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»

Виконав:

Студент групи ІА-31

Калиновський В.О

Перевірив:

Мягкий Михайло Юрійович

Київ-2025

Тема: Основи проектування.

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.

Завдання

• Ознайомитись з короткими теоретичними відомостями.

• Проаналізувати тему та спроєктувати діаграму варіантів використання відповідно до обраної теми лабораторного циклу.

• Спроєктувати діаграму класів предметної області.

• Вибрати 3 варіанти використання та написати за ними сценарії використання.

• На основі спроєктованої діаграми класів предметної області розробити основні класи та структуру бази даних системи. Класи даних повинні реалізувати шаблон Repository для взаємодії з базою даних.

• Нарисувати діаграму класів для реалізованої частини системи.

• Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму варіантів використання відповідно, діаграму класів системи, вихідні коди класів системи, а також зображення структури бази даних.

Варіант

Графічний редактор (proxy, prototype, decorator, bridge, flyweight, SOA) повинен вміти створювати / редагувати растрові (або векторні на розсуд студента) зображення в 2-3 основних популярних 107 форматах (bmp, png, jpg), мати панель інструментів для створення графічних примітивів, вибору кольорів, нанесення тексту, додавання найпростіших візуальних ефектів (ч/б растр, інфрачервоний растр, 2-3 на вибір учня), роботи з шарами.

Теоретичні відомості

Вступ до мови UML

Мова UML є загальноцільовою мовою візуального моделювання, яка розроблена для специфікації, візуалізації, проєктування та документування компонентів програмного забезпечення, бізнес-процесів та інших систем. Мова UML є досить строгим та потужним засобом моделювання, який може бути ефективно використаний для побудови концептуальних, логічних та графічних 18 моделей складних систем різного цільового призначення. Ця мова увібрала в себе найкращі якості та досвід методів програмної інженерії, які з успіхом використовувалися протягом останніх років при моделюванні великих та

складних систем. Діаграма (diagram) – графічне уявлення сукупності елементів моделі у формі зв'язкового графа, вершинам і ребрам (дугам) якого приписується певна семантика. Нотація канонічних діаграм є основним засобом розробки моделей мовою UML.

У нотації мови UML визначено такі види діаграм:

• варіантів використання (use case diagram);

• класів (class diagram);

• кооперації (collaboration diagram);

• послідовності (sequence diagram);

• станів (statechart diagram);

• діяльності (activity diagram);

• компонентів (component diagram);

• розгортання (deployment diagram).

Хід роботи

1. **Use-case diagram**

- **Користувач (Actor)** — головний учасник системи, який взаємодіє з графічним редактором для створення, редагування та збереження зображень.
- **Графічний редактор** — програмне забезпечення, що дозволяє користувачу створювати, редагувати та зберігати растрові зображення.

Вхід до системи

- **Опис:** користувач авторизується або входить у систему для отримання доступу до функцій графічного редактора.
- **Результат:** користувач отримує можливість створювати, відкривати або видаляти зображення.

Створення растрового зображення

- **Опис:** користувач створює нове зображення для редагування.
- **Зв’язок:** включає варіант **«Редагування»** (<<include>>), оскільки після створення зображення користувач переходить до редагування.

Відкриття зображення

- **Опис:** користувач відкриває наявне зображення для подальшої роботи.
- **Зв’язок:** включає варіант **«Редагування»** (<<include>>).

Видалення зображення

- **Опис:** користувач може видалити непотрібне зображення із системи.
- **Результат:** вибране зображення видаляється із бібліотеки або пам’яті програми.

Редагування зображення

- **Опис:** основний варіант використання, який охоплює більшість дій користувача під час роботи з графічним файлом.
- **Зв’язки:**
    - Включає (<<include>>) варіант **«Зберегти зображення»**.
    - Розширюється (<<extend>>) варіантами:
        - **Вибрати колір**
        - **Вибрати фігуру**
        - **Додати текст**
        - **Вибрати візуальний ефект**
        - **Додати шар**

Вибрати колір

- **Опис:** користувач обирає колір для малювання, заповнення або редагування елементів.
- **Тип зв’язку:** <<extend>> від варіанту «Редагування».

Вибрати фігуру

- **Опис:** користувач обирає геометричну фігуру (лінія, коло, прямокутник тощо) для малювання.
- **Тип зв’язку:** <<extend>> від варіанту «Редагування».

Додати текст

- **Опис:** користувач може вставити текстові елементи у зображення.
- **Тип зв’язку:** <<extend>> від варіанту «Редагування».

Вибрати візуальний ефект

- **Опис:** користувач застосовує фільтри чи ефекти до зображення (розмиття, тіні, яскравість тощо).
- **Тип зв’язку:** <<extend>> від варіанту «Редагування».

Додати шар

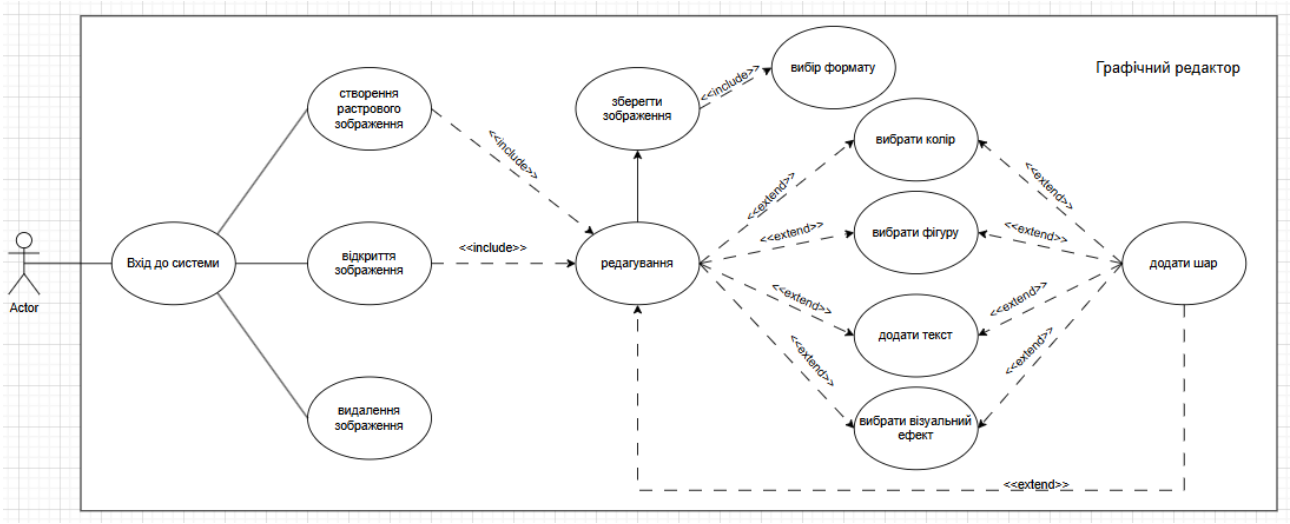
- **Опис:** користувач створює новий шар для розділення елементів зображення.
- **Тип зв’язку:** <<extend>> від варіанту «Редагування».

Зберегти зображення

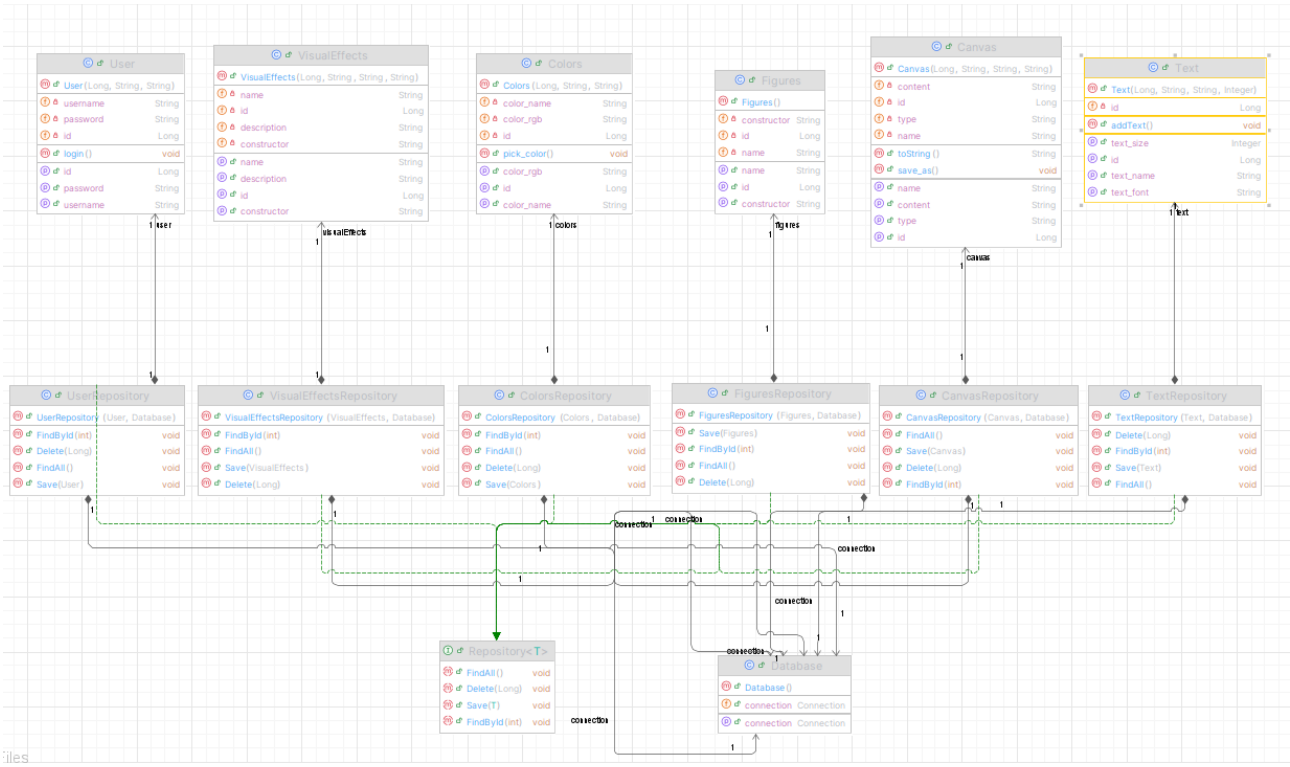
- **Опис:** користувач зберігає створене або відредаговане зображення.
- **Зв’язки:** включає (<<include>>) варіант **«Вибір формату»**.

Вибір формату

- **Опис:** користувач обирає формат файлу для збереження зображення (наприклад, PNG, JPEG, BMP).
- **Тип зв’язку:** <<include>> у варіант «Зберегти зображення».



2. Діаграма класів



1. Клас User (Користувач)(містить змінні username, password, id)
    - Призначення: зберігає інформацію про користувача, який працює в програмі.
    - Основні завдання:
        - зберігання налаштувань користувача (обрані кольори, попередні проєкти, інтерфейсні параметри);

        - ініціація основних дій (створення нового файлу, відкриття, збереження);
        - взаємодія з іншими модулями через інтерфейс користувача.
2. Клас Canvas (Полотно) (містить змінні id, type, content, name)
    - Призначення: головний робочий простір, на якому користувач створює або редагує зображення.
    - Основні завдання:
        - відображення об’єктів (фігури, текст, шари);
        - реалізація операцій малювання, стирання, оновлення полотна;
        - взаємодія з класами Figures, Text, Colors, VisualEffects.
3. Клас Colors (Кольори) (містить змінні id, color_rgb, color_name)
    - Призначення: відповідає за вибір, збереження та зміну кольорів.
    - Основні завдання:
        - зберігання поточного кольору пензля й фону;
        - робота з палітрою кольорів;
        - надання кольору іншим об’єктам, наприклад фігурам або тексту.
4. Клас Figures (Фігури) (містить змінні id, name, constructor)
    - Призначення: керує створенням і відображенням графічних примітивів.
    - Основні завдання:
        - підтримка різних типів фігур (лінії, кола, прямокутники, полігони тощо);
        - зберігання параметрів (координати, розмір, колір);
        - редагування або видалення намальованих елементів.
5. Клас VisualEffects (Візуальні ефекти) (містить змінні id, name, description, constructor)
    - Призначення: застосовує різні ефекти до всього зображення або окремих шарів.
    - Основні завдання:

- реалізація ефектів (чорно-білий, інфрачервоний, змінення яскравості, контрастності);
- забезпечення попереднього перегляду результату;
- взаємодія з полотном (Canvas) і шарами.

6. Клас Text (Текст) (містить змінні id, text_name, text_size, text_font)
    - Призначення: відповідає за роботу з текстовими елементами на зображенні.
    - Основні завдання:
        - додавання, редагування та переміщення тексту;
        - налаштування шрифту, розміру, кольору;
        - інтеграція тексту з іншими графічними елементами.

Системні класи (робота з даними)

1. Інтерфейс Repository<T>
    - Призначення: визначає стандартні методи для взаємодії з даними будь-якого типу.
    - Типові методи:
        - findAll() — отримання всіх записів;
        - findById(int id) — пошук об’єкта за ідентифікатором;
        - save(T entity) — збереження або оновлення даних;
        - delete(int id) — видалення запису.
    - Завдяки інтерфейсу реалізується єдиний підхід до збереження інформації для всіх сутностей (користувачів, кольорів, фігур тощо).
2. Клас UserRepository
    - Призначення: реалізує інтерфейс Repository для збереження даних користувача.
    - Основні завдання:
        - збереження інформації про поточного користувача (налаштування, історія, останні файли);
        - отримання списку користувачів або їхніх профілів;

        - взаємодія з базою даних через клас Database.
3. Клас ColorsRepository
    - Призначення: відповідає за роботу з кольоровими палітрами та налаштуваннями кольорів.
    - Основні завдання:
        - збереження обраних користувачем кольорів;
        - отримання стандартних палітр;
        - синхронізація кольорів між сесіями роботи користувача.
4. Клас FiguresRepository
    - Призначення: керує збереженням і відновленням даних про фігури на полотні.
    - Основні завдання:
        - збереження координат, розмірів, кольорів і типу фігури;
        - відновлення збереженого стану після перезапуску програми;
        - зв’язок із класом Canvas для виведення фігур.
5. Клас VisualEffectsRepository
    - Призначення: зберігає інформацію про ефекти, застосовані до зображення.
    - Основні завдання:
        - фіксування параметрів ефектів (тип, інтенсивність, порядок застосування);
        - забезпечення можливості скасування або повторного застосування ефектів.

6. Клас TextRepository
    - Призначення: відповідає за збереження текстових елементів, доданих користувачем.
    - Основні завдання:
        - зберігання тексту, його позиції, шрифту, кольору;

- відновлення текстових шарів при відкритті збереженого проєкту.

7. Клас Database (База даних)
   - Призначення: забезпечує з’єднання з базою даних та виконання запитів.
   - Основні завдання:
     - створення, відкриття та закриття підключення;
     - виконання SQL-запитів до таблиць користувачів, кольорів, фігур тощо;
     - централізоване керування збереженням усіх даних програми.

3. Сценарії використання

“Додавання фігури та збереження файлу”

Передумови: Користувач має доступ до редактора.

Постумови: У разі успішного виконання, користувач додає необхідну фігуру та зберігає файл у вибраному форматі.

Взаємодіючі сторони: Користувач, редактор.

Короткий опис: Цей варіант використання визначає процес створення та збереження зображення, що намальоване користувачем.

Основний потік подій.

1. Користувач створює зображення

2. Користувач зберігає створене ним зображення в одному з трьох доступних форматів.

Винятки

Виняток №1: Неправильне ім'я/пароль. Якщо під час виконання Основного потоку виявиться, що користувач ввів неправильне ім'я та пароль, система виводить повідомлення про помилку. Користувач може повернутися до початку основного потоку або відмовитись від входу в систему, при цьому виконання варіанта використання завершується.

Примітки відсутні.

“Вхід користувача до системи графічного редактора”

Передумови: Немає.

Постумови: У разі успішного виконання, користувач входить до системи. У протилежному випадку стан системи не змінюється.

Взаємодіючі сторони: Студент, редактор.

Короткий опис: Цей варіант використання визначає вхід користувача до системи редактора.

Основний потік подій.

1. Система запитує ім'я користувача та пароль.

2. Користувач вводить ім'я та пароль.

3. Система перевіряє ім'я та пароль, після чого відкривається доступ до системи. Якщо ім'я та пароль неправильні, Виняток №1.

Винятки

Виняток №1: Неправильне ім'я/пароль. Якщо під час виконання Основного потоку виявиться, що користувач ввів неправильне ім'я та пароль, система виводить повідомлення про помилку. Користувач може повернутися до початку основного потоку або відмовитись від входу в систему, при цьому виконання варіанта використання завершується.

Примітки Відсутні.

“Додавання шару та деталей на нього”

Передумови: Користувач здійснив вхід до системи та створив або відкрив зображення.

Постумови: У разі успішного виконання, користувач додає шар. У протилежному випадку стан системи не змінюється.

Взаємодіючі сторони: користувач, редактор.

Короткий опис: Цей варіант використання визначає процес додавання шару користувачем

Основний потік подій.

1. Користувач у відкритому зображенні додає шар.

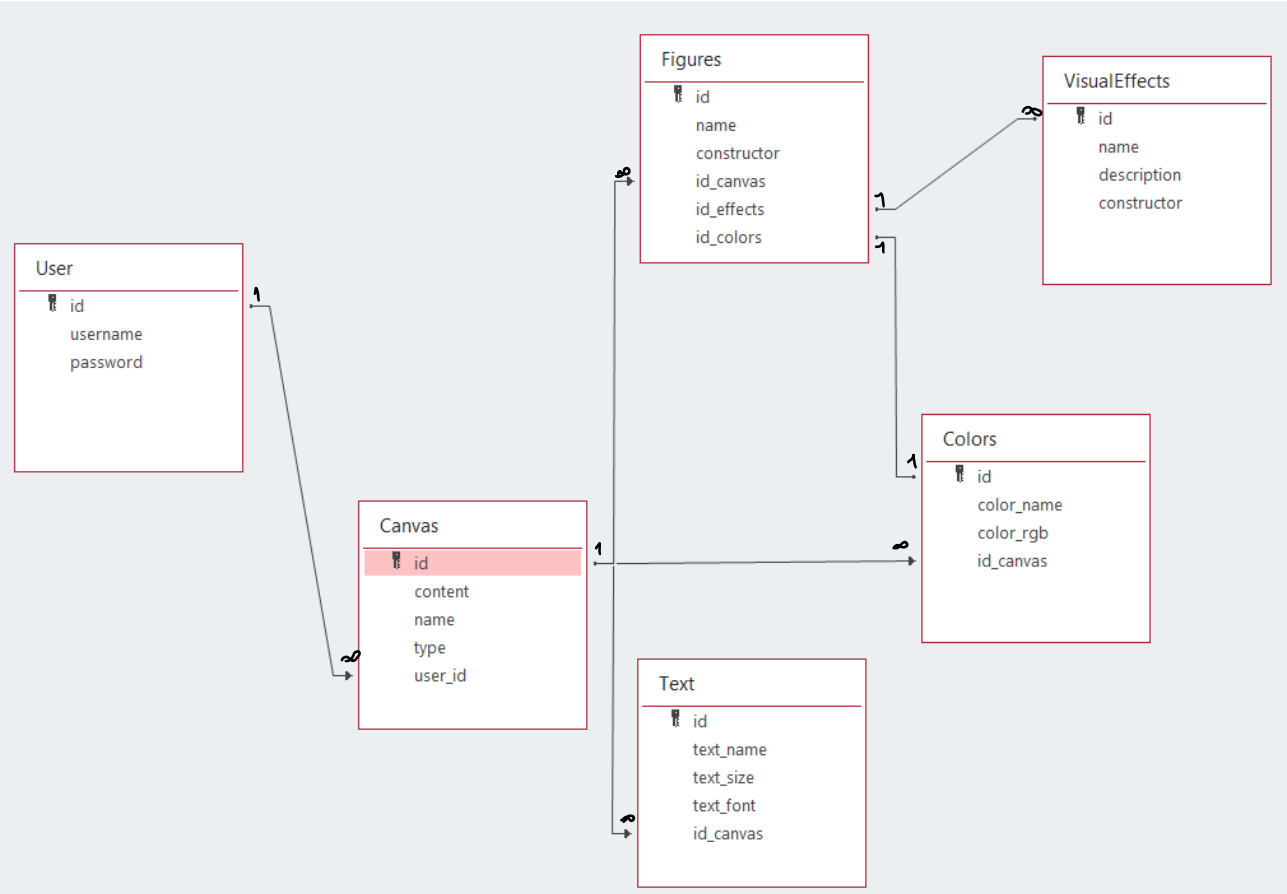
2. Користувач на відкритому шарі наносить потрібні йому деталі.

Винятки

Виняток №1: Неправильне ім'я/пароль. Якщо під час виконання Основного потоку виявиться, що користувач ввів неправильне ім'я та пароль, система виводить повідомлення про помилку. Користувач може повернутися до початку основного потоку або відмовитись від входу в систему, при цьому виконання варіанта використання завершується.

Примітки Відсутні.

4. Структура бази даних системи



Контрольні запитання

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

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

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

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

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

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

9. Основні типи зв’язків між класами:

Асоціація (association) — зв’язок між об’єктами класів;

Агрегація (aggregation) — «ціле-частина», але частини можуть існувати окремо;

Композиція (composition) — «ціле-частина», але частини не існують без цілого;

Наслідування (generalization) — зв’язок між батьківським і дочірнім класами;

Залежність (dependency) — один клас використовує інший тимчасово.

include — один варіант обов’язково включає інший;

extend — один варіант може розширювати інший у певних умовах;

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

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

11. Агрегація позначається порожнім ромбом біля “цілого”.

Композиція позначається заповненим ромбом біля “цілого”.

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

13. 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. 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.

Висновок:

Я обрав зручну систему побудови UML-діаграм та побудував діаграми варіантів використання для системи графічного редактора, розробив сценарії варіантів використання та збудував діаграми класів предметної області.