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

Лабораторна робота № 8

із дисципліни «Технології розроблення програмного забезпечення»

Тема: «Вступ до паттернів проектування»

Виконав
Студент групи ІА-31:
Олея М. С.

Перевірив:
Мягкий М. Ю.

Київ 2025

Зміст

1. Мета:

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

2. Теоретичні відомості:

Composite (Композит): Шаблон для побудови деревоподібних ієрархій "частина-ціле", що дозволяє уніфіковано обробляти як окремі об’єкти (листки), так і їхні контейнери. Ключова ідея: Уніфікована обробка листків і контейнерів. Приклад: Проєкт → Функція → User Story → Task.

Flyweight (Легковаговик): Шаблон для зменшення витрат пам’яті шляхом спільного використання об’єктів з однаковим внутрішнім станом. Ключова ідея: Поділ стану на внутрішній (shared) і зовнішній (context). Приклад: Букви в тексті, графічні примітиви.

Interpreter (Інтерпретатор): Шаблон для реалізації інтерпретації граматики мови, використовуючи рекурсивне дерево абстрактного синтаксису (AST). Ключова ідея: Рекурсивне AST-дерево для обробки виразів. Приклад: Пошук за шаблоном, скриптова мова.

Visitor (Відвідувач): Шаблон для додавання нових операцій до структури об’єктів без зміни їхніх класів. Ключова ідея: Відокремлення логіки операцій від структури об’єктів. Приклад: Розрахунок цін у кошику (з урахуванням знижки або без неї).

3. Завдання:

1) Ознайомитись 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. Поданий звіт повинен містити: діаграму класів, яка представляє використання шаблону в реалізації системи, навести фрагменти коду по реалізації цього шаблону.

4. Хід роботи:

Варіант - 8

Powershell terminal (strategy, command, abstract factory, bridge, interpreter,

client-server)

Термінал для powershell повинен нагадувати типовий термінал з можливістю налаштування кольорів синтаксичних конструкцій, розміру вікна, фону вікна, а також виконання команд powershell і виконуваних файлів, а також працювати в декількох вікнах терміналу (у вкладках або одночасно шляхом розділення вікна)

Призначення та доцільність:

У системі "PowerShell Terminal" виникла потреба підтримки складних командних конструкцій, зокрема конвеєрної обробки (Piping), де результат виконання лівої команди стає вхідними даними для правої (наприклад, echo "text" | upper). Реалізація такого функціоналу через прості маніпуляції рядками (RegEx або Split) призводить до жорсткого коду, який важко розширювати (наприклад, додати підтримку кількох пайпів або інших операторів).

Паттерн Interpreter (Інтерпретатор) є доцільним, оскільки він дозволяє:

- Побудувати граматику мови: Ми визначаємо правила, як інтерпретувати символи (наприклад, | як оператор передачі даних).
- Створити дерево виразів (AST): Команда перетворюється на структуру об'єктів, де кожен вузол знає, як виконати свою частину роботи.
- Розширювати синтаксис: Додавання нових операторів (наприклад, && або >) вимагає лише створення нових класів виразів, не ламаючи існуючий парсер.

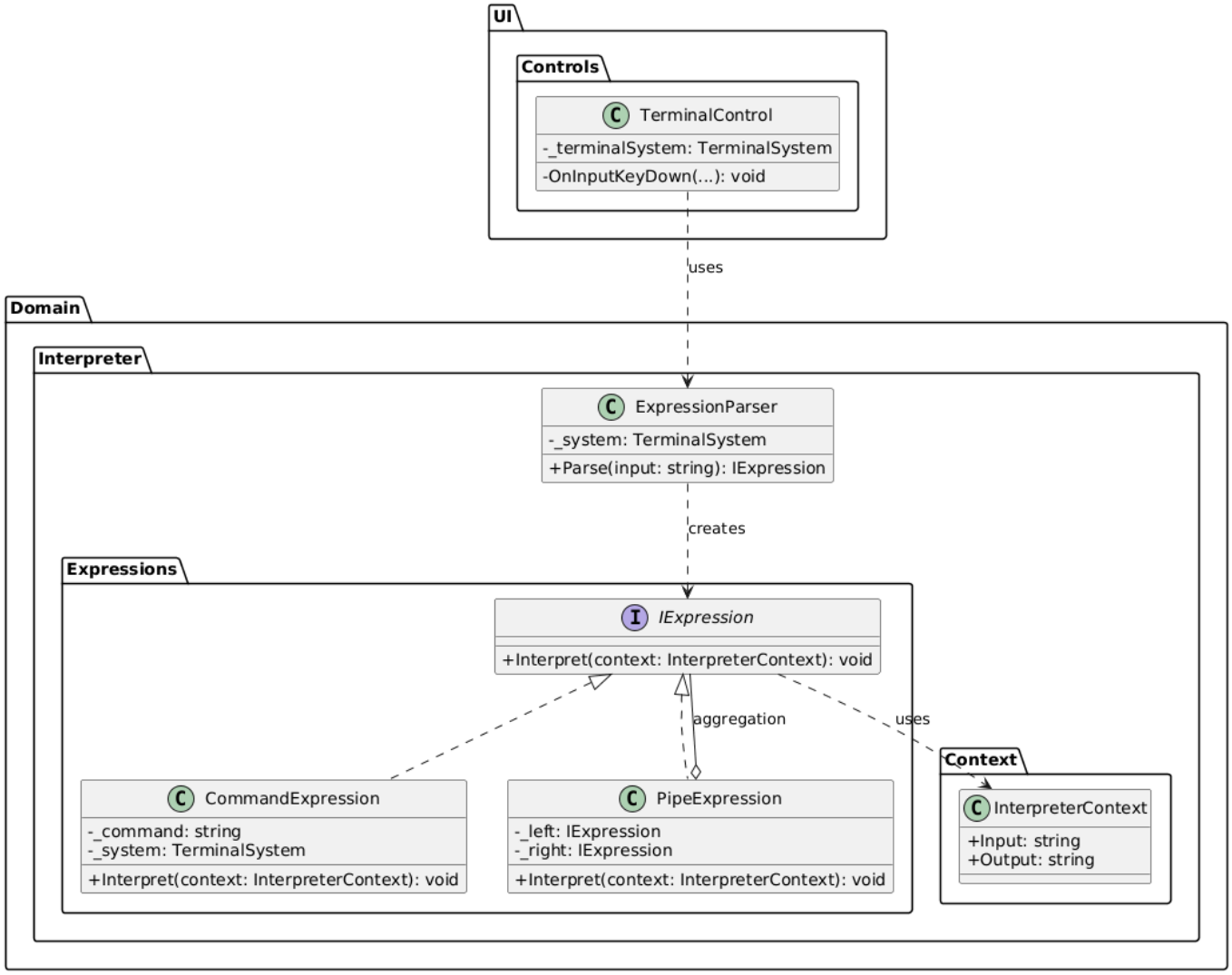


Рисунок 1 - Структура патерну Interpreter

Діаграма, зображена на рис.1 відображає структуру патерну, що складається з таких елементів:

1. Context (InterpreterContext): Клас, що зберігає глобальний стан під час інтерпретації. Він передає дані (Output) від одного виразу до іншого.
2. Abstract Expression (IExpression): Інтерфейс для всіх вузлів дерева. Оголошує метод Interpret.
3. Terminal Expression (CommandExpression): Реалізує елементарну одиницю граматики — конкретну команду (наприклад, echo "hello"). Вона безпосередньо виконує дію через TerminalSystem.
4. Non-terminal Expression (PipeExpression): Реалізує правило граматики для символу |. Цей вираз містить два дочірніх вирази (_left і _right). Він спочатку запускає лівий вираз, бере його вивід і передає його як вхідні дані для правого виразу.
5. Client/Parser (ExpressionParser): Аналізує введений рядок і будує з нього дерево об'єктів (Abstract Syntax Tree). Якщо він бачить |, він створює PipeExpression, інакше — звичайний CommandExpression.

Програмна реалізація:

а) Інтерфейс Виразу (IExpression.cs)

```
namespace PowerShellTerminal.App.Domain.Interpreter
{
    public interface IExpression
    {
        void Interpret(InterpreterContext context);
    }
}
```

б) Контекст (InterpreterContext.cs)

```
public class InterpreterContext
    {
        public string Input { get; set; } = string.Empty;

        public string Output { get; set; } = string.Empty;

        public InterpreterContext(string input)
        {
            Input = input;
        }
    }
```

в) Нетермінальний вираз Пайпу (PipeExpression.cs)

```
public class PipeExpression : IExpression
    {
        private readonly IExpression _left;
        private readonly IExpression _right;

        public PipeExpression(IExpression left, IExpression right)
        {
            _left = left;
            _right = right;
        }

        public void Interpret(InterpreterContext context)
        {
            _left.Interpret(context);

            var pipeContext = new InterpreterContext(context.Output);

            pipeContext.Output = context.Output;

            _right.Interpret(pipeContext);

            context.Output = pipeContext.Output;
```

```
        }
    }
```

г) Парсер

```
public class ExpressionParser
    {
        private readonly TerminalSystem _system;

        public ExpressionParser(TerminalSystem system)
        {
            _system = system;
        }

        public IExpression Parse(string commandLine)
        {
            if (commandLine.Contains("|"))
            {
                var parts = commandLine.Split('|', 2);
                var leftStr = parts[0];
                var rightStr = parts[1];

                IExpression leftExpr = ParseTerminal(leftStr);
                IExpression rightExpr = ParseTerminal(rightStr);

                return new PipeExpression(leftExpr, rightExpr);
            }

            return ParseTerminal(commandLine);
        }

        private IExpression ParseTerminal(string commandPart)
        {
            var trimmed = commandPart.Trim();

            if (trimmed.Equals("upper", System.StringComparison.OrdinalIgnoreCase))
            {
                return new ToUpperExpression();
            }

            return new SystemCommandExpression(_system, trimmed);
        }
    }
```

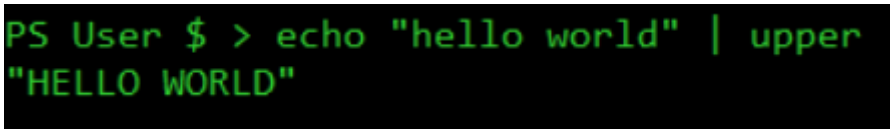


Рисунок 2 – Використання пайпу

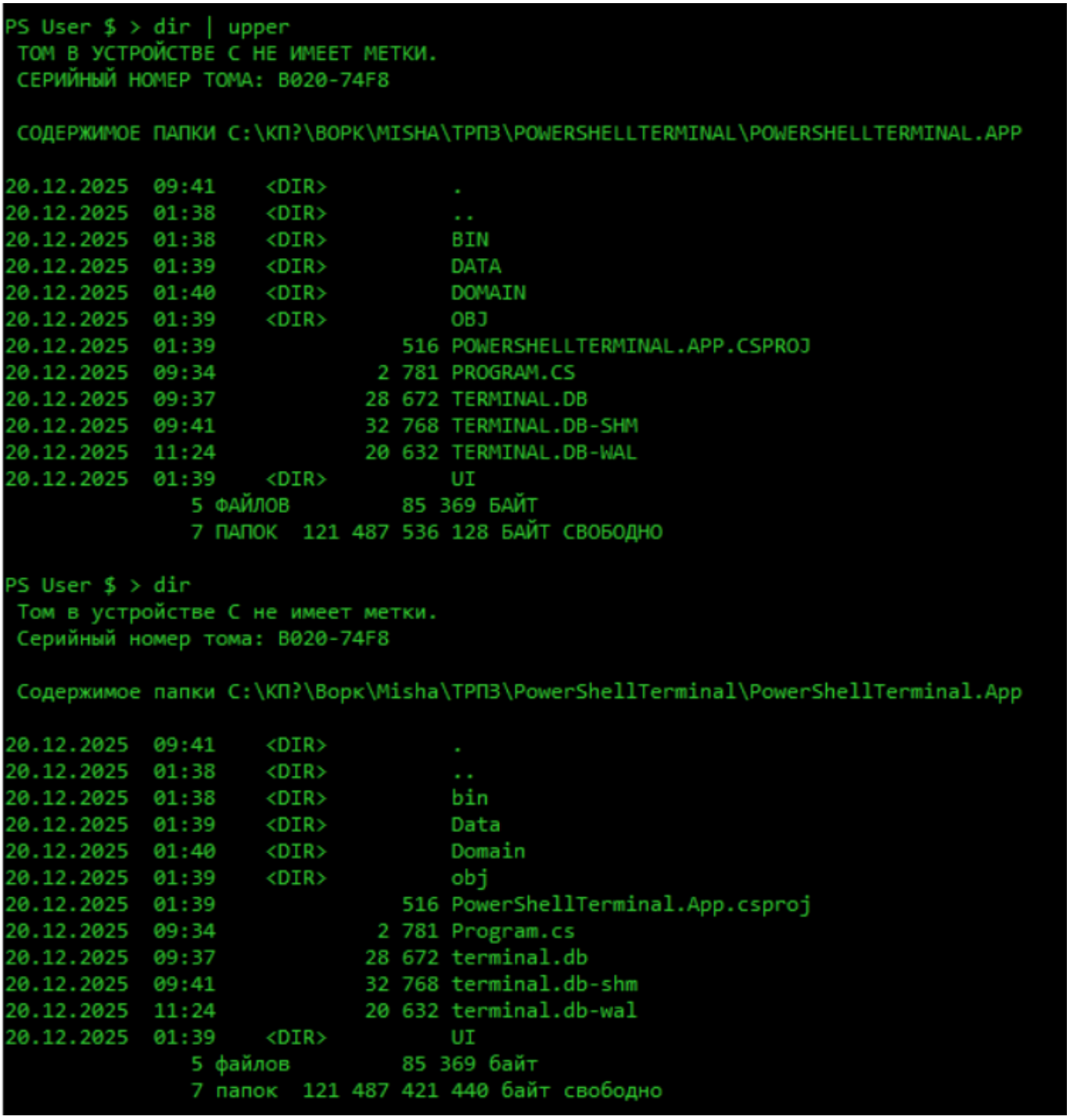


Рисунок 3 – Комбінація з системною командою

5. Висновок

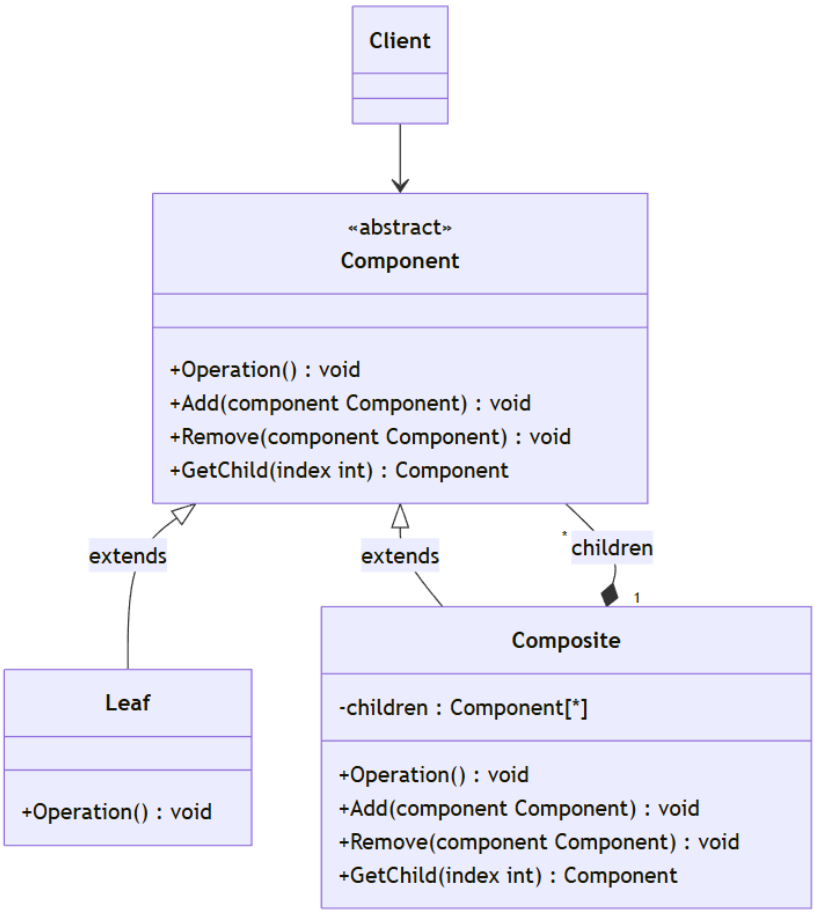
У ході виконання лабораторної роботи було досліджено та реалізовано поведінковий патерн проєктування «Інтерпретатор» (Interpreter). Цей патерн було застосовано для розбору та виконання командних рядків, що містять оператор конвеєра (|). Було розроблено граматику, де окремі команди виступають

термінальними виразами, а операція пайпу — нетермінальним. Реалізація класу ExpressionParser дозволила автоматично перетворювати текстовий ввід користувача в абстрактне синтаксичне дерево (AST). Це забезпечило гнучку обробку даних: результат виконання лівої частини виразу динамічно передається у праву частину через об'єкт контексту (InterpreterContext). Такий підхід робить систему розширюваною, дозволяючи в майбутньому легко додавати нові оператори та правила обробки без зміни основної логіки терміналу.

6. Контрольні питання:

1. Яке призначення шаблону «Композит»? Шаблон використовується для складання об’єктів у деревоподібну структуру для подання ієрархій типу «частина — ціле». Дозволяє уніфіковано обробляти поодинокі об’єкти та композиції.

2. Нарисуйте структуру шаблону «Композит».



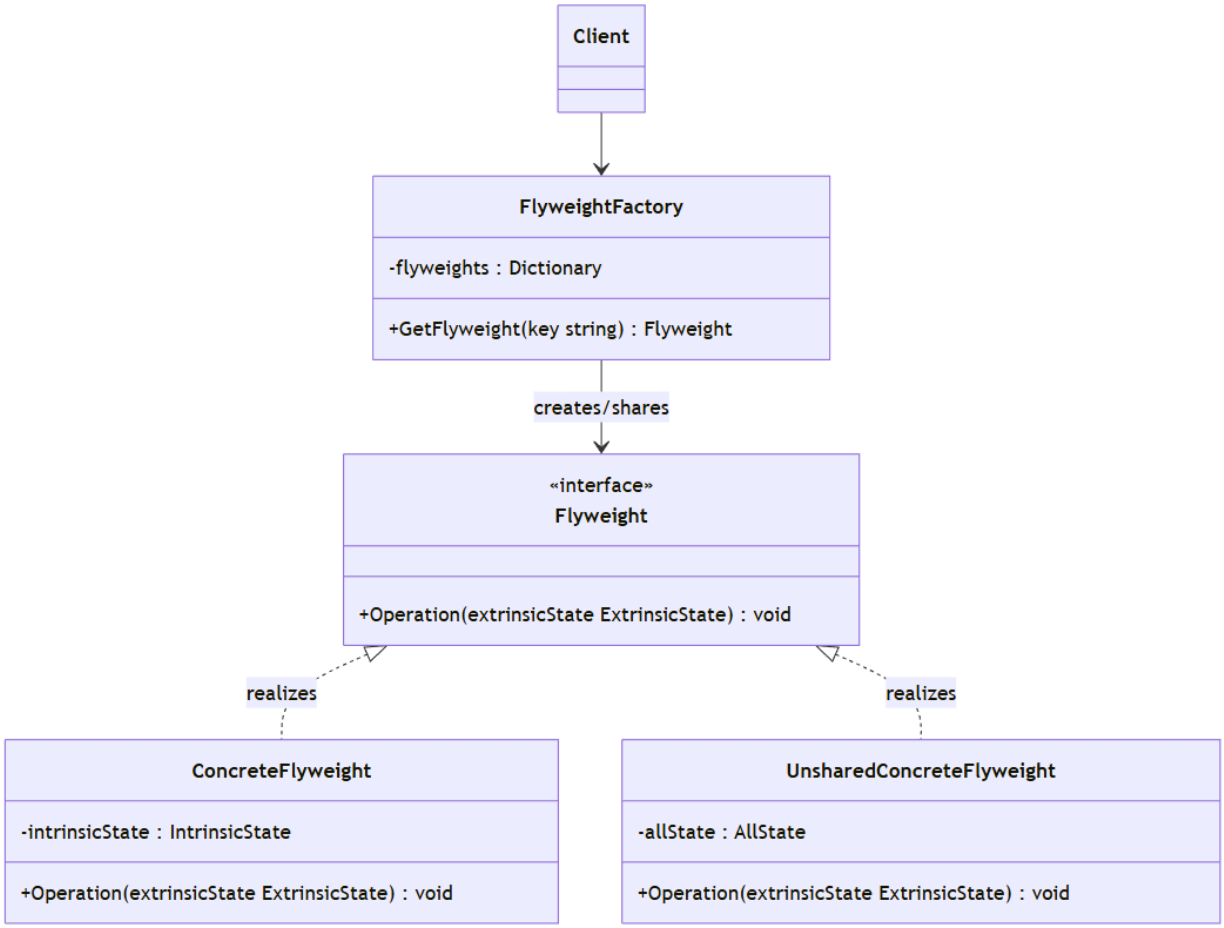
3. Які класи входять в шаблон «Композит», та яка між ними взаємодія?

- Component — абстрактний клас/інтерфейс з операціями.

- Leaf — кінцевий елемент, реалізує операції.
- Composite — контейнер, містить children, рекурсивно делегує операції. Взаємодія: клієнт працює з Component, не розрізняючи Leaf і Composite.

4. Яке призначення шаблону «Легковаговик»? Зменшення кількості об’єктів у пам’яті шляхом поділу спільного (внутрішнього) стану між кількома екземплярами.

5. Нарисуйте структуру шаблону «Легковаговик».



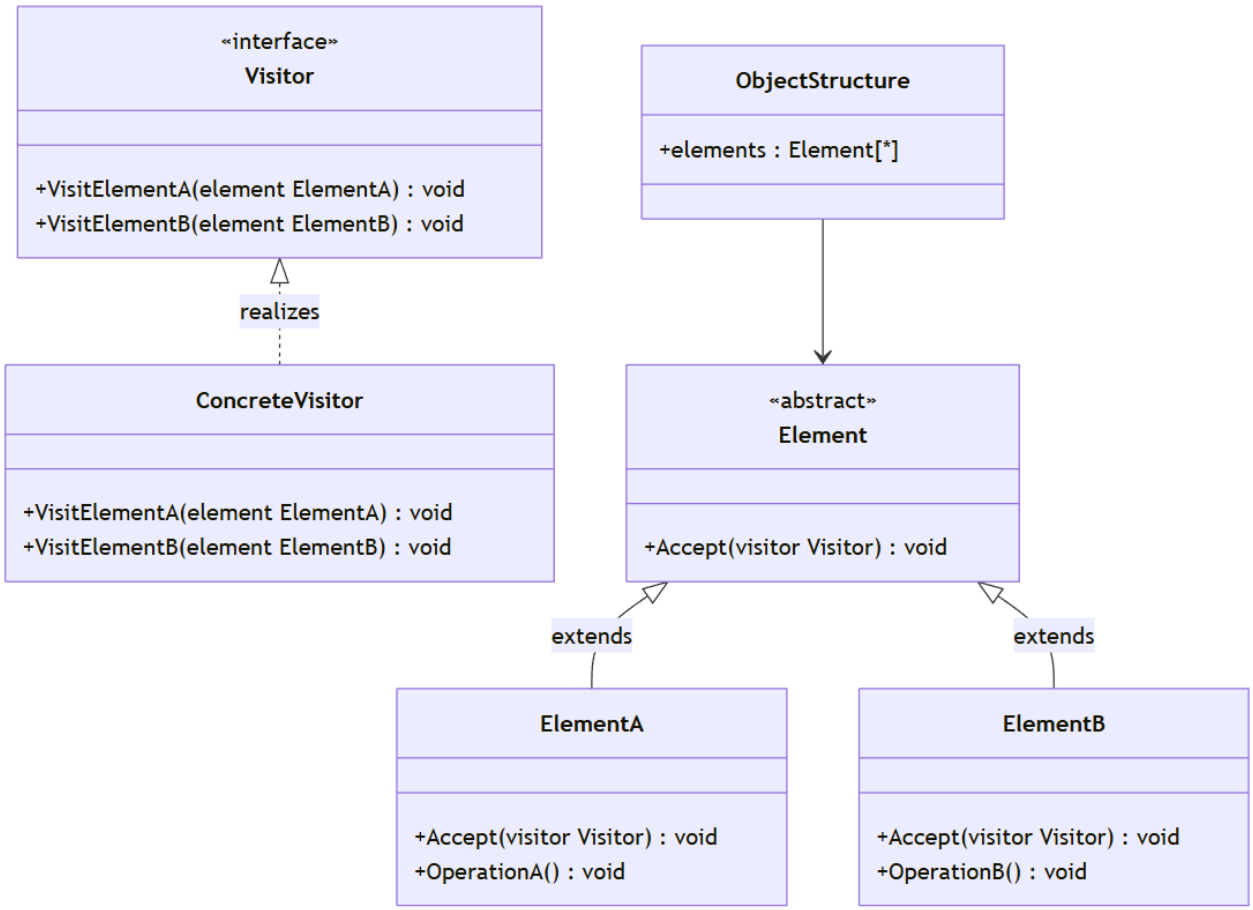
6. Які класи входять в шаблон «Легковаговик», та яка між ними взаємодія?

- Flyweight — інтерфейс з операцією, що приймає зовнішній стан.
- ConcreteFlyweight — зберігає внутрішній стан, спільний.
- UnsharedConcreteFlyweight — не поділюваний варіант.
- FlyweightFactory — кешує та повертає Flyweight за ключем. Взаємодія: клієнт отримує Flyweight з фабрики, передає зовнішній стан у метод.

7. Яке призначення шаблону «Інтерпретатор»? Подання граматики мови та інтерпретатора для обчислення виразів у термінах абстрактного синтаксичного дерева (AST).

8. Яке призначення шаблону «Відвідувач»? Дозволяє додавати нові операції над елементами ієрархії без зміни їх класів, відокремлюючи логіку від структури.

9. Нарисуйте структуру шаблону «Відвідувач».



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

- Visitor — інтерфейс з методами Visit... для кожного типу елемента.
- ConcreteVisitor — реалізує операції.
- Element — інтерфейс з Accept(Visitor).
- ConcreteElement — викликає відповідний Visit... у відвідувача.
- ObjectStructure — колекція елементів. Взаємодія: елемент викликає visitor.Visit(this), відвідувач виконує логіку за типом.