Міністерство освіти і науки України
Національний технічний університет України “Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки Кафедра інформаційних систем та технологій

Лабораторна робота №2

Технології розроблення програмного забезпечення

Тема: «Основи проектування»

Виконала:
Студентка групи ІА-34
Сизоненко А.О
Перевірив:
Мягкий Михайло Юрійович

Київ-2025

Мета: Обрати зручну систему побудови UML-діаграм та навчитися будувати діаграми варіантів використання для системи що проєктується, розробляти сценарії варіантів використання та будувати діаграми класів предметної області.

Тема роботи: Текстовий редактор (strategy, command, observer, template method, flyweight, SOA)

Текстовий редактор повинен вміти розпізнавати текстові файли в будь-якій кодуванні, мати розширені функції редагування: макроси, сніппети, підказки, закладки, перехід на рядок / сторінку, підсвічування синтаксису (для однієї мови програмування або розмітки на розсуд студента).

Теоретичні відомості

UML (Unified Modeling Language) — це універсальна мова візуального моделювання для специфікації, візуалізації, проєктування та документування систем. Вона дозволяє будувати концептуальні, логічні та фізичні моделі складних систем, враховуючи об'єктно-орієнтований аналіз та проєктування (ООАП). Моделі поділяються на уявлення (views): статичні та динамічні, які деталізуються на різних рівнях абстракції (концептуальному, логічному, фізичному).

Діаграми UML — основний спосіб представлення моделей графічно. Основні види діаграм: варіанти використання (use case), класи (class), кооперації (collaboration), послідовності (sequence), станів (statechart), діяльності (activity), компонентів (component), розгортання (deployment).

(Use Case Diagram)

Діаграма варіантів використання показує вимоги до системи та її функціональні можливості. Вона визначає межі системи та є вихідною концептуальною моделлю, яка потім використовується для створення інших діаграм (наприклад, діаграми класів).

Основні елементи:

Актор (actor) — зовнішній суб'єкт, який взаємодіє з системою (людина, програма, пристрій).

Варіант використання (use case) — послідовність дій, що система виконує для актора, зображується еліпсом.

Відносини визначають взаємозв'язки між елементами:

Асоціація — звичайний зв'язок між актором і варіантом використання (суцільна лінія, може бути спрямована або ненаправлена).

Узагальнення (generalization) — успадкування властивостей одного елемента іншим.

Залежність (dependency) — зміна одного елемента впливає на інший;

Include — один варіант використання включає інший;

Extend — один варіант розширює базовий за певних умов.

Сценарії використання — покрокові текстові описи процесів взаємодії користувачів і системи. Містять передумови, постумови, основний потік подій, винятки та примітки.

Діаграми класів

Діаграми класів показують статичну структуру системи, включаючи класи, атрибути, операції та відносини між ними.

Атрибути мають видимість: public (+), package (~), protected (#), private (-).

Відносини:

1. **Асоціація** — зв’язок між класами, може мати назву та множинність;
2. **Узагальнення** — спадкування властивостей;
3. **Агрегація** — відношення частина-ціле (слабке);
4. **Композиція** — відношення частина-ціле (тісне, елементи не існують без цілого).

Діаграми класів використовуються для моделювання даних, архітектури, логіки програмних компонентів і навігації екранів.

Логічна структура бази даних

Бази даних мають **логічну** та **фізичну** моделі:

Фізична модель — організація даних у файлах на диску;

Логічна модель — структура таблиць, зв’язків, індексів для програмного використання.

Нормальні форми забезпечують мінімальну надмірність та уникнення логічних помилок:

1. 1НФ — кожен атрибут містить одне значення;
2. 2НФ — кожен неключовий атрибут повністю залежить від ключа;
3. 3НФ — відсутні транзитивні функціональні залежності неключових атрибутів від ключових;

Хід роботи

1. Діаграма варіантів використання. Use Case diagram

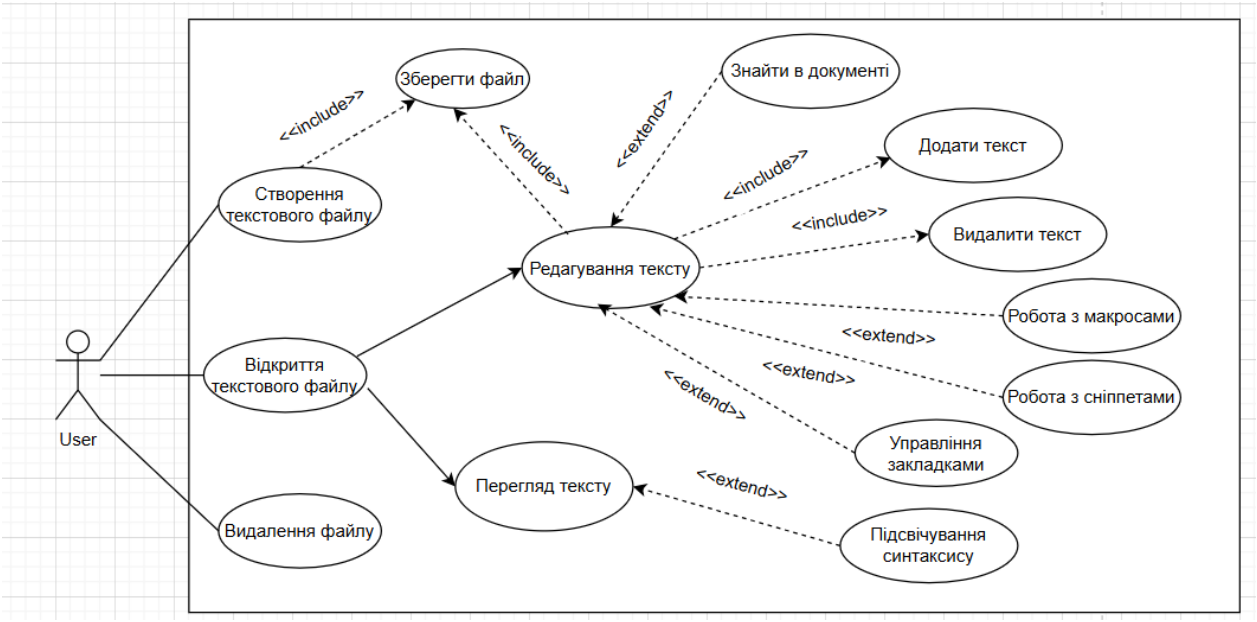


Рисунок 1 – Use Case діаграма

Основним актором є User, який може виконувати різні операції з текстовими файлами. Користувач має можливість створювати, відкривати, переглядати, редагувати та видаляти файли.

Основні варіанти використання системи:

- Створення текстового файлу – створення нового документа для подальшої роботи.
- Відкриття текстового файлу – доступ до вже існуючого файлу для редагування або перегляду.
- Редагування тексту – внесення змін до документа, що включає:
    - Збереження файлу (*include*)
    - Пошук в документі (*include*)
    - Додавання тексту (*include*)
    - Видалення тексту (*include*)
    - Робота з макросами (*extend*)
    - Робота з сніпетами (*extend*)
    - Управління закладками (*extend*)
    - Підсвічування синтаксису (*extend*)
- Перегляд тексту – перегляд вмісту документа без редагування, що має розшинення:
    - Підсвічування синтаксису (*extend*)
- Видалення файлу – повне видалення непотрібного текстового файлу.

2. Розробка діаграми класів для предметної області.

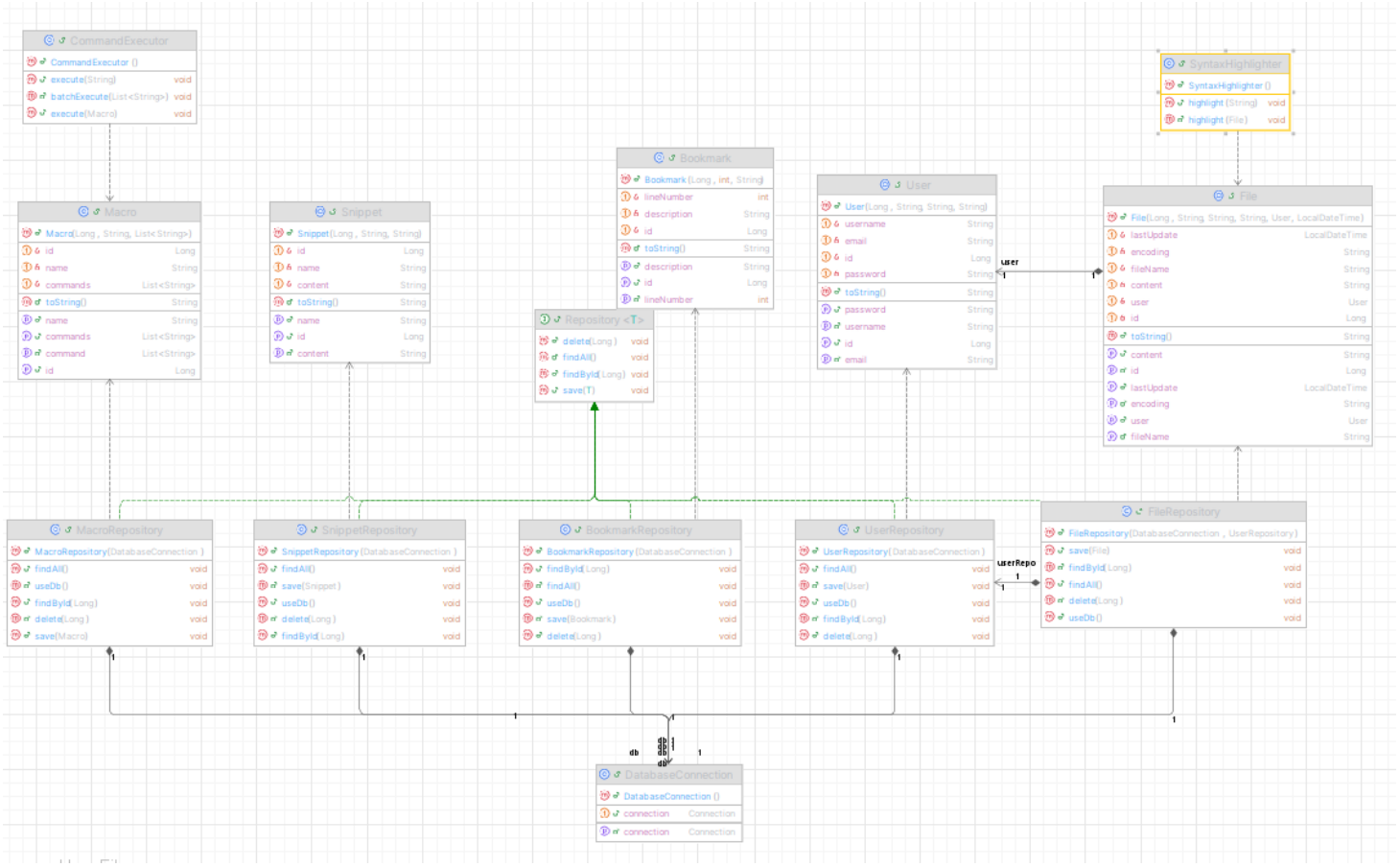


Рисунок 2 –діаграма класів

Основні компоненти діаграми:

1. Repository Pattern

Repository<T> – базовий інтерфейс для CRUD-операцій: Забезпечує розділення бізнес-логіки додатка і логіки роботи з базою даних.

save(T t) – для збереження об'єкта.
findById(Long id) – для пошуку об'єкта за ідентифікатором.
delete(Long id) – для видалення об'єкта.
findAll() – для отримання списку всіх об'єктів.

2. Репозиторії

FileRepisitory – забезпечує CRUD-операції для моделі File:

save(File file)
findById(Long id)
delete(Long id)
findAll()

UserRepisitory – забезпечує CRUD-операції для моделі User

save(File file)
findById(Long id)
delete(Long id)
findAll()

SnippetRepository – CRUD-операції для моделі Snippet:
    save(Snippet snippet)
    findById(Long id)
    delete(Long id)
    findAll()
MacroRepository – CRUD-операції для моделі Macro:
    save(Macro macro)
    findById(Long id)
    delete(Long id)
    findAll()
BookmarkRepository – CRUD-операції для моделі Bookmark:
    save(Bookmark bookmark)
    findById(Long id)
    delete(Long id)
    findAll()

3. Моделі

File – модель, яка представляє текстовий файл.
Поля:
    id (Long) – ідентифікатор файлу.
    fileName (String) – назва файлу.
    encoding (String) – кодування файлу.
    content (String) – вміст файлу.
    lastupdate – для часу останнього редагування.

Snippet – модель для опису сніппета.
Поля:
    id (Long) – ідентифікатор сніппета.
    name (String) – назва сніппета.
    content (String) – вміст сніппета.

Macro – модель для опису макросів.
Поля:
    id (Long) – ідентифікатор макросу.
    name (String) – назва макросу.
    commands (List<String>) – список команд макросу.

Bookmark – модель для опису закладок.
Поля:
    id (Long) – ідентифікатор закладки.
    lineNumber (int) – номер рядка.
    description (String) – опис закладки.

User – модель яка представляє користувача.

    id (Long) – ідентифікатор користувача.
    username (String) – ім’я користувача.
    Password (String) – пароль користувача (в хеші).

email (String) – email користувача.

4. Зв'язки між класами

File і Snippet – сніппети можуть бути використані у файлі для швидкої вставки шаблонного тексту.
File і Macro – макроси можуть бути застосовані для редагування тексту.
File і Bookmark – закладки дозволяють зберігати важливі позиції у тексті.
File і SyntaxHighlighter – кожен файл може мати підсвічування синтаксису.
File і User – кожен файл належить конкретному користувачу.
Macro і CommandExecutor – макрос виконує команди макросів.

5. Utility-класи

SyntaxHighlighter – клас для підсвічування синтаксису:
highlight(String language, String content): String – підсвічує текст для заданої мови програмування.
CommandExecutor – утиліти для виконання команд:
execute(String command): void – виконує одну команду.
batchExecute(List<String> commands): void – виконує список команд.

6. База даних та з'єднання

DatabaseConnection – клас для управління з'єднанням із базою даних:
getConnection() – встановлює та повертає з'єднання з базою даних.
closeConnection(Connection connection) – закриває активне з'єднання.

Загальна структура

1. Моделі – описують основні об’єкти текстового редактора, такі як файли, сніппети, макроси та закладки.
2. Репозиторії – забезпечують CRUD-операції для моделей, забезпечуючи розділення логіки доступу до даних і бізнес-логіки.
3. Utility-класи – допоміжні класи для виконання операцій з файлами, підсвічуванням синтаксису та виконанням команд.
4. Зв’язки – моделі інтегровані між собою, забезпечуючи повну функціональність текстового редактора.

3. Сценарії використання

Сценарій 1 “Редагування текстового файлу”
Сторони взаємодії: користувач, текстовий редактор
Передумови: користувач має доступ до редагування текстового файлу.
Основний сценарій подій:
1. Користувач відкриває або створює файл.
2. Вносить зміни до тексту (додає, видаляє, додає закладки)
3. Використовує додаткові функції (додає сніппети, використовує макроси або пошук по документу)

4. Зберігає внесені до файлу зміни.

Постумови: Файл містить відредагований текст.

Винятки:

Некоректний формат тексту: Система повідомляє про помилку форматування.

Помилка збереження: Якщо система не може зберегти файл, користувачу пропонується обрати інше місце або виправити проблему.

Сценарій 2 “Робота з макросами”

Сторони взаємодії: користувач, текстовий редактор
Передумови: користувач відкрив документ для редагування.
Основний сценарій:

1. Користувач запускає функцію «Робота з макросами».
2. Записує послідовність дій (наприклад, вставку шаблонного тексту).
3. Зберігає макрос.
4. Виконує макрос у документі.

Альтернативи: Користувач редагує/видаляє існуючий макрос.
Постумови: Збережено макроси, які можна застосовувати для автоматизації подальшої роботи.
Винятки:

Помилка під час запису макросу – дії не зберігаються.

Некоректне ім’я макросу – ім’я вже існує або містить заборонені символи.

Помилка виконання макросу – макрос містить недопустимі дії у документі.

Немає макросів для редагування/видалення – операція недоступна.

Сценарій 3 “Управління закладками”

Сторони взаємодії: користувач, текстовий редактор
Передумови: користувач відкрив документ для редагування.
Основний сценарій:

1. Користувач переходить до потрібного місця тексту.
2. Активує функцію додавання закладок через контекстне меню.
3. Додає закладку.
4. Користувач переходить до існуючої закладки.

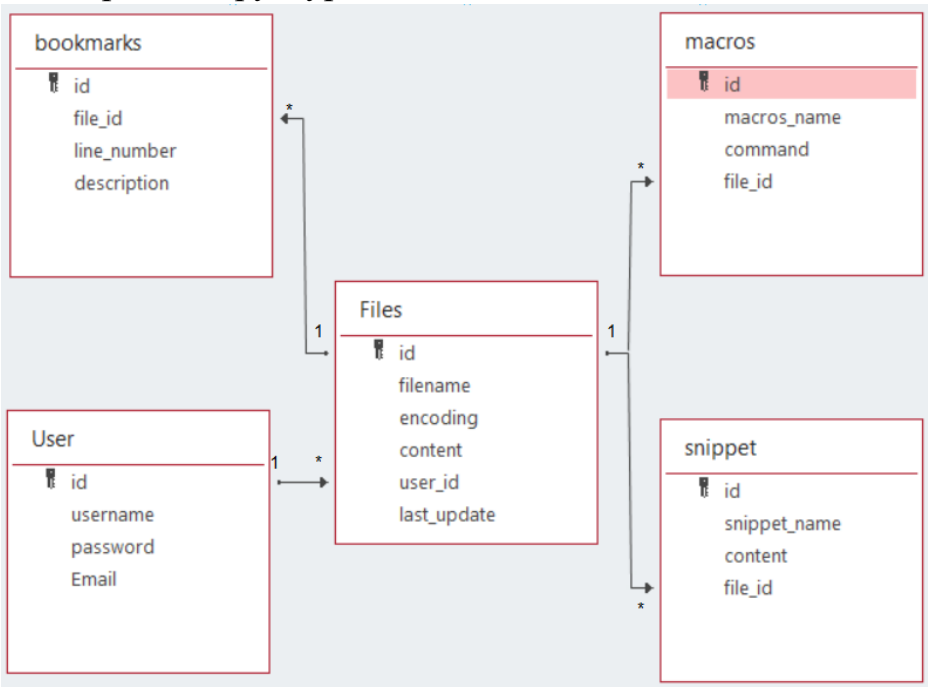
Альтернативи: Користувач редагує/видаляє існуючу закладку.
Постумови: Закладку збережено та можна використати для переходу до конкретного місця в документі.
Винятки:

Некоректне місце для закладки – обране місце тексту не допустиме для закладки.

Ім’я закладки вже існує – нова закладка не зберігається.

Закладка не знайдена – при спробі перейти до неіснуючої або видаленої закладки.

4. Розробка структури бази даних.



User

id — унікальний ідентифікатор користувача (PK).
username — ім’я користувача.
password — хеш пароля.
email — електронна пошта.

Files

id — унікальний ідентифікатор файлу (PK).
filename — назва файлу.
encoding — кодування файлу.
content — вміст файлу (текст).
user_id — зовнішній ключ на User.id (власник/автор).
last_update — дата/час останнього редагування.

bookmarks

id — унікальний ідентифікатор закладки (PK).
file_id — зовнішній ключ на Files.id (до якого файлу прив’язана).
line_number — номер рядка в файлі.
description — короткий опис або примітка.

snippet

id — унікальний ідентифікатор сніпета (PK).

snippet_name — назва сніпета.
content — текст або код сніпета.
file_id — зовнішній ключ на File.id.

macros

id — унікальний ідентифікатор макроса (PK).
macros_name — назва макроса.
command — команда або опис дії, що виконує макрос.
file_id — зовнішній ключ на File.id.

Опис зв'язків

User 1 → N
File — один користувач може мати багато файлів.
File 1 → N
bookmarks — у файлі може бути багато закладок.
File 1 → N
snippet — у файлі можуть бути багато сніпетів.
File 1 → N
macros — у файлі можуть бути багато макросів.

Висновок: У лабораторній роботі я опрацювала теоретичні відомості та створила діаграму прецедентів, діаграми класів і структуру бази даних для системи. Діаграма прецедентів відображають основні сценарії та взаємодію користувачів із системою, діаграми класів показують об'єкти та їхні зв'язки, а структура бази даних визначає таблиці та відносини для зберігання даних. Це забезпечує чітку основу для реалізації системи.

Відповіді на контрольні питання

1. Що таке UML?

UML (Unified Modeling Language) — уніфікована мова моделювання, яка використовується для візуалізації, опису, проєктування та документування програмних систем.

2. Що таке діаграма класів UML?

Діаграма класів — це структурна діаграма UML, яка показує класи системи, їх атрибути, методи та зв'язки між ними.

3. Які діаграми UML називають канонічними?

Канонічні (основні) діаграми UML — це ті, що входять до стандартного набору UML. Вони поділяються на:

Структурні: діаграма класів, об’єктів, компонентів, розгортання, пакетів.

Поведінкові: діаграма варіантів використання, діяльності, станів, послідовності, комунікації тощо.

4. Що таке діаграма варіантів використання?

Це діаграма, яка показує взаємодію користувачів (акторів) із системою через різні варіанти використання (use cases) — тобто, які функції системи доступні користувачу.

5. Що таке варіант використання?

Варіант використання — це опис певної функціональної можливості системи, яку виконує користувач (актор), щоб досягти своєї мети.

6. Які відношення можуть бути відображені на діаграмі використання?

Include (включення) — один варіант завжди виконує інший.

Extend (розширення) — додатковий варіант виконується лише за певних умов.

Generalization (узагальнення) — актор або варіант використання наслідує властивості іншого.

7. Що таке сценарій?

Сценарій — це конкретна послідовність дій, яка описує, як саме актор взаємодіє із системою для реалізації певного варіанту використання.

8. Що таке діаграма класів?

Діаграма класів відображає структуру системи у вигляді класів, їх атрибутів, методів і зв’язків між ними (асоціації, наслідування, залежності тощо).

9. Які зв’язки між класами ви знаєте?

Асоціація — зв’язок між об’єктами двох класів.

Агрегація — “ціле–частина”, але частини можуть існувати окремо.

Композиція — “ціле–частина”, але частини не можуть існувати без цілого.

Наслідування (узагальнення) — один клас успадковує властивості іншого.

Залежність — один клас використовує інший тимчасово.

10. Чим відрізняється композиція від агрегації?

Композиція — сильний зв’язок: якщо знищується ціле, знищуються і частини.

Агрегація — слабкий зв’язок: частини можуть існувати незалежно.

11. Чим відрізняється зв'язки типу агрегації від зв'язків композиції на діаграмах класів?

Агрегація позначається порожнім ромбом біля “цілого”.

Композиція позначається заповненим ромбом біля “цілого”.

12. Що являють собою нормальні форми баз даних?

Нормальні форми — це правила, які допомагають структурувати таблиці бази даних для усунення надлишкових даних і забезпечення цілісності.

1НФ — кожен атрибут містить одне значення;
2НФ — кожен неключовий атрибут повністю залежить від ключа;
3НФ — відсутні транзитивні функціональні залежності неключових атрибутів від ключових

13. Що таке фізична модель бази даних? Логічна?

Логічна модель — опис структури даних (таблиці, зв’язки, ключі) незалежно від конкретної СУБД.

Фізична модель — конкретна реалізація логічної моделі в певній СУБД (типи даних, індекси, схеми, оптимізація).

14. Який взаємозв'язок між таблицями БД та програмними класами?

Кожна таблиця бази даних зазвичай відповідає класу у програмі. Рядки таблиці — це об’єкти класу. Стовпці таблиці — це поля (атрибути) класу. Зв’язки між таблицями — відповідають асоціаціям між класами.