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 №7

Технології розроблення програмного забезпечення

«ШАБЛОН «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD»»

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

Принципи проектування програмного забезпечення

1. **Don't Repeat Yourself (DRY)**
   Уникання повторень у коді зменшує його розмір, підвищує читабельність і спрощує підтримку. Повторення призводять до дублювання помилок і складнощів у модернізації. Рефакторинг, наприклад, винесення спільного коду в методи чи класи, допомагає впроваджувати цей принцип.
2. **Keep It Simple, Stupid (KISS)**
   Системи мають бути простими і складатися з невеликих компонентів, кожен з яких виконує одну функцію. Простота полегшує розуміння, зменшує кількість помилок і підвищує надійність.
3. **You Only Load It Once (YOLO)**
   Ініціалізацію та завантаження конфігураційних даних потрібно виконувати лише раз на старті програми. Це зменшує затримки, пов’язані з багаторазовим доступом до ресурсів, таких як файлові системи.
4. **Принцип Парето (80/20)**
   80% результату досягається за 20% зусиль. У програмуванні це може стосуватися оптимізації, виправлення багів або розподілу навантаження.

Ефективність зусиль важлива, проте доведення до ідеалу потребує значних ресурсів.

5. **You Ain’t Gonna Need It (YAGNI)**
   Уникання надмірного функціоналу, який може не знадобитися. Простота рішення краще за універсальність, якщо вона не виправдовує витрати часу і ресурсів.

MEDIATOR

Шаблон «Mediator» координує взаємодію між об'єктами через центральний об'єкт, замінюючи прямі зв’язки між ними. Це спрощує структуру та забезпечує керованість системи.

FACADE

«Facade» надає єдиний уніфікований інтерфейс для доступу до складної підсистеми, приховуючи її внутрішню реалізацію. Це спрощує роботу з підсистемою, зменшує залежності та запобігає «спагеті-коду».

BRIDGE

«Bridge» розділяє інтерфейс і його реалізацію, дозволяючи розвивати їх незалежно. Це забезпечує гнучкість при додаванні нових абстракцій і способів їх реалізації.

TEMPLATE METHOD

Крок 2. Реалізація шаблону проєктування для майбутньої системи

Для реалізації програми IRClient використано шаблон проєктування Template Method, який визначає загальну структуру алгоритму в базовому класі, дозволяючи підкласам реалізувати окремі кроки алгоритму. У цьому випадку клас IRCCommandHandler містить шаблонний метод handleCommand, який задає порядок виконання дій: парсинг команди, її виконання та логування.

Підклас JoinCommandHandler, реалізовує специфічні кроки алгоритму (parseCommand, executeCommand та logCommand), залишаючи незмінною загальну послідовність виконання. Це забезпечує гнучкість у реалізації різних типів команд і дозволяє дотримуватися єдиної структури обробки команд, спрощуючи підтримку та розширення програми.

```java
package org.example.Template;

public abstract class IRCCommandHandler {  3 usages  1 inheritor
    public final void handleCommand(String command) {  1 usage
        parseCommand(command);
        executeCommand();
        logCommand(command);
    }

    protected abstract void parseCommand(String command);  1 usage  1 implementation
    protected abstract void executeCommand();  1 usage  1 implementation
    protected abstract void logCommand(String command);  1 usage  1 implementation
}
```

Рис. 1 – Код класу IRCCommandHandler

```java
package org.example.Template;

import org.example.AbstractFactory.JoinCommand;

import java.util.logging.Logger;

public class JoinCommandHandler extends IRCCommandHandler {  3 usages
    private static final Logger logger = Logger.getLogger(JoinCommandHandler.class.getName());  1 usage
    private String channel;  3 usages

    @Override  1 usage
    protected void parseCommand(String command) {
        channel = command.split( regex: " ")[1];
    }

    @Override  1 usage
    protected void executeCommand() {
        JoinCommand joinCommand = new JoinCommand(channel,  message: "Joining channel " + channel);
        joinCommand.comply();
    }
    @Override  1 usage
    protected void logCommand(String command) {
        logger.info( msg: "Command executed: " + command);
    }
}
```

Рис. 2 – Код класу JoinCommandHandler

У даному випадку цей шаблон дозволяє централізовано керувати обробкою IRC-команд, зберігаючи загальну послідовність дій, таку як парсинг, виконання і логування команд, але даючи можливість підкласам реалізувати конкретні деталі для різних типів команд, наприклад, для JoinCommandHandler. За допомогою цього підходу ми отримуємо наступні переваги:

1. Спрощення підтримки та розширення: Логіка виконання команд розділена на загальну частину (шаблонний метод) і специфічні частини

(реалізація в підкласах). Це дозволяє додавати нові типи команд без зміни існуючого коду, що спрощує підтримку.

2. Зменшення дублювання коду: Загальна структура алгоритму виноситься в базовий клас, що дозволяє уникнути дублювання однакових дій у кожному підкласі.
3. Гнучкість та розширюваність: Підкласи можуть змінювати лише окремі частини алгоритму, при цьому базова структура залишатиметься незмінною. Це дозволяє легко змінювати або додавати нові кроки без порушення загальної логіки.
4. Контроль за виконанням алгоритму: Оскільки базовий клас визначає порядок виконання дій, він надає контроль над виконанням алгоритму, гарантуючи правильний порядок кроків.

Це дозволяє централізовано керувати логікою, знижуючи дублювання коду та спрощуючи підтримку. Шаблон забезпечує гнучкість при додаванні нових варіантів поведінки без зміни загальної логіки, що робить його ефективним для розширюваних систем.

Крок 3. Зображення структури шаблону

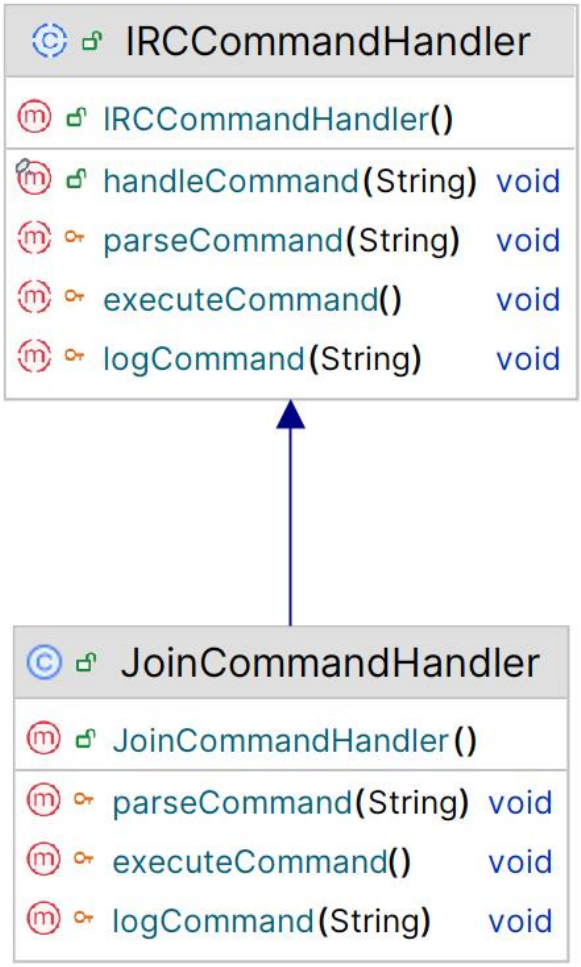

Рис. 3 – Структура шаблону Template Method

Структура шаблону Template Method включає абстрактний клас, який визначає загальну структуру алгоритму у вигляді шаблонного методу. Цей метод є публічним і визначає послідовність кроків, що складають алгоритм обробки команди. Кроки, які мають змінювану частину, визначені як абстрактні методи, що дозволяє підкласам реалізувати їх відповідно до специфіки команди. Конкретні підкласи успадковують абстрактний клас і надають реалізацію для кожного абстрактного методу, додаючи при цьому свою логіку.

В абстрактному класі також можуть бути реалізовані загальні методи, які використовуються в усіх підкласах. Це забезпечує повторне використання коду та централізоване управління частинами алгоритму, які не змінюються. Завдяки такій структурі, базовий клас контролює порядок виконання, а підкласи реалізують деталі. Всі підкласи слідують єдиній схемі обробки, що зменшує складність при розширенні програми.

Структура шаблону дозволяє чітко розмежувати логіку алгоритму і специфіку його виконання, що допомагає спрощувати модифікацію та підтримку коду. Загальні частини алгоритму, які часто використовуються, виносяться в абстрактний клас, що знижує дублювання коду, а специфічні реалізації відокремлюються у підкласах, що забезпечує гнучкість. Це робить програму більш модульною, полегшуючи додавання нових типів команд або зміну існуючих без порушення загальної структури.

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

Висновок: У ході виконання даної лабораторної роботи було досліджено структуру, призначення, переваги та недоліки шаблонів проєктування «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD». Дані патерни мають свої особливі переваги та недоліки, а також використовуються для досягнення різних цілей. Для реалізації частини майбутньої системи було обрано шаблон Template Method, який найкраще підходить для реалізації частини майбутньої системи у даному випадку. Далі було реалізовано обраний патерн, розглянуто його призначення, структуру та досліджено його переваги у використанні.