Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 2

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Основи проектування»

«Web-browser»

Виконала:
Студент групи - ІА-32
Самойленко С. Д.

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.

Теоретичні відомості

UML та його діаграми UML (Unified Modeling Language) - уніфікована мова візуального моделювання, що використовується для аналізу, проєктування та документування програмних систем. UML дозволяє описувати систему на різних рівнях: від концептуального до фізичного. Основні діаграми UML

- Діаграма варіантів використання (Use Case Diagram) - показує вимоги до системи та взаємодію користувачів із нею.
- Діаграма класів (Class Diagram) - описує статичну структуру системи: класи, їх атрибути, методи та зв’язки.

Діаграма варіантів використання

Діаграма use case відображає функціональність системи з точки зору користувача. Основні елементи:

- Актори (Actor) - користувачі або зовнішні системи.
- Варіанти використання (Use Case) - дії або послуги, які система надає актору (наприклад: вхід, перегляд даних, створення транзакції).

Типи відносин:

- Асоціація - прямий зв’язок актора з варіантом використання.
- Include - один сценарій завжди включає інший (обов’язковий).
- Extend - сценарій може бути розширений додатковим (необов’язковим).
- Узагальнення - спадкування ролей або функціоналу.

Для уточнення роботи системи складаються сценарії використання (use case scenarios), які описують:

- передумови та постумови;
- учасників;
- короткий опис;
- основний перебіг подій;
- винятки.

Діаграма класів

Діаграма класів показує структуру системи: класи, їх атрибути, методи та зв’язки між ними.

Клас містить:

- назву;
- атрибути (дані);
- методи (операції).

Види зв’язків:

- Асоціація - загальний зв’язок між класами.
- Узагальнення (успадкування) - зв’язок між батьківським і дочірнім класом.
- Агрегація - відношення «ціле-частина», де частини можуть існувати окремо.
- Композиція - сильне відношення «ціле-частина», де частини не існують без цілого.

Логічна структура бази даних

Проєктування бази даних часто виконується на основі діаграми класів. Виділяють:

- Фізичну модель - організація файлів і способів зберігання.
- Логічну модель - таблиці, атрибути, ключі, зв’язки.

Щоб уникнути надмірності даних застосовують нормалізацію:

- 1НФ - кожен атрибут має лише одне атомарне значення.
- 2НФ - усі неключові атрибути залежать від усього первинного ключа.
- 3НФ - немає транзитивних залежностей (атрибутів, що залежать від інших неключових атрибутів).
- НФ Бойса-Кодда (BCNF) - посилена форма 3НФ, кожна залежність визначається ключем.

Хід роботи

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.

1. **Розробка діаграми варіантів використання**

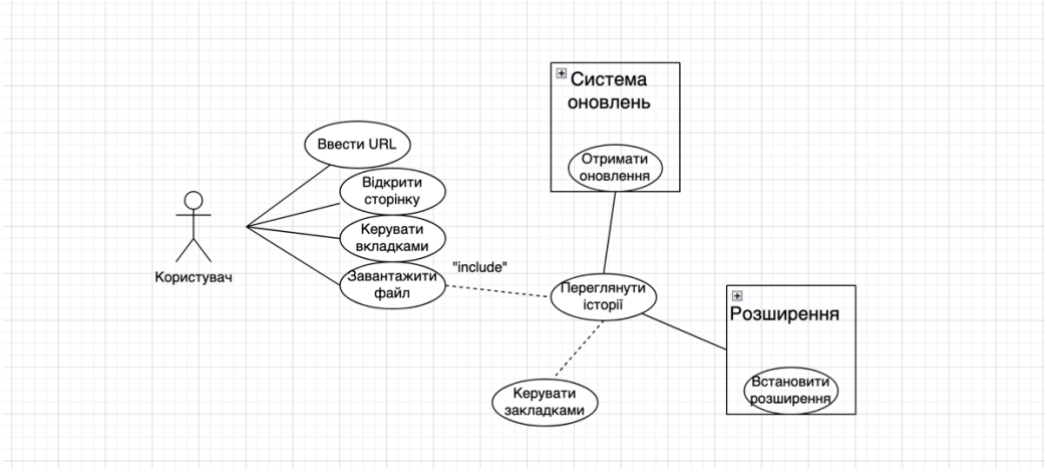

Рисунок 1 – Use case діаграма

2. Сценарії використання

Сценарій 1. Відкриття веб-сторінки

Передумови:

Користувач має підключення до Інтернету.

Постумови:

Відображено HTML-сторінку.

Учасники:

Користувач, BrowserEngine, NetworkModule, Renderer.

Основний перебіг:

1. Користувач вводить URL.
2. BrowserEngine передає URL у NetworkModule.
3. NetworkModule надсилає HTTP-запит.
4. Отримує HTML-відповідь.
5. Renderer парсить HTML/CSS.
6. Сторінка відображається користувачу.

Винятки:

- Сервер недоступний → Показати “No Internet”.
- Невірний URL → Показати помилку.

Сценарій 2. Завантаження файлу

Передумови:

Браузер має дозвіл на збереження файлів.

Постумови:

Файл збережено у «Downloads».

Учасники:

Користувач, DownloadManager, NetworkModule.

Основний перебіг:

1. Користувач натискає на файл.
2. NetworkModule надсилає запит.
3. Відповідь надходить у DownloadManager.
4. Файл записується блоками на диск.
5. Статус показується у списку завантажень.

Винятки:

- Недостатньо місця → Download failed.
- Втрата мережі → Pause / Retry.

Сценарій 3. Перегляд історії

Передумови:

База історії містить дані.

Постумови:

Користувач бачить список відвіданих сторінок.

Учасники:

Користувач, HistoryRepository, BrowserEngine.

Основний перебіг:

1. Користувач відкриває “History”.
2. BrowserEngine звертається до HistoryRepository.
3. Репозиторій повертає список записів.
4. Дані відображаються в UI.

Винятки:

- База пошкоджена → показати часткові дані.

3. Діаграма класів

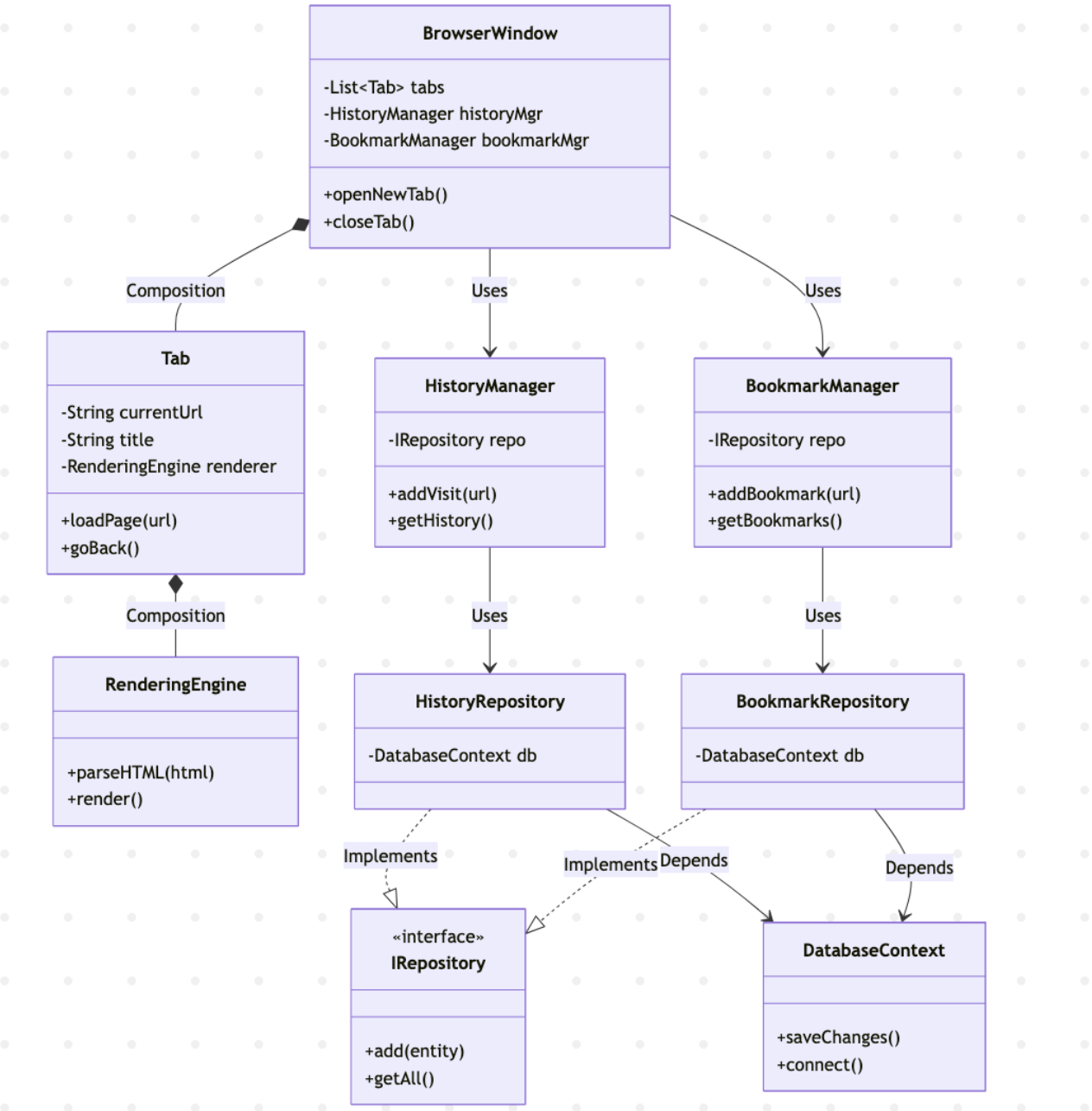

Рисунок 2 – Діаграма класів системи

1. **BrowserWindow (Головне вікно)**

- Містить: Список вкладок (Tab), Менеджер закладок (BookmarkManager), Менеджер історії (HistoryManager).
- Методи: openNewTab(), closeTab().

2. **Tab (Вкладка)**

- Зв'язок: Композиція з BrowserWindow (вкладка не існує без вікна).
- Поля: currentUrl, title.
- Методи: loadPage(url), refresh(), goBack().

3. **RenderingEngine (Двигун відображення)**

- Зв'язок: Композиція з Tab (кожна вкладка має свій рушій для малювання сторінки).
- Методи: parseHTML(), renderLayout().

4. **NetworkClient (Мережевий модуль)**

- Зв'язок: Асоціація з Tab. Вкладка використовує його для завантаження даних.
- Методи: sendGetRequest(url).

5. **HistoryManager**

- Зв'язок: Асоціація з BrowserWindow.
- Роль: Керує списком відвіданих сторінок.
- Використовує: HistoryRepository.

6. **BookmarkManager**

- Зв'язок: Асоціація з BrowserWindow.
- Роль: Керує збереженими закладками.
- Використовує: BookmarkRepository.

7. **IRepository<T> (Інтерфейс)**

- Загальний інтерфейс для роботи з БД (як у тебе в завданні).

8. **HistoryRepository та BookmarkRepository**

- Зв'язок: Реалізація інтерфейсу IRepository.
- Зв'язок: Залежність від DatabaseContext.

5. **Основні класи та структура бази даних системи:**

id	user_id	url	title	visit_time
1	1	https://kpi.ua	КПІ ім. Ігоря Сікорського	2025-11-21 10:05:00
2	1	https://google.com	Google Пошук	2025-11-21 10:10:00

Рисунок 3 – Атрибути та їх типи даних для таблиці History

id	user_id	url	title	folder_name
1	1	https://campus.kpi.ua	Електронний кампус	Навчання
2	1	https://github.com	GitHub	Робота

Рисунок 4 – Атрибути та їх типи даних для таблиці Bookmarks

id	user_id	file_...	file_path	file_size	status
1	1	lab2_...	/downloads/lab2...	102400	Completed
2	1	instal...	/downloads/temp...	5242880	Failed

Рисунок 5 – Атрибути та їх типи даних для таблиці Downloads

id	username	password	email	created_at
1	student_ia32	pass123	student@kpi.ua	2025-11-21 10:00:00

Рисунок 6 – Атрибути та їх типи даних для таблиці Users

id	user_id	theme	homepage	language
1	1	Dark	https://google.com	UKR

Рисунок 7 – Атрибути та їх типи даних для таблиці Settings

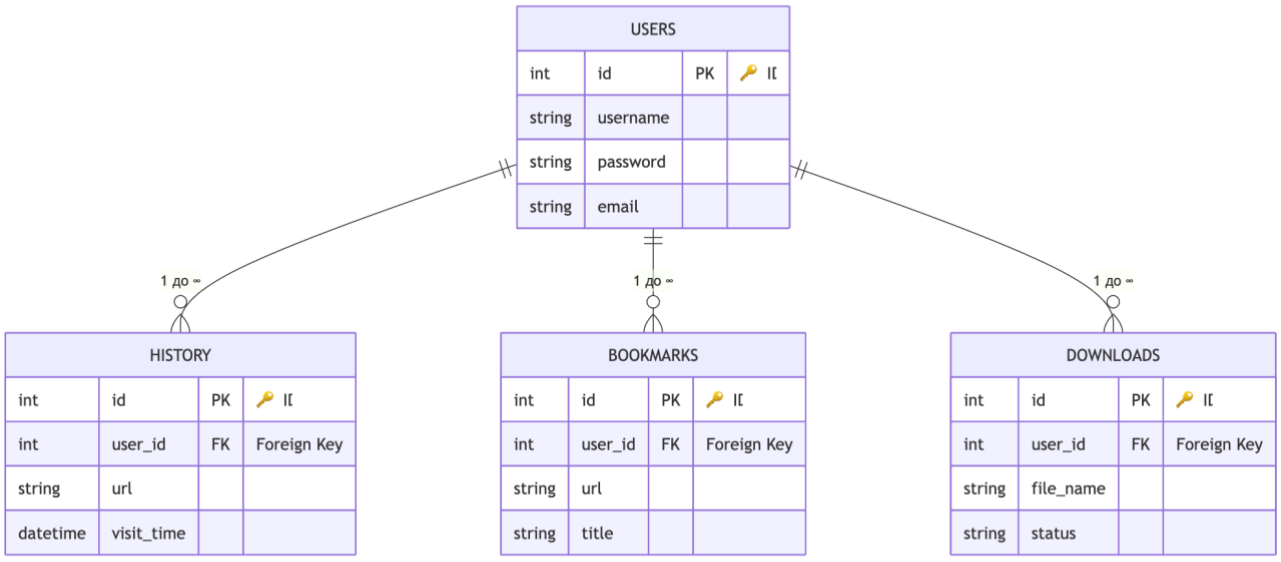

Рисунок 8 – Структура бази даних для предметної області

6. Вихідні коди класів системи

```
// 1. Універсальний інтерфейс
public interface IRepository<T> {
    void add(T entity);
    T getById(int id);
    List<T> getAll();
    void delete(int id);
}

// 2. Сутності (Entity Classes)
public class HistoryItem {
    private int id;
    private String url;
    private String title;
    private Date visitTime;

    public HistoryItem(String url, String title) {
        this.url = url;
        this.title = title;
        this.visitTime = new Date();
    }
}

public class Bookmark {
    private int id;
    private String url;
    private String title;

    public Bookmark(String url, String title) {
        this.url = url;
        this.title = title;
    }
}

// 3. Контекст бази даних (Імітація)
public class DatabaseContext {
    private List<HistoryItem> historyTable = new ArrayList<>();
    private List<Bookmark> bookmarksTable = new ArrayList<>();

    public List<HistoryItem> getHistory() { return historyTable; }
    public List<Bookmark> getBookmarks() { return bookmarksTable; }

    public void saveChanges() {
        System.out.println("Дані збережено в локальну БД.");
    }
}

// 4. Реалізація репозиторіїв
public class HistoryRepository implements IRepository<HistoryItem> {
```

```
    private DatabaseContext context;

    public HistoryRepository(DatabaseContext context) {
        this.context = context;
    }

    @Override
    public void add(HistoryItem entity) {
        context.getHistory().add(entity);
        context.saveChanges();
    }

    @Override
    public List<HistoryItem> getAll() {
        return context.getHistory();
    }

    @Override
    public HistoryItem getById(int id) {
        return context.getHistory().stream().findFirst().orElse(null);
    }

    @Override
    public void delete(int id) { /* Implementation */ }
}

// 5. Логіка Браузера

public class RenderingEngine {
    public void render(String htmlContent) {
        System.out.println("Відображення HTML: " + htmlContent);
    }
}

public class Tab {
    private String currentUrl;
    private String title;
    private RenderingEngine renderer;
    private HistoryRepository historyRepo;

    public Tab(HistoryRepository repo) {
        this.renderer = new RenderingEngine();
        this.historyRepo = repo;
    }

    public void navigateTo(String url) {
        this.currentUrl = url;
        this.title = "Page: " + url; // Спрощено
        System.out.println("Завантаження " + url + "...");
```

```
        renderer.render("<html><body>Content of " + url + "</body></html>");
        HistoryItem visit = new HistoryItem(url, this.title);
        historyRepo.add(visit);
    }
}

public class BrowserWindow {
    private List<Tab> tabs = new ArrayList<>();
    private DatabaseContext dbContext;
    private HistoryRepository historyRepo;

    public BrowserWindow() {
        this.dbContext = new DatabaseContext();
        this.historyRepo = new HistoryRepository(dbContext);
    }

    public void openNewTab(String url) {
        Tab tab = new Tab(historyRepo);
        tabs.add(tab);
        tab.navigateTo(url);
    }

    public void showHistory() {
        List<HistoryItem> history = historyRepo.getAll();
        System.out.println("--- Історія переглядів ---");
        for (HistoryItem item : history) {
            System.out.println(item.visitTime + ": " + item.url);
        }
    }
}

// 6. Головний клас для демонстрації
public class Main {
    public static void main(String[] args) {
        BrowserWindow browser = new BrowserWindow();

        browser.openNewTab("google.com");
        browser.openNewTab("kpi.ua");

        browser.showHistory();
    }
}
```

Висновок: під час виконання лабораторної роботи, ми навчилися основам проектування, обрали зручну систему побудови UML-діаграм та навчилися будувати діаграми варіантів використання для системи що проєктується, розробляти сценарії варіантів використання та будувати діаграми класів предметної області.