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).

Для панелі адміністратора необхідно генерувати зведений звіт: загальна тривалість усіх треків у базі, кількість проведених ефірів, кількість треків у чергах. Ці дані розкидані по різних класах сутностей (Track, DjStream, PlaybackQueue). Додавання методів типу CalculateStats() безпосередньо в ці класи порушило б принцип єдиної відповідальності (SRP), оскільки сутності не повинні відповідати за логіку звітності. Реалізовано патерн Відвідувач. Створено клас ListeningStatsVisitor, який містить алгоритми підрахунку. Сутності реалізують

метод Accept, дозволяючи відвідувачу "зайти" і зчитати необхідні дані. Клієнт (AdminController) ініціює обхід колекцій і передає заповнений об'єкт відвідувача у View для відображення.

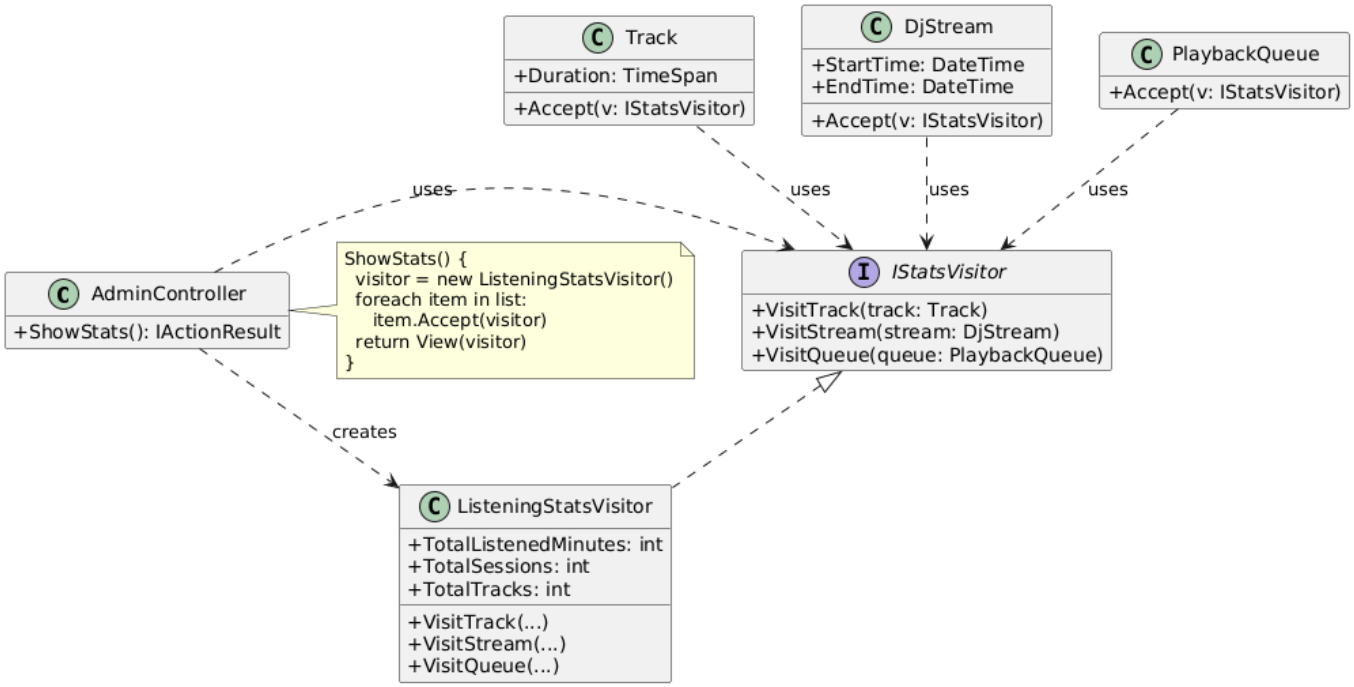


Рисунок 1 - Структура патерну Відвідувач у системі "Online Radio Station"

- AdminController (Client): Контролер, який отримує дані з БД, створює екземпляр ListeningStatsVisitor і запускає обхід елементів.
- IStatsVisitor: Інтерфейс, що визначає методи відвідування для кожного типу сутності.
- ListeningStatsVisitor (ConcreteVisitor): Накопичує статистику (сумує хвилини, лічильники) у своїх властивостях.
- Track, DjStream... (Elements): Сутності, які мають метод Accept, що дозволяє відвідувачу виконати над ними дію.

```
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

```
[HttpGet]
0 references
public async Task<IActionResult> ShowStats()
{
    var visitor = new ListeningStatsVisitor();
    var allTracks = await _trackRepo.GetAll().ToListAsync();
    var allStreams = await _streamRepo.GetAll().ToListAsync();
    var allQueueItems = await _queueRepo.GetAll().ToListAsync();
    foreach (var track in allTracks) track.Accept(visitor);
    foreach (var stream in allStreams) stream.Accept(visitor);
    foreach (var item in allQueueItems) item.Accept(visitor);
    return View(visitor);
}
```

Рисунок 5 - Клієнтський код (ShowStats в AdminController.cs)

```
<div class="row">
    <div class="col-md-4">
        <div class="card text-white bg-primary mb-3">
            <div class="card-header">Загальна тривалість</div>
            <div class="card-body">
                <h1 class="card-title">@Model.TotalListenedMinutes</h1>
                <p class="card-text">хвилин контенту в базі (всі треки)</p>
            </div>
        </div>
    </div>
    <div class="col-md-4">
        <div class="card text-white bg-success mb-3">
            <div class="card-header">Всього треків у плейлистах</div>
            <div class="card-body">
                <h1 class="card-title">@Model.TotalTracks</h1>
                <p class="card-text">всього треків додано до плейлистів</p>
            </div>
        </div>
    </div>
    <div class="col-md-4">
        <div class="card text-white bg-info mb-3">
            <div class="card-header">Проведено DJ-сесій</div>
            <div class="card-body">
                <h1 class="card-title">@Model.TotalSessions</h1>
                <p class="card-text">зареєстровано стрімів від Діджеїв</p>
            </div>
        </div>
    </div>
</div>
```

Рисунок 6 - Відображення

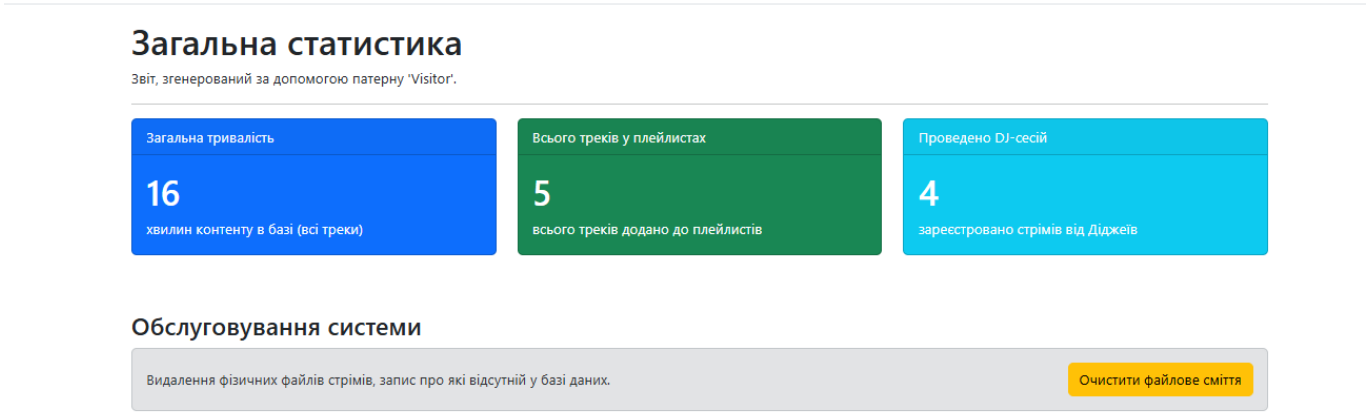


Рисунок 7 – Згенерована сторінка статистики на основі даних, зібраних об'єктом ListeningStatsVisitor

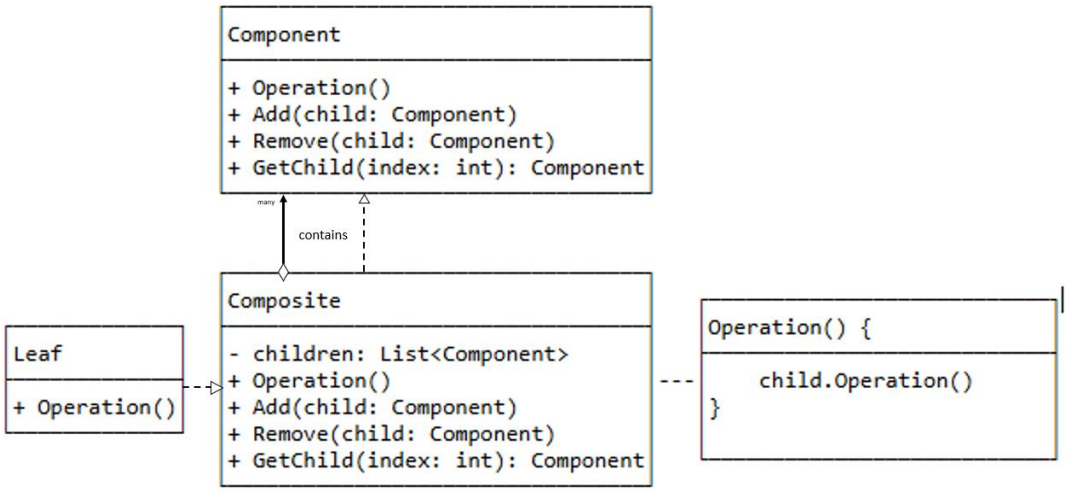
4. Висновок

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. Клієнтський код (контролер) лише ініціює процес обходу, отримуючи на виході готовий об'єкт зі звітом, який легко відобразити на веб-сторінці..

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».

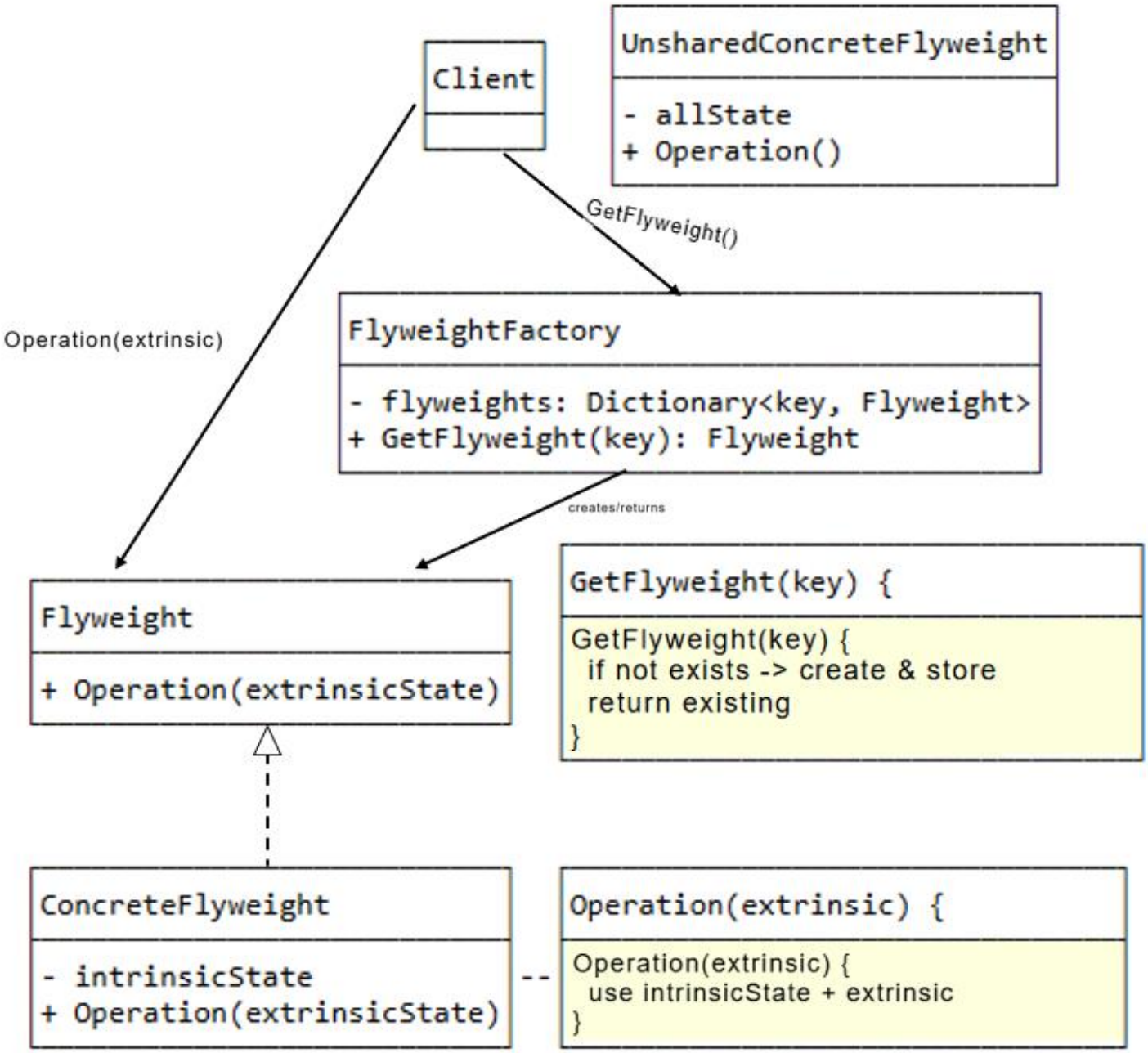


3. Які класи входять в шаблон «Композит», та яка між ними взаємодія?

- 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».



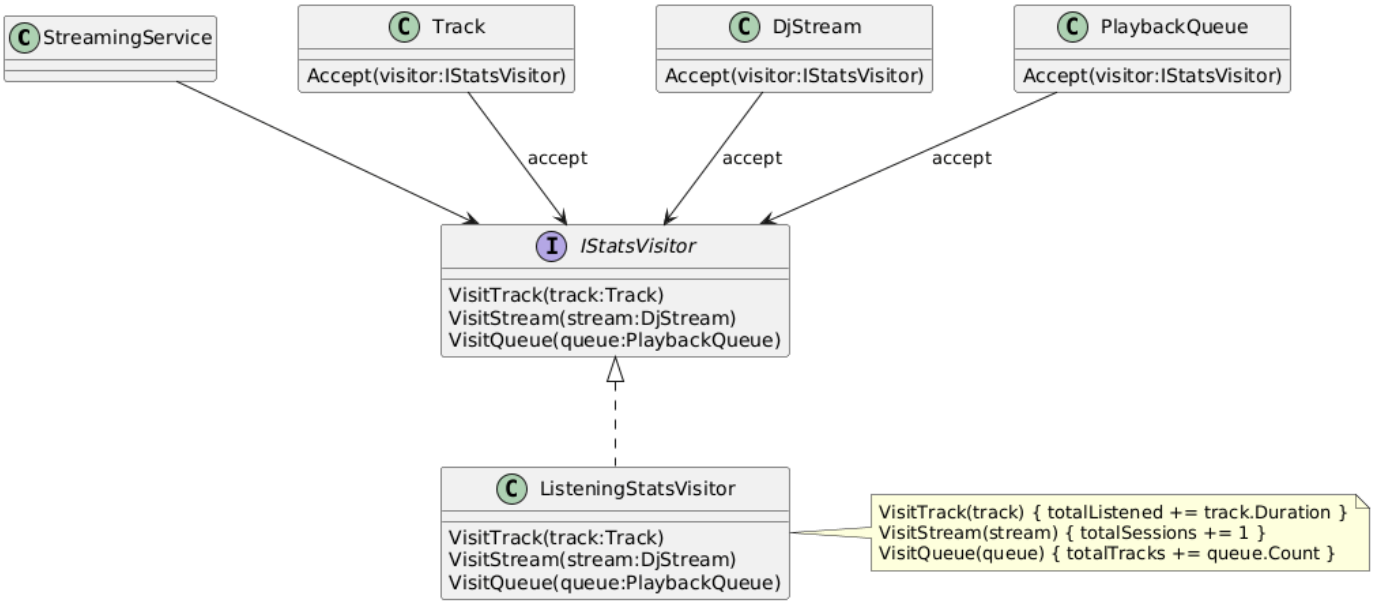
6. Які класи входять в шаблон «Легковаговик», та яка між ними взаємодія?

- 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.