Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №8

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Патерни проєктування»

Варіант: E-mail клієнт

Виконав:
студент групи ІА-32
Карповець Роман

Київ - 2025

Тема: Патерни проєктування.

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

Тема проєкту: E-mail клієнт

Опис: 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.

Зміст

Теоретичні відомості

Шаблон «Interpreter»

Призначення: Даний шаблон використовується для подання граматики і 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.
Шаблон зручно використовувати в разі невеликої граматики (інакше розростеться кількість використовуваних класів) і відносно простого контексту (без взаємних залежностей і т.п.).

Даний шаблон визначає базовий каркас інтерпретатора, який за допомогою рекурсії повертає результат обчислення пропозиції на основі результатів окремих елементів.

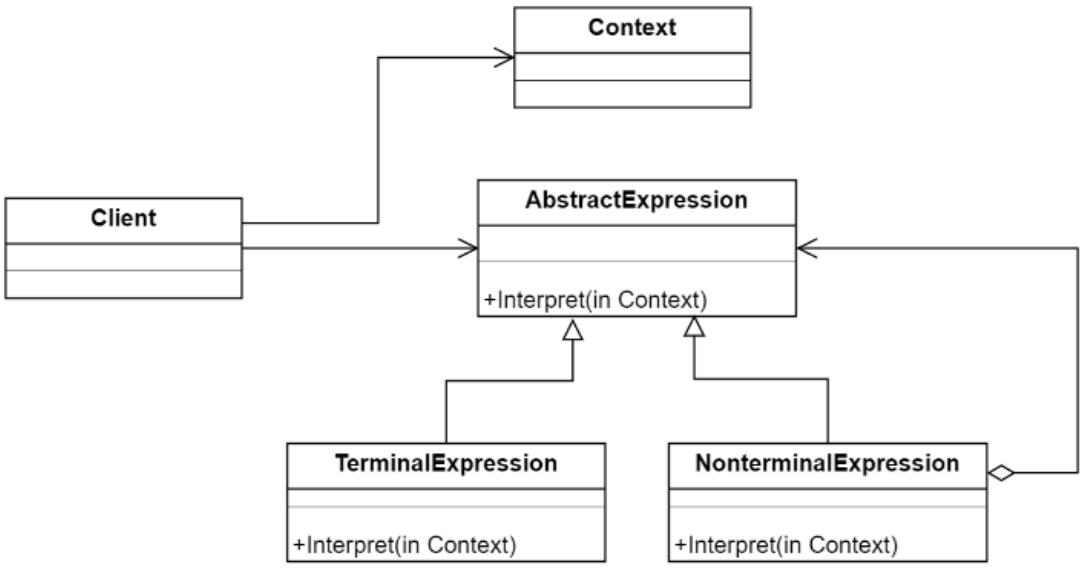

Рис. 1. Структура патерна «Інтерпретатор»

При використанні даного шаблону дуже легко реалізовується і

розширюється граматика, а також додаються нові способи інтерпретації виразів Проблема: Стоїть задача пошуку рядків по зразку, як така, що часто зустрічається. Або ж загалом є якась задача, яка дуже часто змінюється.

Рішення: Може бути вирішена шляхом створення інтерпретатора, який визначає граматику мови. «Клієнт» будує речення у вигляді абстрактного синтаксичного дерева, у вузлах якого знаходяться об’єкти класів «НетермінальнийВираз» і «ТермінальнийВираз» (рекурсивне), потім «Клієнт» ініціалізує контекст і виражає операцію Розібрати(Контекст). На кожному вузлі типу «НетермінальнийВираз» визначається операція Розібрати для кожного підвиразу. Для класу «ТермінальнийВираз» операція Розібрати визначає базу рекурсії. «АбстрактнийВираз» визначає абстрактну операцію Розібрати, загальну для всіх вузлів в абстрактному синтаксичному дереві. «Контекст» містить інформацію, глобальну по відношенню до інтерпретатора.

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.

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.

Хід роботи

Реалізація патерну Interpreter

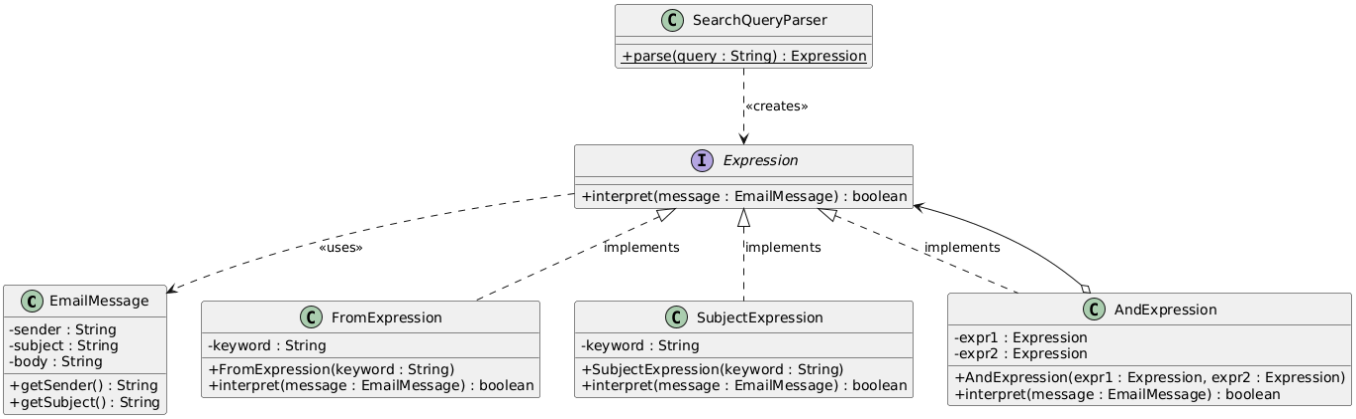

Рис. 2. Діаграма класів з використанням патерну Interpreter

Фрагмент коду з реалізацією патерну Interpreter

Expression.java

```java
package com.emailclient.service.interpreter;

import com.emailclient.model.EmailMessage;

public interface Expression {
    boolean interpret(EmailMessage message);
}
```

FromExpression.java

```java
package com.emailclient.service.interpreter;

import com.emailclient.model.EmailMessage;

public class FromExpression implements Expression {
    private String data;

    public FromExpression(String data) {
        this.data = data.toLowerCase(); // Робимо пошук нечутливим до регістру
    }

    @Override
    public boolean interpret(EmailMessage message) {
        if (message.getSender() == null) return false;
        return message.getSender().toLowerCase().contains(data);
    }
```

```
}
```

AndExpression.java

```java
package com.emailclient.service.interpreter;

import com.emailclient.model.EmailMessage;

public class AndExpression implements Expression {
    private Expression expr1;
    private Expression expr2;

    public AndExpression(Expression expr1, Expression expr2) {
        this.expr1 = expr1;
        this.expr2 = expr2;
    }

    @Override
    public boolean interpret(EmailMessage message) {
        // Повертає true, тільки якщо ОБИДВІ умови виконуються
        return expr1.interpret(message) && expr2.interpret(message);
    }
}
```

SubjectExpression.java

```java
package com.emailclient.service.interpreter;

import com.emailclient.model.EmailMessage;

public class SubjectExpression implements Expression {
    private String data;

    public SubjectExpression(String data) {
        this.data = data.toLowerCase();
    }

    @Override
    public boolean interpret(EmailMessage message) {
        if (message.getSubject() == null) return false;
        return message.getSubject().toLowerCase().contains(data);
    }
}
```

SearchQueryExpression.java

```java
package com.emailclient.service.interpreter;

public class SearchQueryParser {

   /**
    * Перетворює рядок типу "from:ivan subject:hello" у об'єкт Expression
    */
   public static Expression parse(String query) {
       Expression finalExpression = null;
       String[] tokens = query.split(" "); // Розбиваємо запит по пробілах

       for (String token : tokens) {
           Expression currentExpr = null;

           // Розпізнаємо команди
           if (token.toLowerCase().startsWith("from:")) {
               String searchTerm = token.substring(5); // Беремо текст після "from:"
               currentExpr = new FromExpression(searchTerm);
           }
           else if (token.toLowerCase().startsWith("subject:")) {
               String searchTerm = token.substring(8); // Беремо текст після "subject:"
               currentExpr = new SubjectExpression(searchTerm);
           }

           // Будуємо ланцюжок (дерево)
           if (currentExpr != null) {
               if (finalExpression == null) {
                   finalExpression = currentExpr;
               } else {
                   // Якщо вже є умова, додаємо нову через AND
                   finalExpression = new AndExpression(finalExpression, currentExpr);
               }
           }
       }
       return finalExpression;
   }
}
```

Висновок: я вивчив структуру шаблонів «Composite», «Flyweight» (Пристосуванець), «Interpreter», «Visitor» та навчився застосовувати їх в реалізації програмної системи.