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:
Трегуб К. В.

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:

Composite — застосовується для побудови деревоподібних структур типу «частина–ціле», дозволяючи однаково працювати як з окремими елементами, так і з їх контейнерами. Приклад — структура проєкту: функції, user stories, задачі.

Flyweight — зменшує використання пам’яті шляхом розділення спільного внутрішнього стану між багатьма об’єктами, залишаючи зовнішній стан у контексті. Приклад — відображення символів тексту чи простих графічних елементів.

Interpreter — використовується для опису та виконання граматики спеціалізованої мови за допомогою рекурсивного дерева виразів. Приклади — пошук за шаблоном або прості скриптові мови.

Visitor — дає можливість додавати нові операції над об’єктами без зміни їхніх класів, відокремлюючи їхню структуру від логіки обробки. Приклад — обчислення вартості товарів у кошику з урахуванням різних правил знижок.

3. Хід роботи:

1) Ознайомитись з короткими теоретичними відомостями.
2) Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
3) Реалізувати один з розглянутих шаблонів за обраною темою.
4) Реалізувати не менше 3-х класів відповідно до обраної теми.
5) Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму класів, яка представляє використання шаблону в реалізації системи, навести фрагменти коду по реалізації цього шаблону.

Тема :

Beauty Salon Booking System (State, Chain of Responsibility, Observer, Facade, Composite, Client-Server) — веб-застосунок для бронювання послуг салону

краси, що підтримує управління життєвим циклом запису, проходження валідованого процесу створення бронювання, автоматичне сповіщення користувачів про зміни статусу, централізовану обробку оплати через фасад та роботу як з окремими послугами, так і з пакетними пропозиціями. Система побудована за архітектурою клієнт–сервер з розмежуванням бізнес-логіки, управління станом і відображенням інтерфейсу.

Салон краси часто пропонує клієнтам не лише окремі послуги (наприклад, "Стрижка"), але й комплексні пакети (наприклад, "Стрижка + Миття голови + Укладка" або "SPA-день"). З точки зору клієнтського коду (системи бронювання), робота з пакетом і з окремою послугою має бути однаковою: нам потрібно знати загальну ціну та тривалість. Без використання патерну довелося б писати складну логіку перевірок: "Якщо це пакет, то просумуй елементи, а якщо послуга — візьми ціну".

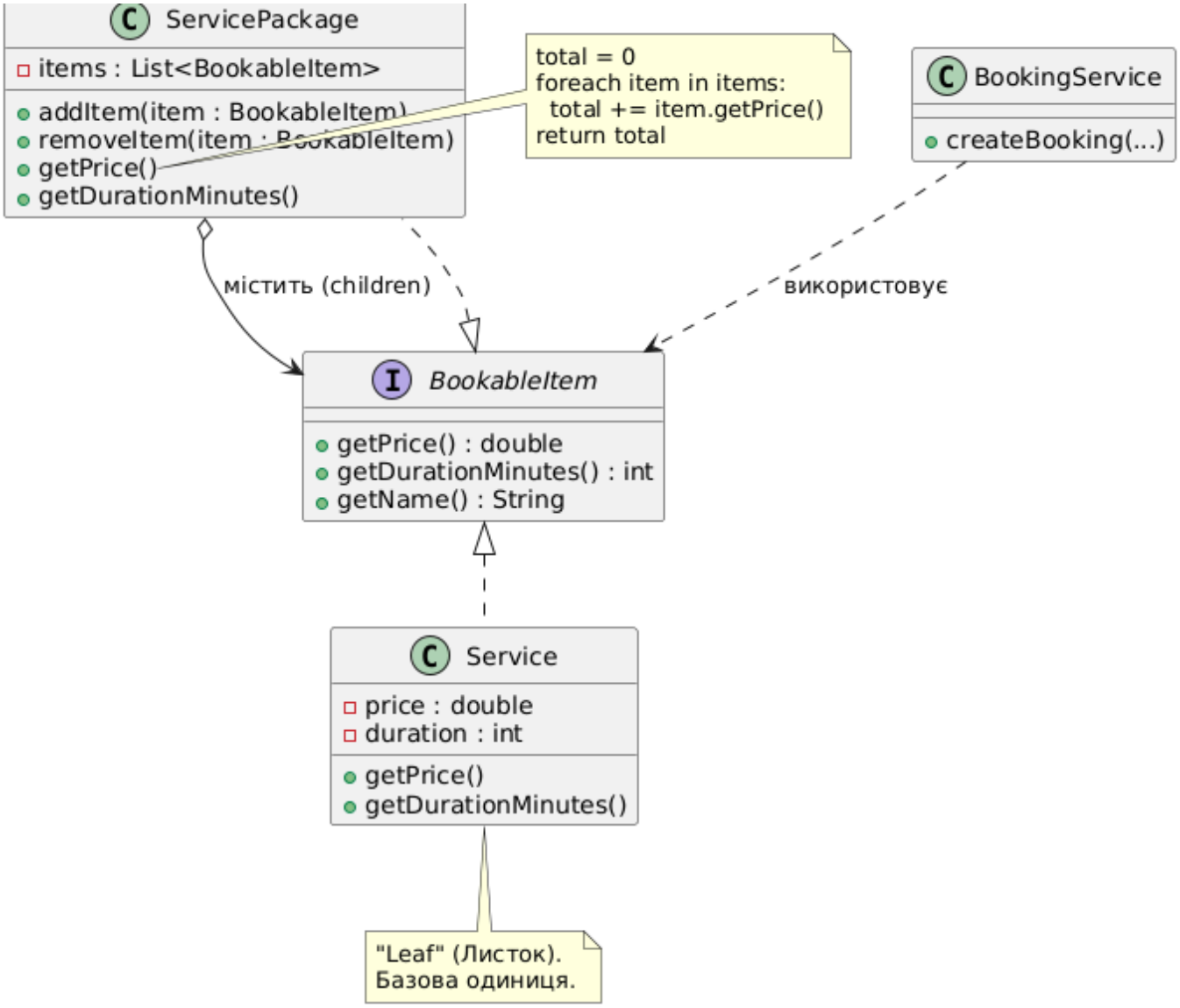


Рисунок 1 - Структура патерну Composite

- **BookableItem (Component)** – спільний інтерфейс для всіх об'єктів, які можна забронювати. Він оголошує ключові методи: getPrice() та getDurationMinutes(). Це дозволяє клієнту ігнорувати відмінності між простими та складеними об'єктами.
- **Service (Leaf)** – представляє просту послугу (Листок дерева). У нього немає підоб'єктів. Він виконує основну роботу: повертає свою вартість та тривалість, що зберігаються в базі даних.
- **ServicePackage (Composite)** – представляє пакет послуг (Вузол дерева). Він зберігає список дочірніх елементів типу BookableItem. При виклику getPrice(), він не має власної ціни, а делегує виклик усім дочірнім елементам, підсумовує результати і повертає загальну суму.
- **BookingService (Client)** – працює з об'єктами через інтерфейс BookableItem. Йому байдуже, чи бронює клієнт одну послугу, чи складний пакет — він просто викликає getPrice() для розрахунку суми замовлення

```java
package com.beautysalon.booking.composite;

public interface BookableItem {
    double getPrice();
    int getDurationMinutes();
    String getName();
}
```

Рисунок 2 - Інтерфейс Компонента (BookableItem.java)

```java
public class ServicePackage implements BookableItem {
    private String name;
    private List<BookableItem> items = new ArrayList<>();

@Override
public String getName() {
    return this.name;
}

@Override
public double getPrice() {
    double totalPrice = 0;
    for (BookableItem item : items) {
        totalPrice += item.getPrice();
    }
    return totalPrice;
}

@Override
public int getDurationMinutes() {
    int totalDuration = 0;
    for (BookableItem item : items) {
        totalDuration += item.getDurationMinutes();
    }
    return totalDuration;
}
```

Рисунок 3 - ServicePackage.java

```
BookableItem finalItem;
BookableItem baseService = context.getService();

if (allInclusive) {
    ServicePackage vipPackage = new ServicePackage("VIP-пакет: " + baseService.getName());
    vipPackage.addItem(baseService);

    com.beautysalon.booking.entity.Service addons =
        new com.beautysalon.booking.entity.Service(name: "VIP-додатки (Косметика, Масаж, Напої

    vipPackage.addItem(addons);
    finalItem = vipPackage;
} else {
    finalItem = baseService;
}

Booking newBooking = new Booking();
newBooking.setClient(context.getClient());
newBooking.setMaster(context.getMaster());
newBooking.setService(context.getService());
newBooking.setBookingDate(context.getDateTime().toLocalDate());
newBooking.setBookingTime(context.getDateTime().toLocalTime());
newBooking.setTotalPrice(finalItem.getPrice());
newBooking.setStatus(BookingStatus.PENDING);
```

Рисунок 4 - Використання в бізнес-логіці (BookingService.java)

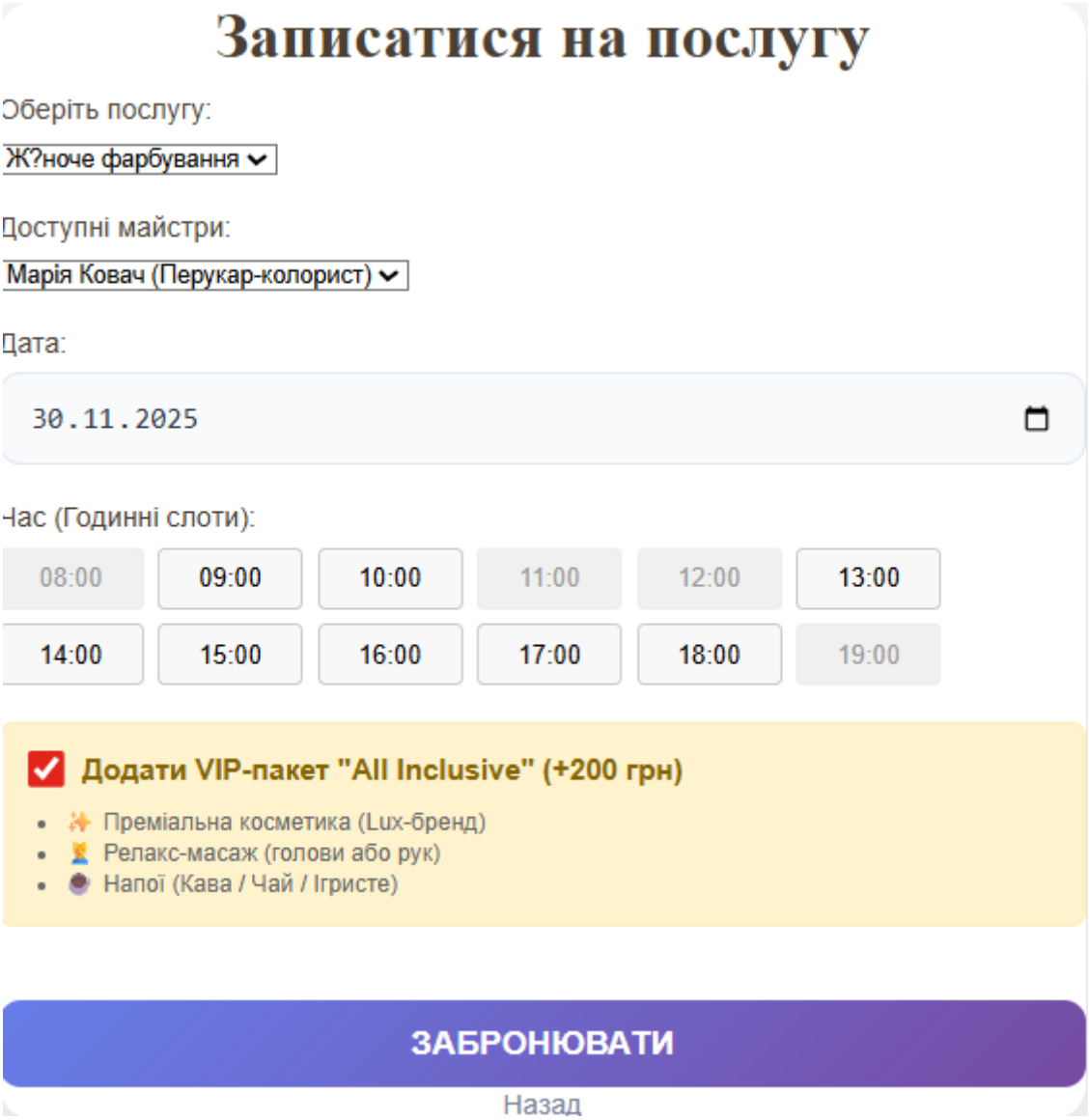


Рисунок 5 – при створенні бронювання вмикаємо пакет з віп-послугами

Рисунок 6 – Бачимо що цвартість бронювання на 200 грн збільшилась, ніж встановлена ціна

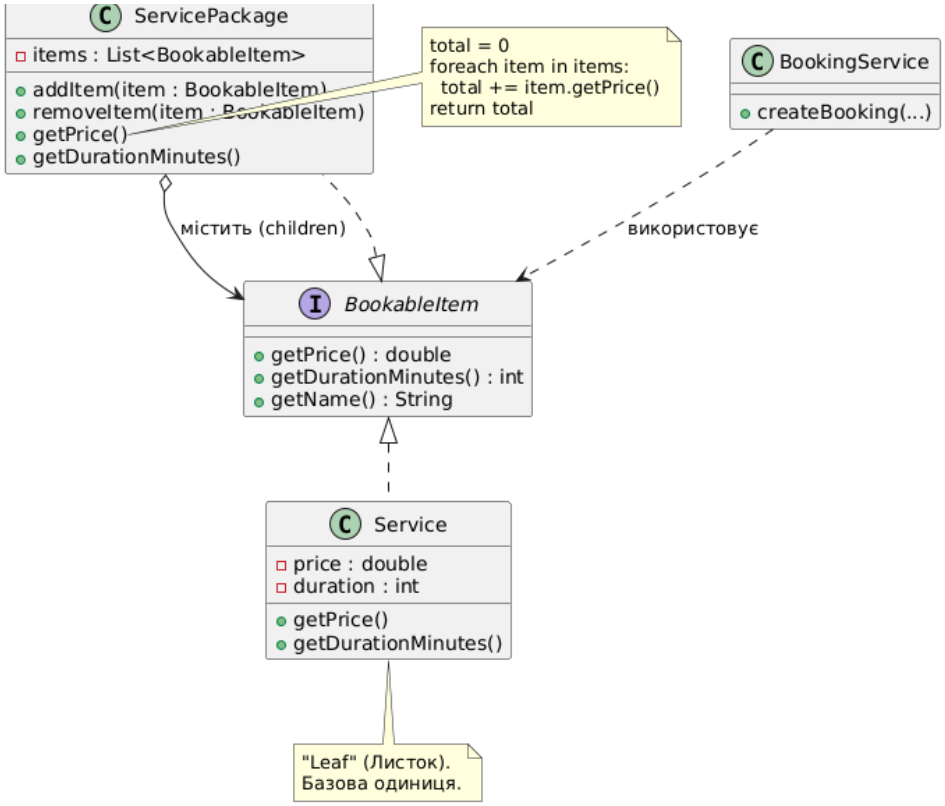
4. Висновок

У ході виконання лабораторної роботи були опрацьовані структурні патерни проєктування, зокрема Composite, Flyweight, Interpreter та Visitor. Для системи бронювання було реалізовано патерн Composite, який дав змогу сформувати гнучкий механізм пакетних пропозицій на кшталт «All Inclusive». Прості послуги (Service) були об’єднані у складені структури (ServicePackage) та представлені через спільний інтерфейс. Застосування цього патерну забезпечило уніфіковану роботу з різними типами об’єктів: клієнтському коду більше не потрібно розрізняти, чи є елемент окремою послугою, чи цілим пакетом. Такий підхід підвищив гнучкість системи — нові комплексні пропозиції можуть легко формуватися шляхом комбінування існуючих послуг. Крім того, дерево пакетів може мати будь-яку глибину, а обчислення вартості чи тривалості виконуються коректно завдяки рекурсивній природі Composite, що робить систему масштабованою та зручною в супроводі.

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».

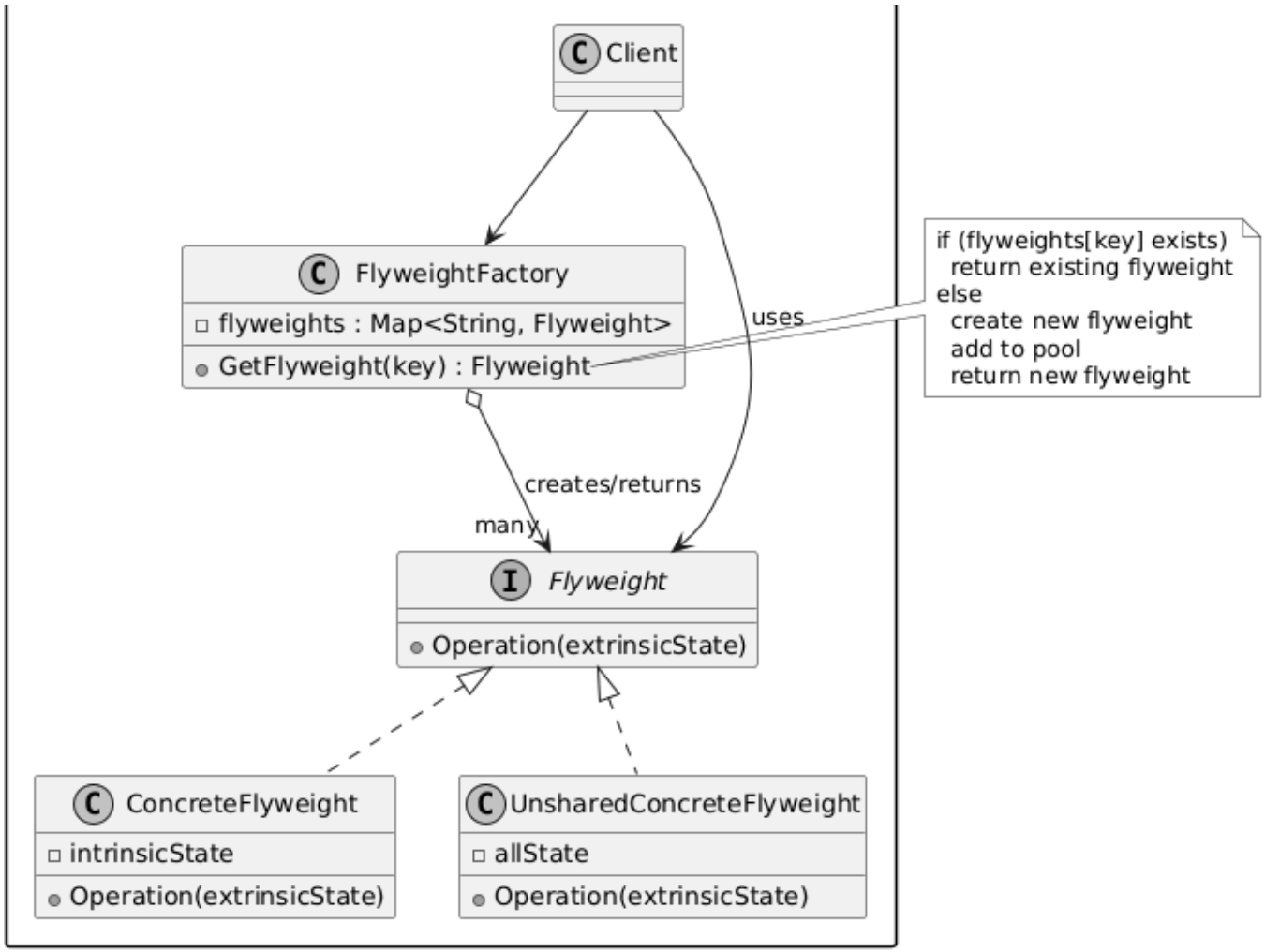


3. Які класи входять в шаблон «Композит», та яка між ними взаємодія?

- 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?

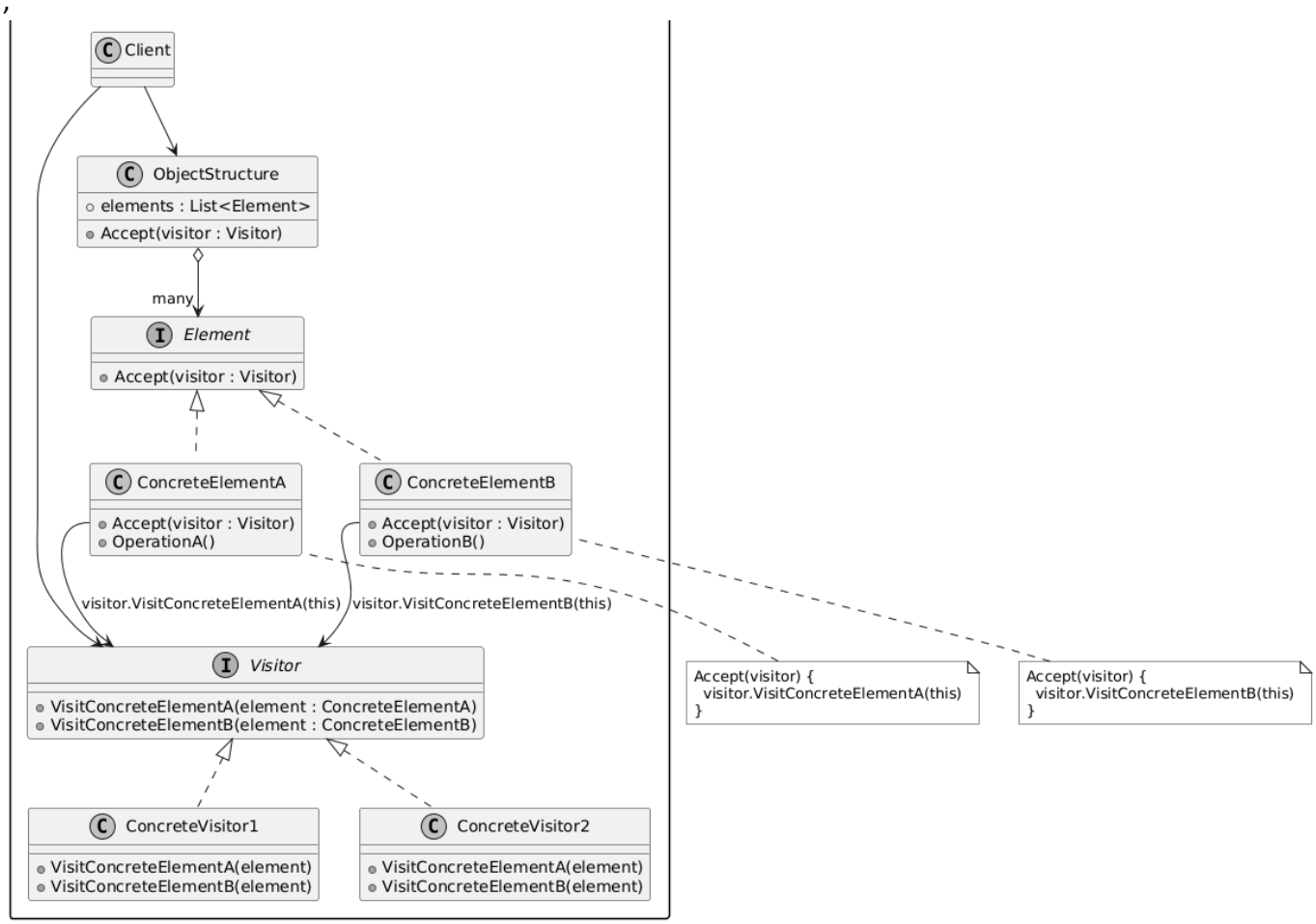
- Flyweight — інтерфейс з операцією, що приймає зовнішній стан.
- ConcreteFlyweight — зберігає внутрішній стан, спільний.
- UnsharedConcreteFlyweight — не поділюваний варіант.

- 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.