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

Тема: Патерни проектування.

Виконала
студентка групи ІА–32:
Слюсарева А.А.

Київ 2025

ЗМІСТ

ЛАБОРАТОРНА РОБОТА № 8

Тема: Патерни проектування.

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

8.1. 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.

8.2. Теоретичні відомості

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).

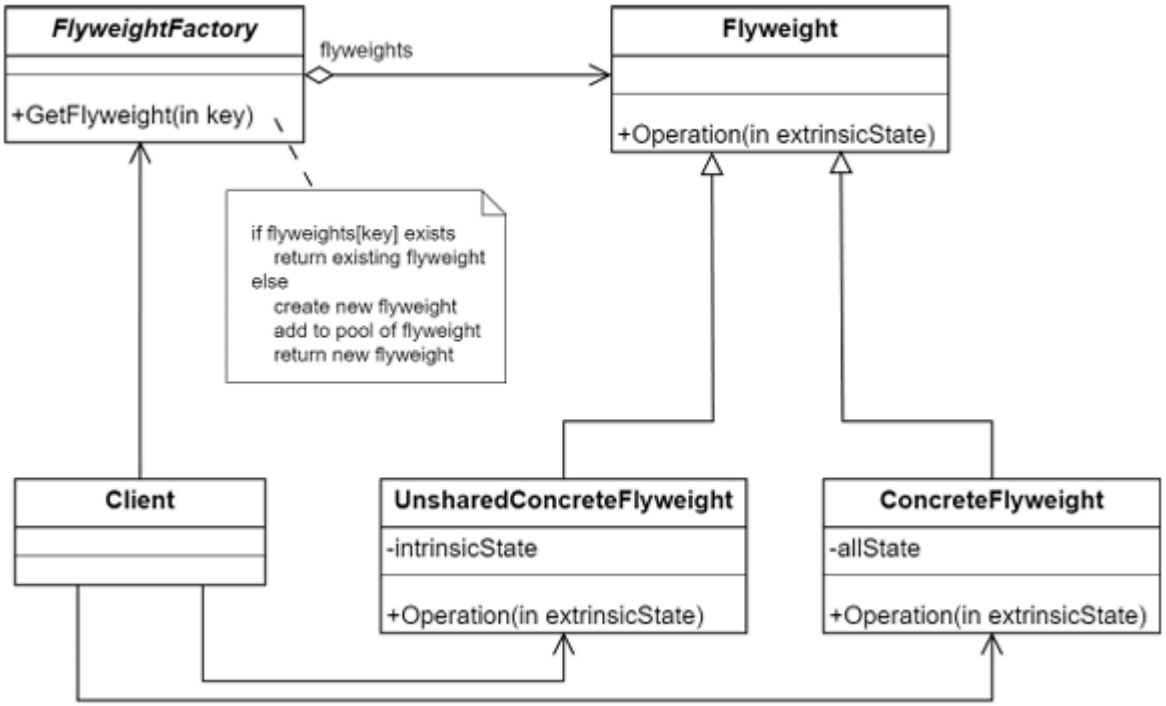


Рисунок 8.2. Структура патерну Flyweight (Легковаговик)

Хорошим прикладом є наступне зображення на рисунку 3.

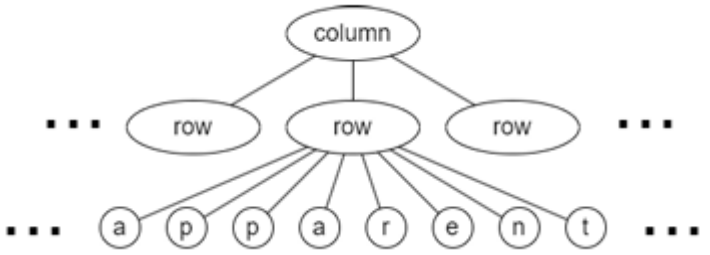


Рисунок 8.3. Приклад ситуації де варто використовувати Flyweight

Здається, ніби для кожної літери існує окремий об'єкт. Насправді фізично об'єкт всього один, існує лише безліч посилань на нього. (рисунку 8.4)

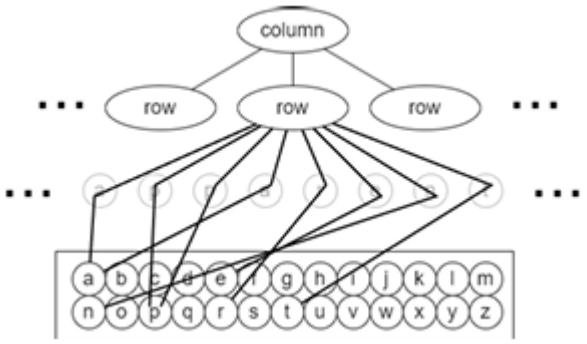


Рисунок 8.4. Використання патерну Flyweight

Даний шаблон дуже добре застосовувати у випадках, коли використовується безліч однакових об'єктів (наприклад, графічних примітивів).

Переваги та недоліки:

+ Заощаджує оперативну пам’ять.

- Витрачає процесорний час на пошук/обчислення контексту.

- Ускладнює код програми внаслідок введення безлічі додаткових класів.

8.3 Хід роботи

Діаграма класів:

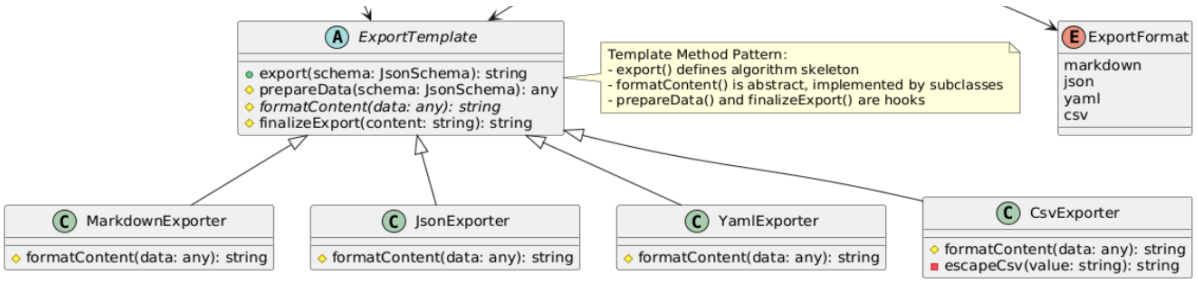


Рис 1. Діаграма класів системи

Фрагменти коду:

```
import { Injectable } from '@angular/core';
import { SchemaPropertyMetadata } from
'../models/flyweight.model';

@Injectable({
 providedIn: 'root'
})
export class FlyweightFactoryService {
 private metadata: Map<string,
SchemaPropertyMetadata> = new Map();

 getMetadata(dataType: string, format?: string):
SchemaPropertyMetadata {
   const key = `${dataType}:${format || 'default'}`;

   if (!this.metadata.has(key)) {
     this.metadata.set(key,
this.createMetadata(dataType, format));
   }

   return this.metadata.get(key)!;
 }

 private createMetadata(dataType: string, format?:
string): SchemaPropertyMetadata {
   const validators: string[] = [];
```

```
  switch (dataType) {
    case 'string':
      validators.push('isString');
      if (format === 'email')
validators.push('isEmail');
      if (format === 'url')
validators.push('isURL');
      break;
    case 'number':
      validators.push('isNumber');
      break;
    case 'boolean':
      validators.push('isBoolean');
      break;
    case 'array':
      validators.push('isArray');
      break;
    case 'object':
      validators.push('isObject');
      break;
  }

  return {
    dataType,
    format,
    validators
  };
}
```

```
  getPoolSize(): number {
    return this.metadata.size;
  }
}
```

У реалізації використано патерн Легковаговик (Flyweight), який дозволяє уникати створення великої кількості однакових об’єктів, розділяючи спільний внутрішній стан між багатьма екземплярами. Клас FlyweightFactoryService виступає фабрикою та пулом легковагових об’єктів: він кешує метадані властивостей JSON-схеми, щоб при повторних запитах не створювати нові об’єкти, якщо їхній внутрішній стан (dataType, format) збігається. Фабрика повертає вже існуючі метадані, а унікальний зовнішній стан передається ззовні під час використання. Це зменшує кількість об’єктів у пам’яті та оптимізує продуктивність при роботі з великою кількістю однотипних полів.

8.4 Висновок

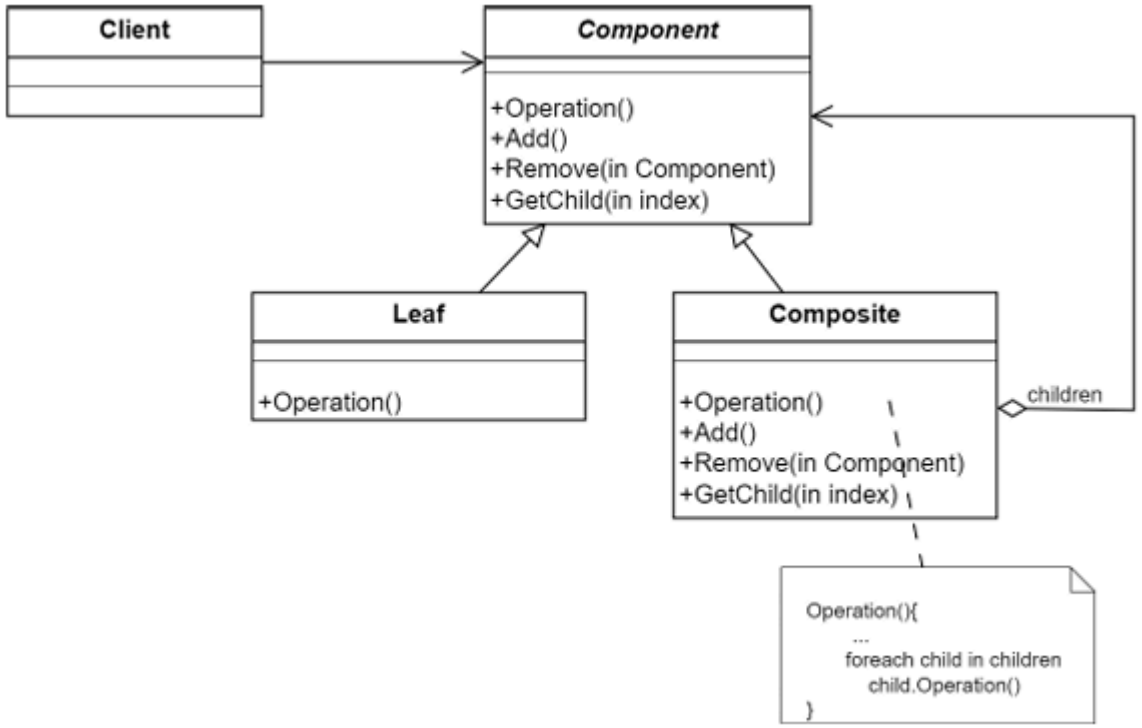
Під час даної роботи було реалізовано патерн Шаблонний метод у додатку JSON Tool.

8.5 Контрольні запитання

1. **Яке призначення шаблону «Композит»?**

   Композит дозволяє працювати з окремими об’єктами і групами об’єктів однаково, створюючи деревоподібну структуру. Він спрощує роботу з ієрархіями, де клієнт може не розрізняти одиничні елементи і цілі колекції.

2. **Нарисуйте структуру шаблону «Композит».**



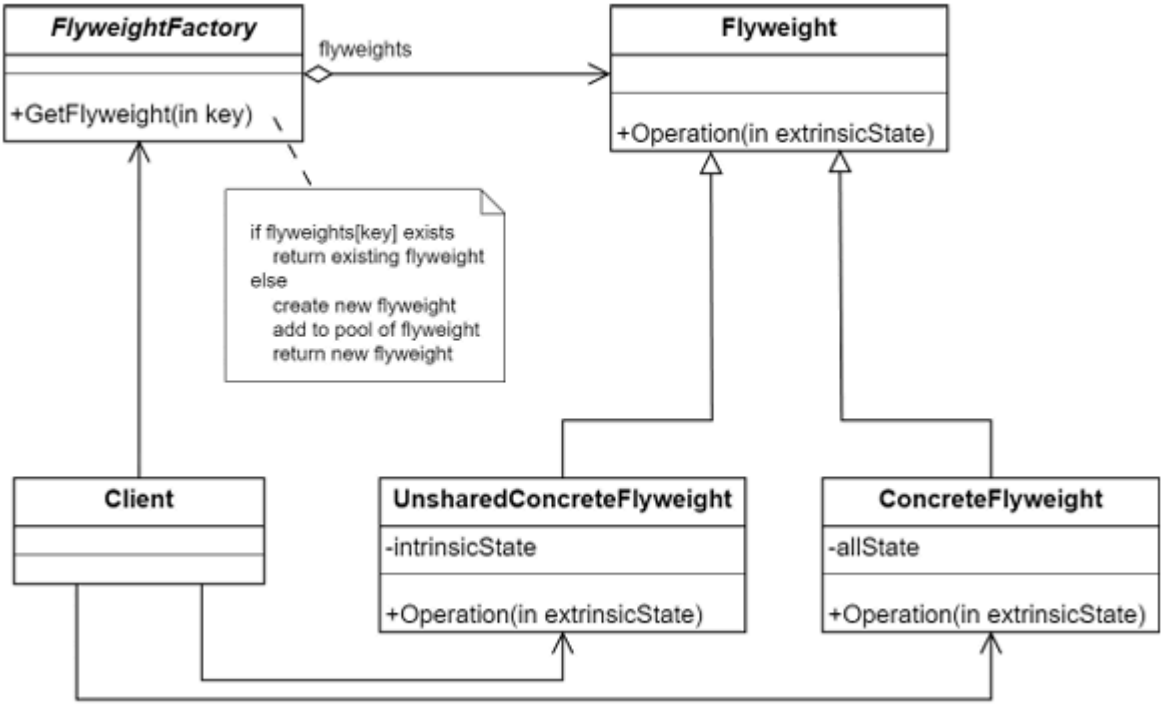
3. **Які класи входять в шаблон «Композит», та яка між ними взаємодія?**

   У шаблон входять Component, Leaf і Composite. Component визначає спільний інтерфейс. Leaf представляє окремий елемент, Composite містить інші компоненти і делегує їм виконання операцій, забезпечуючи роботу всієї структури як єдиного об’єкта.

4. **Яке призначення шаблону «Легковаговик»?**

   Легковаговик зменшує використання пам’яті, розділяючи спільний стан між багатьма об’єктами і зберігаючи унікальний стан зовні. Це дозволяє створювати дуже велику кількість дрібних об’єктів без великих витрат.

5. **Нарисуйте структуру шаблону «Легковаговик».**



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

   У шаблон входять Flyweight, ConcreteFlyweight, UnsharedFlyweight та FlyweightFactory. Фабрика повертає вже створені легковагові об’єкти, а клієнт додає зовнішній унікальний стан. Внутрішній спільний стан зберігається в легковагових об’єктах і розділяється між клієнтами.

7. **Яке призначення шаблону «Інтерпретатор»?**

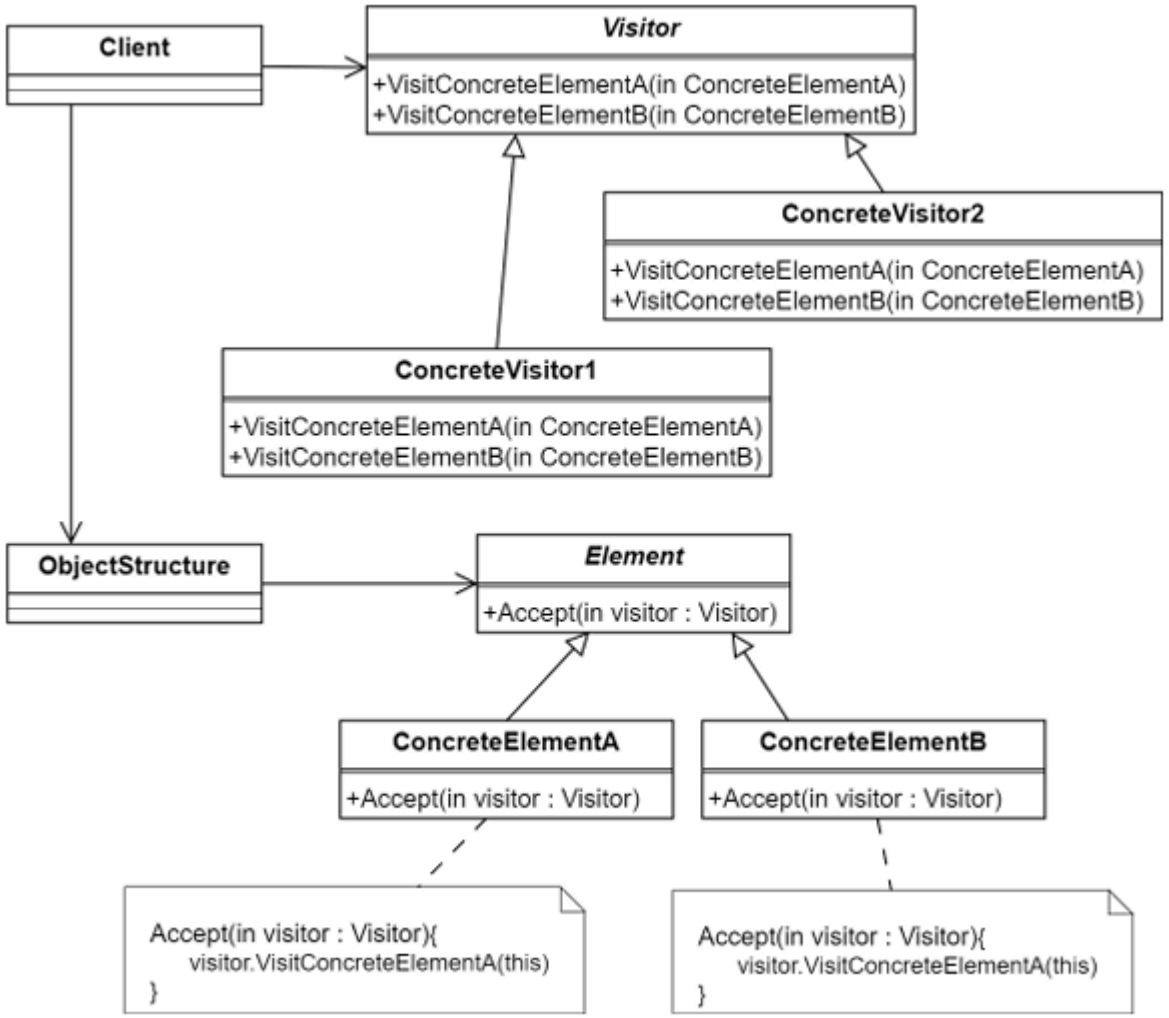
   Інтерпретатор описує граматику простої мови та надає механізм для інтерпретації виразів цієї мови, представляючи кожне правило граматики окремим класом.

8. **Яке призначення шаблону «Відвідувач»?**

   Відвідувач дозволяє додавати нові операції до об’єктів складної структури без зміни цих об’єктів. Операції винесені у окремий клас, який проходить

по елементах структури.

9. **Нарисуйте структуру шаблону «Відвідувач».**



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

У шаблон входять Visitor, ConcreteVisitor, Element, ConcreteElement та ObjectStructure. Element приймає відвідувача через метод accept, а Visitor визначає операції для кожного типу елементів. ObjectStructure зберігає колекцію елементів і дозволяє відвідувачу проходити їх.