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 *№8*

із дисципліни ***«Технології розробки програмного забезпечення»***

Тема: ***«Патерни проектування»***

Виконав:

Студент групи ІА-34

Ковальчук Станіслав

Перевірив:

асистент кафедри ІСТ

Мягкий Михайло Юрійович

Київ — 2025

Тема: Патерни проектування.

Мета: Вивчити структуру шаблонів «Composite», «Flyweight» (Пристосуванець), «Interpreter», «Visitor» та навчитися застосовувати їх в реалізації програмної системи.

Застосунок: Office communicator (strategy, adapter, abstract factory, bridge, composite, client-server)

Мережевий комунікатор для офісу повинен нагадувати функціонал програми Skype з можливостями голосового / відео / конференц-зв'язку, відправки текстових повідомлень і файлів (можливо, оффлайн), веденням організованого списку груп / контактів.

Короткі теоретичні відомості:

1. Шаблон «Компонувальник» (Composite)

Призначений для організації об'єктів у деревоподібну структуру для представлення ієрархії «частина-ціле».

- **Перевага:** Уніфікація обробки. Він дозволяє клієнту однаково працювати як з поодинокими об'єктами (**листами**), так і з цілими групами об'єктів (**композитами**) через єдиний інтерфейс.
- **Застосування:** Найбільш доцільний для обробки складних ієрархічних структур, таких як дерева елементів інтерфейсу або файлові системи.

2. Шаблон «Легковаговик» (Flyweight)

Використовується для оптимізації пам'яті шляхом спільного використання об'єктів різними частинами програми.

- **Ключова концепція:** Розподіл стану на **внутрішній** (незмінні дані, наприклад, символ шрифту) та **зовнішній** (контекстні дані, наприклад, координати символу на сторінці).
- **Результат:** Внутрішній стан зберігається всередині об'єкта-легковаговика, а зовнішній передається йому ззовні, що суттєво зменшує кількість створюваних екземплярів.

3. Шаблон «Інтерпретатор» (Interpreter)

Визначає граматику та механізм виконання для спеціалізованих мов (наприклад, скриптів або математичних виразів).

- **Принцип роботи:** Граматика описується через термінальні та нетермінальні символи. Клієнт формує **абстрактне синтаксичне дерево (АСД)**.
- **Механізм:** Кожен вузол дерева (вираз) інтерпретується окремо. Батьківські вузли рекурсивно викликають обробку дочірніх елементів, обчислюючи кінцевий результат на основі отриманого контексту.

4. Шаблон «Відвідувач» (Visitor)

Дозволяє додавати нові операції до існуючих класів без зміни їхнього початкового коду.

- **Плюси:** Спрощує впровадження нових функцій та групує споріднені операції в одному місці.
- **Мінуси:** Ускладнює розширення самої ієрархії об'єктів (додавання нового класу елемента вимагає оновлення всіх існуючих «відвідувачів»).
- **Ефективність:** Найкраще підходить для виконання різнотипних операцій над об'єктами складної структури.

Хід роботи:

1. Повна діаграма класів (без шаблону).

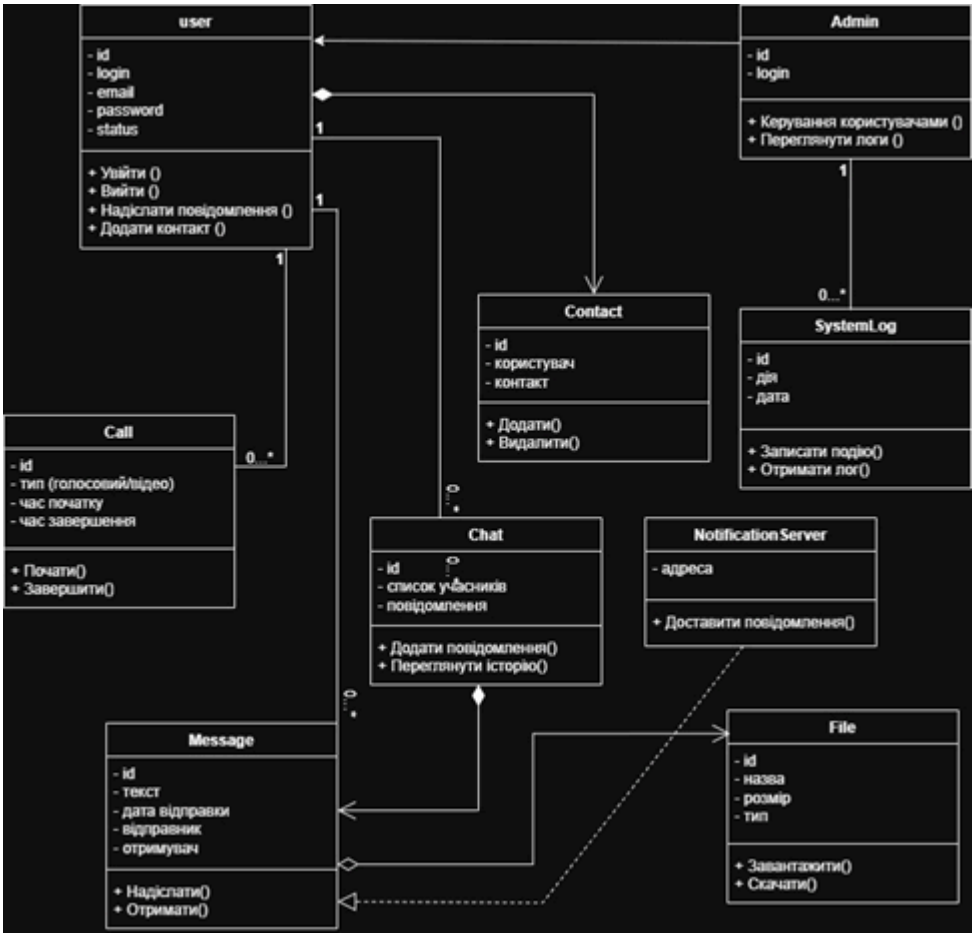


Рис. 8.1 – Діаграма класів системи

2. Діаграма класів з використанням шаблону «Composite»:

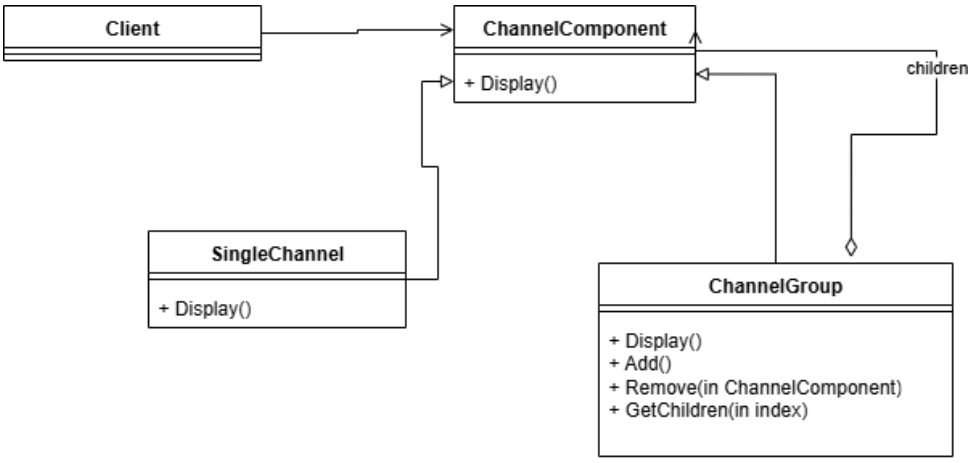
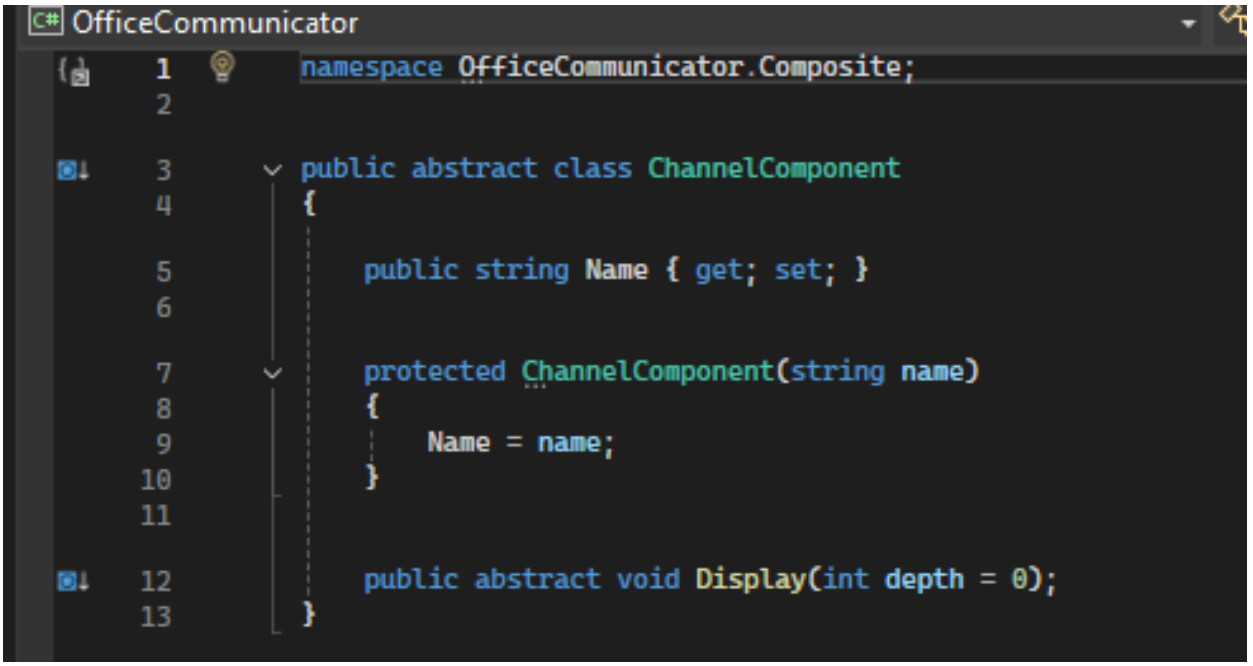


Рис. 8.2 – Діаграма класів системи з використанням шаблону «Composite»

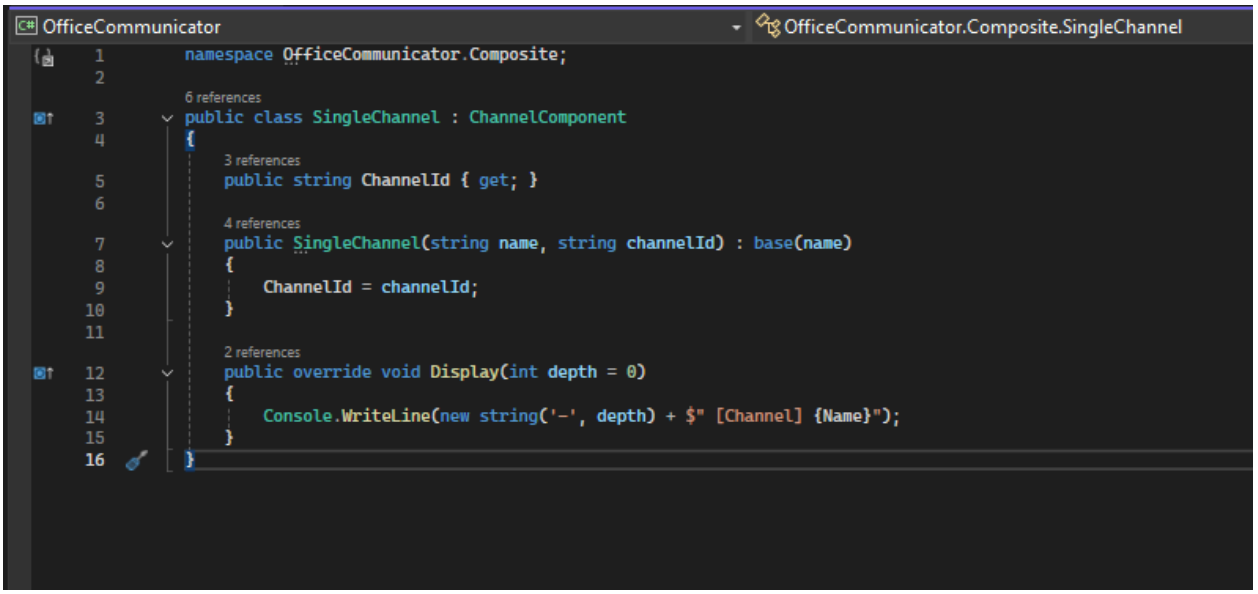
Програмна реалізація

Для реалізації деревоподібної структури каналів (патерн **Composite**) було розроблено наступні класи.

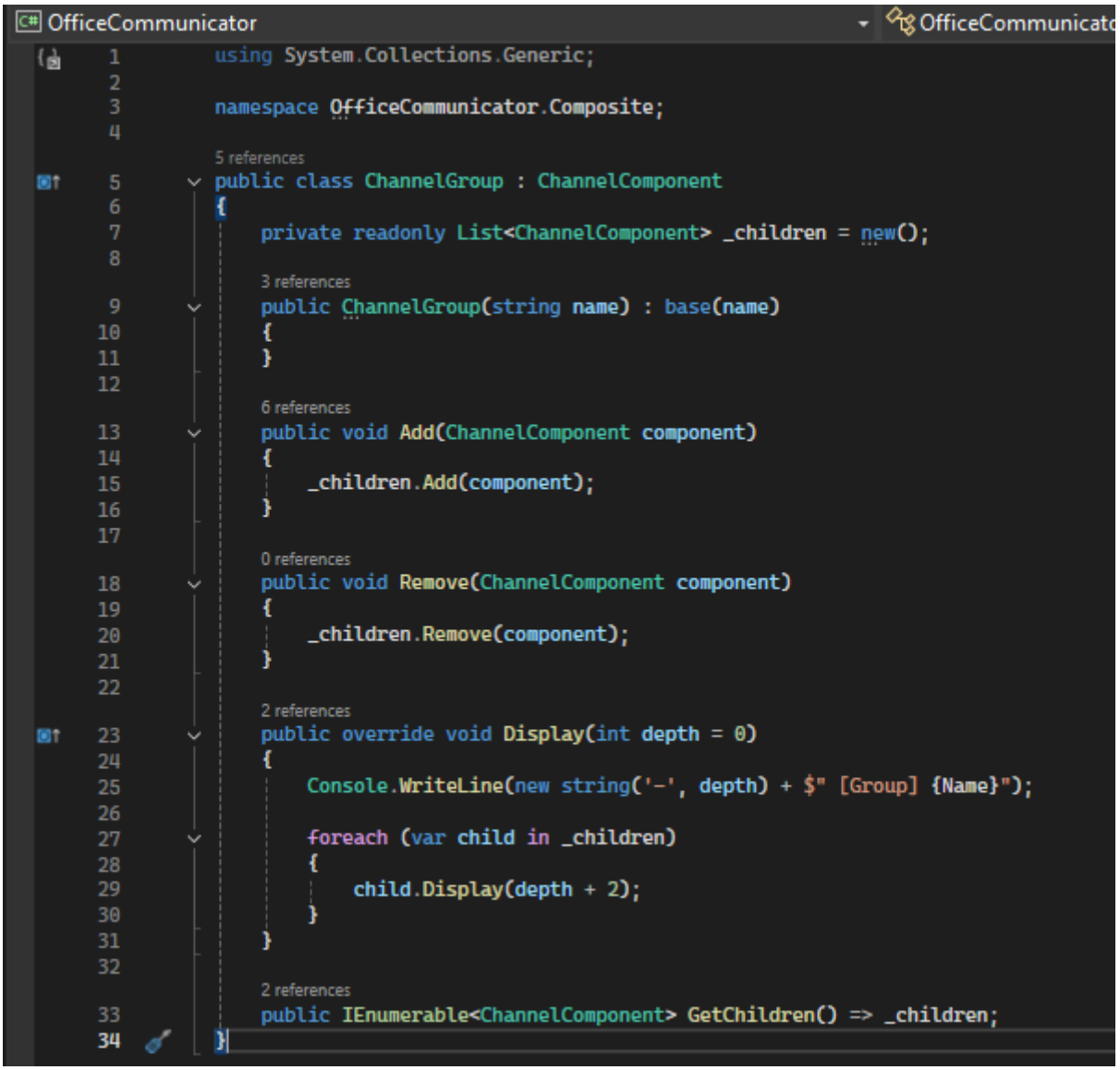
Абстрактний компонент (ChannelComponent.cs) Абстрактний клас, що визначає загальний інтерфейс для всіх об'єктів ієрархії (як груп, так і окремих каналів).



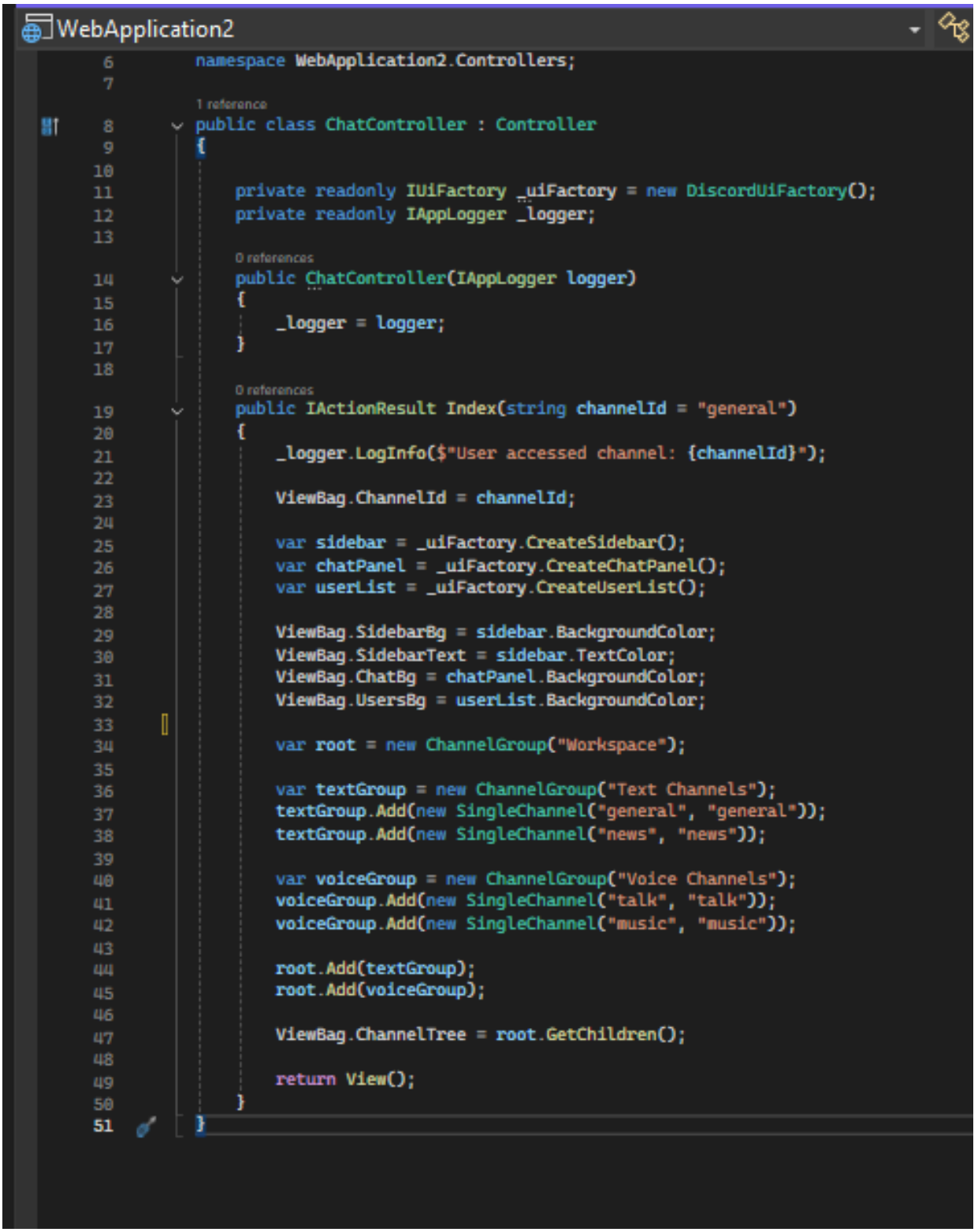
Лист дерева (SingleChannel.cs) Клас представляє окремий канал (чат). Він є кінцевим вузлом дерева ("листом") і не може містити інших компонентів.



Композит (ChannelGroup.cs) Клас представляє групу каналів. Він містить список дочірніх компонентів (_children) і реалізує методи для керування ними (Add, Remove). Метод Display рекурсивно викликає відображення для всіх нащадків.



Приклад використання (ChatController.cs) Демонстрація створення деревоподібної структури та роботи з нею через єдиний інтерфейс ChannelComponent.



Висновок: Під час виконання лабораторної роботи я поглибив свої знання про структурні патерни проєктування, зосередившись на вивченні та практичному застосуванні патерну Composite.

Головною метою роботи була організація зручної ієрархічної структури чатів у системі "Office Communicator", де канали можуть бути згруповані у папки та підпапки.

У ході програмної реалізації було виконано наступне:

1. Розроблено єдиний інтерфейс компонентів: Створено абстрактний клас ChannelComponent, який визначає загальні методи (наприклад, Display) для всіх елементів ієрархії. Це дозволило клієнтському коду працювати з простими та складними елементами однаково.
2. Реалізовано прості елементи (Leaf): Створено клас SingleChannel, що представляє окремий чат (кінцевий вузол дерева).
3. Реалізовано складені елементи (Composite): Створено клас ChannelGroup, який містить список дочірніх компонентів та реалізує механізми управління ними (додавання, видалення).

Результат: Застосування патерну Composite дозволило створити гнучку деревоподібну структуру. Завдяки цьому:

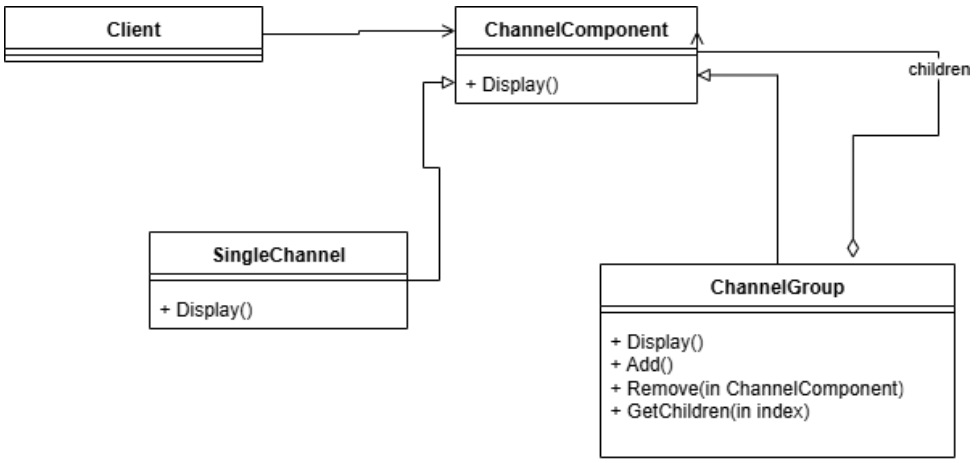
- Спростився клієнтський код (контролер), оскільки йому не потрібно знати, чи працює він з окремим каналом, чи з цілою групою.
- Система стала легко розширюваною: додавання нових типів каналів або нових рівнів вкладеності не потребує зміни основного коду програми.
- Забезпечено зручну візуалізацію структури чатів у інтерфейсі користувача.

Відповіді на контрольні питання:

1. Яке призначення шаблону «Композит»?

   Шаблон «Композит» дозволяє клієнтам працювати з окремими об'єктами та їх групами однаковим чином. Він використовується для створення деревоподібних структур, де окремі об'єкти та їх групи (композити) обробляються однаково. Це дозволяє створювати складні структури, які містять інші об'єкти чи підструктури, зберігаючи при цьому простоту доступу до елементів.

2. Нарисуйте структуру шаблону «Композит».



3. Які класи входять в шаблон «Композит», та яка між ними взаємодія?

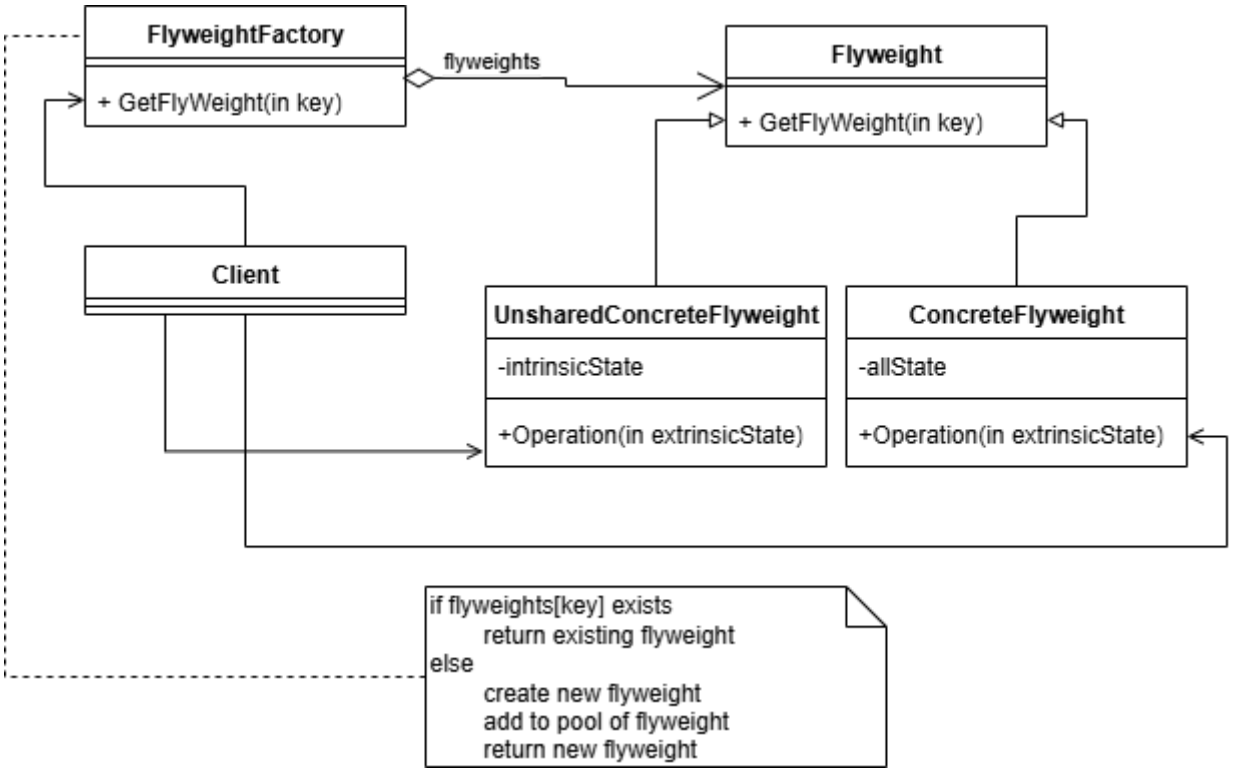
   Шаблон «Композит» складається з таких основних класів:

   - **Компонент (Component)** — це абстракція, яка визначає інтерфейс для всіх об'єктів, які будуть оброблятися в деревоподібній структурі. Це може бути як окремий об'єкт, так і група об'єктів.
   - **Лист (Leaf)** — це клас, який представляє конкретний об'єкт, що не має дочірніх елементів. Лист реалізує методи компонента.
   - **Композит (Composite)** — це клас, який представляє групу компонентів. Він зберігає колекцію дочірніх компонентів і делегує їм виконання операцій, таких як додавання, видалення та виконання операцій.

4. Яке призначення шаблону «Легковаговик»?

Шаблон «Легковаговик» (Flyweight) дозволяє зекономити пам'ять, використовуючи спільні об'єкти для збереження даних, які не змінюються, замість створення нових об'єктів для кожного запиту. Це дуже корисно, коли потрібно працювати з великими кількостями подібних об'єктів, зберігаючи ресурси.

5. Нарисуйте структуру шаблону «Легковаговик».



6. Які класи входять в шаблон «Легковаговик», та яка між ними взаємодія?

Шаблон «Легковаговик» складається з таких класів:

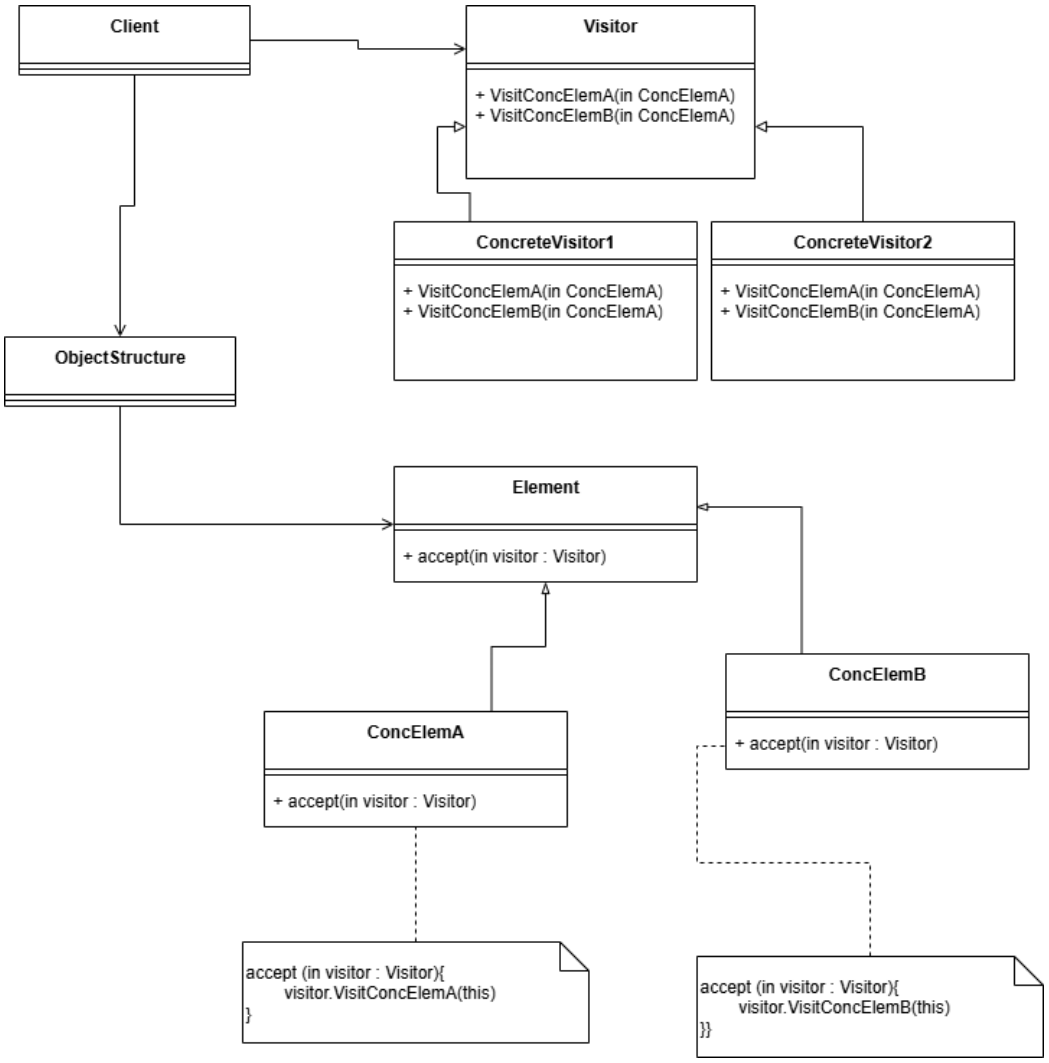
- **Легковаговик (Flyweight)** — це інтерфейс, який визначає методи для роботи з спільними об'єктами.
- **Конкретний легковаговик (ConcreteFlyweight)** — це клас, який реалізує легковаговик і зберігає спільні дані.
- **Фабрика легковаговиків (FlyweightFactory)** — це клас, який відповідає за створення і керування об'єктами легковаговиків. Він забезпечує наявність лише одного екземпляра для кожного унікального об'єкта.

- **Клієнт (Client)** — це клас, який використовує легковаговики для зберігання спільних даних і звертається до фабрики для отримання екземплярів.

7. Яке призначення шаблону «Інтерпретатор»?

   Шаблон «Інтерпретатор» дозволяє визначити граматику для мови та створити інтерпретатор, який використовує цю граматику для інтерпретації виразів. Це корисно для створення мов для специфічних задач або для роботи з доменними мовами.
8. Яке призначення шаблону «Відвідувач»?

   Шаблон «Відвідувач» дозволяє додавати нову поведінку об'єктам, не змінюючи їх самих. Це досягається шляхом використання класу «відвідувача», який може виконувати операції над елементами об'єкта, не змінюючи структуру об'єкта.
9. Нарисуйте структуру шаблону «Відвідувач».



10. Які класи входять в шаблон «Відвідувач», та яка між ними взаємодія?

Шаблон «Відвідувач» включає такі класи:

- **Елемент (Element)** — це інтерфейс або абстрактний клас, який визначає метод для прийому відвідувача. Він має посилання на методи, які дозволяють відвідувачу взаємодіяти з елементами.
- **Конкретний елемент (ConcreteElement)** — це конкретна реалізація елемента, яка реалізує метод прийому відвідувача та визначає, як відвідувач взаємодіє з цим елементом.
- **Відвідувач (Visitor)** — це інтерфейс, який визначає методи для кожного конкретного елемента. Відвідувач визначає операції, які можуть бути виконані над елементами.

- **Конкретний відвідувач (ConcreteVisitor)** — це клас, який реалізує інтерфейс відвідувача і визначає конкретні операції, які можуть бути виконані на елементах.