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
Технології розроблення програмного забезпечення
«Патерни проектування»

Тема: Редактор зображень

Виконала:

студентка групи ІА-32

Ейсмонт А.В.

Перевірив:

Мягкий М.Ю.

Київ 2025

Зміст

Тема: Патерни проектування.

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

Завдання:

- Ознайомитись з короткими теоретичними відомостями.
- Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
- Реалізувати один з розглянутих шаблонів за обраною темою.
- Реалізувати не менше 3-х класів відповідно до обраної теми.
- 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

Теоретичні відомості:

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 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.

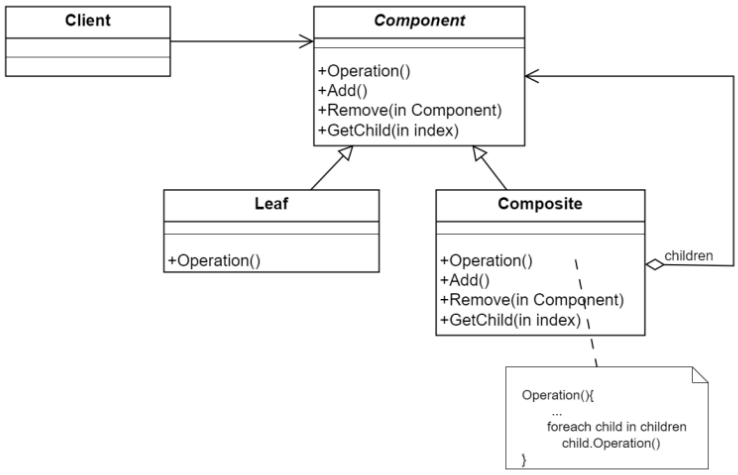


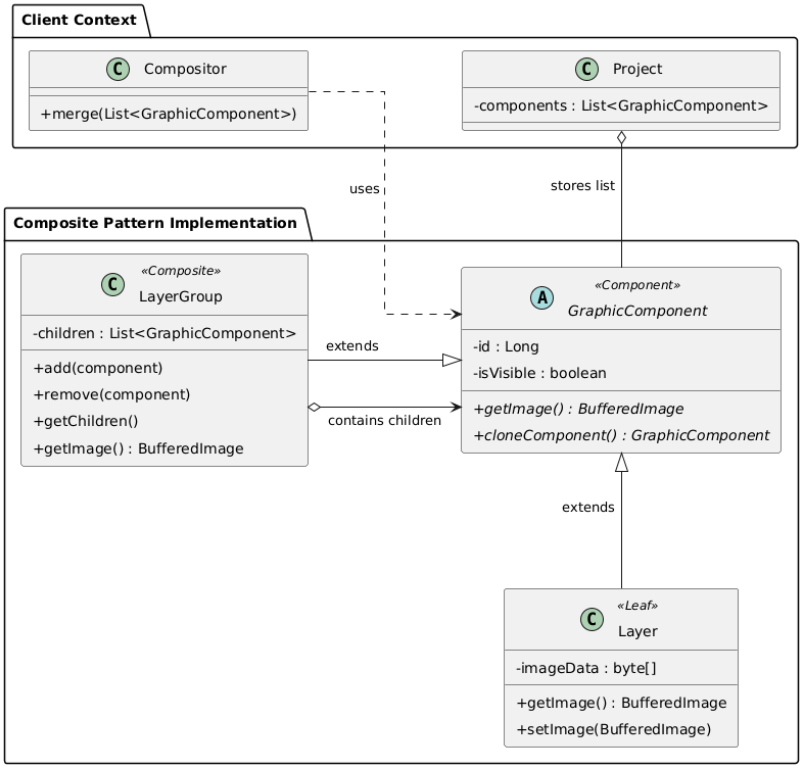
Рисунок 8.1. Структура патерна «Компонувальник»

Проблема: Ви розробляєте систему керування проєктами. Кожен проєкт складається із наборів функцій, кожна функція з userstory, а кожна userstory в свою чергу із задач по її реалізації. Ви реалізуєте функціонал відображення оціночної вартості робіт по кожній із функцій, а також відображення чи всі userstory були оцінені. Це потрібно бізнес-аналітики, щоб розуміти, що всі userstory розробниками були розглянуті і оцінені, а також обговорити необхідність реалізації того чи іншого функціоналу на основі попередньої оцінки. 91 Рішення: Крашим підходом в даній ситуації буде використання патерну Компоновщик. Класи що представляють функції, userstory, задачі будуть наслідуватися від одного інтерфейсу ITask, функції (Feature) та Userstory будуть складними об'єктами і міститимуть колекції об'єктів ITask, а задачі (Task) будуть представляли кінцеві об'єкти без дочірніх елементів. Для розрахунку оціночної вартосі робіт, в ITask інтефейс додаємо метод GetEstimatedPoints(). В класах-компоновщиках методи GetEstimatedPoints() реалізовуємо як обхід всіх дочірніх елементів та сумування результатів відповідей GetEstimatedPoints(). Таким чином, візуальні форми будуть працювати з колекцією елементів ITask і їм не потрібно буде знати конкретні типи дочірніх класів з якими вони працюють. В результаті логіка візуальних форм виходить достатньо простою і вона не буде містити бізнес-логіки розрахунку загальної оцінки по проєктам, а просто викликає метод GetEstimatedPoints() не замислюючись містить цей об'єкт дочірні об'єкти чи ні. Переваги та недоліки: + Спрощує представлення деревоподібної структури. + Додає гнучкості в роботі з складними об’єктами та

рекурсивними операціями. + Дозволяє додавати та видаляти об’єкти в ієрархії без впливу на клієнтський код. - Потрібні додаткові зусилля для початкового впровадження. - Вимагає гарно спроєктованого загального інтерфейсу.

Хід роботи:

Діаграма класів:



Опис реалізації патерну «Composite»

У цій лабораторній роботі я використала патерн **Composite (Компонувальник)**, щоб реалізувати функцію групування шарів. Головна ідея полягає в тому, щоб програма могла працювати з **групою шарів** так само просто, як і зі **звичайним шаром**. Для цього було вибудовано структуру у вигляді дерева:

- **Component:** Створено абстрактний клас GraphicComponent (у пакеті entity). Він визначає спільний контракт для всіх елементів графіки: методи getImage() для отримання відображення та cloneComponent() для копіювання.
- **Leaf:** Клас Layer представляє базовий елемент (шар із зображенням). Він зберігає реальні дані (byte[] imageData) і реалізує метод getImage(), повертаючи відновлене зображення. Це кінцевий вузол дерева, який не може мати дочірніх елементів.

– **Composite:** Клас LayerGroup представляє контейнер (папку). Він містить список дочірніх компонентів List<GraphicComponent>. Метод getImage() у групі реалізовано рекурсивно: він створює пусте полотно, проходить по всіх дочірніх елементах, викликає у них getImage() і послідовно накладає отримані зображення одне на одне. Таким чином, група динамічно формує своє візуальне представлення на основі вмісту.

Завдяки цьому підходу мій сервіс (EditorService) не повинен перевіряти, що саме він обробляє — одиночний файл чи складну папку. Він просто викликає один і той самий метод для будь-якого елемента. У базі даних це все зберігається в одній таблиці, де спеціальна колонка вказує, чи це шар, чи група.

Фрагменти програмного коду:

GraphicComponent.java

```
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "component_type")
@Table(name = "components")
public abstract class GraphicComponent {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    // Поліморфний метод, що декларує спільну поведінку.  Клієнтський код використовує його для отримання візуального представлення незалежно від конкретної реалізації об'єкта.
    public abstract BufferedImage getImage();

    // Метод для реалізації патерну Prototype у контексті Composite
    public abstract GraphicComponent cloneComponent();
```

```
}
```

Layer.java

```
@Entity
@DiscriminatorValue("LEAF")
public class Layer extends GraphicComponent {

    @Lob
    @Column(columnDefinition = "BLOB")
    private byte[] imageData

    @Override
    public BufferedImage getImage() {
        // Безпосередня реалізація: відновлення зображення з потоку байтів
        try {
            return ImageIO.read(new ByteArrayInputStream(imageData));
        } catch (IOException e) {
            throw new RuntimeException("Помилка десеріалізації зображення", e);
        }
    }

    // ... реалізація cloneComponent()
}
```

LayerGroup.java

```
@Entity
@DiscriminatorValue("GROUP")
public class LayerGroup extends GraphicComponent {
    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    private List<GraphicComponent> children = new ArrayList<>();
```

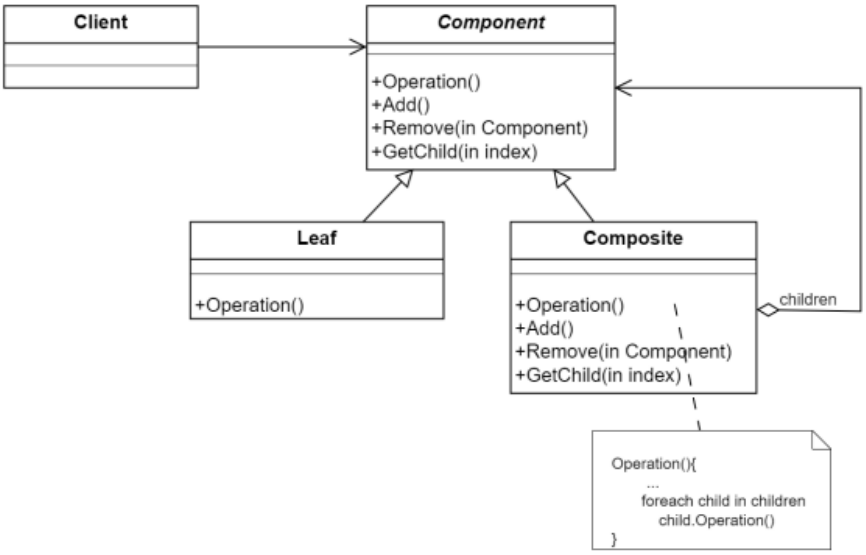
```
    @Override
    public BufferedImage getImage() {
        if (children.isEmpty()) return null;
        BufferedImage result = new BufferedImage(...);
        Graphics2D g = result.createGraphics();

        // Рекурсивний обхід дерева:  група викликає getImage() у кожного нащадка, якщо нащадок - це група, виклик піде далі вглиб (рекурсія).
        for (GraphicComponent child : children) {
            if (child.isVisible()) {
                g.drawImage(child.getImage(), 0, 0, null);
            }
        }
        g.dispose();
        return result;
    }

    // Методи керування дочірніми елементами
    public void add(GraphicComponent component) {
        children.add(component);
    }
}
```

Питання до лабораторної роботи:

1. Яке призначення шаблону «Композит»?

Шаблон «Композит» використовується для складання об'єктів у деревоподібну структуру для подання ієрархій типу «частина-ціле». Цей шаблон дозволяє клієнтам уніфіковано обробляти як поодинокі об'єкти, так і складні композиції об'єктів (об'єкти з вкладеністю).

2. Нарисуйте структуру шаблону «Композит».



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

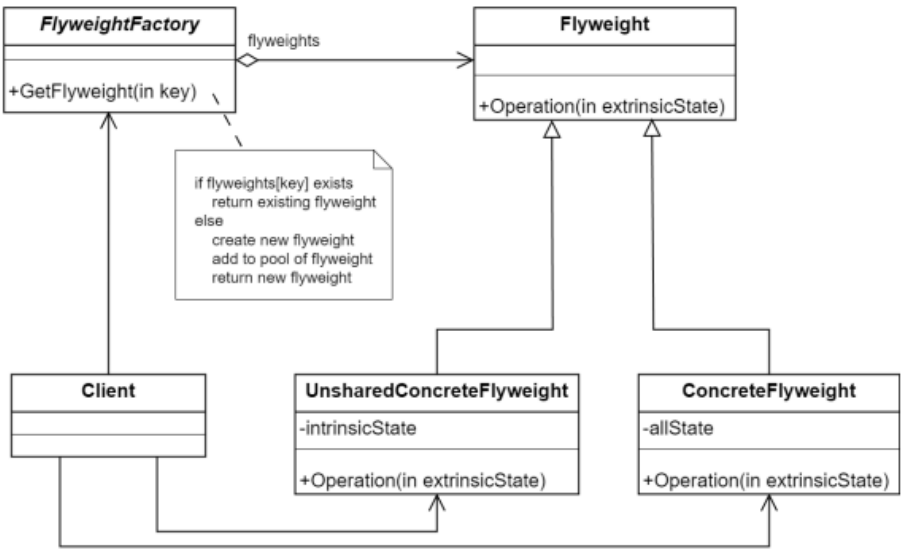
Структура шаблону складається з абстрактного класу Component, який оголошує інтерфейс для всіх об'єктів у композиції, класу Leaf, що представляє кінцеві об'єкти без нащадків, та класу Composite, який містить дочірні компоненти та керує ними .

Взаємодія між класами відбувається таким чином, що клієнт маніпулює об'єктами через інтерфейс Component; якщо отримувачем запиту є Leaf, він виконує операцію, а якщо Composite, то він перенаправляє запит своїм дочірнім елементам, часто виконуючи додаткові операції до або після цього .

4. Яке призначення шаблону «Легковаговик»?

Шаблон «Легковаговик» використовується для зменшення кількості об'єктів у додатку шляхом поділу (шарингу) цих об'єктів між різними ділянками програми. Ключовою є концепція поділу стану на «внутрішній» (зберігається в об'єкті) та «зовнішній» (передається клієнтом) .

5. Нарисуйте структуру шаблону «Легковаговик».



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

Структура шаблону включає FlyweightFactory для створення та управління пулом легковаговиків, інтерфейс Flyweight, через який діють конкретні легковаговики (ConcreteFlyweight), та клієнта, який зберігає посилання на легковаговиків і обчислює зовнішній стан .

Взаємодія полягає в тому, що клієнт звертається до фабрики за об'єктом, і якщо такий вже існує, фабрика повертає його, інакше створює новий; після цього клієнт викликає операції об'єкта, передаючи йому необхідний зовнішній стан .

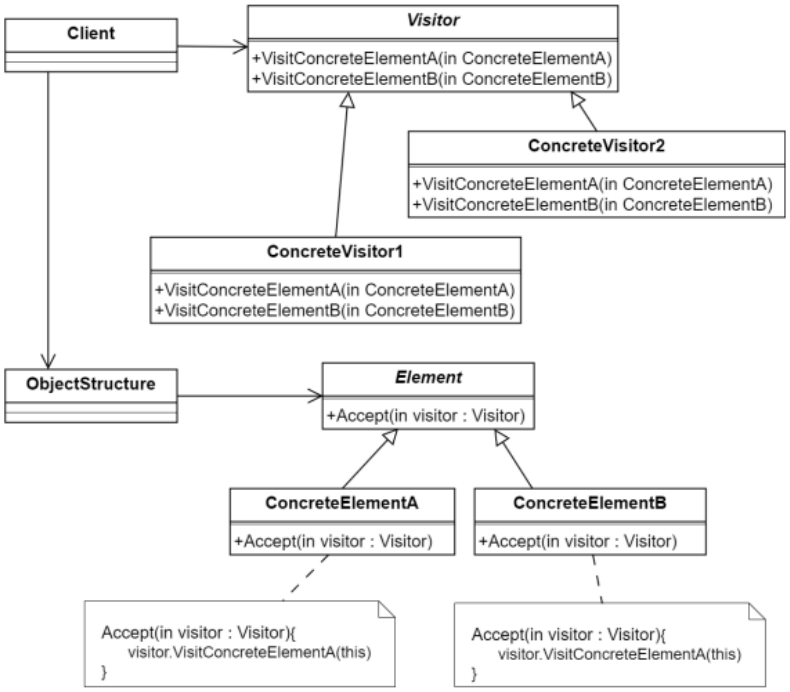
7. Яке призначення шаблону «Інтерпретатор»?

Даний шаблон використовується для подання граматики певної мови та інтерпретатора для цієї мови. Граматика представлена класами для термінальних і нетермінальних символів. Шаблон будує абстрактне синтаксичне дерево, де кожен вузол — це вираз, який інтерпретується в певному контексті .

8. Яке призначення шаблону «Відвідувач»?

Шаблон «Відвідувач» дозволяє визначати нові операції над елементами об'єктної структури без зміни класів цих елементів. Це дозволяє групувати однотипні операції, що застосовуються до різнотипних об'єктів, в одному класі-відвідувачі.

9. Нарисуйте структуру шаблону «Відвідувач».



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

Структура шаблону охоплює інтерфейс Visitor з методами для відвідування кожного типу конкретного елемента, класи ConcreteVisitor, що реалізують ці операції, та ієрархію елементів Element з методом Accept, який приймає відвідувача .

Взаємодія базується на тому, що клієнт, маючи структуру об'єктів, передає екземпляр відвідувача кожному елементу через метод Accept, а елемент у свою чергу викликає відповідний метод Visit у відвідувача, передаючи себе як аргумент, що дозволяє відвідувачу виконати потрібну дію над цим елементом .

Вихідний код:
https://github.com/eismonta/TRPZ_2025/tree/ab5c4d8ae1699b878b9cbbfb4d40039979f1c13c/image-editor

Висновок:
Під час виконання лабораторної роботи я ознайомилась з структурою шаблонів «Composite», «Flyweight» (Пристосуванець), «Interpreter», «Visitor» та практично реалізувала патерн Composite у своєму графічному редакторі.