МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

ЛАБОРАТОРНА РОБОТА №8

ТЕМА: «Патерни проектування»

Виконала:

Студентка групи ІА-34

Потейчук С.А.

Перевірив:

Мягкий М.Ю.

Київ 2025

Зміст

Теоретичні відомості

Шаблон «Composite» (Компонувальник)

Призначення: Шаблон «Компонувальник» дозволяє уніфіковано працювати з окремими об'єктами та їх ієрархічними композиціями. Він організує об'єкти в деревоподібні структури, що представляють ієрархії "частина-ціле".

Проблема: У системі управління проектами потрібно оперувати складними ієрархіями: проект, що складається з функцій, що складаються з user stories, що складаються з задач. Необхідно реалізувати єдиний спосіб роботи з будь-яким елементом цієї структури для розрахунку оціночної вартості.

Рішення: Створюється загальний інтерфейс ITask з методом GetEstimatedPoints(). Класи Feature, UserStory та Task реалізують цей інтерфейс. Складні об'єкти (Feature, UserStory) делегують виклик метода своїм дочірнім елементам, а прості об'єкти (Task) повертають власне значення.

Переваги: уніфікована робота з ієрархіями, спрощення коду клієнта, легке додавання нових типів елементів.

Недоліки: необхідність ретельного проектування загального інтерфейсу, додаткові зусилля на початкову реалізацію.

Шаблон «Flyweight» (Легковаговик)

Призначення: Шаблон «Легковаговик» використовується для ефективної підтримки великої кількості дрібних об'єктів шляхом розділення спільного стану між ними. Він дозволяє значно зменшити використання пам'яті.

Концепція станів:

- Внутрішній стан: Незмінні дані, спільні для багатьох об'єктів (наприклад, символ літери, колір, шрифт).
- Зовнішній стан: Змінні дані, унікальні для кожного використання (наприклад, позиція на екрані).

Приклад: У текстових редакторах кожен символ може бути представлений одним об'єктом-легковаговиком, який зберігає внутрішній стан (код символу, шрифт), а зовнішній стан (позиція, колір) зберігається окремо для кожного входження.

Переваги: значна економія пам'яті при роботі з великою кількістю об'єктів.

Недоліки: збільшення складності коду, додаткові витрати часу на пошук та управління станами.

Шаблон «Interpreter» (Інтерпретатор)

Призначення: Шаблон «Інтерпретатор» використовується для визначення граматики мови та створення інтерпретатора для обробки виразів цієї мови. Він особливо ефективний для реалізації мов з простими граматиками.

Проблема: Потрібно реалізувати систему пошуку рядків за шаблоном або обробки складних виразів, які часто змінюються. Наприклад, система фільтрації даних за різними критеріями.

Рішення: Створюється абстрактне синтаксичне дерево, де кожен вузол представляє елемент граматики. Основними компонентами є:

- AbstractExpression - абстрактний клас з методом Interpret()
- TerminalExpression - термінальні символи граматики
- NonterminalExpression - нетермінальні символи, що містять посилання на інші вирази
- Context - містить глобальну інформацію для інтерпретатора

Переваги: легке розширення граматики, зручність реалізації складних правил.

Недоліки: складність підтримки для граматик з великою кількістю правил.

Шаблон «Visitor» (Відвідувач)

Призначення: Шаблон «Відвідувач» дозволяє додавати нові операції до ієрархії класів без зміни самих класів. Він відокремлює алгоритми від структури об'єктів, над якими вони оперують.

Проблема: У системі інтернет-магазину з різними типами товарів (електроніка, напої, хімія) виникає потреба реалізувати різну логіку обробки для кожного типу товару без ускладнення самих класів товарів.

Рішення: Створюється окрема ієрархія відвідувачів, кожен з яких реалізує методи обробки для кожного типу товару. Основні компоненти:

- Visitor - інтерфейс з методами visit() для кожного типу елемента
- ConcreteVisitor - конкретні реалізації операцій
- Element - інтерфейс елементів, що приймають відвідувача
- ConcreteElement - конкретні класи елементів

Переваги: легке додавання нових операцій, групування споріднених операцій.

Недоліки: складність додавання нових типів елементів, порушення інкапсуляції.

Тема: Патерни проектування.

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

Хід роботи

7. Редактор зображень (state, prototype, memento, facade, composite, client server)

Редактор зображень має такі функціональні можливості: відкриття/збереження зображень у найпопулярніших форматах, застосування ефектів, наприклад поворот, розтягування, стиснення, кадрування зображення, можливість створення колажів шляхом «нашарування» зображень.

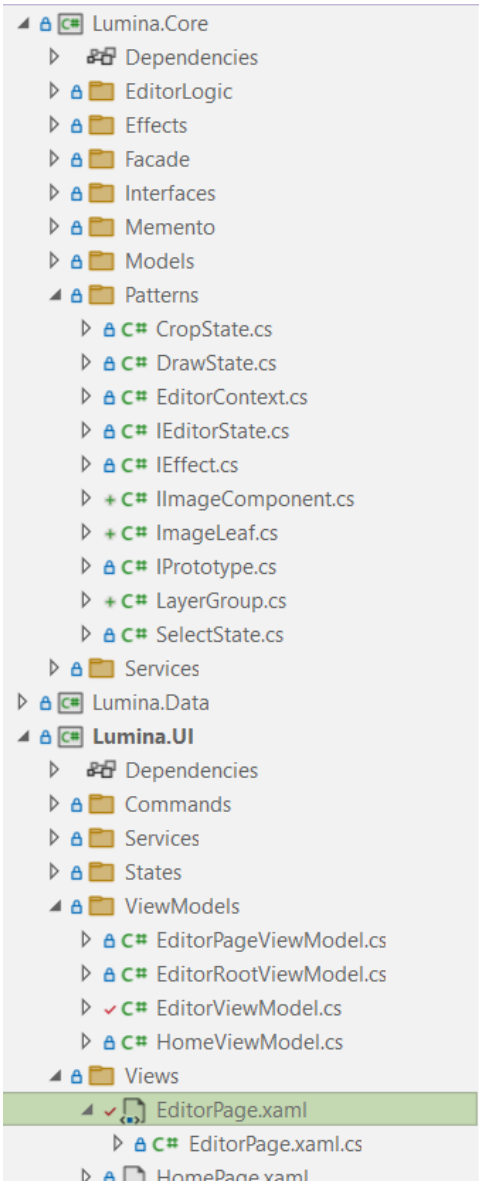


Рисунок 1 – Структура проєкту

Загальна структура проєкту:

- Lumina.Core – Бізнес-логіка, сервіси, моделі.
- Lumina.Data – Робота з базою даних, реалізації репозиторіїв, контекст EF Core, сутності та мапери.
- Lumina.UI – Графічний інтерфейс (WPF), користувацькі дії, візуальне відображення, керування станами.

Lumina.UI:

- App.xaml / App.xaml.cs – Точка входу. Ініціалізація головного вікна.
- MainWindow.xaml / MainWindow.xaml.cs – Головне вікно, у якому завантажуються сторінки (HomePage, EditorPage).
- HomePage.xaml / HomePage.xaml.cs – Головна сторінка з кнопками “Open Image”, “Create Collage”, “Go to Editor”, “Exit”.
- EditorPage.xaml / EditorPage.xaml.cs – Головна сторінка редактора: полотно, панель інструментів, ефекти, стани.
- ViewModels/ – ViewModel-и для зв’язку UI та Core.
- State/ – Реалізація патерну State для поведінки редактора.

Патерн Composite дозволяє працювати зі складними об’єктами (композитами) і простими об’єктами (листами) однаково.

У контексті редактора зображень:

- Листи (ImageLeaf) – це окремі шари (ImageLayer), які можна редагувати, переміщувати, застосовувати ефекти.
- Композиції (LayerGroup) – це групи шарів, які можуть містити інші групи та листи.

Інтерфейс IImageComponent дозволяє однаково працювати з листами і групами, наприклад, застосовувати ефекти чи дублювати. Таким чином можна

додавати нові шари чи групи шарів, приміняти ефекти як до окремого шару, так і до групи, а також легко дублювати групу шарів або окремий шар.

```
7 references
public interface IImageComponent
{
    9 references
    string Name { get; set; }
    3 references
    Task RenderAsync();
    4 references
    Task ApplyEffectAsync(string effectName, string? parameters = null);
}
```

Рисунок 2 – клас IImageComponent.cs

IImageComponent – це загальний інтерфейс для листів і груп шарів. Він забезпечує однаковий метод ApplyEffectAsync, щоб застосовувати ефекти незалежно від типу компонента.

```
4 references
public class ImageLeaf : IImageComponent
{
    5 references
    public string Name { get; set; }
    6 references
    public ImageLayer Layer { get; set; }

    2 references
    public ImageLeaf(string name, ImageLayer layer)
    {
        Name = name;
        Layer = layer;
    }

    2 references
    public Task RenderAsync()
    {
        Console.WriteLine($"Rendering layer '{Name}' at ({Layer.X},{Layer.Y}) size {Layer.Width}x{Layer.Height}");
        return Task.CompletedTask;
    }

    3 references
    public Task ApplyEffectAsync(string effectName, string? parameters = null)
    {
        Console.WriteLine($"Applying effect '{effectName}' to layer '{Name}'");
        return Task.CompletedTask;
    }
}
```

Рисунок 3 – клас ImageLeaf.cs

ImageLeaf представляє окремий шар зображення. Він містить дані про позицію, розмір, ротацію та застосовані ефекти (ImageLayer), а також реалізує метод ApplyEffectAsync.

```
6 references
public class LayerGroup : IImageComponent
{
    4 references
    public string Name { get; set; }
    private readonly List<IImageComponent> _children = new();

    2 references
    public LayerGroup(string name)
    {
        Name = name;
    }

    3 references
    public void Add(IImageComponent component)
    {
        _children.Add(component);
    }

    0 references
    public void Remove(IImageComponent component)
    {
        _children.Remove(component);
    }

    2 references
    public async Task RenderAsync()
    {
        Console.WriteLine($"Rendering group '{Name}' with {_children.Count} child components...");
        foreach (var child in _children)
            await child.RenderAsync();
    }

    3 references
    public async Task ApplyEffectAsync(string effectName, string? parameters = null)
    {
        Console.WriteLine($"Applying effect '{effectName}' to group '{Name}'");
        foreach (var child in _children)
            await child.ApplyEffectAsync(effectName, parameters);
    }
}
```

Рисунок 4 – клас LayerGroup.cs

LayerGroup може містити будь-яку кількість листів та інших груп.

Методи Add та Remove дозволяють динамічно змінювати структуру.

Метод ApplyEffectAsync рекурсивно застосовує ефекти до всіх дочірніх компонентів.

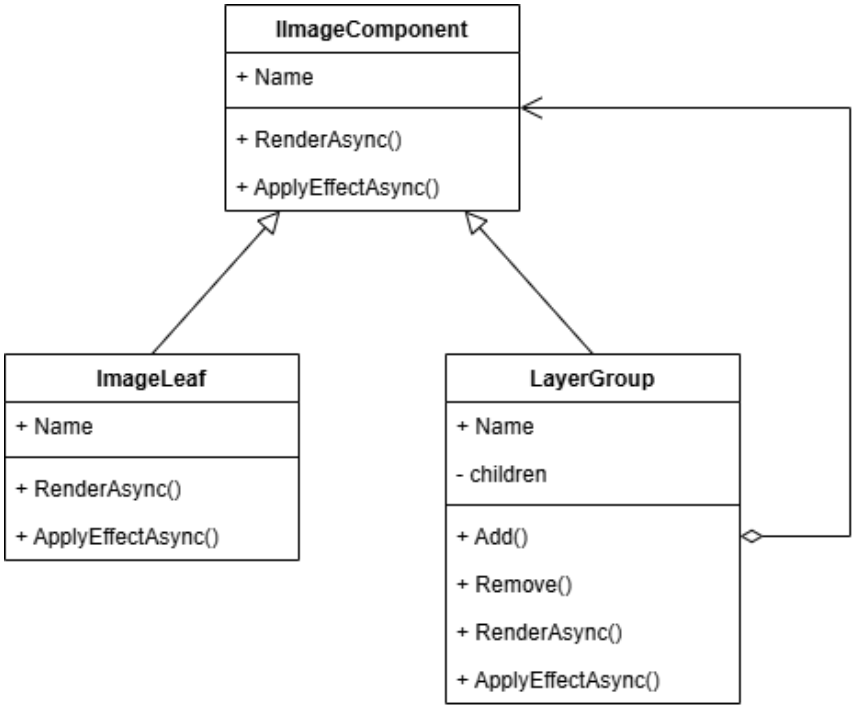


Рисунок 8 – Діаграма класів, яка представляє використання шаблону в реалізації системи

Переваги використання Composite в редакторі

- Уніфікація роботи з шарами та групами. Можна викликати один і той же метод ApplyEffectAsync для окремого шару або для групи шарів без перевірок типу.
- Гнучкість структури. Легко створювати багаторівневі колажі з груп шарів і підгруп.
- Розширюваність. Додаючи новий тип шару (наприклад, векторний шар або текстовий шар), не потрібно міняти логіку UI чи застосування ефектів.
- Масштабованість. Ефекти можна застосовувати рекурсивно до всіх елементів групи, не пишучи окремий код для кожного шару.

Висновки

Патерн Composite дозволяє об’єднувати окремі об’єкти і групи об’єктів у єдину ієрархічну структуру, що значно спрощує роботу з комплексними системами, як робота з шарами та колажами в редакторі зображень.

Завдяки використанню спільного інтерфейсу IImageComponent, методи для застосування ефектів (ApplyEffectAsync), дублювання чи видалення шарів реалізуються однаково як для окремих шарів (ImageLeaf), так і для груп шарів (LayerGroup).

Composite дозволяє будувати багаторівневі колажі, де групи можуть містити як інші групи, так і окремі шари. Це забезпечує зручну організацію об’єктів сцени та полегшує їх обробку.

Застосування ефектів, дублювання чи переміщення компонентів здійснюється рекурсивно, що зменшує кількість дубльованого коду та підвищує масштабованість системи.

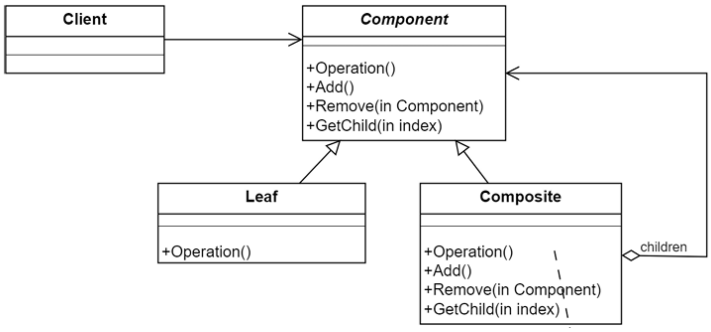
У лабораторній роботі було реалізовано інтерфейс IImageComponent, класи ImageLeaf та LayerGroup, а також демонстровано застосування ефектів до складної ієрархії шарів. Це дозволяє ефективно управляти композиціями у графічному редакторі.

Контрольні питання

1. Яке призначення шаблону «Композит»?

Призначення шаблону «Композит» – організувати об'єкти в деревоподібні структури для представлення ієрархій "частина-ціле". Він дозволяє уніфіковано працювати з окремими об'єктами та їх композиціями.

2. Нарисуйте структуру шаблону «Композит».



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

Класи:

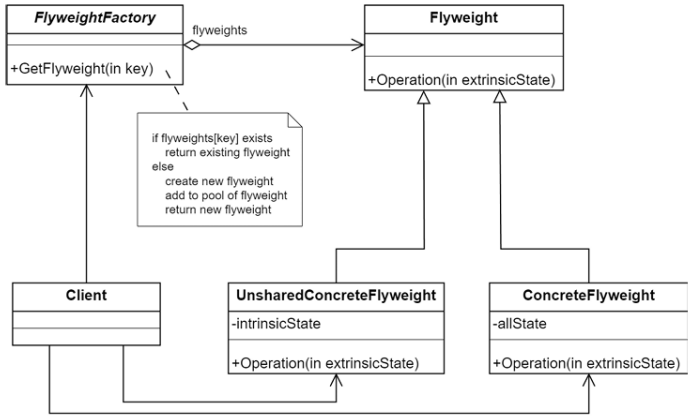
- Component (Компонент): Визначає інтерфейс для всіх елементів композиції
- Leaf (Листок): Представляє кінцеві об'єкти без дочірніх елементів
- Composite (Композит): Містить колекцію дочірніх компонентів та реалізує операції з ними

Взаємодія: Клієнт працює з усіма об'єктами через інтерфейс Component. Композит делегує операції своїм дочірнім компонентам, а Листок виконує операції безпосередньо.

4. Яке призначення шаблону «Легковаговик»?

Призначення шаблону «Легковаговик» – ефективно підтримувати велику кількість дрібних об'єктів шляхом розділення спільного стану між ними, зменшуючи використання пам'яті.

5. Нарисуйте структуру шаблону «Легковаговик».



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

Класи:

- Flyweight (Легковаговик): Містить внутрішній стан (спільний)
- ConcreteFlyweight:
- FlyweightFactory: Створює та керує легковаговиками
- Client (Клієнт): Зберігає зовнішній стан та використовує легковаговики

Взаємодія: Клієнт звертається до фабрики за легковаговиком, передаючи зовнішній стан. Фабрика повертає існуючий або створює новий легковаговик.

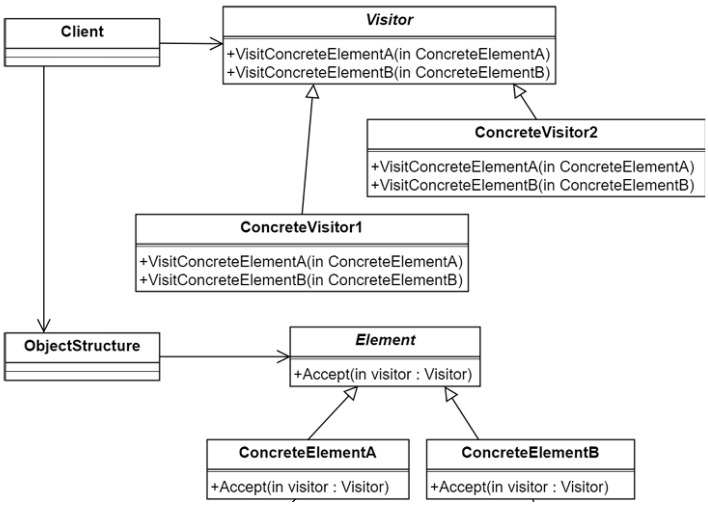
7. Яке призначення шаблону «Інтерпретатор»?

Призначення шаблону «Інтерпретатор» – визначити граматику мови та створити інтерпретатор для обробки виразів цієї мови. Він використовується для реалізації мов з простими граматиками.

8. Яке призначення шаблону «Відвідувач»?

Призначення шаблону «Відвідувач» – додавати нові операції до ієрархії класів без зміни самих класів. Він відокремлює алгоритми від структури об'єктів.

9. Нарисуйте структуру шаблону «Відвідувач».



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

Класи:

- Visitor (Відвідувач): Оголошує методи visit для кожного типу елемента
- ConcreteVisitor: Реалізує конкретні операції
- Element (Елемент): Оголошує метод accept для прийняття відвідувача
- ConcreteElement: Реалізує метод accept

Взаємодія: Клієнт створює відвідувача та передає його елементам. Елемент викликає метод accept, передаючи себе відвідувачу, який виконує потрібну операцію.