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»»

Виконала:

студентка групи ІА-23

Єрмак Д. Р

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

Хід роботи:

Крок 1. Теоретичні відомості

Шаблони роботи з базами даних у корпоративних додатках дозволяють ефективно організувати доступ до даних, що зберігаються в БД, особливо в масштабних системах з великою кількістю користувачів та взаємопов'язаних програм.

Об'єктно-реляційне відображення (ORM)

Цей шаблон дозволяє відображати ряди з таблиць БД на об'єкти в коді. Кожен об'єкт містить дані та логіку доступу до них, що дозволяє зручно маніпулювати даними в об'єктно-орієнтованому стилі. Однак з ростом складності запитів та взаємодій з БД, логіка доступу часто виноситься в окремі класи.

Репозиторій

Шаблон репозиторію дозволяє відокремити логіку доступу до бази даних від бізнес-логіки. Репозиторії відповідають за виконання всіх операцій з БД (збереження, видалення, пошук) і забезпечують тестованість коду, оскільки дозволяють змінювати логіку роботи з даними без впливу на решту системи.

Часто використовується базовий клас для репозиторіїв, який абстрагує загальні операції, такі як з’єднання з БД.

Маппери

Ці об'єкти або методи здійснюють перетворення даних з об'єктів на формат, який може бути використаний для зберігання в базі даних або передачі через мережу. Маппери відповідають за відповідність між властивостями об'єктів і колонками таблиць БД, а також можуть виправляти невідповідності типів даних при передачі об'єктів.

COMPOSITE

Шаблон Composite дозволяє організувати об'єкти в деревоподібну структуру для представлення ієрархій типу «частина цілого», де окремі об'єкти (компоненти) та їх групи можуть оброблятися однаково. Це дає можливість рекурсивно виконувати операції, наприклад, на форму або її дочірні елементи, забезпечуючи уніфіковане управління ієрархіями об'єктів.

FLYWEIGHT

Шаблон Flyweight використовується для зменшення кількості об'єктів в додатку, шляхом поділу їх між різними ділянками програми. Він дозволяє створювати поділювані об'єкти, де внутрішній стан зберігається в самому об'єкті, а зовнішній стан зберігається в контексті використання, що дозволяє ефективно управляти ресурсами при роботі з великою кількістю схожих об'єктів.

INTERPRETER

Шаблон Interpreter використовується для подання граматики мови і створення інтерпретатора, який обчислює значення виразів на основі абстрактного синтаксичного дерева. Кожен елемент граматики інтерпретується окремо, а батьківські вирази рекурсивно обчислюють дочірні елементи, що дозволяє легко розширювати граматику та додавати нові способи інтерпретації. Шаблон зручний для невеликих граматик і простих контекстів.

VISITOR

Шаблон Visitor дозволяє додавати нові операції для об'єктів без зміни їх структури, забезпечуючи розділення логіки операцій і даних елементів. Він зручний для групування однотипних операцій, але ускладнює додавання нових елементів в ієрархію, оскільки вимагає змін у всіх відвідувачах.

Крок 2. Реалізація шаблону проєктування для майбутньої системи

Для реалізації програми IRClient використано шаблон проєктування Composite, який дозволяє організувати об'єкти у вигляді деревоподібної структури та уніфіковано працювати як з окремими об'єктами, так і з групами. У цьому випадку основний інтерфейс IRCComponent визначає спільний контракт для роботи з окремими командами (IRCCommand) і групами команд (IRCCommandGroup).

Клас IRCCommand реалізує базову операцію execute, яка виконує команду IRC і логує результат. Водночас клас IRCCommandGroup відповідає за об'єднання кількох команд чи груп у ієрархічну структуру. Він реалізує метод execute, який рекурсивно викликає виконання для всіх дочірніх елементів, забезпечуючи єдину послідовність обробки.

Цей підхід дозволяє легко масштабувати програму шляхом додавання нових груп команд чи окремих команд, зберігаючи простоту роботи через загальний інтерфейс. Шаблон забезпечує гнучкість у реалізації складних ієрархій команд і спрощує підтримку, дозволяючи управляти як одиничними, так і об'єднаними елементами.

```
package org.example.Composite;

public interface IRCComponent {  10 usages  2 implementations
    void execute();  2 usages  2 implementations

    default void addComponent(IRCComponent component) {  4 usages  1 override
        throw new UnsupportedOperationException("addComponent not supported");
    }

    default void removeComponent(IRCComponent component) {  no usages  1 override
        throw new UnsupportedOperationException("removeComponent not supported");
    }

    default IRCComponent getChild(int index) {  no usages  1 override
        throw new UnsupportedOperationException("getChild not supported");
    }
}
```

Рис. 1 – Код інтерфейсу IRCComponent

```java
package org.example.Composite;

import java.util.logging.Logger;

public class IRCCommand implements IRCComponent {  8 usages
    private static final Logger logger = Logger.getLogger(IRCCommand.class.getName());  1 usage
    private final String command;  2 usages

    public IRCCommand(String command) {  4 usages
        this.command = command;
    }

    @Override  2 usages
    public void execute() {
        logger.info( msg: "Executing command: " + command);
    }
}
```

Рис. 2 – Код класу IRCCommand

```java
package org.example.Composite;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

public class IRCCommandGroup implements IRCComponent {  4 usages
    private final List<IRCComponent> components = new ArrayList<>();  6 usages
    public IRCCommandGroup(String name) {components.add(new IRCCommand(name));}  2 usages
    @Override  4 usages
    public void execute() {
        for (IRCComponent component : components) {
            component.execute();
        }
    }
    @Override  4 usages
    public void addComponent(IRCComponent component) {
        if (!Objects.isNull(component)) {
            components.add(component);
        }
    }
    @Override  1 usage
    public void removeComponent(IRCComponent component) {
        components.remove(component);
    }
    @Override  1 usage
    public IRCComponent getChild(int index) {
        if (index < 0 || index >= components.size()) {
            throw new IndexOutOfBoundsException("Invalid index: " + index);
        }
        return components.get(index);
    }
}
```

Рис. 3 – Код класу IRCCommandGroup

У даній реалізації використано шаблон проєктування Composite, який дозволяє працювати з окремими об'єктами та їх групами однаковим чином. У цьому випадку об'єкти IRCCommand представляють індивідуальні команди, а IRCCommandGroup — групи команд, які можуть містити як окремі команди, так і інші групи.

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

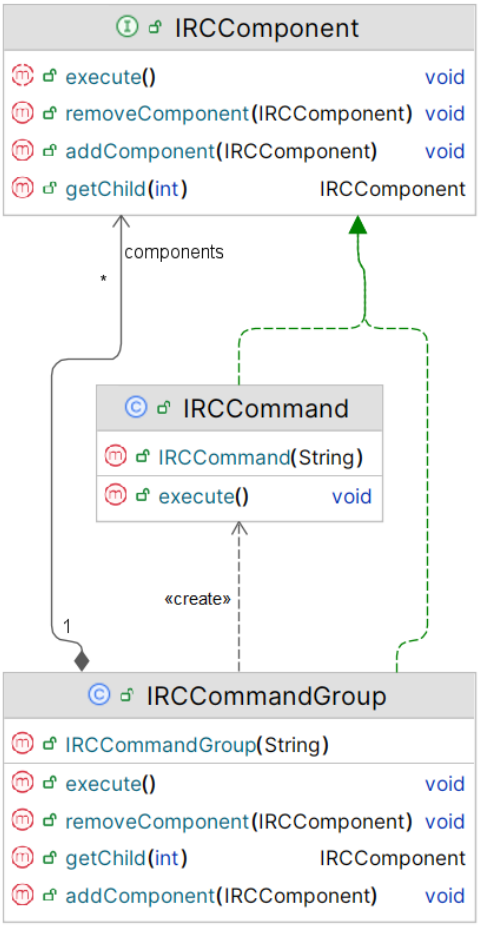

Рис. 4 – Структура шаблону Composite

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.

Код можна переглянути в даному репозиторії:
https://github.com/konrelityw/irc_chat

Висновок: Під час виконання даної лабораторної роботи було досліджено структуру, призначення, основні властивості, переваги та недоліки шаблонів проєктування «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR». Наведені шаблони мають власні особливі переваги та недоліки, а також використовуються для досягнення різних цілей. Для реалізації частини майбутньої системи було обрано шаблон Composite, який найкраще підходить у даному випадку. Далі було реалізовано обраний шаблон проєктування, розглянуто його призначення, структуру та досліджено його переваги та недоліки у використанні.