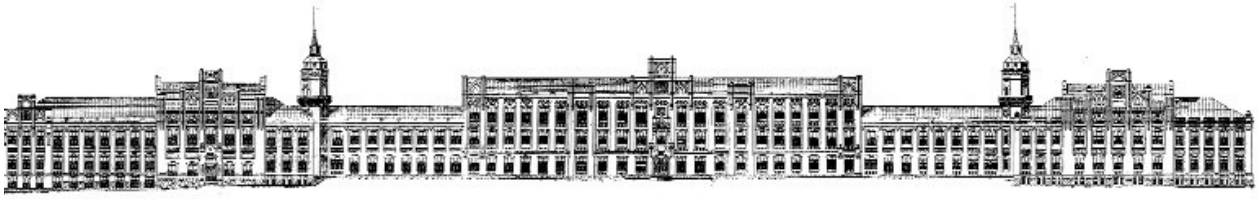

Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №8

з дисципліни «Технології розроблення

програмного забезпечення»

Тема: «Патерни проектування»

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

Зміст

Table of Contents

Вступ

Темою роботи є «Вступ до патернів проєктування». Мета цієї лабораторної це вивчити структуру шаблонів «Composite», «Flyweight» (Пристосуванець), «Interpreter», «Visitor» та навчитися застосовувати їх в реалізації програмної системи.

Як об’єкт проєктування обрано тему «Web crawler»: програмний сканер, що розпізнає структуру сторінок сайту, переходить за посиланнями, збирає інформацію за заданим терміном, видаляє несемантичні елементи (рекламу, скрипти тощо), зберігає очищені дані у вигляді структурованого набору HTML-файлів і веде статистику відвідань та метадані.

1 Теоретичні відомості

1.1. Шаблон «Composite»

Шаблон «Composite» Призначення: Шаблон використовується для складання об'єктів в деревоподібну структуру для подання ієрархій типу «частина цілого». Даний шаблон дозволяє уніфіковано обробляти як поодинокі об'єкти, так і об'єкти з вкладеністю [6]. Простим прикладом може служити складання компонентів всередині звичайної форми. Форма може містити дочірні елементи (поля для введення тексту, цифр, написи, малюнки тощо); дочірні елементи можуть в свою чергу містити інші елементи. Наприклад, при виконанні операції розтягування форми необхідно, щоб вся ієрархія розтягнулася відповідним чином. В такому випадку 90 форма розглядається як композитний об'єкт і операція розтягування застосовується до всіх дочірніх елементів рекурсивно. Даний шаблон зручно використовувати при необхідності подання та обробки ієрархій об'єктів. Крім того, патерн «Composite» (Компонувальник) краще використовувати, коли ви представляєте структуру даних у вигляді дерева.

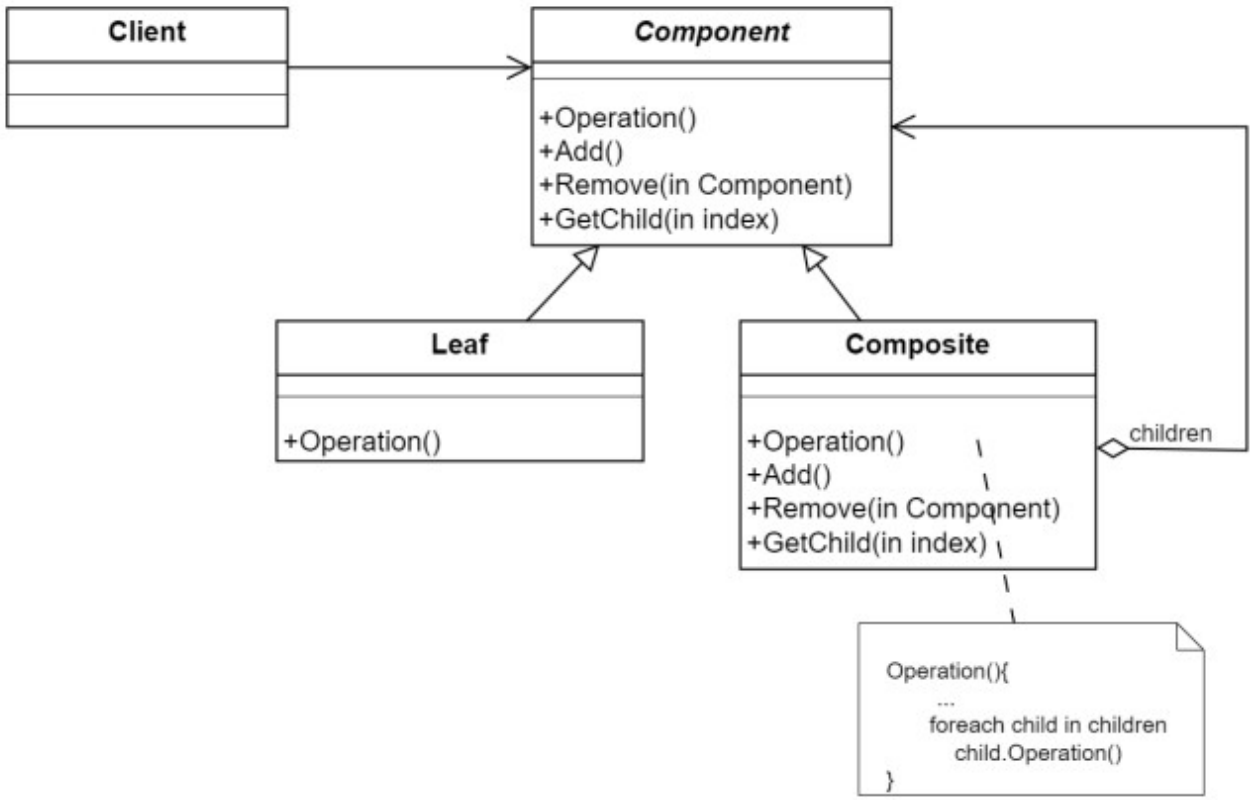

Рисунок 1.1. Структура патерну «Компонувальник»

1.2. Шаблон «Flyweight»

Призначення: Шаблон використовується для зменшення кількості об'єктів в додатку шляхом поділу цих об'єктів між ділянками додатку. Flyweight являє собою поділюваний об'єкт. Дуже важливою є концепція «внутрішнього» і «зовнішнього» станів. Внутрішній стан відображає дані, характерні саме поділюваному об'єкту (наприклад, код букви); зовнішній стан несе інформацію про його застосування в додатку (наприклад, рядок і стовпчик). Внутрішній стан зберігається в самому поділюваному об'єкті, зовнішній – в об'єктах додатку (контексту використання поділюваного об'єкта).

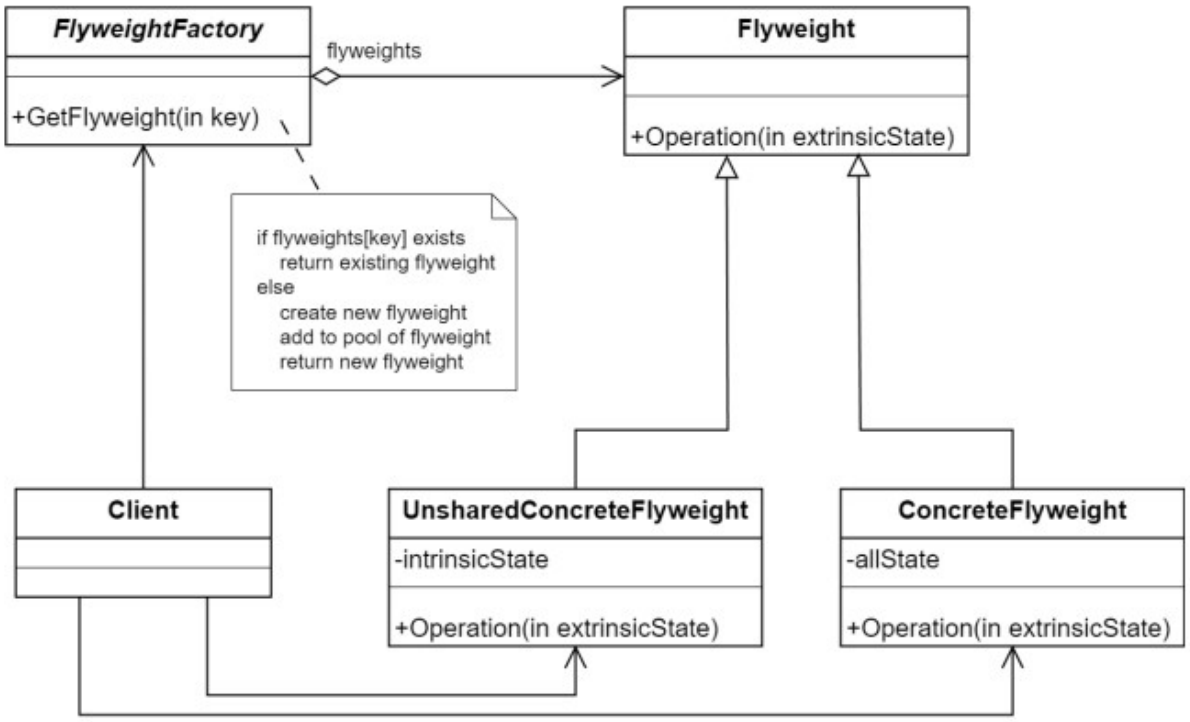

Рисунок 1.2. Структура патерну Flyweight (Легковаговик)

1.3. Шаблон «Interpreter»

Призначення: Даний шаблон використовується для подання граматики і інтерпретатора для вибраної мови (наприклад, скриптової). Граматика мови представлена термінальними і нетермінальними символами, кожен з яких інтерпретується в контексті використання. Клієнт передає контекст і сформовану пропозицію в використовувану мову в термінах абстрактного синтаксичного дерева (деревоподібна структура, яка однозначно визначає ієрархію виклику підвиразів), кожен вираз інтерпретується окремо з використанням контексту. У разі наявності дочірніх виразів, батьківський вираз

інтерпретує спочатку дочірні (рекурсивно), а потім обчислює результат власної операції.

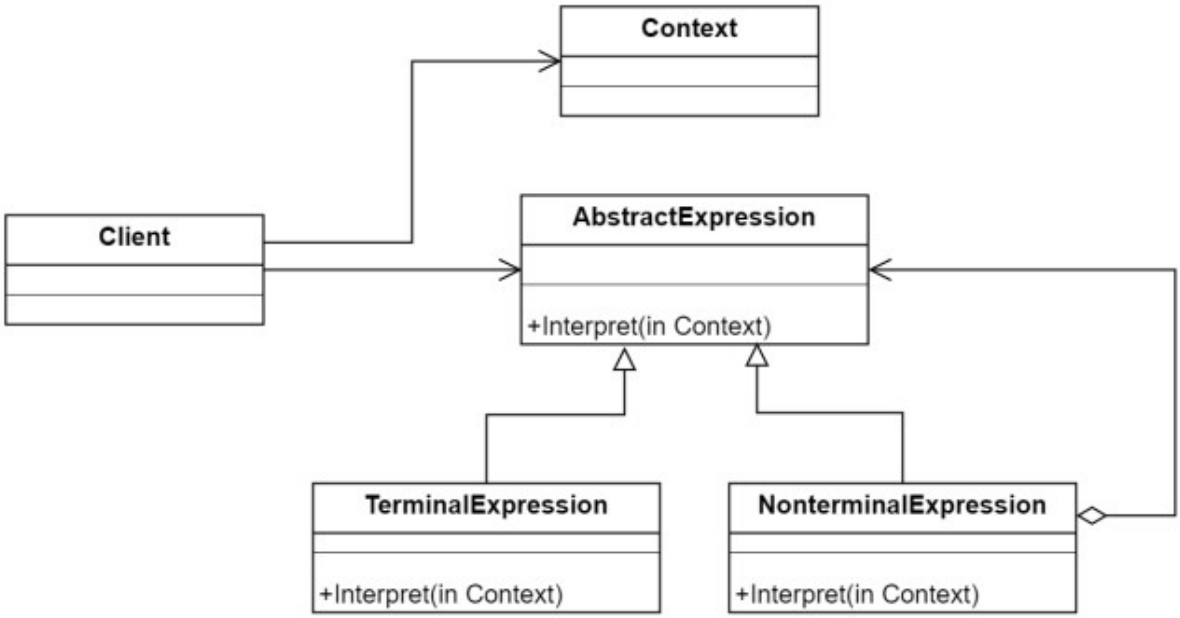

Рисунок 1.3. Структура патерна «Інтерпретатор»

1.4. Шаблон «Visitor»

Призначення: Шаблон відвідувач дозволяє вказувати операції над елементами без зміни структури конкретних елементів. Таким чином вкрай зручно додавати нові операції, проте дуже важко додавати нові елементи в ієрархію (необхідно додавати відповідні методи для обробки їх відвідувань в кожного відвідувача). Даний шаблон дозволяє групувати однотипні операції, що застосовуються над різнотипними об'єктами.

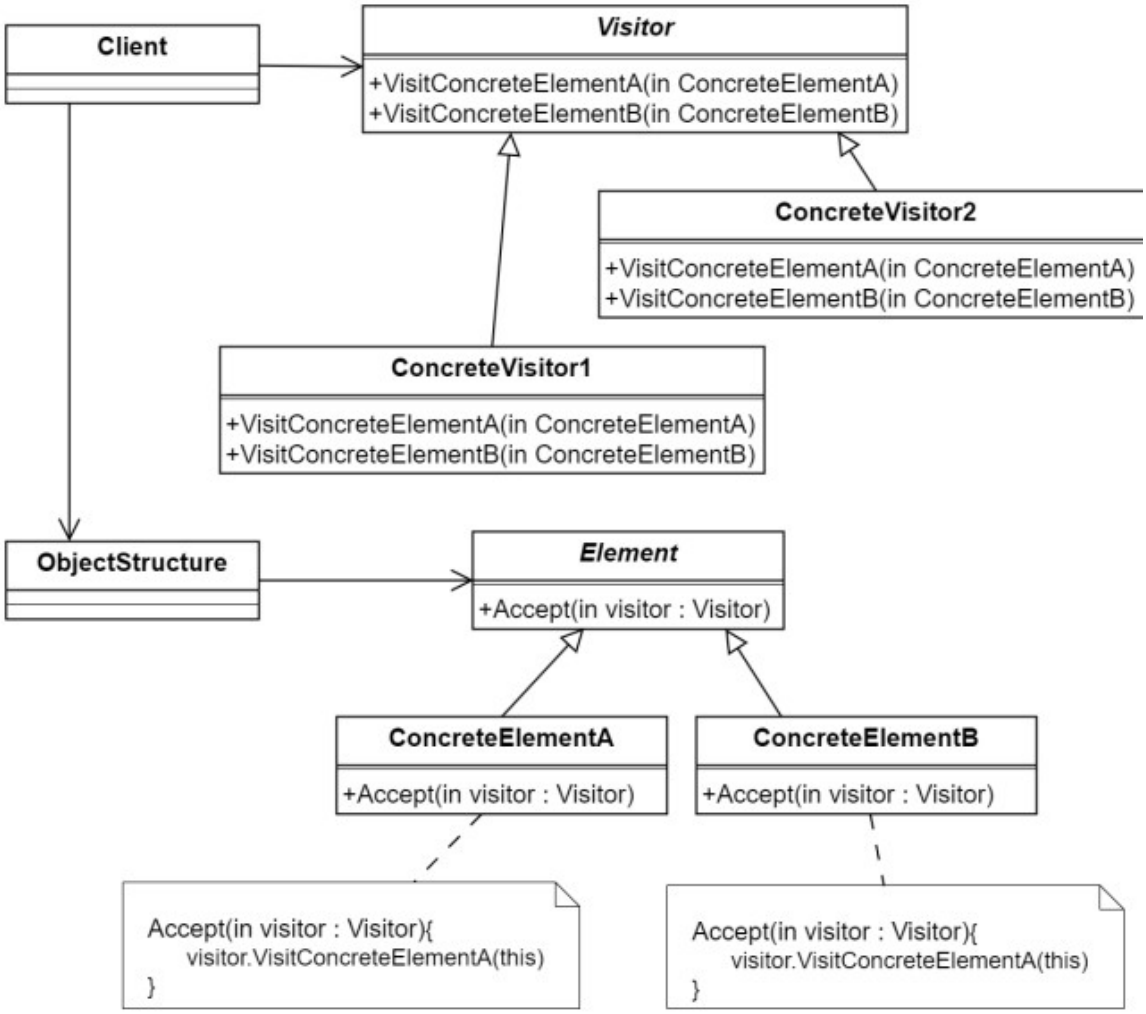

Рисунок 1.4.Структура патерна «Відвідувач»

2 Хід роботи

2.1. Використання Composite для побудови і виводу дерева сайту (site map) на основі результатів обходу.

Завдання

- Описати інтерфейс компонента SiteComponent.
- Реалізувати композит SiteGroup та листок PageLeaf.
- Побудувати дерево за URL-адресами та виводити його в консоль.

2.2 Діаграма класів (фрагмент)

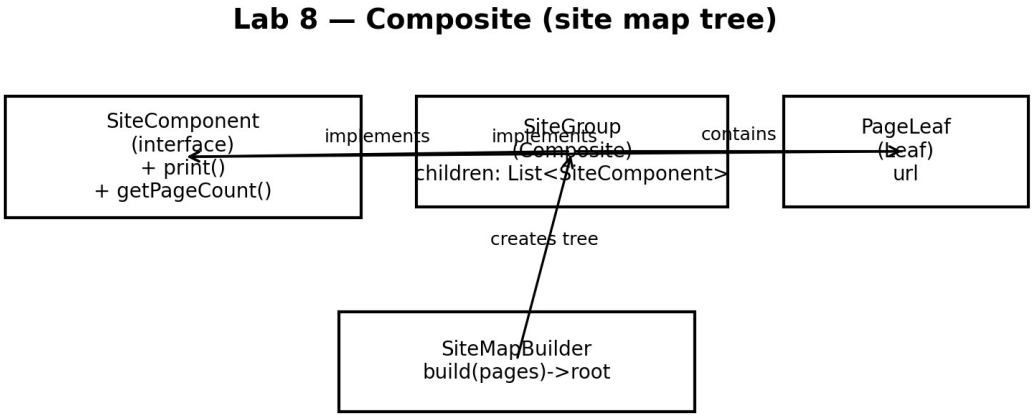

Рисунок 2.2 - Діаграма класів (фрагмент)

2.3. Фрагмент коду (Java)

SiteComponent (інтерфейс компонента)

```
package ua.kpi.webcrawler.composite;

public interface SiteComponent {
   String getName();
   int getPageCount();
   void print(String indent);
}
```

SiteMapBuilder: побудова дерева

```
   public SiteGroup build(List<PageData> pages) {
      SiteGroup root = new SiteGroup("root");
      Map<String, SiteGroup> hosts = new HashMap<>();
```

```
    for (PageData p : pages) {
        String url = p.getUrl();
        try {
            URI uri = new URI(url);
            String host = uri.getHost() == null ? "unknown-host" : uri.getHost();
            SiteGroup hostGroup = hosts.computeIfAbsent(host, h -> {
                SiteGroup g = new SiteGroup(h);
                root.add(g);
                return g;
            });

            String path = uri.getPath() == null ? "" : uri.getPath();
            if (path.isEmpty() || "/".equals(path)) {
                hostGroup.add(new PageLeaf(url));
                continue;
            }

            String[] parts = path.split("/");
            SiteGroup current = hostGroup;
            for (String part : parts) {
                if (part == null || part.isBlank()) continue;
                SiteGroup next = findOrCreateGroup(current, part);
                current = next;
            }
            current.add(new PageLeaf(url));
        } catch (Exception e) {
            // fallback
            root.add(new PageLeaf(url));
        }
    }

    return root;
}

private SiteGroup findOrCreateGroup(SiteGroup parent, String name) {
    for (SiteComponent c : parent.getChildren()) {
        if (c instanceof SiteGroup && c.getName().equals(name)) {
            return (SiteGroup) c;
        }
    }
    SiteGroup g = new SiteGroup(name);
    parent.add(g);
    return g;
```

```
    }
}
```

Висновки

У ході виконання лабораторної роботи було використано Composite для побудови і виводу дерева сайту (site map) на основі результатів обходу. Composite дозволив представити результат обходу у вигляді дерева (host → сегменти шляху → сторінки) та уніфіковано обробляти як групи, так і окремі сторінки.

Контрольні запитання

1. Яке призначення шаблону «Композит»?

Шаблон «Композит» (Composite) призначений для побудови деревоподібних структур об’єктів і роботи з ними однаковим способом. Він дозволяє клієнту однаково обробляти як окремі об’єкти (листки), так і їхні групи (композиції).

2. Нарисуйте структуру шаблону «Композит».

Структура включає:

- Component — спільний інтерфейс для всіх елементів;
- Leaf — «листок», одиночний елемент без дітей;
- Composite — складений елемент, який містить колекцію Component;
- Client — працює з об’єктами через інтерфейс Component.

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

У шаблон входять Component, Leaf, Composite та Client. Client звертається до будь-якого елемента як до Component. Leaf виконує операцію напряму. Composite зберігає список дочірніх Component і делегує їм виконання операцій (наприклад, викликає операцію для кожного дочірнього елемента).

4. Яке призначення шаблону «Легковаговик»?

Шаблон «Легковаговик» (Flyweight) призначений для економії пам’яті при роботі з великою кількістю схожих об’єктів шляхом винесення спільного стану в спільно використовувані об’єкти та передачі змінного (зовнішнього) стану ззовні.

5. Нарисуйте структуру шаблону «Легковаговик».

Структура включає:

- Flyweight — інтерфейс легковаговика;
- ConcreteFlyweight — зберігає внутрішній (спільний) стан;
- FlyweightFactory — створює та кешує легковаговики;
- Client — запитує легковаговики у фабрики і передає зовнішній стан під час виклику методів.

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

У шаблон входять Flyweight/ConcreteFlyweight, FlyweightFactory і Client. Client звертається до FlyweightFactory, щоб отримати легковаговик для певного ключа. Фабрика повертає вже існуючий об’єкт або створює новий і кешує його. Під час виклику операцій клієнт передає зовнішній стан, який не зберігається всередині легковаговика.

7. Яке призначення шаблону «Інтерпретатор»?

Шаблон «Інтерпретатор» (Interpreter) призначений для опису граматики простої мови та побудови механізму інтерпретації (обчислення) виразів цією мовою. Він часто застосовується для розбору і виконання правил, фільтрів, простих мов запитів або математичних виразів.

8. Яке призначення шаблону «Відвідувач»?

Шаблон «Відвідувач» (Visitor) дозволяє додавати нові операції над об’єктною структурою (наприклад, деревом) без зміни класів елементів цієї структури. Логіка операцій переноситься в окремі класи-відвідувачі.

9. Нарисуйте структуру шаблону «Відвідувач».

Структура включає:

- Visitor — інтерфейс з методами visit для різних типів елементів;
- ConcreteVisitor — конкретні реалізації операцій;
- Element — інтерфейс елемента з методом accept(Visitor);
- ConcreteElementA/B — конкретні елементи структури;
- ObjectStructure (необов’язково) — колекція/дерево елементів, по яких проходить відвідувач.

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

У шаблон входять Element/ConcreteElement, Visitor/ConcreteVisitor та, за потреби, ObjectStructure. Клієнт створює ConcreteVisitor і запускає обхід структури. Кожен елемент викликає accept(visitor), а всередині accept відбувається виклик visitor.visit(this). Це забезпечує подвійне диспетчеризування: виконується правильний метод visit залежно від типу елемента.