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 №5

Технології розроблення програмного забезпечення

«ШАБЛОНИ «ADAPTER», «BUILDER», «COMMAND», «CHAIN OF RESPONSIBILITY», «PROTOTYPE»»

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 функціоналу робочої програми у вигляді класів та їх взаємодій для досягнення конкретних функціональних можливостей.
3. Застосування одного з розглянутих шаблонів при реалізації програми.

Зміст

Крок 1. Теоретичні відомості 2

Крок 2. Реалізація шаблону проєктування для майбутньої системи 3

Крок 3. Зображення структури шаблону 7

Хід роботи:

Крок 1. Теоретичні відомості

Патерни проєктування – це важлива складова сучасної розробки програмного забезпечення, яка надає готові рішення для типових завдань у програмуванні. Використання шаблонів знижує складність систем, сприяє їх масштабованості, запобігає дублюванню коду та робить його більш зрозумілим і легким у підтримці.

З іншого боку, анти-патерни, відомі також як "пастки", представляють собою приклади поширених, але неефективних підходів, які можуть працювати у певних ситуаціях, але загалом призводять до зниження продуктивності та якості коду. Вивчення анти-патернів є важливим для професійного розвитку, оскільки допомагає розробникам уникати типових помилок та виявляти слабкі місця у системах.

Цей термін виник у контексті інформатики, як протилежність до "шаблонів проєктування". Вивчення та застосування шаблонів у програмуванні є частиною хорошої практики, а уникнення анти-патернів допомагає розробникам створювати більш якісне програмне забезпечення.

ADAPTER

Шаблон Adapter застосовується для узгодження несумісних класів, створюючи проміжний клас, який перетворює інтерфейс одного класу в той, що зрозумілий

іншому. Це дає змогу взаємодіяти об’єктам, які інакше не могли б працювати разом, без змін у їхньому початковому коді.

BUILDER

Шаблон Builder використовується для покрокового створення складних об'єктів, ізолюючи процес їх побудови від кінцевого представлення. Це підвищує гнучкість коду та спрощує його підтримку, особливо коли об’єкт має багато параметрів або різні конфігурації.

COMMAND

Command перетворює звичайний виклик методу в клас. Це дозволяє передавати дії як параметри, створюючи черги команд, а також підтримувати скасування та повторне виконання операцій. Основна перевага цього підходу в тому, що він зменшує залежність між відправником і виконавцем дії.

CHAIN OF RESPONSIBILITY

Шаблон Chain of Responsibility організовує обробку запитів за допомогою послідовності обробників. Кожен обробник вирішує, чи обробляти запит, чи передає його наступному в ланцюгу. Це забезпечує гнучкість у зміні послідовності обробки запитів та допомагає реалізувати логіку більш динамічно.

PROTOTYPE

Prototype використовується для створення нових об'єктів шляхом копіювання вже існуючих. Використовується, коли на процес створення об'єкта витрачається багато ресурсів, а копіювання допомагає значно скоротити витрати. Шаблон Prototype дозволяє зберігати стан об'єкта, це актуально для складних конфігурацій.

Крок 2. Реалізація шаблону проєктування для майбутньої системи

Для реалізації програми IRClient використано шаблон проєктування Builder, оскільки він спрощує створення об’єкта IRCCOnnection з багатьма параметрами. За допомогою IRCConnectionBuilder можна поетапно налаштувати необхідні параметри з'єднання, такі як server і port, перед створенням об'єкта IRCConnection. Це знижує ймовірність помилок у налаштуванні та робить код гнучким для змін.

Builder дозволяє відокремити процес налаштування від створення об'єкта. Завдяки цьому при створенні з'єднання можна уникнути перевантажених

конструкторів і змінювати параметри без необхідності вносити зміни в основний клас IRCConnection.

```java
package org.example.Builder;

import org.example.IRCConnection;

public class IRCConnectionBuilder {  5 usages
    private String server;  2 usages
    private int port;  2 usages

    public IRCConnectionBuilder setServer(String server) {  1 usage
        this.server = server;
        return this;
    }

    public IRCConnectionBuilder setPort(int port) {  1 usage
        this.port = port;
        return this;
    }

    public IRCConnection build() {  1 usage
        IRCConnection connection = new IRCConnection();
        connection.connectToServer(server, port);
        return connection;
    }
}
```

Рис. 1 – Код класу IRCConnectionBuilder

```java
package org.example;

import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;

public class IRCConnection {  14 usages
    private static final Logger logger = Logger.getLogger(IRCConnection.class.getName());  1 usage
    private Socket socket;  5 usages
    private PrintWriter writer;  2 usages
    private String currentChannel;  4 usages

    public void connectToServer(String server, int port) {  1 usage
        try {
            socket = new Socket(server, port);
            writer = new PrintWriter(socket.getOutputStream(),  autoFlush: true);
            System.out.println("Connected to " + server + " on port " + port);
        } catch (IOException e) {
            logger.log(Level.SEVERE,  msg: "Unable to connect to " + server + " on port " + port, e);
        }
    }

    public void joinChannel(String channel) {  1 usage
        if (isConnected()) {
            sendCommand("JOIN " + channel);
            currentChannel = channel;
            System.out.println("Joined channel: " + channel);
        } else {
            System.err.println("Not connected to any server.");
        }
    }

    public void sendMessage(String message) {  1 usage
        if (isConnected() && currentChannel != null) {
            sendCommand("PRIVMSG " + currentChannel + " :" + message);
            System.out.println("Sent message to " + currentChannel + ": " + message);
        } else {
            System.err.println("No channel joined or not connected to any server.");
        }
    }

    public void sendMetadata(String metadata) {  1 usage
        if (isConnected()) {
            sendCommand("METADATA " + metadata);
            System.out.println("Metadata: " + metadata);
        } else {
            System.err.println("Not connected to any server.");
        }
    }

    private void sendCommand(String command) {  3 usages
        if (isConnected()) {
            writer.println(command);
            System.out.println("Sent command: " + command);
        } else {
            System.err.println("Not connected to any server.");
        }
    }

    private boolean isConnected() { return socket != null && socket.isConnected() && !socket.isClosed(); }
}
```

Рис. 2 – Код класу IRCConnection

Клас IRCConnectionBuilder та IRCConnection реалізовують шаблон Builder, що є важливим для нашого випадку, оскільки його використання забезпечує:

1. Поетапне налаштування: Завдяки використанню шаблону Builder, налаштування параметрів з'єднання (сервер та порт) стає зручним і зрозумілим процесом. Кожен параметр можна змінювати окремо, що дозволяє створювати гнучкі конфігурації з'єднання.
2. Чистота коду: Builder дозволяє уникнути надмірної кількості параметрів у конструкторах класу IRCConnection, що значно спрощує його використання та підвищує читабельність коду. Це особливо важливо, коли кількість параметрів зростає.
3. Ізоляція логіки створення об'єкта: Використання шаблону Builder дозволяє ізолювати логіку створення об'єкта від основного класу, що знижує залежність між компонентами програми. В результаті, це полегшує зміну і підтримку коду в майбутньому.

У нашому випадку шаблон Builder забезпечує зручне і гнучке створення об'єкта IRCConnection, дозволяючи ефективно управляти налаштуванням з'єднання без зайвих складнощів. Це значно підвищує масштабованість і зручність роботи з кодом.

Патерн Builder в реалізації класів IRCConnectionBuilder та IRCConnection реалізується через створення класу IRCConnectionBuilder, який дозволяє поетапно налаштовувати параметри з'єднання. Методи setServer() та setPort() надають можливість встановлювати ці параметри окремо, при цьому кожен метод повертає сам об'єкт IRCConnectionBuilder, що дозволяє викликати їх у ланцюжку. Після налаштування параметрів метод build() створює об'єкт IRCConnection і ініціалізує його з переданими значеннями. Така структура дозволяє ізолювати логіку створення об'єкта від основного класу.

Крок 3. Зображення структури шаблону

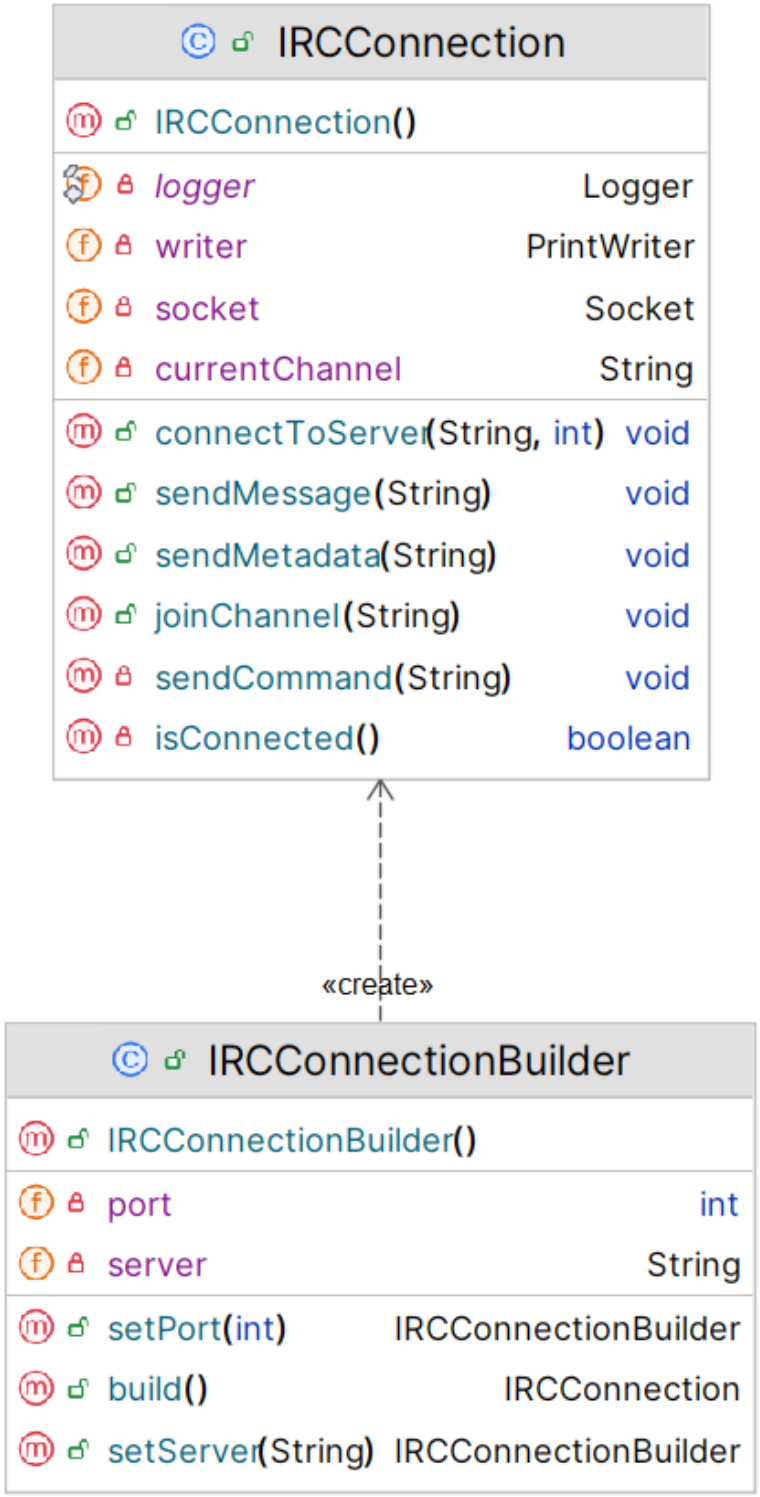

Рис. 3 – Структура шаблону Builder

Шаблон Builder в реалізації класів IRCConnectionBuilder та IRCConnection відокремлює процес створення об'єкта IRCConnection від його представлення, що є важливим для зручності налаштування та створення об'єкта з різними параметрами. Це дозволяє гнучко налаштовувати підключення до сервера, змінюючи лише окремі параметри без необхідності створювати складні конструкторами чи модифікувати сам клас IRCConnection.

Однак, у певних випадках, використання Builder може призвести до надмірної складності, якщо об'єкти, які створюються, є дуже простими або мають лише кілька параметрів. У таких ситуаціях використання шаблону Builder може виглядати зайвим, оскільки клас може бути створений без потреби у поетапному налаштуванні. Однак, перевагами цього шаблону є те, що він дозволяє створювати об'єкти з різними параметрами за допомогою одного і того ж коду без необхідності в складних конструкторах, полегшує розширення класів та забезпечує більш чистий і зрозумілий код, що робить його чудовим вибором для складних об'єктів з багатьма параметрами.

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

Висновок: У ході виконання даної лабораторної роботи було розглянуто структуру, призначення, переваги та недоліки шаблонів проєктування «ADAPTER», «BUILDER», «COMMAND», «CHAIN OF RESPONSIBILITY», «PROTOTYPE». Розглянуті патерни програмування мають свої переваги та недоліки, а також використовуються для досягнення різних цілей. Для реалізації частини майбутньої системи було обрано патерн, який найкраще підійде у даному випадку. Далі було реалізовано обраний шаблон, розглянуто його структуру та досліджено його переваги для використання для майбутньої системи.