Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут» імені Ігоря Сікорського
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота № 8

із дисципліни: «Технології розроблення програмного забезпечення»

на тему: «Патерни проектування.**»**

Виконав:
студент групи ІА-34
Вінницький Г.Р.
Перевірил:
Мягкий Михайло Юрійович

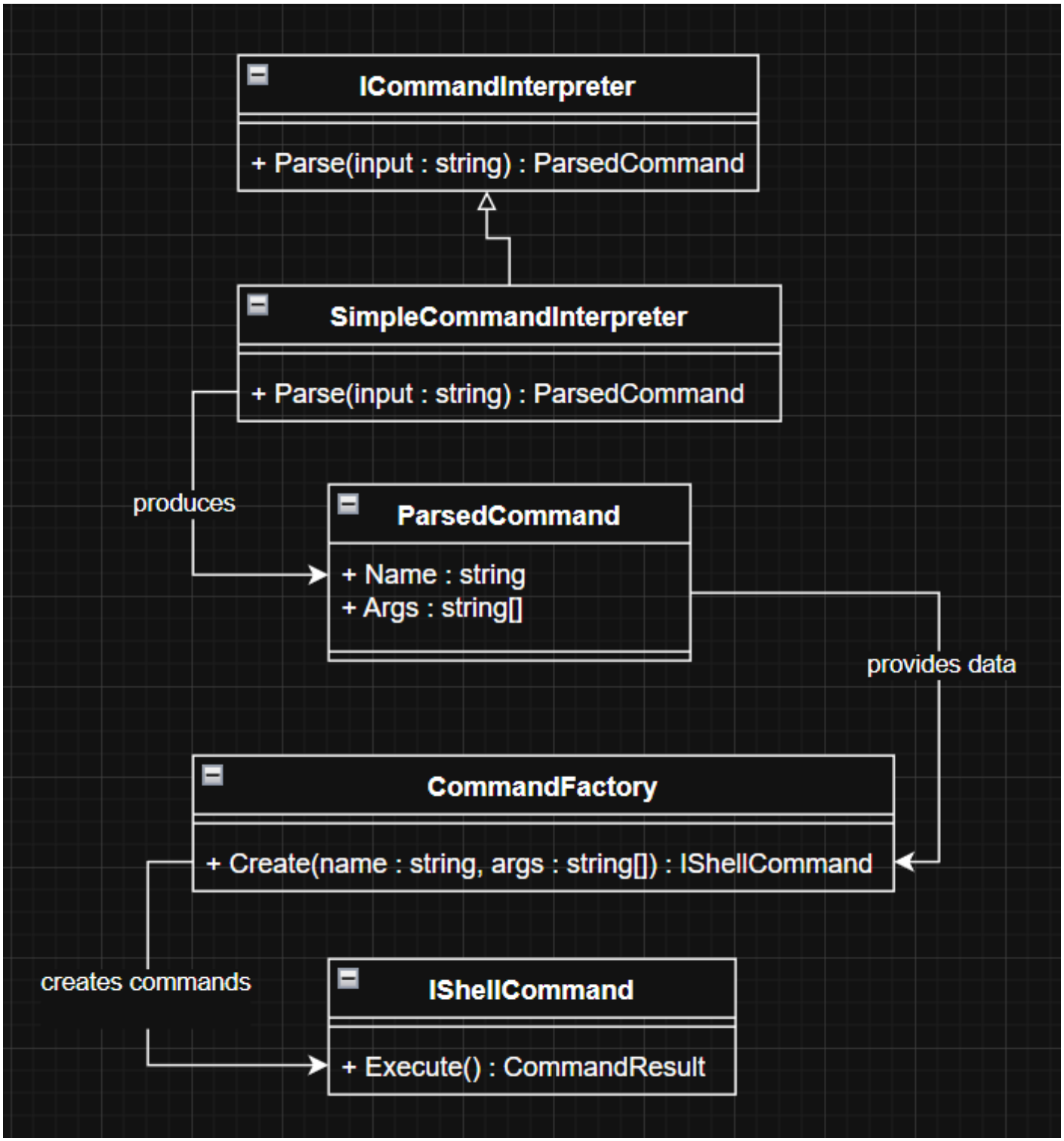
Київ-2025

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

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.

18. **Shell (total commander)** (state, prototype, factory method, template method, interpreter, client-server)

Оболонка повинна вміти виконувати основні дії в системі – перегляд файлів папок в файлової системі, перемикання між дисками, копіювання, видалення, переміщення об'єктів, пошук.



Interpreter Pattern

Опис діаграми (Interpreter)

1. Загальна ідея патерну Interpreter

Патерн Interpreter застосовується для побудови невеликих доменно-орієнтованих мов та механізмів розбору текстових інструкцій.

У твоєму проєкті такою мовою є командна мова файлового менеджера, де кожна команда вводиться текстом:

copy C:\a.txt D:\b.txt

delete log.txt

cd "C:\Users\Admin"

ls

search report

Interpreter дозволяє: описати граматику команд, реалізувати розбір тексту, перетворити текст у модель (ParsedCommand), ізолювати розбір від створення та виконання команд. Тобто файловий менеджер має власну невелику мову, і SimpleCommandInterpreter — це її інтерпретатор.

2. Опис елементів діаграми

Діаграма містить чотири ключові частини патерну:

1. Інтерпретатор команди — ICommandInterpreter, SimpleCommandInterpreter
2. Абстрактне представлення результату — ParsedCommand
3. Фабрика команд (зв’язок із іншим патерном) — CommandFactory
4. Ієрархія команд (Command Pattern) — IShellCommand + реалізації

2.1. ICommandInterpreter — інтерфейс для інтерпретації мови

Це верхньорівневий контракт, який:

1) задає стандартизований метод Parse()
2) дозволяє замінювати реалізацію інтерпретатора
3) відділяє синтаксичний аналіз від решти системи

Інтерфейс надає можливість:

1) створювати різні інтерпретатори (простий, розширений, з підтримкою лапок, з підтримкою пайпів і т. д.)
2) тестувати окремо механізм розбору

Таким чином, система залишається розширюваною.

2.2. SimpleCommandInterpreter — конкретний інтерпретатор

Це реальна реалізація Interpreter у проєкті.

Його відповідальність:

1) отримати текст користувача
2) очистити його
3) розбити на токени
4) визначити назву команди
5) виділити аргументи
6) повернути об’єкт ParsedCommand

2.3. ParsedCommand — результат роботи Interpreter

Це структурований об'єкт, який виступає аналогом AST (Abstract Syntax Tree), але в спрощеній формі.

ParsedCommand містить:

1) Name — ідентифікатор команди (наприклад, "copy")
2) Args — список аргументів

Цей клас розділяє:

1) аналіз мови (Interpreter)
2) логіку створення команд (Factory)

2.4. CommandFactory — компонент, який створює об’єкти команд

Interpreter НЕ створює команд — лише інтерпретує текст.

Створення команд — це відповідальність Factory Method або Prototype Factory.

Factory:

1) отримує ParsedCommand,
2) визначає який клас команди створити,
3) повертає IShellCommand.

Висновок:

У ході виконання лабораторної роботи було детально досліджено шаблон проектування Interpreter та реалізовано його у програмній системі *Shell Total Commander*. Патерн використано для побудови власної міні-мови команд, що забезпечує інтуїтивну взаємодію користувача з файловим менеджером.

У проєкті створено інтерфейс ICommandInterpreter і конкретний інтерпретатор SimpleCommandInterpreter, який виконує синтаксичний аналіз текстових інструкцій. Введений користувачем рядок перетворюється у структуру ParsedCommand, що є внутрішнім абстрактним представленням команди, незалежним від способу її виконання.

Таке розділення дозволяє:

1) підтримувати чітку ізоляцію логіки інтерпретації та логіки виконання команд;
2) легко розширювати мову новими командами без змін інтерпретатора;
3) підвищувати масштабованість і тестованість системи;

4) інтегрувати патерни Factory Method, Command та Prototype в єдину архітектуру;
5) забезпечити користувачу можливість взаємодіяти з системою природною текстовою мовою.

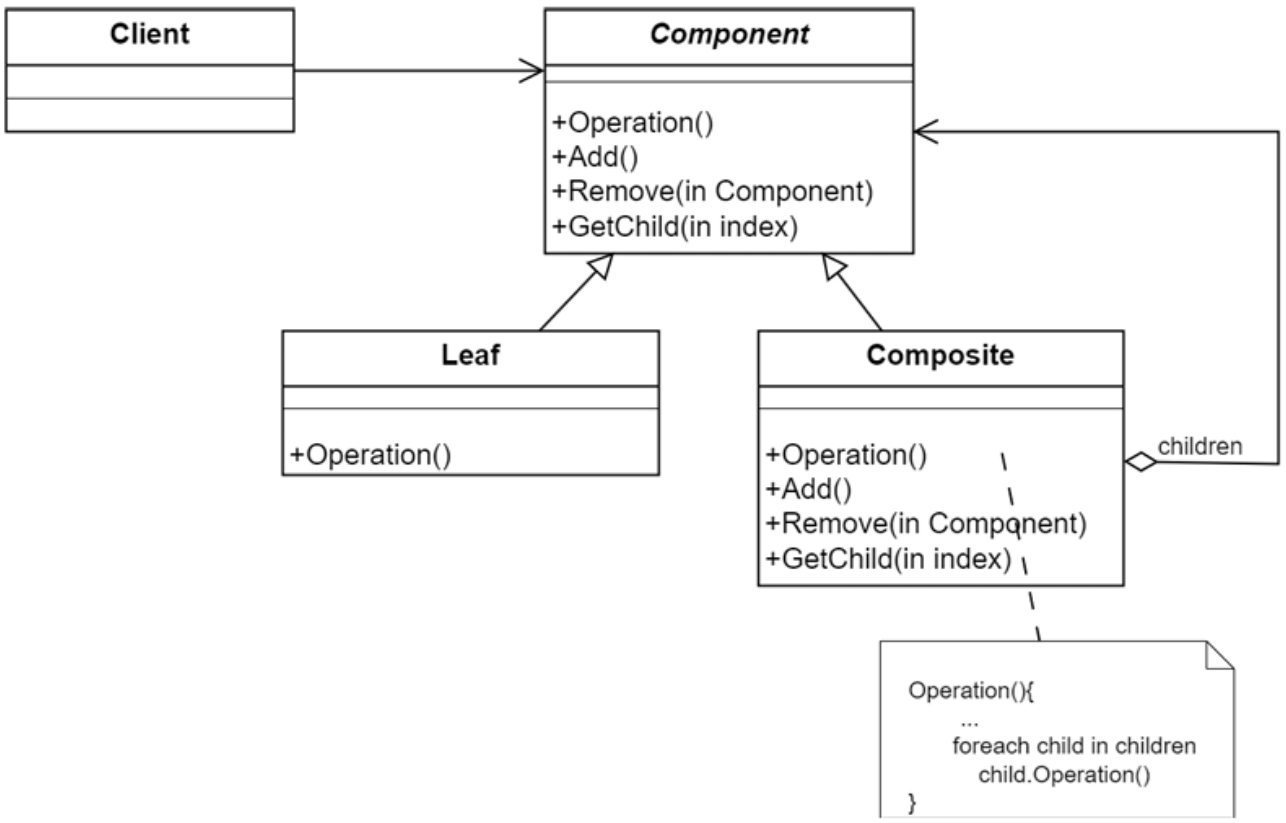
Таким чином, застосування патерну Interpreter у курсовому проєкті є повним, коректним та демонструє всі переваги структурного підходу до побудови командних інтерфейсів.

Питання до лабораторної роботи:

1. Яке призначення шаблону «Композит»?

Шаблон «Композит» використовується для представлення ієрархічних структур у вигляді дерева, де окремі об’єкти та групи об’єктів обробляються однаково. Він дозволяє клієнтам працювати з окремими елементами та з їх композиціями через єдиний інтерфейс, не розрізняючи «листки» та «вузли». Основна мета — забезпечити можливість рекурсивної роботи з вкладеними структурами, не ускладнюючи код клієнта.

2. Нарисуйте структуру шаблону «Композит».



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

У шаблон входять три основні типи класів:

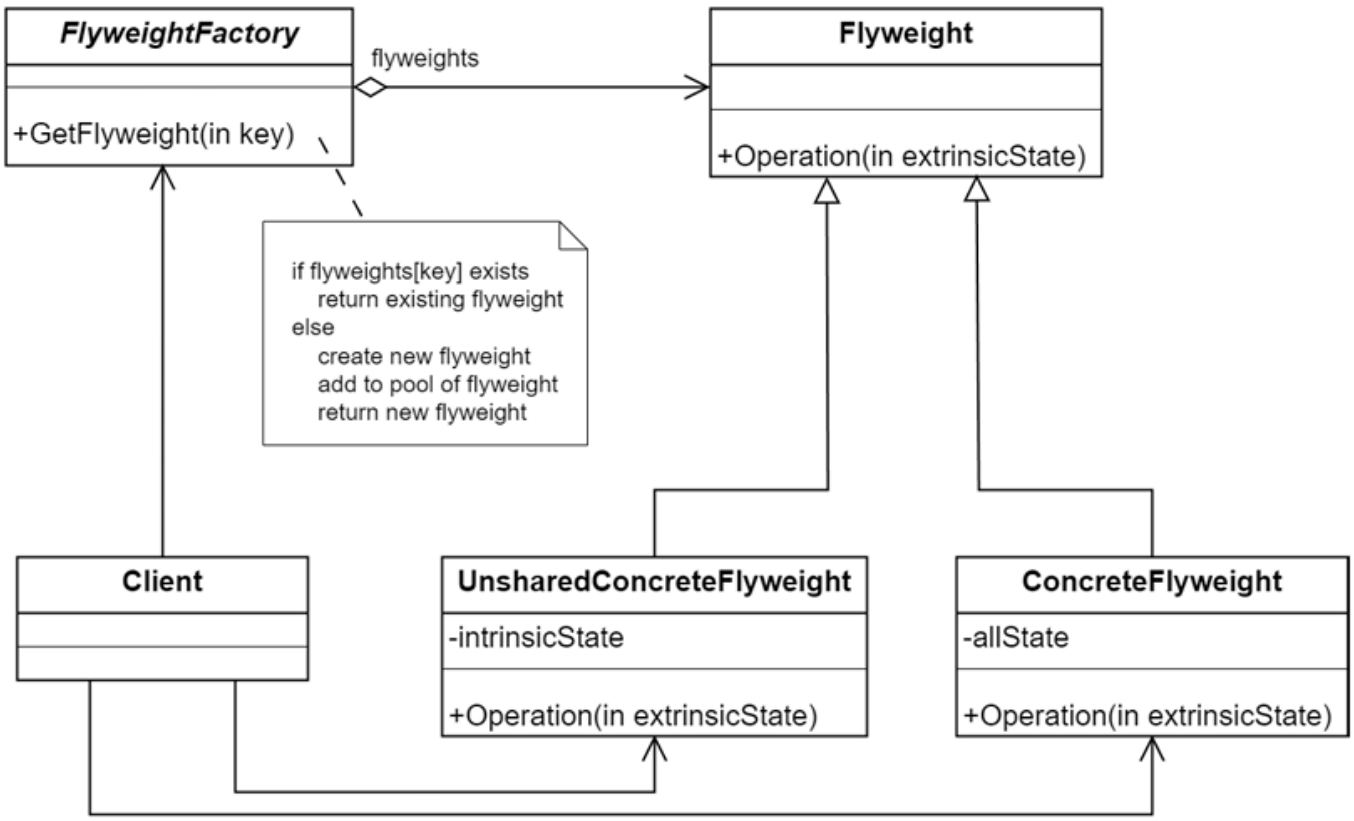
Component — визначає інтерфейс операцій, спільний для листків і вузлів;

Leaf — реалізує поведінку кінцевого елемента, який не має дочірніх компонентів;
Composite — містить колекцію дочірніх компонентів і делегує операції кожному з них.
Взаємодія полягає в тому, що Composite викликає ті ж методи, що й Leaf, але передає виклики своїм дочірнім елементам. Клієнт не знає, працює він з Composite чи Leaf — це робить структуру прозорою.

4. Яке призначення шаблону «Легковаговик»?

Шаблон «Легковаговик» застосовується для оптимізації використання пам’яті шляхом спільного використання однакових об’єктів. Він дозволяє зберігати тільки один екземпляр об’єкта, який повторюється в системі багато разів. Змінні частини стану виносять у зовнішній контекст, тоді як незмінні зберігаються у внутрішньому стані легковаговика. Цей шаблон часто використовується там, де кількість однотипних об’єктів може бути дуже великою.

5. Нарисуйте структуру шаблону «Легковаговик»



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

До шаблону входять такі класи:

Flyweight — абстрактний базовий тип об’єктів, що можуть бути спільно використані;
ConcreteFlyweight — конкретний об’єкт, який має внутрішній незмінний стан;
FlyweightFactory — створює та керує колекцією легковагових об’єктів, повертаючи

вже існуючий екземпляр при повторному запиті;

Client — передає зовнішній змінний стан об’єкту під час виконання операцій.

Взаємодія: клієнт звертається до фабрики, отримує або існуючий легковаговик, або новий (якщо немає в кеші), і викликає його методи, доповнюючи їх зовнішнім станом.

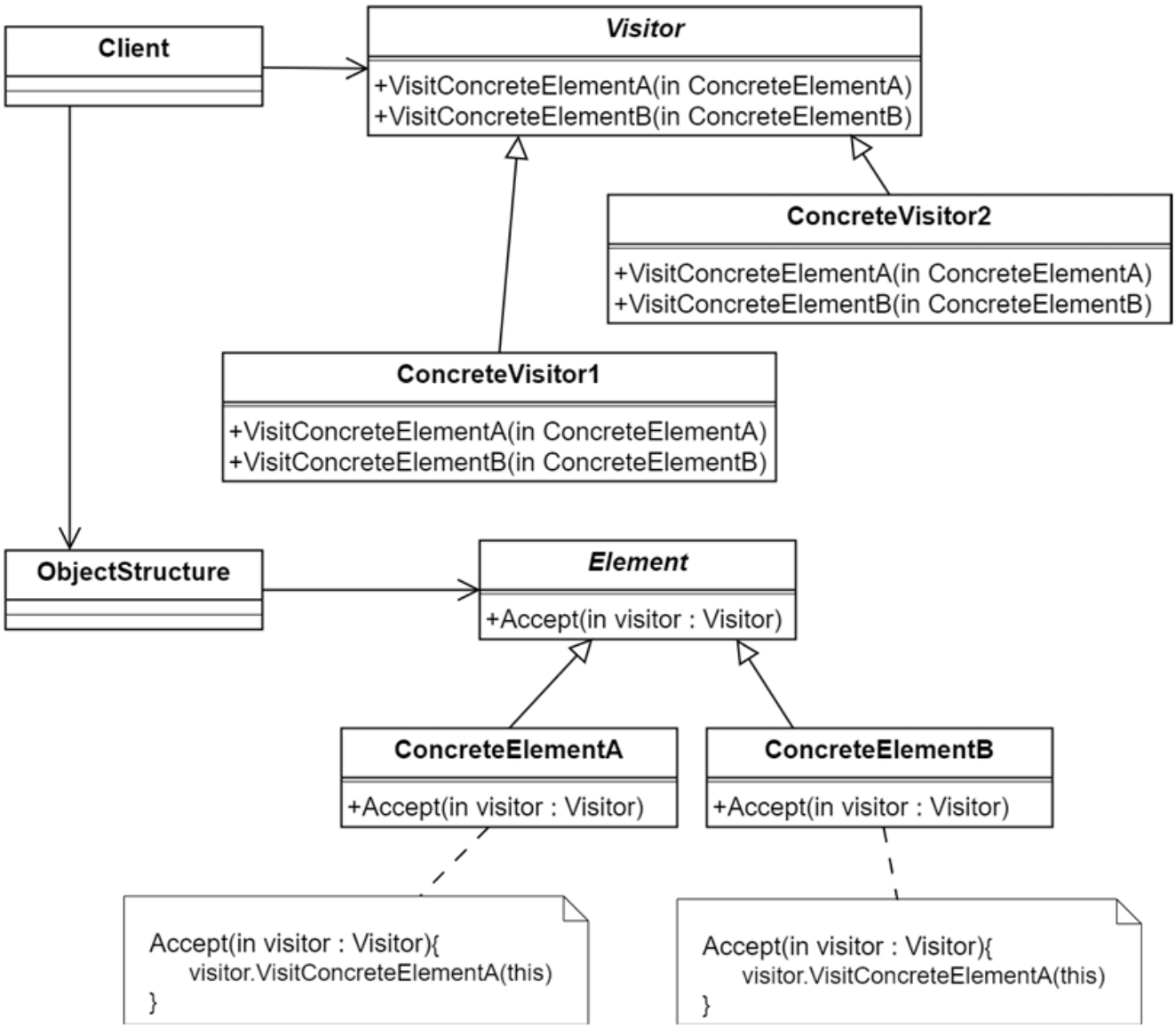
7. Яке призначення шаблону «Інтерпретатор»?

Шаблон «Інтерпретатор» призначений для визначення граматики простої мови та побудови інтерпретатора, який виконує вирази цієї мови. Він використовується, коли потрібно опрацьовувати багато однотипних інструкцій або математичних/логічних виразів. Структура шаблону уможливлює побудову синтаксичного дерева, де кожен вузол представляє частину мови або операцію.

8. Яке призначення шаблону «Відвідувач»?

Шаблон «Відвідувач» дозволяє додавати нові операції до складних структур об’єктів, не змінюючи самих класів цих об’єктів. Він відокремлює алгоритм від структури даних, даючи змогу додавати нові функції, що виконуються над елементами, без зміни їхнього коду. Завдяки цьому шаблон корисний у системах, де потрібно часто додавати нові операції.

9. Нарисуйте структуру шаблону «Відвідувач»



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

У шаблон входять:

Visitor — визначає набір методів для роботи з різними типами елементів;

ConcreteVisitor — містить реалізацію дій, які виконуються над елементами;

Element — оголошує метод accept(), що приймає відвідувача;

ConcreteElement — реалізує accept(), передаючи себе конкретному методу відвідувача;

ObjectStructure — управляє колекцією елементів та дозволяє відвідувачу пройтись по структурі.

Взаємодія: ConcreteElement приймає відвідувача і викликає метод, що відповідає його типу. Visitor отримує доступ до внутрішнього стану елементів і виконує над ними відповідні операції.