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
Технологія розроблення програмного забезпечення

Патерни проектування

Виконав
студент групи ІА-31
Даценко Влад

Перевірив:
Мягкий Михайло
Юрійович

Київ 2025р.

Тема: Патерни проектування.

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

Завдання.

1. Ознайомитися 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.

..25 Installer generator (iterator, builder, factory method, bridge, interpreter, client-server)

Генератор інсталяційних пакетів повинен мати якийсь спосіб налаштування файлів, що входять в установку, установки вікон з інтерактивними можливостями (галочка - створити ярлик на робочому столі; ввести в текстове поле деякі дані, наприклад, ліцензійний ключ і т.д.). Генератор повинен вивести один файл .exe або .msi.

Теоретичні відомості

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.

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 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.

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).

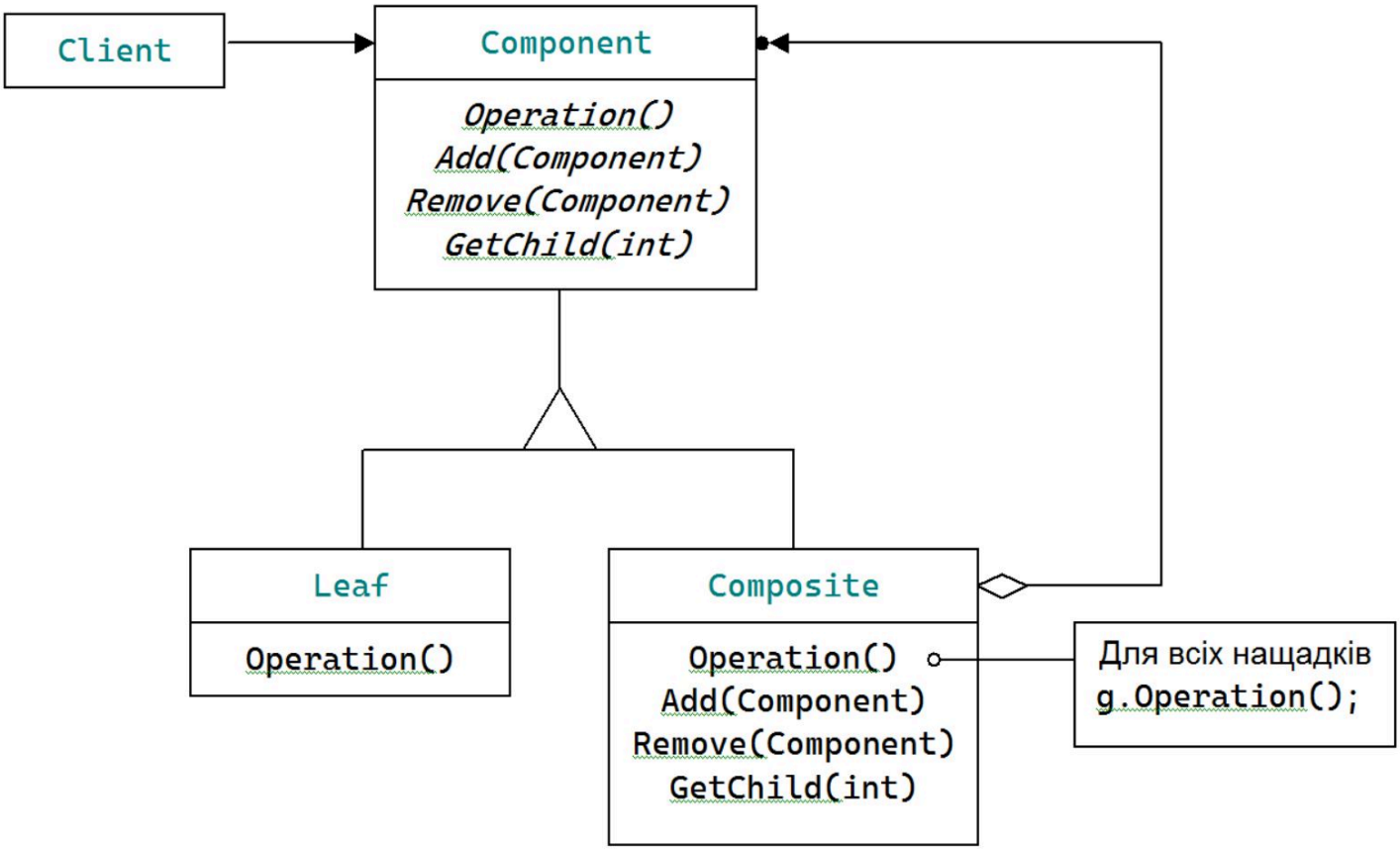
Interpreter

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. Шаблон зручно використовувати в разі невеликої граматики (інакше розростеться кількість використовуваних класів) і відносно простого контексту (без взаємних залежностей і т.п.). Даний шаблон визначає базовий каркас інтерпретатора, який за допомогою рекурсії повертає результат обчислення пропозиції на основі результатів окремих елементів.

Хід роботи

В моїй лабораторній роботі розглянуто патерн Composite.

Загальна структура патерну



Я вирішив використати патерн composite, щоб організувати ієрархічну структуру об’єктів, де як окремі елементи (листки), так і групи елементів (композити) обробляються однаково. Це дозволяє мені створювати складні дерева об’єктів, у яких кожен вузол може бути як простим (Leaf), так і складеним (Composite).

Інтерфейс Component визначає спільний метод operation(), який реалізують як Leaf, так і Composite. Клас Leaf відповідає за базову операцію — наприклад,

вивід імені елемента. Клас Composite містить список дочірніх компонентів (які можуть бути як Leaf, так і інші Composite) і реалізує методи для додавання, видалення та обробки всіх дочірніх елементів.

Завдяки такій структурі я можу легко додавати нові типи елементів або групувати їх у довільні дерева, не змінюючи код клієнта. Наприклад, CompositeDemo створює дерево з кількох листків і підкомпозитів, а виклик operation() для кореневого Composite рекурсивно обробляє всю структуру.

Таке рішення спрощує роботу з ієрархічними даними, дозволяє розширювати функціонал без зміни

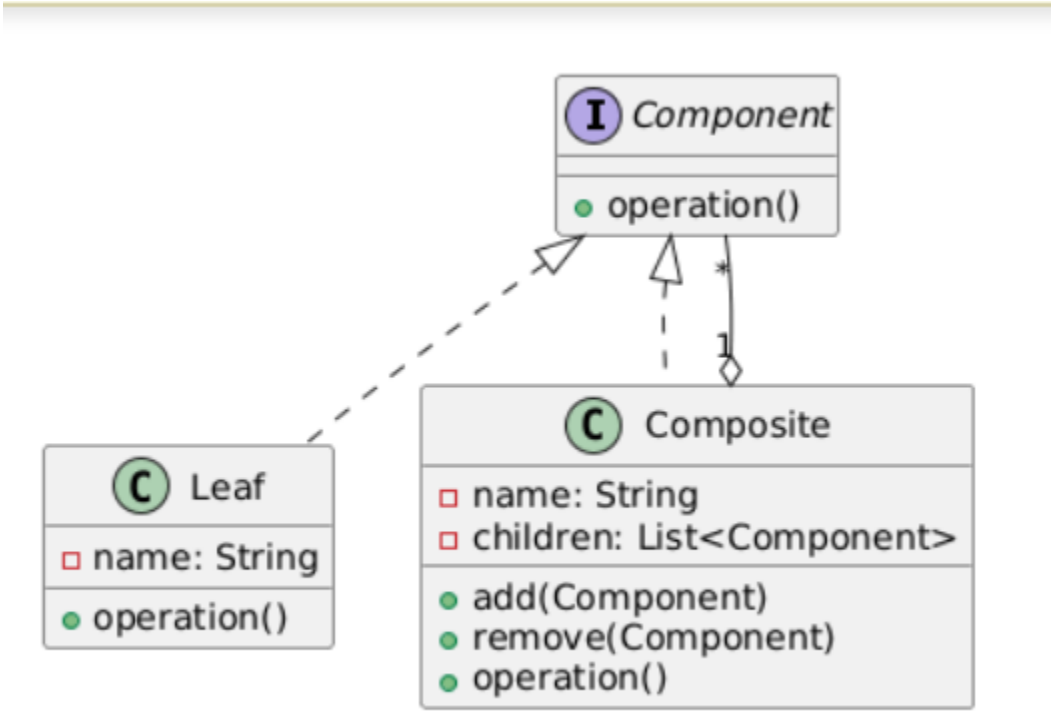


Рисунок 1. Структура патерну Composite

```java
public interface Component {
    void operation();
}
```

Рисунок 2. Інтерфейс Component

```java
public class Leaf implements Component {
    private final String name;
    public Leaf(String name) {
        this.name = name;
    }
    @Override
    public void operation() {
        System.out.println("Leaf: " + name);
    }
}
```

Рисунок 3.Клас Leaf

```java
public class Composite implements Component {
    private final String name;
    private final List<Component> children = new ArrayList<>();
    public Composite(String name) {
        this.name = name;
    }
    public void add(Component component) {
        children.add(component);
    }
    public void remove(Component component) {
        children.remove(component);
    }
    @Override
    public void operation() {
        System.out.println("Composite: " + name);
        for (Component child : children) {
            child.operation();
        }
    }
}
```

Рисунок 4. Клас Composite

```java
public class CompositeDemo {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Component leaf1 = new Leaf(name: "A");
        Component leaf2 = new Leaf(name: "B");
        Composite composite = new Composite(name: "Root");
        composite.add(leaf1);
        composite.add(leaf2);
        Composite subComposite = new Composite(name: "Sub");
        subComposite.add(new Leaf(name: "C"));
        composite.add(subComposite);
        composite.operation();
    }
}
```

Рисунок 5. Клас CompositeDemo

Висновок: У ході виконання лабораторної роботи було реалізовано шаблон проєктування Composite, що дозволяє організувати ієрархічну структуру об’єктів, у якій як окремі елементи (листки), так і групи елементів (композити) обробляються однаково. Основна ідея використання цього патерну полягає у створенні єдиного інтерфейсу для роботи з простими та складеними об’єктами, що значно спрощує керування складними структурами.

Завдяки Composite стало можливим легко додавати нові типи елементів або групувати їх у довільні дерева без зміни коду клієнта. Це забезпечило гнучкість, масштабованість і зрозумілу структуру системи, а також спростило розширення функціоналу. Використання шаблону Composite в даній роботі є виправданим і продемонструвало свою ефективність для роботи з ієрархічними даними та організації взаємодії між різними компонентами системи.