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
Технології розроблення програмного забезпечення
«ШАБЛОНИ «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR»»

Виконав:
студент групи

ІА-24 Чайка А.П

Перевірив:

Мягкий М. Ю.

Київ 2024

Тема лабораторних робіт:

IRC client (singleton, builder, abstract factory, template method, composite, client-server)

Клієнт для IRC-чатів з можливістю вказівки порту і адреси з'єднання, підтримка базових команд (підключення до чату, створення чату, установка імені, реєстрація, допомога і т.д.), отримання метаданих про канал.

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.
2. Реалізувати частину функціонала робочої програми у вигляді класів і їх взаємодій для досягнення конкретних функціональних можливостей.
3. Застосування одного з даних шаблонів при реалізації програми.

Зміст

Крок 1. Теоретичні відомості 2

Крок 2. Реалізація шаблону проєктування для майбутньої системи 4

Крок 3. Зображення структури шаблону 8

Крок 4. Зображення результату виконання 8

Хід роботи:

Крок 1. Теоретичні відомості

1. Composite (Компоновщик)

Шаблон Composite використовується для представлення ієрархічної структури, де окремі об’єкти та групи об’єктів обробляються однаково. Основна ідея — створити дерево об’єктів, кожен вузол якого є або окремим елементом, або контейнером інших елементів.

Переваги:

- Спрощує роботу з ієрархічними структурами.
- Легко додаються нові типи компонентів.

Недоліки:

- Ускладнення коду через створення спільного інтерфейсу для всіх компонентів.

Де використовується:

- Файлові системи, де папки містять файли та інші папки.

- Графічні редактори для обробки груп об’єктів.

2. Flyweight (Приспособленець)

Шаблон Flyweight зменшує використання пам’яті, дозволяючи ділити загальний стан об’єктів між багатьма екземплярами. Це досягається за рахунок розділення стану на внутрішній (shared) і зовнішній (унікальний для кожного об’єкта).

Переваги:

- Зниження споживання пам’яті за рахунок повторного використання об’єктів.

Недоліки:

- Ускладнення коду через необхідність відокремлення стану.
- Підвищення вартості доступу до зовнішнього стану.

Де використовується:

- Розробка текстових редакторів (букви та шрифти можуть бути приспособленцями).
- Ігри, де потрібно відображати багато схожих об’єктів (наприклад, дерева в лісі).

3. Interpreter (Інтерпретатор)

Шаблон Interpreter використовується для розробки мов програмування або виконання команд на основі правил граматики. Кожне правило описується як клас, а синтаксичне дерево будується відповідно до виразу.

Переваги:

- Легко додавати нові правила граматики.
- Зручний для опису складних операцій з використанням власної мови.

Недоліки:

- Погана масштабованість для великих мов.
- Ускладнення підтримки для складних граматик.

Де використовується:

- SQL-подібні запити.
- Генератори коду.
- Парсери.

4. Visitor (Відвідувач)

Шаблон Visitor дозволяє додавати нові операції для класів без зміни їхньої структури. Це досягається створенням окремого об’єкта (відвідувача), який реалізує необхідну поведінку для кожного типу об’єкта.

Переваги:

- Спрощення додавання нових операцій.
- Відокремлення логіки від структури класів.

Недоліки:

- Додавання нових типів об’єктів вимагає змін у всіх відвідувачах.
- Може ускладнювати код у простих ієрархіях.

Де використовується:

- Обхід складних структур (наприклад, дерева).
- Системи обробки файлів, де різні типи файлів потребують різної обробки.
- Комп’ютерні графіки для відображення елементів на екрані

Крок 2. Реалізація шаблону проєктування для майбутньої системи

Цей код демонструє використання шаблону проектування Composite (Компонувальник) для реалізації системи команд в IRC-клієнті. Шаблон Composite дозволяє створювати структури об'єктів у вигляді дерев, де кожен об'єкт може бути як окремим елементом, так і контейнером для інших елементів. Це дозволяє клієнтам працювати з окремими об'єктами та їхніми групами однаково.

Опис класів та інтерфейсу
1.Інтерфейс IRCComponent:

```
1  public interface IRCComponent {  5 usages
2      void execute();  1 usage  2 implementations
3  }
```

Рис. 1 – Код інтерфейсу IRCComponent

Інтерфейс IRCComponent визначає метод execute(), який повинні реалізувати всі компоненти. Це дозволяє виконувати команди незалежно від того, чи є вони окремими командами або групами команд.

2.Клас IRCCommand

```
1   public class IRCCommand implements IRCComponent {  no usages  new *
2       private String command;  2 usages
3
4       public IRCCommand(String command) { this.command = command; }
7
8       @Override  1 usage  new *
9       public void execute() {
10          System.out.println("Executing command: " + command);
11      }
12  }
```

Рис. 2 – Код класу IRCCommand

Клас IRCCommand реалізує інтерфейс IRCComponent і представляє окрему команду. Він має поле command, яке зберігає текст команди, та метод execute(), який виводить команду на консоль.

3.Клас IRCCommandGroup:

```java
import java.util.ArrayList;
import java.util.List;

public class IRCCommandGroup implements IRCComponent {  no usages  new *
    private List<IRCComponent> components = new ArrayList<>();  2 usages

    public void addComponent(IRCComponent component) { components.add(component); }

    @Override  1 usage  new *
    public void execute() {
        for (IRCComponent component : components) {
            component.execute();
        }
    }
}
```

Рис. 3 – Код класу IRCCommandGroup

Клас IRCCommandGroup також реалізує інтерфейс IRCComponent і представляє групу команд. Він містить список компонентів components, які можуть бути як окремими командами, так і іншими групами команд. Метод addComponent() дозволяє додавати нові компоненти до групи, а метод execute() виконує всі команди, що містяться в групі.

Використання в системі IRCClient

У системі IRCClient цей шаблон дозволяє створювати складні команди, що складаються з простих команд та інших груп команд. Наприклад, можна створити групу команд для певної дії, яка включає кілька окремих команд, і виконати їх усі разом. Це демонструє гнучкість та зручність використання шаблону Composite для управління командами в IRC-клієнті.

Приклад використання:

```java
public class Main {  new *
    public static void main(String[] args) {  new *
        IRCCommand command1 = new IRCCommand("JOIN #channel");
        IRCCommand command2 = new IRCCommand("PRIVMSG #channel :Hello everyone!");
        IRCCommand command3 = new IRCCommand("PART #channel");

        IRCCommandGroup group = new IRCCommandGroup();
        group.addComponent(command1);
        group.addComponent(command2);
        group.addComponent(command3);

        group.execute();
    }
}

```

Рис. 4 – Код Main

Переваги використання цього шаблону:

1. Централізоване управління структурою: Шаблон дозволяє створювати складні ієрархії команд, де кожен елемент може бути оброблений однаковим способом через інтерфейс IRCComponent. Це спрощує роботу з групами команд, оскільки не потрібно розрізняти, чи це окрема команда, чи група.
2. Гнучкість розширення: Логіка додавання, видалення та отримання дочірніх елементів реалізована в класі IRCCommandGroup. Це дозволяє легко модифікувати структуру, додаючи нові команди або групи, без порушення існуючого коду.
3. Зменшення дублювання коду: Загальні операції над компонентами, такі як виконання команд (execute), реалізовані однаково для окремих команд (IRCCommand) і груп команд (IRCCommandGroup), що уніфікує роботу з ними.
4. Простота підтримки та розширення: Шаблон дозволяє додавати нові типи компонентів, не змінюючи існуючий код, що полегшує підтримку системи. Наприклад, можна реалізувати новий клас команд або модифікувати поведінку груп без змін у клієнтському коді.
5. Рекурсивна обробка: Група команд (IRCCommandGroup) може містити інші групи чи команди. Виконання (execute) обробляється рекурсивно, що дозволяє ефективно обробляти ієрархічні структури команд.

Цей шаблон дозволяє об'єднувати команди у групи та керувати їхньою ієрархією уніфіковано, знижуючи складність коду та полегшуючи підтримку.

Крок 3. Зображення структури шаблону

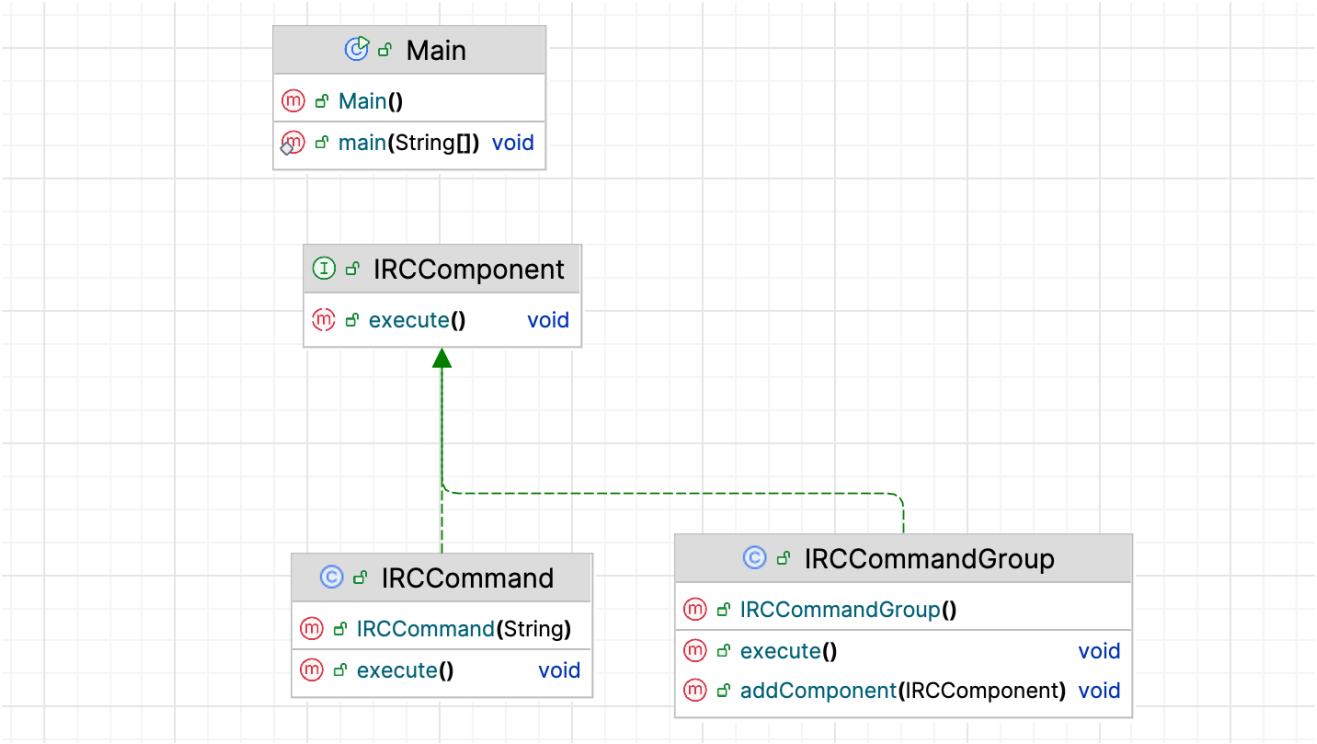

Рис. 5 – Структура шаблону Composite

Крок 4. Зображення результату виконання

```
Executing command: JOIN #channel
Executing command: PRIVMSG #channel :Hello everyone!
Executing command: PART #channel
```

Рис. 6 –Результат виконання

Шаблон Composite забезпечує побудову деревоподібної структури з об’єктів, що можуть бути як простими, так і складеними. У цьому прикладі IRCComponent визначає єдиний інтерфейс для роботи з окремими командами та їх групами. Листові компоненти, як-от IRCCommand, реалізують базову функціональність, виконуючи команду, але не підтримують операції з дочірніми елементами. Складені компоненти, такі як IRCCommandGroup, зберігають колекцію дочірніх елементів і забезпечують їх обробку, надаючи можливість додавання, видалення та доступу до вкладених об’єктів.

Загальна структура забезпечує уніфікований підхід до роботи з об’єктами незалежно від їх складності. Метод execute в IRCCommandGroup викликає виконання для кожного дочірнього елемента, що дозволяє рекурсивно

обробляти всю ієрархію. Такий підхід спрощує керування деревоподібними структурами, знижує дублювання коду та підвищує гнучкість програми, оскільки дозволяє легко додавати нові типи компонентів і розширювати функціональність без зміни загальної структури.

Висновок: У ході виконання лабораторної роботи було досягнуто наступних результатів:

Ознайомлення з теоретичними відомостями: Було розглянуто основні концепції та принципи роботи шаблонів проектування «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR». Це дозволило зрозуміти, як ці шаблони можуть бути використані для вирішення різних завдань у програмуванні.

Реалізація функціонала робочої програми: Було реалізовано частину функціонала програми у вигляді класів та їх взаємодій. Зокрема, було створено класи IRCComponent, IRCCommand, IRCCommandGroup, які дозволяють виконувати команди в IRC-клієнті.

Застосування шаблону «COMPOSITE»: Для реалізації програми було використано шаблон проектування «COMPOSITE». Цей шаблон дозволив створити структуру команд у вигляді дерева, де кожен об'єкт може бути як окремою командою, так і групою команд. Це забезпечило гнучкість та зручність управління командами в IRC-клієнті.

Загалом, виконання лабораторної роботи дозволило закріпити знання про шаблони проектування та їх застосування на практиці. Реалізація функціонала програми з використанням шаблону «COMPOSITE» продемонструвала його ефективність та зручність у вирішенні завдань, пов'язаних з управлінням командами.