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

«Основи проектування.»

Виконала:
студентка групи ІА-32
Красоха В.О.

Перевірив:
Мягкий М.Ю.

Київ 2025

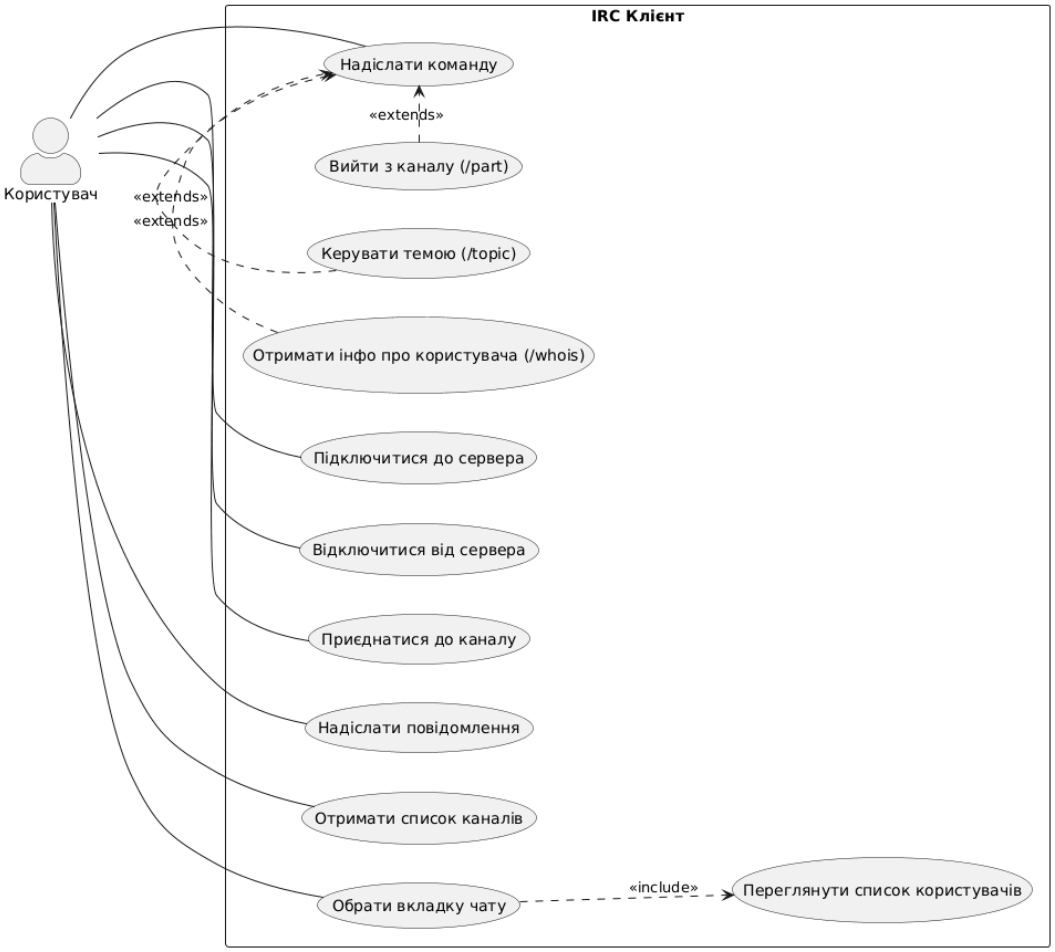
Тема: Основи проектування.

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.

Хід роботи

Тема: IRC-Клієнт

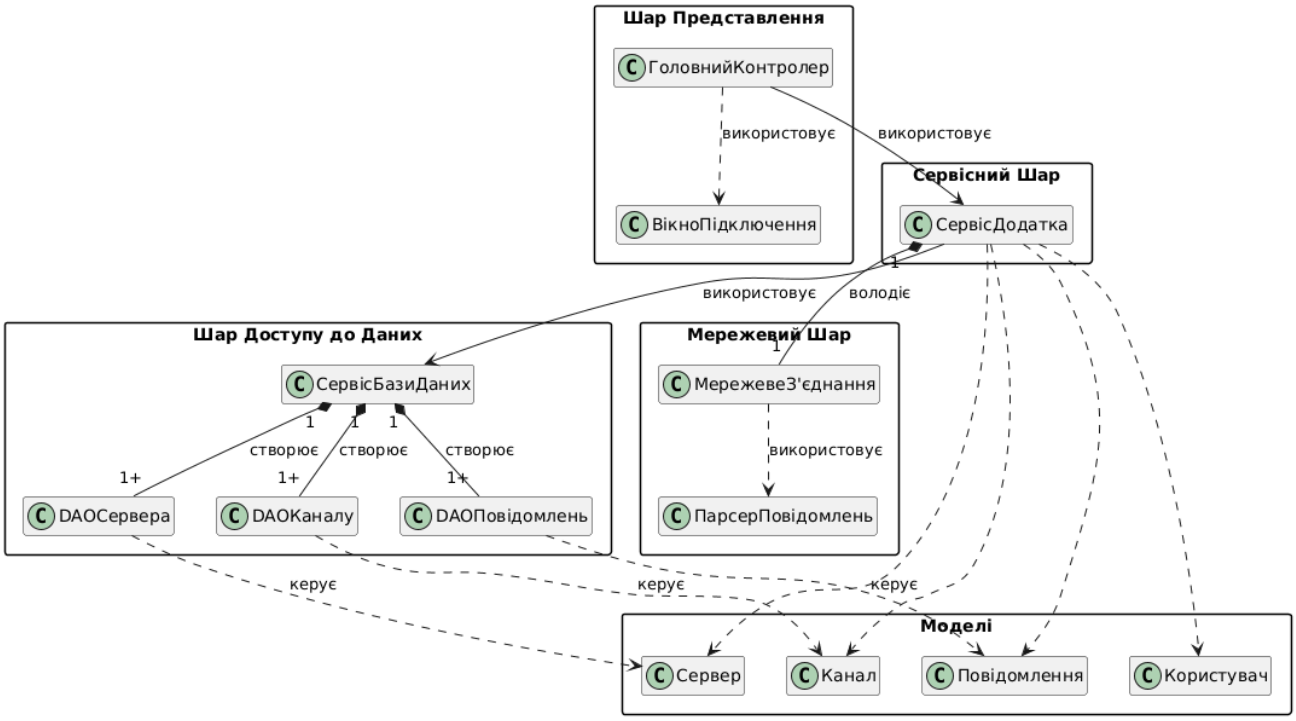
Діаграма варіантів використання:



Для моделювання взаємодії користувача з системою побудовано діаграму варіантів використання. Вона визначає єдиного актора, "Користувача", та набір його взаємодій. До основних варіантів належать "Підключитися до сервера", "Надіслати повідомлення" та "Приєднатися до каналу". Також визначено специфічні варіанти, що розширюють базові, наприклад, "Надіслати команду", та автоматичні процеси, що включаються, як-от "Завантажити історію повідомлень" при приєднанні до каналу. На основі цих варіантів використання розроблено детальні сценарії, що описують покрокову логіку взаємодії,

включно з основним успішним потоком та альтернативними сценаріями, такими як обробка помилок або введення команди замість повідомлення.

Діаграма класів системи:



Архітектура Repository Pattern

- Repository<T>: Базовий інтерфейс, що визначає методи save(T entity), findById(int id), findAll(), delete(int id).

- MessageRepository: Спадкує базовий інтерфейс; додає метод findByChannel(int channelId) для завантаження історії конкретного чату.
- ServerRepository: Спадкує базовий інтерфейс; додає метод findActiveServers() для відображення поточної панелі серверів.
- ChannelRepository: Реалізує доступ до збережених каналів, до яких користувач приєднувався раніше.

Моделі (Предметна область IRC-клієнта)

- User { id, nickname, realName, status }: Сутність користувача системи (як поточного клієнта, так і учасників каналів).
- Server { id, name, host, port, nickname }: Модель, що зберігає параметри підключення до IRC-сервера.
- Message (Abstract) { id, sender, timestamp, content }: Базовий абстрактний клас для всіх повідомлень. Реалізує патерн Prototype (метод clone) для можливості пересилання або повторного використання об'єктів.
    - ChannelMessage { channelId }: Конкретна реалізація повідомлення, прив'язаного до конкретного каналу.
    - PrivateMessage { receiverId }: Повідомлення для приватних діалогів.

Зв’язки між класами

- Server 1 — 0..* Channel: (Композиція) Об'єкт сервера володіє списком каналів, які активні під час поточної сесії.
- Channel 1 — 0..* Message: (Агрегація) Канал містить історію повідомлень, яка завантажується з бази даних SQLite.
- User 1 — 0..* Message: (Асоціація) Один користувач може бути автором багатьох повідомлень у різних каналах.
- Message <|— ChannelMessage, PrivateMessage: (Успадкування) Спеціалізовані типи повідомлень є нащадками базового класу Message.

Сервіси та Utility-класи

- IRCService: Головний сервіс бізнес-логіки. Використовує IRCConnection для мережевої роботи та Repository для збереження даних. Методи: connect(), joinChannel(name), sendMessage(text).
- DatabaseService: Реалізований за патерном Singleton. Гарантує єдине підключення до файлу ircclient.db через JDBC та ініціалізує структуру таблиць при першому запуску.
- MessageParser: Utility-клас для розбору вхідних сирих IRC-рядків у об'єкти типу Message.
- MainController: Контролер JavaFX, що пов’язує FXML-вигляд із сервісами. Реалізує обробку подій натискання кнопок та введення команд.

Структура бази даних:

Для зберігання даних додатка, таких як профілі підключень та історія повідомлень, використовується реляційна база даних SQLite. Структура бази даних визначається на етапі ініціалізації сервісу даних і складається з трьох пов'язаних таблиць.

Перша таблиця, servers, призначена для зберігання профілів серверів, до яких підключався користувач.

Таблиця 2.1 - Servers

Назва поля	Тип SQLite	Призначення
id	INTEGER	Первинний ключ (Primary Key), автоінкремент
name	VARCHAR(100)	Описова назва сервера, надана користувачем
host	VARCHAR(255)	Адреса (хост) сервера для підключення
port	INT	Порт сервера
nickname	VARCHAR(50)	Нікнейм, що використовувався для цього підключення

Друга таблиця, channels, зберігає канали, до яких приєднувався користувач, і має зовнішній ключ, що посилається на таблицю servers.

Таблиця 2.2 - Channels

Назва поля	Тип SQLite	Призначення
id	INTEGER	Первинний ключ (Primary Key), автоіключ
server_id	INT	Зовнішній ключ (Foreign Key) до servers(id)
name	VARCHAR(100)	Назва каналу (напр., #java)
joined_at	DATETIME	Час останнього приєднання до каналу

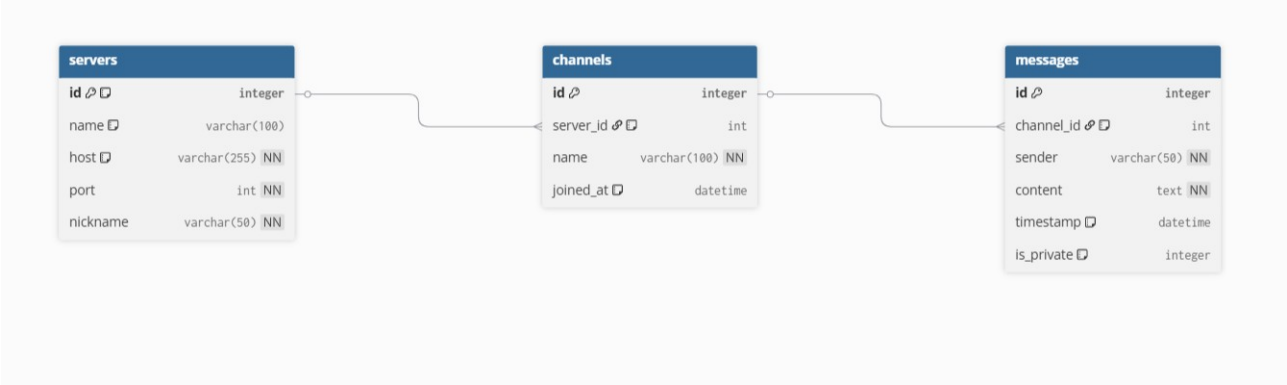
Третя таблиця, messages, призначена для кешування історії повідомлень. Вона пов'язана із таблицею channels.

Таблиця 2.3 - Messages

Назва поля	Тип SQLite	Призначення
id	INTEGER	Первинний ключ (Primary Key), автоінкремент
channel_id	INT	Зовнішній ключ (Foreign Key) до channels(id)
sender	VARCHAR(50)	Відправник повідомлення (нікнейм)
content	TEXT	Тіло повідомлення

Продовження таблиці 2.3

Назва поля	Тип SQLite	Призначення
timestamp	DATETIME	Часова мітка відправки повідомлення
is_private	INTEGER	Булевий прапор (0 або 1), що вказує на приватне повідомлення



Коди класів:

Channel.java:

```
package org.ircclient.model;

import java.time.LocalDateTime;

public class Channel {
    private Integer id;
    private Integer serverId;
    private String name;
    private LocalDateTime joinedAt;
    private String topic;

    public Channel() {
    }

    public Channel(Integer serverId, String name) {
        this.serverId = serverId;
        this.name = name;
        this.joinedAt = LocalDateTime.now();
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
```

```
    }

    public Integer getServerId() {
        return serverId;
    }

    public void setServerId(Integer serverId) {
        this.serverId = serverId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public LocalDateTime getJoinedAt() {
        return joinedAt;
    }

    public void setJoinedAt(LocalDateTime joinedAt) {
        this.joinedAt = joinedAt;
    }

    public String getTopic() {
        return topic;
    }

    public void setTopic(String topic) {
        this.topic = topic;
    }

    @Override
    public String toString() {
        return name;
    }
}
```

Message.java:

```
package org.ircclient.model;

import java.time.LocalDateTime;

public class Message {
    private Integer id;
    private Integer channelId;
    private String channelName;
    private String sender;
    private String content;
    private LocalDateTime timestamp;
    private Boolean isPrivate;
    private MessageType type;

    public enum MessageType {
        NORMAL,      // Звичайне повідомлення
        SYSTEM,      // Системне повідомлення
        ERROR,       // Помилка
        JOIN,        // Користувач приєднався
```

```
    PART,       // Користувач вийшов
    QUIT,       // Користувач відключився
    TOPIC,      // Зміна теми
    NOTICE      // Notice повідомлення
}

public Message() {
    this.timestamp = LocalDateTime.now();
    this.type = MessageType.NORMAL;
    this.isPrivate = false;
}

public Message(String sender, String content) {
    this();
    this.sender = sender;
    this.content = content;
}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public Integer getChannelId() {
    return channelId;
}

public void setChannelId(Integer channelId) {
    this.channelId = channelId;
}

public String getChannelName() {
    return channelName;
}

public void setChannelName(String channelName) {
    this.channelName = channelName;
}

public String getSender() {
    return sender;
}

public void setSender(String sender) {
    this.sender = sender;
}

public String getContent() {
    return content;
}

public void setContent(String content) {
    this.content = content;
}

public LocalDateTime getTimestamp() {
    return timestamp;
}

public void setTimestamp(LocalDateTime timestamp) {
```

```
        this.timestamp = timestamp;
    }

    public Boolean getIsPrivate() {
        return isPrivate;
    }

    public void setIsPrivate(Boolean isPrivate) {
        this.isPrivate = isPrivate;
    }

    public MessageType getType() {
        return type;
    }

    public void setType(MessageType type) {
        this.type = type;
    }
}
```

Server.java:

```
package org.ircclient.model;

import java.time.LocalDateTime;

public class Server {
    private Integer id;
    private String name;
    private String host;
    private Integer port;
    private String nickname;
    private LocalDateTime createdAt;

    public Server() {
    }

    public Server(String name, String host, Integer port, String nickname) {
        this.name = name;
        this.host = host;
        this.port = port;
        this.nickname = nickname;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getHost() {
```

```
        return host;
    }

    public void setHost(String host) {
        this.host = host;
    }

    public Integer getPort() {
        return port;
    }

    public void setPort(Integer port) {
        this.port = port;
    }

    public String getNickname() {
        return nickname;
    }

    public void setNickname(String nickname) {
        this.nickname = nickname;
    }

    public LocalDateTime getCreatedAt() {
        return createdAt;
    }

    public void setCreatedAt(LocalDateTime createdAt) {
        this.createdAt = createdAt;
    }

    @Override
    public String toString() {
        return name != null ? name : host + ":" + port;
    }
}
```

User.java:

```
package org.ircclient.model;

public class User {
    private String nickname;
    private String username;
    private String hostname;
    private String realName;
    private String modes;

    public User() {
    }

    public User(String nickname) {
        this.nickname = nickname;
    }

    public String getNickname() {
        return nickname;
    }

    public void setNickname(String nickname) {
        this.nickname = nickname;
```

```
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getHostname() {
        return hostname;
    }

    public void setHostname(String hostname) {
        this.hostname = hostname;
    }

    public String getRealName() {
        return realName;
    }

    public void setRealName(String realName) {
        this.realName = realName;
    }

    public String getModes() {
        return modes;
    }

    public void setModes(String modes) {
        this.modes = modes;
    }

    @Override
    public String toString() {
        return (modes != null ? modes : "") + nickname;
    }
}
```

Контрольні запитання:

1.Що таке UML?

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

2. Що таке діаграма класів UML?

Це структурна діаграма UML, яка відображає класи системи, їх атрибути, методи та зв’язки між класами.

3. Які діаграми UML називають канонічними?

Канонічні (основні) діаграми UML:

- діаграма класів
- діаграма варіантів використання
- діаграма послідовностей
- діаграма діяльності
- діаграма станів
- діаграма компонентів
- діаграма розгортання

4. Що таке діаграма варіантів використання?

Це діаграма UML, яка показує взаємодію користувачів (акторів) із системою через варіанти використання.

5. Що таке варіант використання?

Варіант використання — це опис функціональної можливості системи, яка виконується у взаємодії з актором для досягнення певної мети.

6. Які відношення можуть бути відображені на діаграмі використання?

- асоціація (актор ↔ варіант використання)
- include (включення)

- extend (розширення)
- узагальнення (generalization)

7. Що таке сценарій?

Сценарій — це послідовність кроків виконання варіанту використання, яка описує взаємодію актора і системи.

8. Що таке діаграма класів?

Діаграма класів — це UML-діаграма, що описує структуру системи через класи, їх властивості, операції та взаємозв’язки.

9. Які зв’язки між класами ви знаєте?

- асоціація
- агрегація
- композиція
- успадкування (generalization)
- залежність (dependency)
- реалізація (realization)

10. Чим відрізняється композиція від агрегації?

- Агрегація — слабкий зв’язок «ціле–частина», де частина може існувати окремо
- Композиція — сильний зв’язок, де частина не може існувати без цілого

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

- агрегація позначається порожнім ромбом
- композиція позначається заповненим ромбом
- при композиції життєвий цикл частин залежить від цілого

12. Що являють собою нормальні форми баз даних?
Нормальні форми — це правила організації таблиць БД, які зменшують надмірність даних і запобігають аномаліям оновлення (1НФ, 2НФ, 3НФ, BCNF тощо).

13. Що таке фізична модель бази даних? Логічна?

- Логічна модель — опис структури БД: таблиці, поля, зв’язки без прив’язки до СУБД
- Фізична модель — реалізація БД в конкретній СУБД (типи даних, індекси, ключі)

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

- таблиця БД ↔ клас
- поле таблиці ↔ атрибут класу
- запис таблиці ↔ об’єкт класу
- зв’язки між таблицями ↔ зв’язки між класами