

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

**Тема:** Основи проектування.

Виконала  
студентка групи IA-32:  
Слюсарєва А.А.

Київ 2025

## **ЗМІСТ**

[2.1 Завдання](#)

[2.2 Теоретичні відомості](#)

[2.3 Хід роботи](#)

[2.4 Висновок](#)

[2.5 Контрольні запитання](#)

## ЛАБОРАТОРНА РОБОТА № 2

**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.

### 2.1. 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 спроектованої 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.

### 2.2. Теоретичні відомості

#### 1. Мова UML (Уніфікована Мова Моделювання)

**Призначення:** Загальноцільова мова візуального моделювання для специфікації, візуалізації, проєктування та документування ПЗ, бізнес-процесів та інших систем.

**Методологія ООАП (Об'єктно-орієнтованого аналізу та проєктування):**  
Повна модель системи складається з взаємопов'язаних представлень (views) (статичне, динамічне), які розглядаються на різних рівнях абстрагування:

- Концептуальний рівень (початкова модель): Загальне уявлення, без багатьох деталей.
- Логічний рівень: Конкретизація, деталізація структури та поведінки.
- Фізичний рівень: Представлення фізичних компонентів (реалізація).

**Діаграма (Diagram):** Графічне представлення сукупності елементів моделі у формі зв'язного графа. Канонічні діаграми UML (наприклад, класів, варіантів використання, компонентів) фіксують усі уявлення про модель.

## 2. Діаграма Варіантів Використання (Use-Case Diagram)

**Тип моделі:** Концептуальна модель, що відображає вимоги до системи. Не описує внутрішній устрій.

**Призначення:**

- Визначення загальної межі функціональності системи.
- Формулювання загальних вимог до функціональної поведінки.
- Створення основи для аналізу, проєктування, розробки та тестування.

**Основні елементи:**

**Варіант використання (Use Case):** Описує службу, яку система надає актору (еліпс).

**Актор (Actor):** Будь-який зовнішній об'єкт, суб'єкт чи система, що взаємодіє з моделлюємою системою.

**Відношення (Relationship):** Семантичний зв'язок між елементами.

- **Асоціація (Association):** Зв'язок між актором і варіантом використання (суцільна лінія, може бути спрямованою).
- **Узагальнення (Generalization):** Наслідування атрибутів/поведінки (предок-нащадок, не зафарбований трикутник).
- **Залежність (Dependency):** Зміна одного елемента призводить до зміни іншого.
- **Включення (Include):** Один варіант використання завжди включає поведінку іншого (залежність <<include>>, пунктирна стрілка до включаємого). Використовується для повторного використання функціональності.
- **Розширення (Extend):** Варіант використання може розширювати базову поведінку за певних умов (залежність <<extend>>, пунктирна стрілка від розширюючого). Може мати точки розширення (extension point).

**Сценарії використання:** Текстові, покрокові інструкції, що детально описують взаємодію користувача та системи для конкретного варіанта використання (включають передумови, постумови, основний хід подій, винятки).

### 3. Діаграма Класів (Class Diagram)

**Тип моделі:** Логічна/статична модель, що показує структуру системи.

**Елементи:**

**Клас:** Основний будівельний блок. Прямокутник, розділений на 3 області: Назва, Атрибути (властивості, дані), Операції (методи, послуги).

Видимість атрибутів/операцій:

- (Public, відкритий): Видно для будь-якого іншого класу.
- # (Protected, захищений): Видно для нащадків.
- (Private, закритий): Видно лише всередині класу (Інкапсуляція).

Відношення:

- Асоціація (Association): Загальний зв'язок між класами. Має множинність (кількість об'єктів, що беруть участь у зв'язку).
- Узагальнення (Generalization/Наслідування): Зв'язок клас-предок to клас-нащадок. Нашадок успадковує атрибути та операції.
- Агрегація (Aggregation): Відношення "has-a" (частина-ціле). Слабка залежність (порожній ромб на стороні цілого).
- Композиція (Composition): Відношення "owns-a" (ціле володіє частиною). Сильна залежність, частина не може існувати без цілого (зафарбований ромб на стороні цілого).

#### 4. Логічна Структура Бази Даних та Нормальні Форми

Логічна модель БД: Структура таблиць, представлень, індексів, що дозволяє програмування та використання БД.

Нормальні форми (НФ): Властивість відношення (таблиці) у реляційній моделі, що характеризує його з точки зору надмірності та потенційної суперечливості.

Нормалізація: Процес перетворення структури БД для досягнення мінімальної логічної надмірності та усунення аномалій оновлення.

- 1НФ (Перша НФ): Кожен кортеж (рядок) містить лише одне значення для кожного атрибута (стовпця).

- 2НФ (Друга НФ): У 1НФ, і кожен неключовий атрибут функціонально залежить від потенційного ключа.
- 3НФ (Третя НФ): У 2НФ, і відсутні транзитивні функціональні залежності неключових атрибутів від ключових.
- НФ Бойса — Кодда (НФБК): Кожна нетривіальна та неприводима функціональна залежність має як свій детермінант деякий потенційний ключ.

### 2.3 Хід роботи

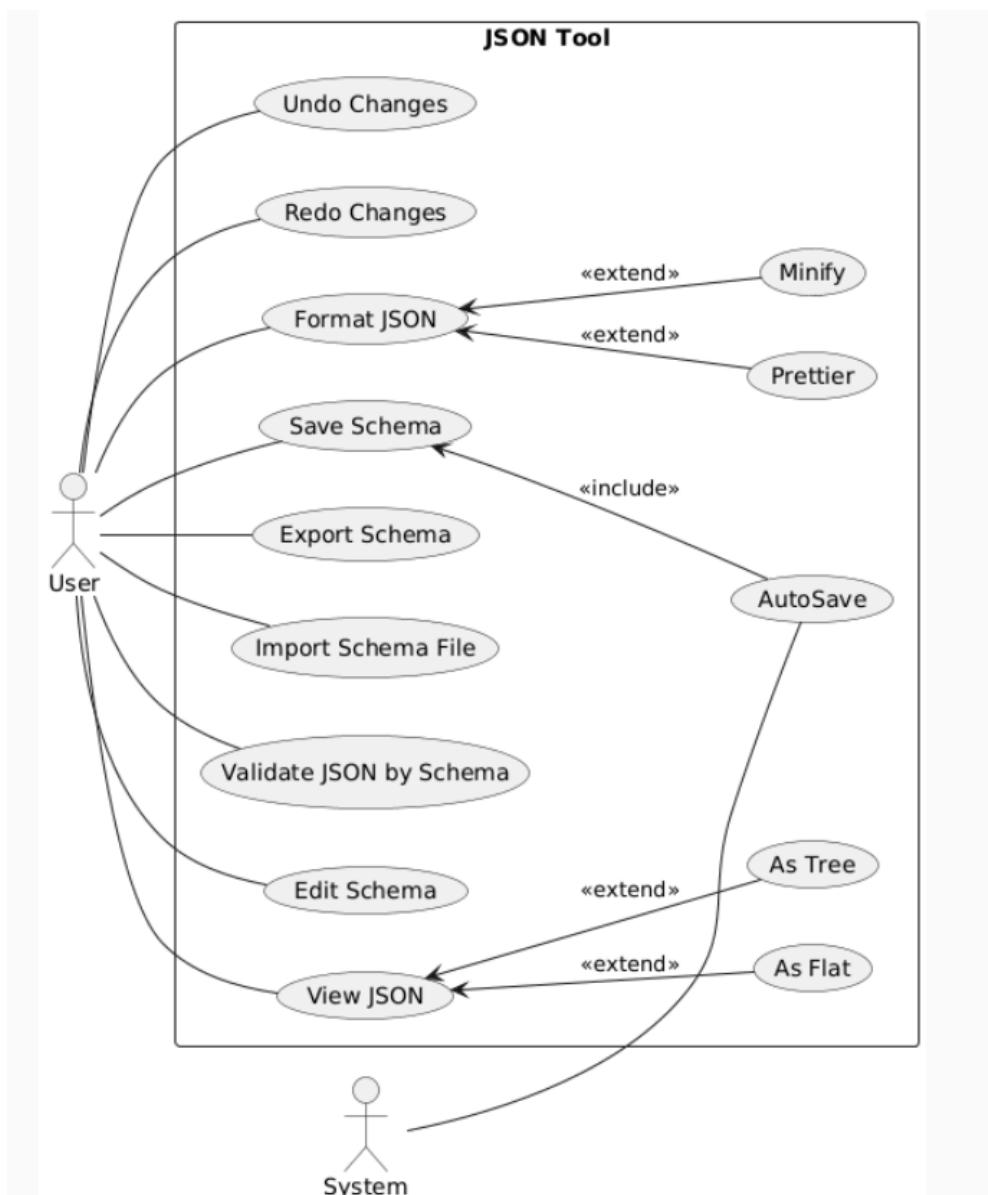


Рис 1. Діаграма прецедентів

Назва	Форматування JSON
Передумови	Користувач має доступ до системи та ввів JSON-текст.
Післяумови	<ul style="list-style-type: none"> <li>• Користувач отримує структурований, відформатований JSON.</li> <li>• У разі помилки синтаксису система повідомляє про некоректний JSON.</li> </ul>
Сторони, що взаємодіють	Користувач, система
Опис	Користувач запускає операцію автоматичного форматування JSON, щоб зробити його читабельним і правильно структурованим.
Послідовність подій	<p>Користувач вводить JSON-текст у редактор.</p> <p>Користувач обирає дію "Format JSON".</p> <p>Система аналізує JSON.</p> <p>Система здійснює форматування JSON згідно стандартних правил.</p> <p>Система показує користувачу відформатований JSON.</p>
Виняткові ситуації	JSON містить синтаксичні помилки, система відображає повідомлення про це.

Таблиця 1. Форматування JSON

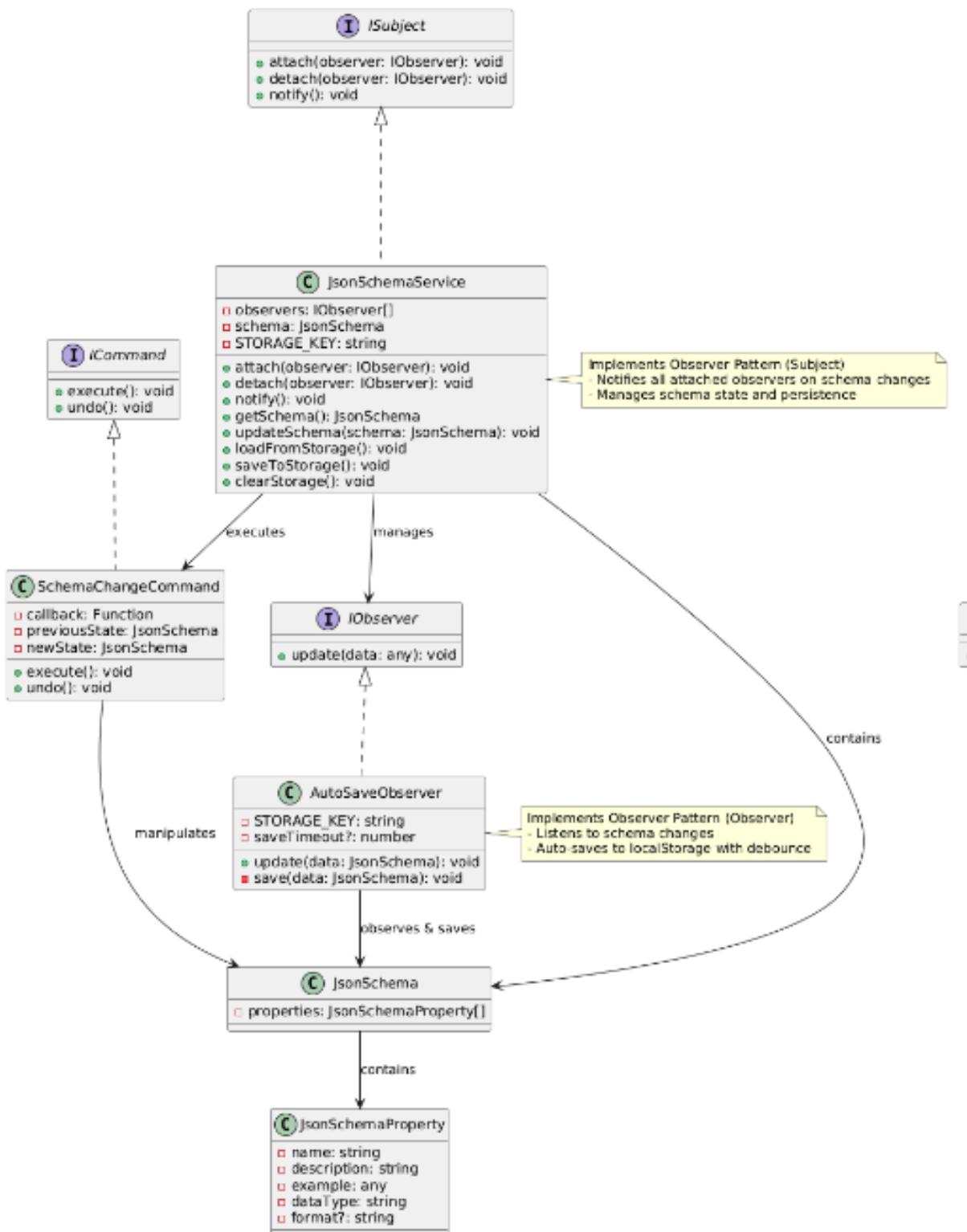
Назва	Скасувати зміни
Передумови	У редакторі JSON були виконані одна або більше змін.
Післяумови	<ul style="list-style-type: none"> <li>• Внесені зміни відмінено.</li> <li>• JSON повернуто до попереднього стану.</li> </ul>
Сторони, що взаємодіють	Користувач, система
Опис	Користувач скасовує останню дію, виконану під час

	редагування JSON.
Послідовність подій	<p>Користувач редагує JSON.</p> <p>Користувач натискає CTRL+Z.</p> <p>Система визначає останню дію у стеку змін.</p> <p>Система відновлює попередній стан JSON.</p>
Виняткові ситуації	Немає дій для скасування

Таблиця 2. Скасувати зміни

Назва	Валідація JSON
Передумови	Користувач має JSON-об'єкт і JSON-схему.
Післяумови	<ul style="list-style-type: none"> <li>Користувач отримує підтвердження про відповідність JSON схемі.</li> <li>У разі помилок система повертає список невідповідностей.</li> </ul>
Сторони, що взаємодіють	Користувач, система
Опис	Користувач перевіряє, чи відповідає JSON-об'єкт обраній JSON-схемі, вводячи їх у систему для валідації.
Послідовність подій	<p>Користувач вводить JSON-об'єкт у систему.</p> <p>Користувач вводить або вибирає JSON-схему.</p> <p>Користувач обирає дію "Validate JSON".</p> <p>Система виконує перевірку JSON згідно правил, описаних у схемі.</p> <p>Система повідомляє користувача про результат.</p>
Виняткові ситуації	Введений JSON або схема містять помилки, система показує повідомлення.

Таблиця 3. Валідація JSON



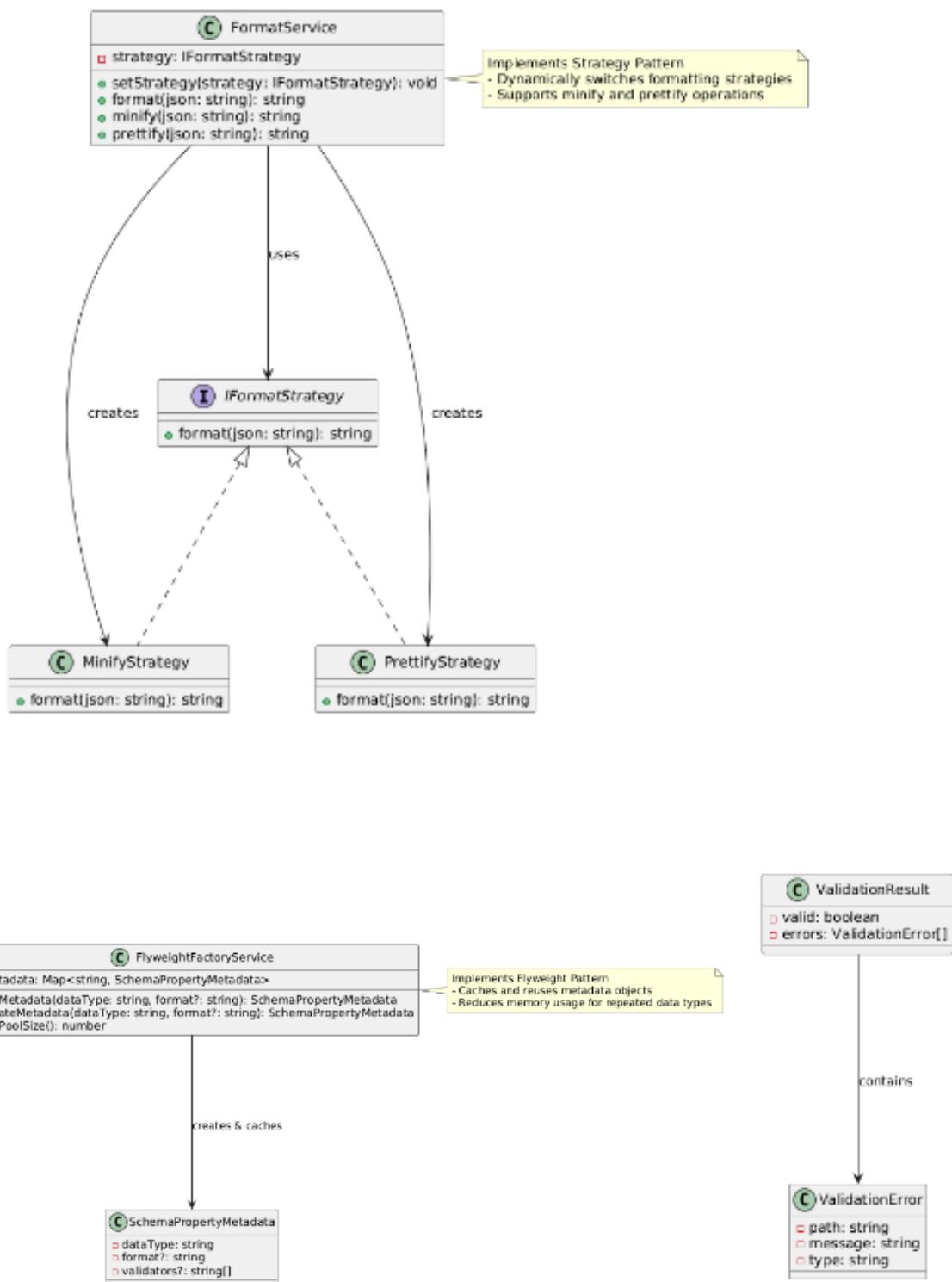


Рис 2. Діаграма класів системи

## **2.4 Висновок**

Під час даної роботи було відпрацьовано діаграми прецедентів і класів.

## **2.5 Контрольні запитання**

### **1. Що таке UML?**

UML – це універсальна мова моделювання, яка використовується для опису, проєктування та документування програмних і бізнес-систем.

### **2. Що таке діаграма класів UML?**

Діаграма класів – це структурна UML-діаграма, що показує класи системи, їх атрибути, методи та зв’язки між ними.

### **3. Які діаграми UML називають канонічними?**

Канонічні діаграми – це базові діаграми UML, до яких належать діаграма класів, варіантів використання, послідовностей, діяльності, компонентів та станів.

### **4. Що таке діаграма варіантів використання?**

Діаграма варіантів використання – це діаграма, що показує взаємодію користувачів з системою через набори можливих дій.

### **5. Що таке варіант використання?**

Варіант використання – це опис конкретної функції системи з погляду користувача та її результату.

### **6. Які відношення можуть бути відображені на діаграмі використання?**

На діаграмі використання відображають відношення асоціації між актором і варіантом, а також залежності типу «include» та «extend».

### **7. Що таке сценарій?**

Сценарій – це послідовність кроків, що описує виконання конкретного варіанта використання в окремому випадку.

## **8. Що таке діаграма класів?**

Діаграма класів – це модель структури системи, яка показує її класи, їх властивості, операції та зв'язки.

## **9. Які зв'язки між класами ви знаєте?**

Основні зв'язки – асоціація, залежність, узагальнення, агрегація, композиція.

## **10. Чим відрізняється композиція від агрегації?**

Композиція – це сильна форма цілого, де частина не може існувати без цілого, агрегація – слабка форма, де частина може існувати окремо.

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

Агрегація позначається порожнім ромбом і показує слабкий зв'язок, композиція позначається зафарбованим ромбом і показує сильну залежність частини від цілого.

## **12. Що являють собою нормальні форми баз даних?**

Нормальні форми – це правила організації таблиць БД, що допомагають усунути дублювання даних і забезпечити цілісність.

## **13. Що таке фізична модель бази даних? Логічна?**

Логічна модель – це опис структури даних без урахування СУБД, фізична модель – це конкретна реалізація таблиць, індексів і типів даних у вибраній СУБД.

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

Таблиця БД зазвичай відповідає класу, рядок таблиці – об'єкту, а стовпці – його властивостям.