декілька колонок, працював з маркованими та нумерованими списками, а також створював та редагував таблиці, формули та графічні примітиви. Ця робота дозволила мені отримати більше навичок у роботі з текстовим редактором та підвищити свою продуктивність при роботі з документами

Відповіді на запитання

3 Який пункт меню призначений для встановлення інтервалу між рядками тексту?

Для встановлення інтервалу між рядками тексту в більшості текстових редакторів використовується пункт меню "Інтервал між рядками" або подібний. Наприклад, у Microsoft Word це може бути знайдено в розділі "Висновки" (або "Розмітка сторінки") у групі "Абзац".

4 Як встановити відстань між абзацами 18 пт? Які засоби форматування абзаців розрізняють?

Для встановлення відстані між абзацами у 18 пунктів існує кілька способів залежно від редактора. У Microsoft Word це можна зробити вибравши вкладку "Висновки" (або "Розмітка сторінки"), потім "Абзац" і встановити значення в полі "Відступ перед" та "Відступ після". Щодо засобів форматування абзаців, їх можна розрізнити за допомогою таких параметрів, як вирівнювання тексту (ліве, праве, по центру, за ширині), відступи абзаців, інтервали між абзацами, номерація, маркування тощо.

5 Як для рядків тексту встановити відступ зліва 6 см і вирівняти текст за лівим краєм?

Для встановлення відступу зліва у 6 см і вирівнювання тексту за лівим краєм в більшості текстових редакторів використовується вкладка "Висновки" (або "Розмітка сторінки") і параметр "Відступи". Треба вибрати вкладку "Абзац" та вказати потрібні значення для відступу зліва.