Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота № 8

із дисципліни «Технології розроблення програмного забезпечення»

Тема: «Патерни проектування»

Виконав
Студент групи ІА-31:
Козир Я. О.

Перевірив:
Мягкий М. Ю.

Київ 2025

Зміст

1. **Мета:**

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

2. **Теоретичні відомості**

Патерн	Призначення	Ключова ідея	Приклад
Composite	Деревоподібні ієрархії "частина-ціле"	Уніфікована обробка листків і контейнерів	Проєкт → Функція → User Story → Task
Flyweight	Зменшення пам’яті через поділ об’єктів	Внутрішній стан (shared) / Зовнішній (context)	Букви в тексті, графічні примітиви
Interpreter	Інтерпретація граматики мови	Рекурсивне AST-дерево	Пошук за шаблоном, скриптова мова
Visitor	Додавання операцій без зміни елементів	Відокремлення логіки від структури	Розрахунок цін у кошику (з/без знижки)

3. **Хід роботи**

Тема :

21. **Online radio station** (iterator, adapter, factory method, facade, visitor, client-server)

Додаток повинен служити сервером для радіостанції з можливістю мовлення на радіостанцію (64, 92, 128, 196, 224 kb/s) в потоковому режимі; вести облік підключених користувачів і статистику відвідувань і прослуховувань; налаштувати папки з піснями і можливість вести списки програвання або playlists (не відтворювати всі пісні).

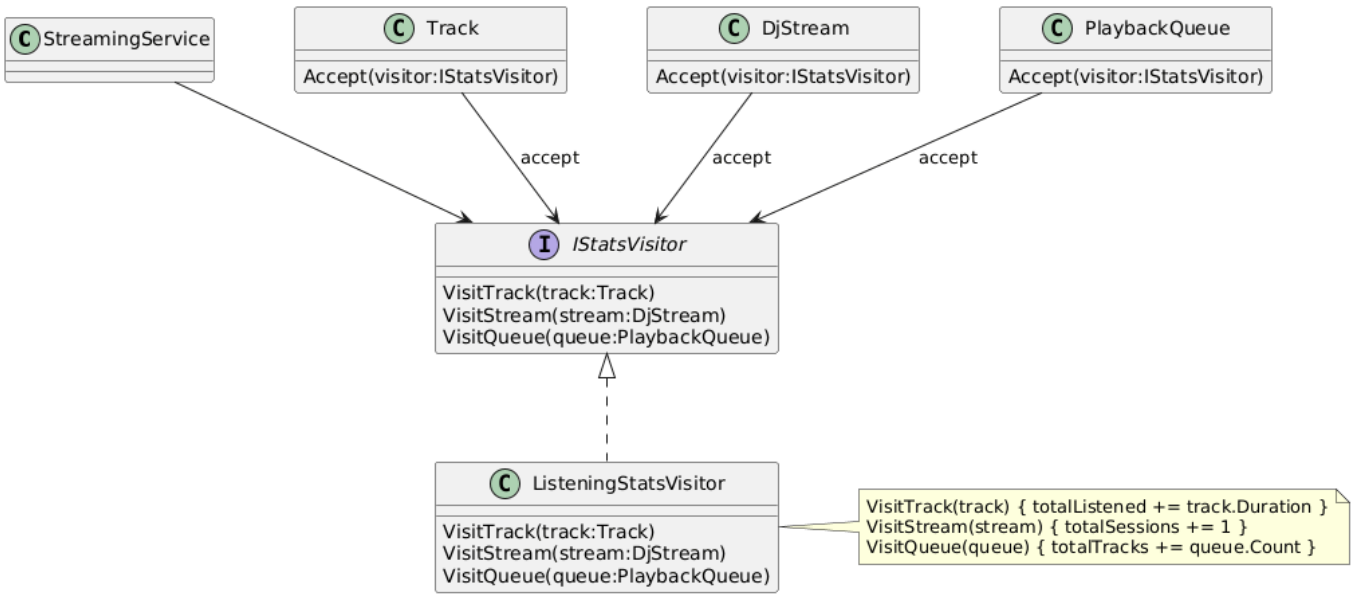


Рисунок 1 - Структура патерну Відвідувач у системі "Online Radio Station"

Для даного проєкту було обрано патерн Visitor (Відвідувач). Шаблон реалізований таким чином:

• IStatsVisitor – це інтерфейс, який описує методи Visit для кожного елемента. Він задає контракт для всіх можливих відвідувачів.

• ListeningStatsVisitor – це конкретна реалізація відвідувача. Він реалізує методи Visit по-своєму: для треку додає тривалість прослуховування, для стріму рахує сесії, для черги — кількість треків.

• Track, DjStream, PlaybackQueue – це елементи, що приймають відвідувача. Кожен клас реалізує метод Accept, який викликає відповідний Visit.

• StreamingService – це контекст, який працює з відвідувачем. Він не знає деталей реалізації, а просто викликає Accept у елементів.

В результаті: Можна легко змінювати алгоритм статистики (прослуховування, підключення, плейлисти), не змінюючи класи елементів. Це і є суть Відвідувача – відокремлення операцій від структури даних, що забезпечує гнучкість і підтримуваність системи.

```
namespace OnlineRadioStation.Domain
{
    4 references
    public interface IStatsVisitor
    {
        2 references
        void VisitTrack(Track track);
        2 references
        void VisitStream(DjStream stream);
        2 references
        void VisitQueue(PlaybackQueue queue);
    }
}
```

Рисунок 2 - Код інтерфейсу IStatsVisitor

```
namespace OnlineRadioStation.Domain
{
    2 references
    public class ListeningStatsVisitor : IStatsVisitor
    {
        2 references
        public int TotalListenedMinutes { get; private set; } = 0;
        2 references
        public int TotalSessions { get; private set; } = 0;
        2 references
        public int TotalTracks { get; private set; } = 0;

        2 references
        public void VisitTrack(Track track)
        {
            TotalListenedMinutes += (int)track.Duration.TotalMinutes;
            Console.WriteLine($"[Stats] Тривалість треку {track.Title}: {track.Duration.TotalMinutes} хв");
        }

        2 references
        public void VisitStream(DjStream stream)
        {
            TotalSessions += 1;
            Console.WriteLine($"[Stats] Сесія стріму {stream.StreamId}: {stream.EndTime - stream.StartTime}");
        }

        2 references
        public void VisitQueue(PlaybackQueue queue)
        {
            TotalTracks += 1; // кількість треків у черзі
            Console.WriteLine($"[Stats] Черга {queue.QueueId}: {queue.QueuePosition} позиція");
        }
    }
}
```

Рисунок 3 - Код класу ListeningStatsVisitor

```
1 reference
public void Accept(IStatsVisitor visitor)
{
    visitor.VisitTrack(this);
}

public void Accept(IStatsVisitor visitor)
{
    visitor.VisitStream(this);
}
}

public void Accept(IStatsVisitor visitor)
{
    visitor.VisitQueue(this);
}
```

Рисунок 4 – Метод Accept в коді класу Track, DjStream, PlaybackQueue

```
public string PrepareTrack(string mp3Path)
{
    if (_audioProcessor == null)
        throw new InvalidOperationException("IAudioProcessor не зареєстровано в DI");

    return _audioProcessor.Process(mp3Path);
}
```

Рисунок 5 - StreamingService.CollectStats

```
public void CollectStats(Track track, DjStream stream, PlaybackQueue queue)
{
    var visitor = new ListeningStatsVisitor();
    track.Accept(visitor);
    stream.Accept(visitor);
    queue.Accept(visitor);
    Console.WriteLine(
        $"[Stats] Загальна статистика: {visitor.TotalListenedMinutes} хв, " +
        $"{visitor.TotalSessions} сесій, {visitor.TotalTracks} треків");
}
```

Рисунок 6 - TestController.VisitorTest

```
[Normalizer] Нормалізую: demo.mp3
[Encoder] Кодую в AAC: demo.mp3.norm.mp3
[Tagger] Додаю метадані: demo.mp3.norm.mp3.aac
[Facade] Обробка завершена: demo.mp3.norm.mp3.aac.tagged.aac
[Stats] Тривалість треку Song 1: 3 хв
[Stats] Сесія стріму 5d0200c1-c7cc-45a6-be07-ccb43607f424: 00:30:00.0002537
[Stats] Черга 4c9999e7-1d45-4bfb-9a59-a578f56108f1: 1 позиція
[Stats] Загальна статистика: 3 хв, 1 сесій, 1 треків
```

Рисунок 7 – Вивід в консолі як результат

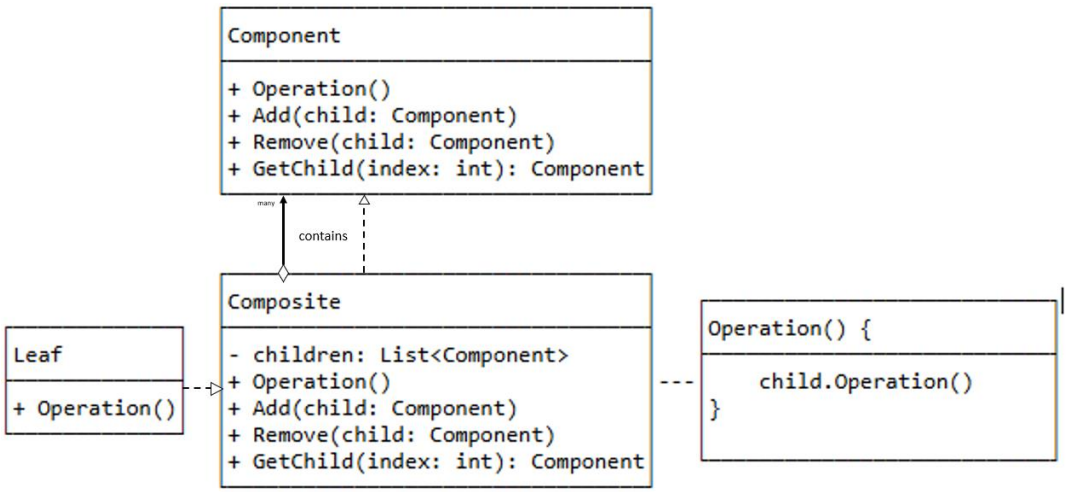
4. Висновок

У ході виконання лабораторної роботи було розглянуто шаблони проєктування, зокрема Composite, Flyweight, Interpreter, Visitor. Отримані знання допомогли зрозуміти їхню роль у створенні гнучкої та масштабованої архітектури. На прикладі веб-застосунку «Online Radio Station» було реалізовано шаблон Visitor для збору статистики (прослуховування, сесії, плейлисти). Логіка статистики винесена в окремий відвідувач (ListeningStatsVisitor), що дозволило відокремити операції від структури даних. Такий підхід спрощує керування статистикою, робить код чистішим і дає змогу легко додавати нові відвідувачі (наприклад, UserStatsVisitor) без зміни елементів. Це підтверджує ефективність патернів проєктування як інструменту для побудови зрозумілого, підтримуваного та готового до розвитку коду.

5. Контрольні питання

1. Яке призначення шаблону «Композит»? Шаблон використовується для складання об’єктів у деревоподібну структуру для подання ієрархій типу «частина — ціле». Дозволяє уніфіковано обробляти поодинокі об’єкти та композиції.

2. Нарисуйте структуру шаблону «Композит».

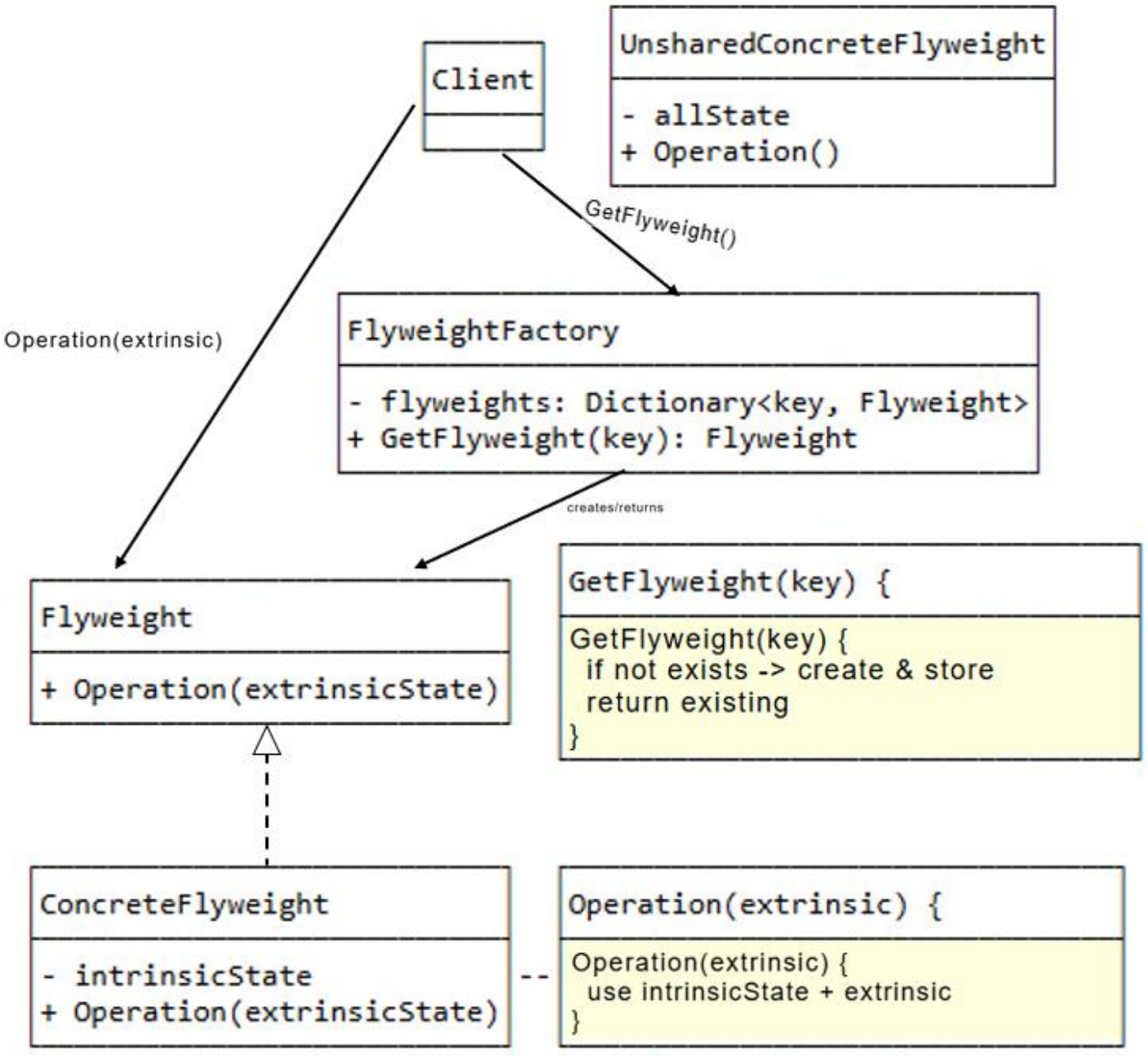


3. Які класи входять в шаблон «Композит», та яка між ними взаємодія?

- Component — абстрактний клас/інтерфейс з операціями.
- Leaf — кінцевий елемент, реалізує операції.
- Composite — контейнер, містить children, рекурсивно делегує операції. Взаємодія: клієнт працює з Component, не розрізняючи Leaf і Composite.

4. Яке призначення шаблону «Легковаговик»? Зменшення кількості об’єктів у пам’яті шляхом поділу спільного (внутрішнього) стану між кількома екземплярами.

5. Нарисуйте структуру шаблону «Легковаговик».



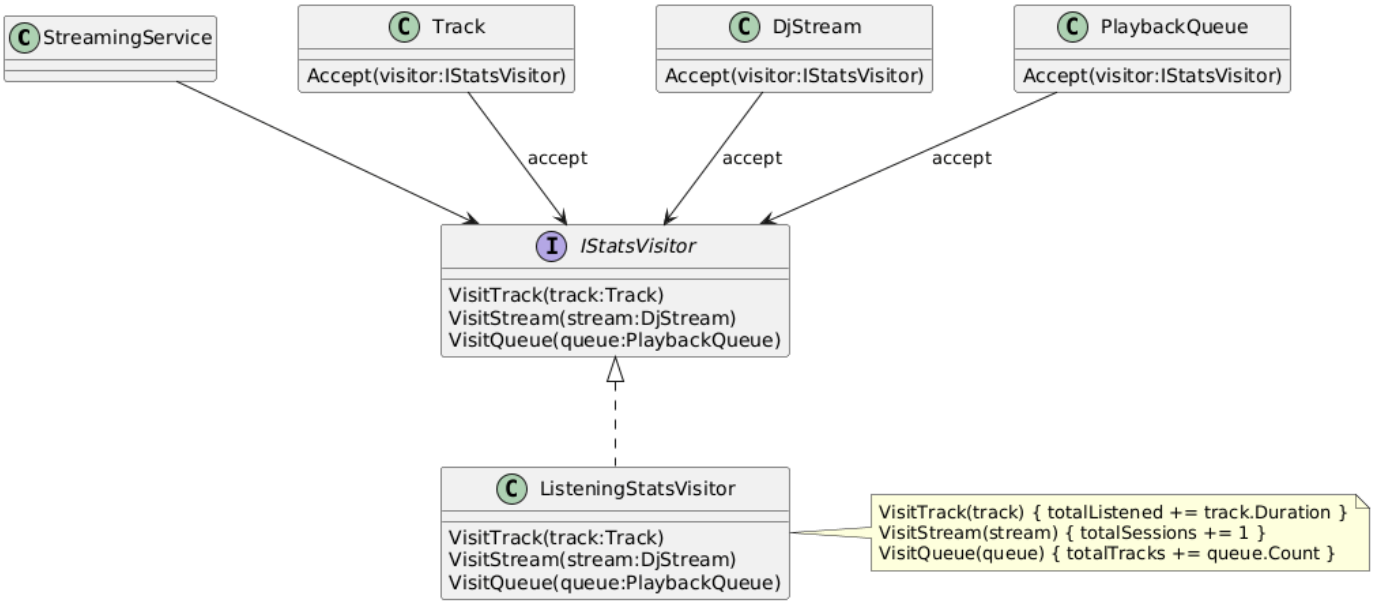
6. Які класи входять в шаблон «Легковаговик», та яка між ними взаємодія?

- Flyweight — інтерфейс з операцією, що приймає зовнішній стан.
- ConcreteFlyweight — зберігає внутрішній стан, спільний.
- UnsharedConcreteFlyweight — не поділюваний варіант.
- FlyweightFactory — кешує та повертає Flyweight за ключем. Взаємодія: клієнт отримує Flyweight з фабрики, передає зовнішній стан у метод.

7. Яке призначення шаблону «Інтерпретатор»? Подання граматики мови та інтерпретатора для обчислення виразів у термінах абстрактного синтаксичного дерева (AST).

8. Яке призначення шаблону «Відвідувач»? Дозволяє додавати нові операції над елементами ієрархії без зміни їх класів, відокремлюючи логіку від структури.

9. Нарисуйте структуру шаблону «Відвідувач».



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

- Visitor — інтерфейс з методами Visit... для кожного типу елемента.
- ConcreteVisitor — реалізує операції.
- Element — інтерфейс з Accept(Visitor).
- ConcreteElement — викликає відповідний Visit... у відвідувача.
- ObjectStructure — колекція елементів. Взаємодія: елемент викликає visitor.Visit(this), відвідувач виконує логіку за типом.