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

Лабораторна робота №8
«Технології розроблення програмного забезпечення»
Тема: « E-mail клієнт»

Виконав:
Студент групи ІА-34
Марченко А.О.
Перевірив:
Мягкий Михайло Юрійович

Київ 2025

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

15. **E-mail клієнт** (singleton, builder, decorator, template method, interpreter, client-server)

Поштовий клієнт повинен нагадувати функціонал поштових програм Mozilla Thunderbird, The Bat і т.д. Він повинен сприймати і коректно обробляти pop3/smtp/imap протоколи, мати функції автонастройки основних поштових провайдерів для України (gmail, ukr.net, i.ua), розділяти повідомлення на папки/категорії/важливість, зберігати чернетки незавершених повідомлень, прикріплювати і обробляти прикріплені файли.

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. 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.

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.

Шаблон «Flyweight»

Призначення: Шаблон використовується для зменшення кількості об'єктів вдодатку шляхом поділу цих об'єктів між ділянками додатку. Flyweight являє собою поділюваний об'єкт.

Переваги та недоліки:

+ Заощаджує оперативну пам’ять.

- Витрачає процесорний час на пошук/обчислення контексту.

- Ускладнює код програми внаслідок введення безлічі додаткових класів.

Шаблон «Interpreter»

Призначення: Даний шаблон використовується для подання граматики інтерпретатора для вибраної мови (наприклад, скриптової) . Граматика мови представлена термінальними і нетермінальними символами, кожен з яких інтерпретується в контексті використання. Клієнт передає контекст і сформовану пропозицію в використовувану мову в термінах абстрактного синтаксичного дерева (деревоподібна структура, яка однозначно визначає ієрархію виклику підвиразів), кожен вираз інтерпретується окремо з використанням контексту. У разі наявності дочірніх виразів, батьківський вираз інтерпретує спочатку дочірні (рекурсивно), а потім обчислює результат власної операції.

Переваги та недоліки:

+ Граматику стає легко розширювати та змінювати, реалізації класів, що описують вузли абстрактного синтаксичного дерева схожі (легко кодуються).

+ Можна легко змінювати спосіб обчислення виразів.

- Супроводження граматики с великою кількістю правил є проблематичним.

Шаблон «Visitor»

Призначення: Шаблон відвідувач дозволяє вказувати операції над елементами без зміни структури конкретних елементів [6]. Таким чином вкрай зручно додавати нові операції, проте дуже важко додавати нові елементи в ієрархію (необхідно додавати відповідні методи для обробки їх відвідувань в кожного відвідувача). Даний шаблон дозволяє групувати однотипні операції, що застосовуються над різнотипними об'єктами.

Приклад з життя: Прикладом може служити написання компілятора. Припустимо, існують різні об'єкти в синтаксисі мови програмування: виклики методів і умовні вирази. Компліятор перед генерацією коду повинен обійти всі вирази (і виклики методів, і умовні вирази) і перевірити безпеку типів, після чого згенерувати відповідний код. Відповідно буде два відвідувачі – для перевірки безпеки типів і для генерації коду. У кожного з них буде по 2 методи – для викликів методів і для умовних операцій. Таким чином при необхідності додавання нових кроків компіляції досить буде визначити нового «відвідувача» і викликати його у відповідний час.

Хід роботи

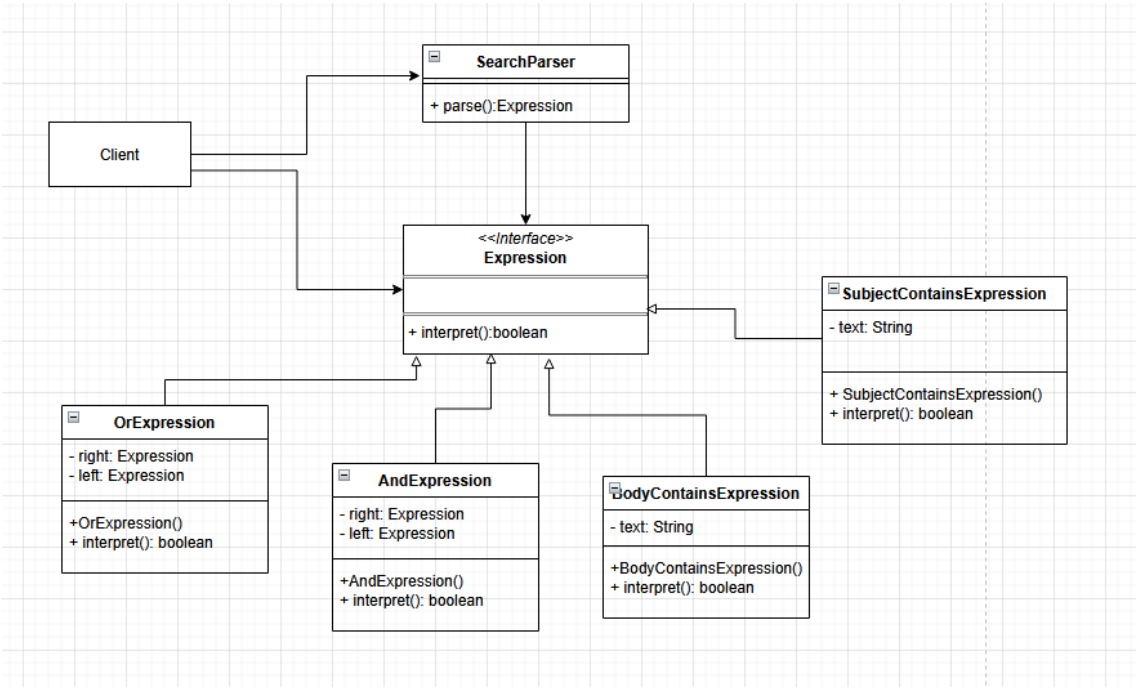


Рис. 8.1-Реалізація патерну Інтерпретатор

На діаграмі зображено реалізацію патерна Інтерпретатор, який дозволяє будувати та виконувати просту мову пошуку для фільтрації листів. У центрі знаходиться інтерфейс Expression, що визначає метод interpret(), який повертає логічне значення — чи відповідає конкретний email певному критерію. Від нього успадковуються два типи виразів. Перший — термінальні вирази, які безпосередньо перевіряють властивості листа: SubjectContainsExpression перевіряє, чи містить тема потрібний текст, а BodyContainsExpression — чи містить цей текст тіло листа. Другий тип — нетермінальні вирази, які комбінують інші вирази у складні логічні конструкції: AndExpression повертає true, якщо обидва підвирази виконуються, а OrExpression — якщо виконується хоча б один.

Клас SearchParser відповідає за розбір текстового пошукового запиту (наприклад, “subject:urgent AND body:report”) і побудову дерева об’єктів Expression, яке відображає логіку запиту. Клієнтський код передає запит у SearchParser, отримує кореневий вираз і викликає його метод interpret() для кожного листа. Таким чином, діаграма демонструє повністю структуровану модель мови пошуку, де кожен елемент запиту представлений окремим класом, а складні умови формуються шляхом композиції об’єктів.

Код патерну:

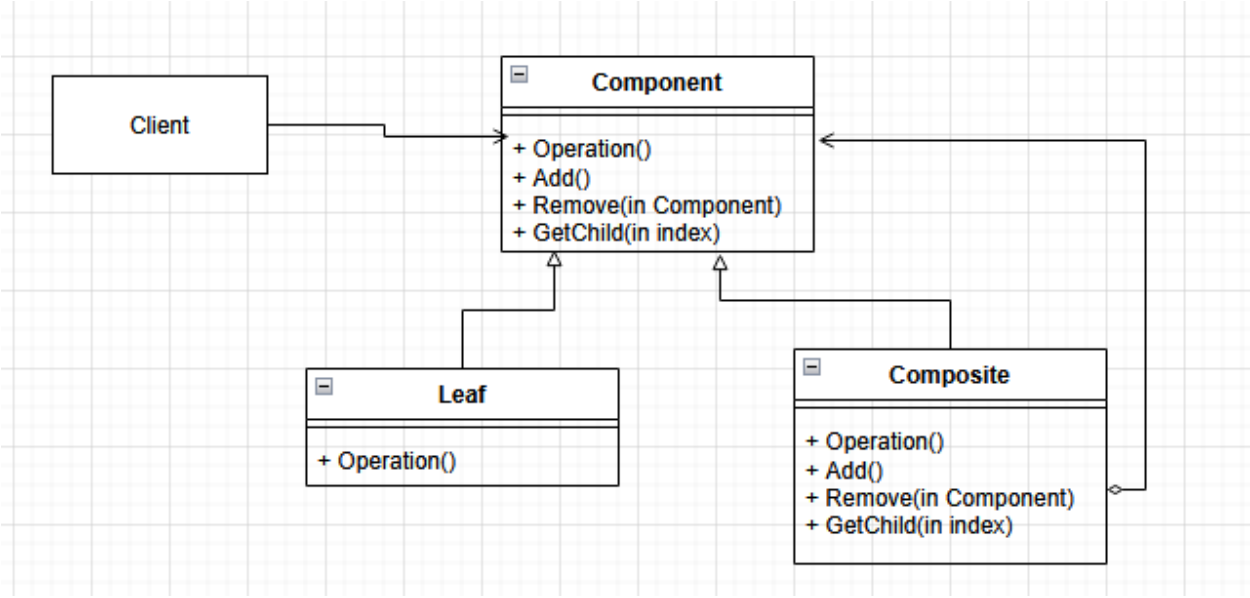
https://github.com/BeautifulBublik/TRPZ/tree/master/EmailClient/src/main/java/com/example/emailclient/service/interpreter

Питання до лабораторної роботи

1. Яке призначення шаблону «Композит»?

   Шаблон «Композит» (Composite) призначений для об’єднання об’єктів у древовидні структури "частина-ціле" та дозволяє клієнтам однаково працювати як з окремими об'єктами (листками), так і з їхніми групами (вузлами) через спільний інтерфейс.

2. Нарисуйте структуру шаблону «Композит».



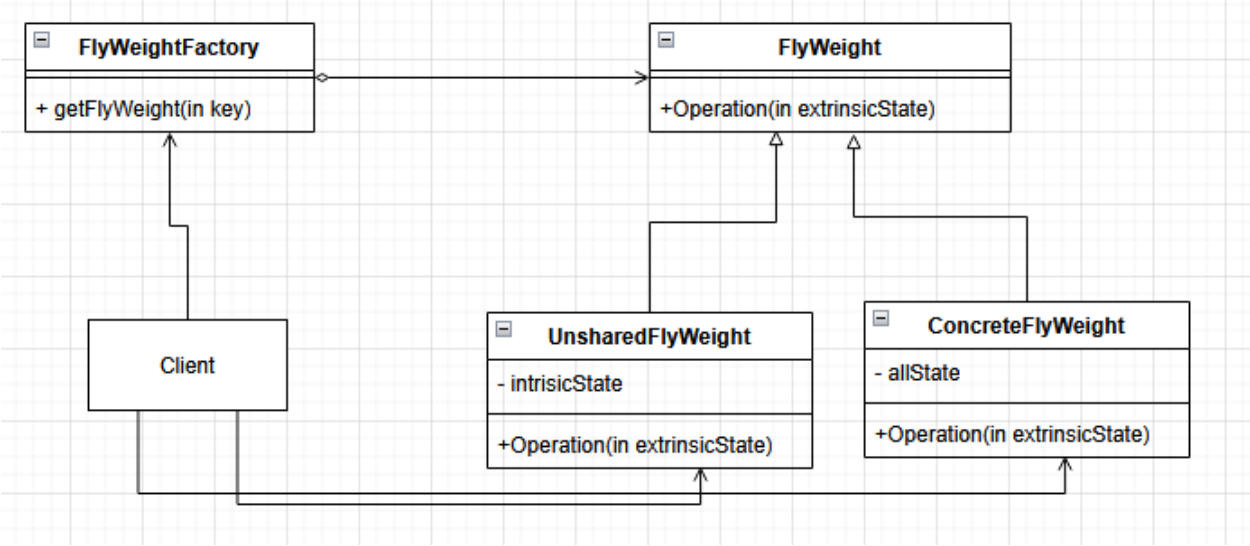
3. Які класи входять в шаблон «Композит», та яка між ними взаємодія?

   Шаблон проєктування Композит (Composite) дозволяє будувати деревоподібні структури, об'єднуючи індивідуальні об'єкти та їх композиції в єдиний клас, що робить їх схожими для клієнта; він включає Компонент (загальний інтерфейс), Листок (прості об'єкти) та Композит (контейнер для листків і інших композитів), взаємодіючи так, що клієнт працює з компонентом, а Композит керує ієрархією, делегуючи операції листкам.

4. Яке призначення шаблону «Легковаговик»?

Призначення шаблону «Легковаговик» полягає в економії пам'яті шляхом спільного використання об'єктів для зберігання даних, які повторюються. Це структурний патерн проєктування, який дозволяє вмістити велику кількість об'єктів в обмеженій оперативній пам'яті, розподіляючи спільний стан між ними замість того, щоб копіювати однакові дані в кожен об'єкт

5. Нарисуйте структуру шаблону «Легковаговик».



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

Flyweight (Легкий об'єкт) інтерфейс або клас, який представляє спільні дані. Це те, що ми хочемо зберегти (наприклад, символ літери, текстура). ConcreteFlyweight (Конкретний Легкий об'єкт) реалізація, що містить спільні, незмінні дані. Client (Клієнт) код, що використовує легкі об'єкти. FlyweightFactory (Фабрика Легких об'єктів) створює та керує пулом легких об'єктів. Context (Контекст) унікальні, змінні дані для кожного об'єкта (наприклад, позиція символу на екрані, колір).

7. Яке призначення шаблону «Інтерпретатор»?

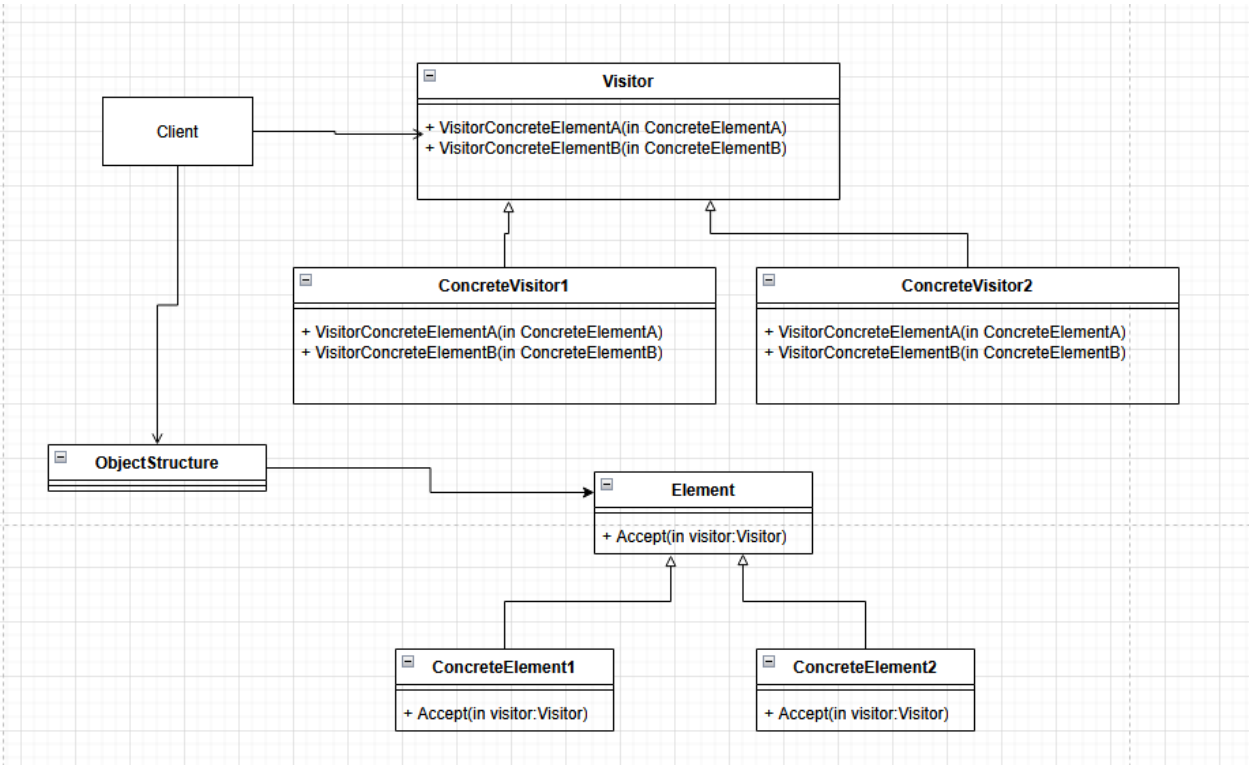
Призначення шаблону «Інтерпретатор» — визначати представлення граматики для заданої мови та створювати інтерпретатор для речень цієї мови, дозволяючи програмам представляти та виконувати інструкції, що описують певний набір правил, часто у формі абстрактних синтаксичних

дерев. Це корисно для реалізації компіляторів, парсерів, систем запитів та будь-яких систем, де потрібно інтерпретувати складні вирази чи команди.

8. Яке призначення шаблону «Відвідувач»?

Призначення шаблону «Відвідувач» полягає в тому, щоб дозволити додавати нові операції до ієрархії класів, не змінюючи самих цих класів. Він переносить логіку операцій до окремого класу «Відвідувач», що дозволяє легко додавати нову функціональність, створюючи похідні класи Відвідувача.

9. Нарисуйте структуру шаблону «Відвідувач».



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

Шаблон «Відвідувач» складається з таких основних класів: Visitor, який визначає методи visit() для різних елементів; ConcreteVisitor, що реалізує конкретні операції; Element, який має метод accept(); ConcreteElement, що викликає у відвідувача відповідний метод visit(this); і ObjectStructure, яка містить набір елементів. Взаємодія відбувається так: структура об’єктів передає відвідувача кожному елементу, елемент викликає у

відвідувача метод visit, і таким чином виконуються потрібні операції без зміни самих класів елементів.

Висновок: У цій задачі я використав патерн Інтерпретатор, тому що він природно підходить для створення власної мови пошуку та її обробки. Пошуковий запит складається з простих термінів на кшталт subject:..., body:..., а також логічних операторів AND і OR. Патерн дозволяє подати кожен елемент запиту у вигляді окремого класу й будувати з них дерево виразів, яке точно відображає структуру запиту. Завдяки цьому пошукова логіка легко розширюється — можна додати нові оператори або типи умов без зміни існуючого коду. Також дослідив структуру шаблонів «Composite», «Flyweight» (Пристосуванець), «Visitor».