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 №4

Технології розроблення програмного забезпечення

«ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»»

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

Зміст

Хід роботи:

Крок 1. Теоретичні відомості

Шаблон проєктування — це формалізований опис типового рішення, що часто використовується при проєктуванні інформаційних систем, має загальновживану назву та рекомендації для застосування в різних ситуаціях. Його основні переваги включають полегшення створення структури системи, виділення значимих елементів та зв’язків, що робить модель більш зрозумілою та простою для вивчення. Шаблони підвищують стійкість системи до змін, спрощують її розширення та вдосконалення. Вони представляють собою ескізи архітектурних рішень, які зручно застосовувати у відповідних обставинах.

Розглянемо шаблони, надані для лабораторної роботи

1. **SINGLETON (Одинак)**

Мета:
Забезпечити існування лише одного екземпляра класу в програмі та надати глобальну точку доступу до нього.

Використання:
Застосовується, коли потрібно контролювати доступ до ресурсу, наприклад, до логування, конфігурації або підключення до бази даних.

2. ITERATOR (Ітератор)
Мета:
Надати послідовний доступ до елементів колекції без розкриття її внутрішньої структури.

Використання:
Використовується для проходження складних структур даних (масивів, списків, дерев), незалежно від їх реалізації.

3. PROXY (Замісник)
Мета:
Надати об’єкт-замісник, який контролює доступ до іншого об’єкта, додаючи додаткову поведінку або функціональність.

Використання:
Застосовується для оптимізації роботи, наприклад, у разі відкладеного завантаження, захисту доступу чи кешування.

4. STATE (Стан)
Мета:
Дозволити об’єкту змінювати свою поведінку залежно від внутрішнього стану, наче він змінює свій клас.

Використання:
Застосовується, коли об’єкт має кілька станів, і його поведінка залежить від поточного стану (наприклад, автомат продажу чи банкомат).

5. STRATEGY (Стратегія)
Мета:
Дозволити вибір алгоритму з кількох можливих варіантів під час виконання програми, ізоляція алгоритмів у окремі класи.

Використання:
Застосовується, коли є кілька способів виконання завдання (наприклад, різні методи сортування чи обчислення).

Крок 2. Реалізація шаблону проєктування для майбутньої системи

```
1   package org.example.Singleton;
2
3   import org.example.IRCConnection;
4
5   public class IRCClient {  12 usages
6       private static IRCClient specimen;  3 usages
7       private final IRCConnection connection;  2 usages
8
9       private IRCClient() { connection = new IRCConnection(); }
12
13      public static synchronized IRCClient getSpecimen() {  3 usages
14          if (specimen == null) {
15              specimen = new IRCClient();
16          }
17          return specimen;
18      }
19
20      public IRCConnection getConnection() { return connection; }
23  }
```

Рис. 1 – Код класу IRCClient

У поданому коді використовується **шаблон Singleton**, який реалізовано в класі IRCClient. Ось як і для чого він використовується:

Де використовується шаблон Singleton?

1. **Статичне поле** instance**:**

Це поле зберігає єдиний екземпляр класу IRCClient. Завдяки цьому в програмі існує лише один об'єкт цього класу.

2. **Приватний конструктор** IRCClient()**:**
   Використання приватного конструктора унеможливлює створення екземплярів класу за межами самого класу. Таким чином, обмежується створення об'єктів IRCClient.

3. **Метод** getInstance()**:**
   Цей статичний метод надає доступ до єдиного екземпляра класу. Якщо екземпляр ще не створено, метод створює його, а якщо вже існує, повертає наявний.

4. **Ключове слово** synchronized**:**
   Додає потокобезпеку до методу getInstance(), що гарантує, що у багатопотоковому середовищі не буде створено кілька екземплярів класу.

Для чого використовується Singleton у цьому коді?

1. **Єдине з'єднання із сервером:**
   Завдяки шаблону Singleton у програмі підтримується лише одне підключення до IRC-сервера через об'єкт IRCConnection. Це економить ресурси й запобігає дублюванню підключень.

2. **Контрольований доступ:**
   Використовуючи метод getInstance(), будь-яка частина програми може отримати доступ до єдиного екземпляра IRCClient, що забезпечує централізоване управління з'єднанням.

3. **Потокобезпечність:**
   У багатопотоковому середовищі гарантується, що завжди існує лише один екземпляр класу IRCClient, навіть якщо до нього одночасно звертаються кілька потоків.

Таким чином, шаблон Singleton забезпечує стабільну й ефективну роботу класу IRCClient, особливо в контексті підтримки з'єднання з сервером.

Крок 3. Зображення структури шаблону

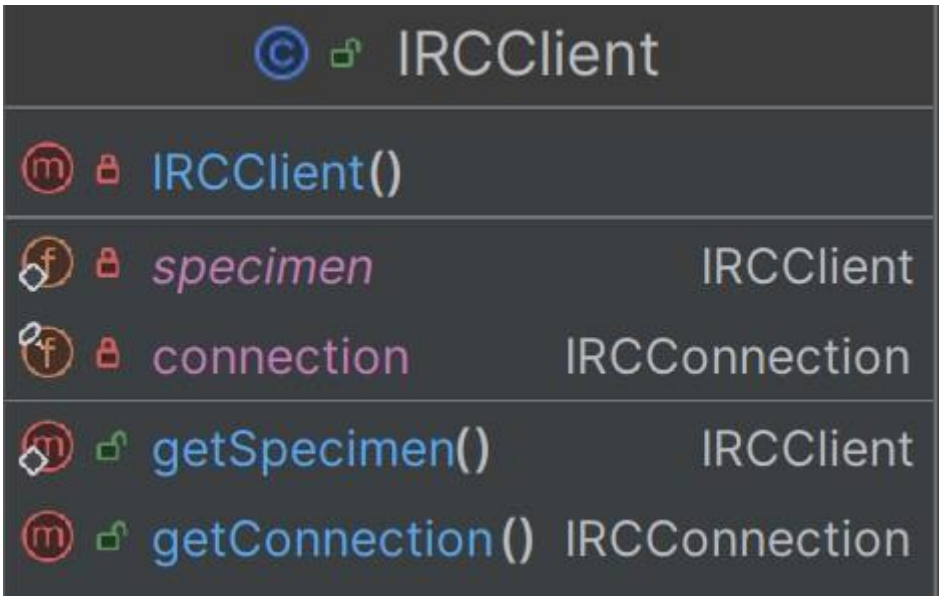

Рис. 2 – Структура шаблону

Висновок: У процесі виконання цієї лабораторної роботи було проаналізовано структуру та призначення шаблонів проєктування: «SINGLETON», «ITERATOR», «PROXY», «STATE» та «STRATEGY». Кожен із цих шаблонів має свої сильні та слабкі сторони. Після аналізу було обрано шаблон, який найкраще відповідає поставленим завданням.

для реалізації майбутньої системи. Останнім етапом була реалізація шаблону та дослідження його структури, переваг у використанні для заданої системи.