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 OFRESPONSIBILITY», «PROTOTYPE»»

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, підтримкабазових 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 їхвзаємодій 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 та анти-патерни в програмуванні

Патерни проєктування – ключовий елемент сучасної розробки програмного забезпечення, що пропонує перевірені рішення для стандартних задач. Їх застосування допомагає зменшити складність систем, сприяє їх масштабуванню, запобігає дублюванню коду та робить його більш зрозумілим і зручним для підтримки.

Анти-патерни, або так звані “пастки”, – це приклади популярних, але недоцільних підходів. Хоча вони можуть працювати у певних умовах, у більшості випадків такі рішення призводять до зниження якості коду і продуктивності. Вивчення анти-патернів допомагає розробникам уникати поширених помилок і вчасно помічати слабкі місця в системі.

Ці поняття виникли в контексті інформатики як антоніми: патерни – це зразки гарної практики, а анти-патерни – приклади того, чого слід уникати. Використання правильних шаблонів та уникнення “пасток” сприяє створенню якісного програмного забезпечення.

Основні шаблони проєктування

Adapter

Шаблон Adapter дозволяє узгоджувати несумісні класи шляхом створення проміжного класу, що адаптує інтерфейс одного класу до очікуваного формату іншого. Це забезпечує сумісність між об’єктами без потреби змінювати їхній початковий код.

Builder
Builder використовується для поетапного створення складних об’єктів, ізолюючи процес побудови від остаточного вигляду об’єкта. Це спрощує підтримку та надає коду більшу гнучкість, особливо якщо об’єкт має багато параметрів або кілька варіантів конфігурації.

Command
Command перетворює виклики методів на об’єкти-класи. Це дозволяє передавати дії як параметри, формувати черги команд, а також додавати функціонал скасування чи повторного виконання операцій. Основна перевага шаблону – зменшення залежності між ініціатором і виконавцем дій.

Chain of Responsibility
Цей шаблон реалізує обробку запитів через послідовність обробників. Кожен обробник вирішує, чи обробляти запит, або передає його далі по ланцюгу. Такий підхід забезпечує гнучкість у налаштуванні обробки запитів та дозволяє динамічно змінювати логіку системи.

Prototype
Prototype дає змогу створювати нові об’єкти шляхом копіювання вже існуючих. Це доцільно у випадках, коли створення об’єкта вимагає значних ресурсів, а копіювання зменшує витрати. Крім того, цей шаблон зберігає стан об’єкта, що є важливим для складних конфігурацій.

Крок 2. Реалізація шаблону проєктування для майбутньої системи

Для реалізації програми IRClient було використано шаблон проєктування Builder, який значно спрощує процес створення об’єкта IRCConnection із численними параметрами. Використання IRCConnectionBuilder дозволяє поступово налаштовувати необхідні параметри з’єднання, такі як server і port, до моменту створення об’єкта. Це знижує ризик помилок у конфігурації та забезпечує гнучкість коду для внесення змін.

Шаблон Builder розділяє процес конфігурування і створення об’єкта, що дає змогу уникнути перевантаження конструкторів. Такий підхід дозволяє легко змінювати налаштування з’єднання без необхідності втручання в код основного класу 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

Класи IRCConnectionBuilder та IRCConnection реалізують шаблон проєктування Builder, що має ключове значення у цьому випадку завдяки кільком важливим перевагам:

- Поетапне налаштування: Шаблон Builder забезпечує зручний і послідовний процес конфігурування параметрів з'єднання, таких як сервер і порт. Кожен параметр можна задавати окремо, що дозволяє створювати гнучкі та індивідуальні конфігурації.
- Чистота коду: Використання Builder дозволяє уникнути перевантажених конструкторів у класі IRCConnection, що спрощує його використання і підвищує читабельність коду, особливо коли кількість параметрів збільшується.
- Ізоляція логіки створення: Шаблон відокремлює логіку створення об'єкта від основного класу, зменшуючи взаємозалежність між компонентами програми. Це полегшує внесення змін і спрощує підтримку коду.

У нашому сценарії Builder забезпечує ефективне та гнучке створення об'єкта IRCConnection, дозволяючи легко налаштовувати параметри з'єднання без зайвих ускладнень. Такий підхід підвищує масштабованість програми та робить роботу з кодом більш зручною.

Реалізація шаблону Builder

Реалізація патерну в класах IRCConnectionBuilder та IRCConnection побудована таким чином:

- Клас IRCConnectionBuilder дозволяє налаштовувати параметри з'єднання покроково через методи setServer() та setPort(). Кожен із цих методів повертає сам об'єкт IRCConnectionBuilder, що дозволяє використовувати їх у вигляді ланцюжка викликів.
- Після конфігурації метод build() створює екземпляр IRCConnection і передає йому задані параметри.

Такий підхід забезпечує ізоляцію логіки створення об'єкта від основного класу, що спрощує внесення змін і робить код більш структурованим і зручним для підтримки.

Крок 3. Зображення структури шаблону

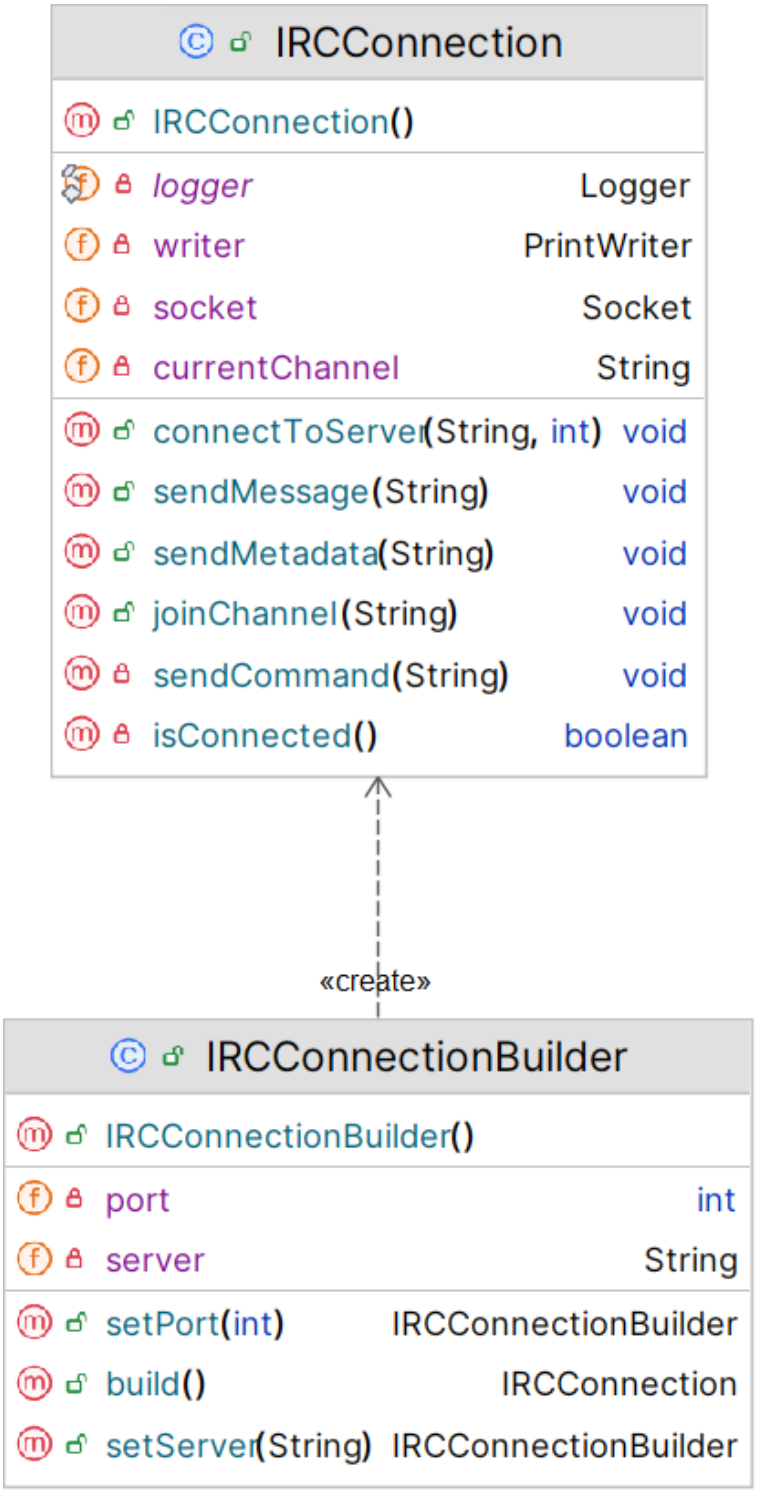

Рис. 3 – Структура шаблону Builder

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 **репозиторії**

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.