



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

з дисципліни «Технології розроблення

програмного забезпечення»

Тема: «Вступ до патернів проєктування»

Виконала
студентка групи IA-32:
Павлюк В. О.

Зміст

Зміст.....	2
Вступ.....	3
1 Теоретичні відомості.....	4
1.1. Поняття шаблону проєктування.....	4
1.2. Шаблон «Singleton».....	4
1.3. Шаблон «Iterator».....	5
.....	5
.....	5
1.4. Шаблон «Proxy».....	6
.....	6
1.5. Шаблон «State».....	6
2 Хід роботи.....	8
2.1. Використання патерну «Proxy» у Web crawler.....	8
2.2 Діаграма класів (фрагмент).....	8
2.3. Фрагмент коду (Java).....	9
Висновки.....	11
Контрольні запитання.....	12

Вступ

Темою роботи є «Вступ до патернів проєктування». Мета цієї лабораторної це Вивчити структуру шаблонів «Singleton», «Iterator», «Proxy», «State», «Strategy» та навчитися застосовувати їх в реалізації програмної системи.

Як об'єкт проєктування обрано тему «Web crawler»: програмний сканер, що розпізнає структуру сторінок сайту, переходить за посиланнями, збирає інформацію за заданим терміном, видаляє несемантичні елементи (рекламу, скрипти тощо), зберігає очищені дані у вигляді структурованого набору HTML-файлів і веде статистику відвідань та метадані.

1 Теоретичні відомості

1.1. Поняття шаблону проєктування

Будь-який патерн проєктування, використовуваний при розробці інформаційних систем, являє собою формалізований опис, який часто зустрічається в завданнях проєктування, вдале рішення даної задачі, а також рекомендації по застосуванню цього рішення в різних ситуаціях [5]. Крім того, патерн проєктування обов'язково має загальновживане найменування. Правильно сформульований патерн проєктування дозволяє, відшукавши одного разу вдале рішення, користуватися ним знову і знову. Варто підкреслити, що важливим початковим етапом при роботі з патернами є адекватне моделювання розглянутої предметної області. Це є необхідним як для отримання належним чином формалізованої постановки задачі, так і для вибору відповідних патернів проєктування.

Відповідне використання патернів проєктування дає розробнику ряд незаперечних переваг. Наведемо деякі з них. Модель системи, побудована в межах патернів проєктування, фактично є структурованим виокремленням тих елементів і зв'язків, які значимі при вирішенні поставленого завдання. Крім цього, модель, побудована з використанням патернів проєктування, більш проста і наочна у вивченні, ніж стандартна модель. Проте, не дивлячись на простоту і наочність, вона дозволяє глибоко і всебічно опрацювати архітектуру розроблюваної системи з використанням спеціальної мови. Застосування патернів проєктування підвищує стійкість системи до зміни вимог та спрощує неминуче подальше доопрацювання системи. Крім того, важко переоцінити роль використання патернів при інтеграції інформаційних систем організації. Також слід зазначити, що сукупність патернів проєктування, по суті, являє собою єдиний словник проєктування, який, будучи уніфікованим засобом, незамінний для спілкування розробників один одним.

1.2. Шаблон «Singleton»

Призначення патерну: «Singleton» (Одинарка) являє собою клас в термінах ООП, який може мати не більше одного об'єкта (звідси і назва «одинарка») [6]. Насправді, кількість об'єктів можна задати (тобто не можна створити більш ніж один об'єктів даного класу). Даний об'єкт найчастіше зберігається як статичне поле в самому класі.

Проблема: Використання одинарка виправдано для наступних випадків:

- може бути не більше N фізичних об'єктів, що відображаються в певних класах;
- необхідно жорстко контролювати всі операції, що проходять через даний клас.

Одинарка вирішує відразу дві проблеми, порушуючи принцип єдиної відповідальності класу.

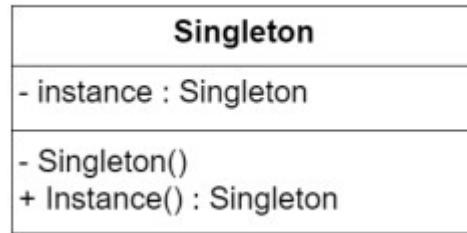


Рисунок 1.2. Структура патерну Одинар

1.3. Шаблон «Iterator»

Призначення: «Iterator» (Ітератор) являє собою шаблон реалізації об'єкта доступу до набору (колекції, агрегату) елементів без розкриття внутрішніх механізмів реалізації. Ітератор виносить функціональність перебору колекції елементів з самої колекції, таким чином досягається розподіл обов'язків: колекція відповідає за зберігання даних, ітератор – за прохід по колекції [6].

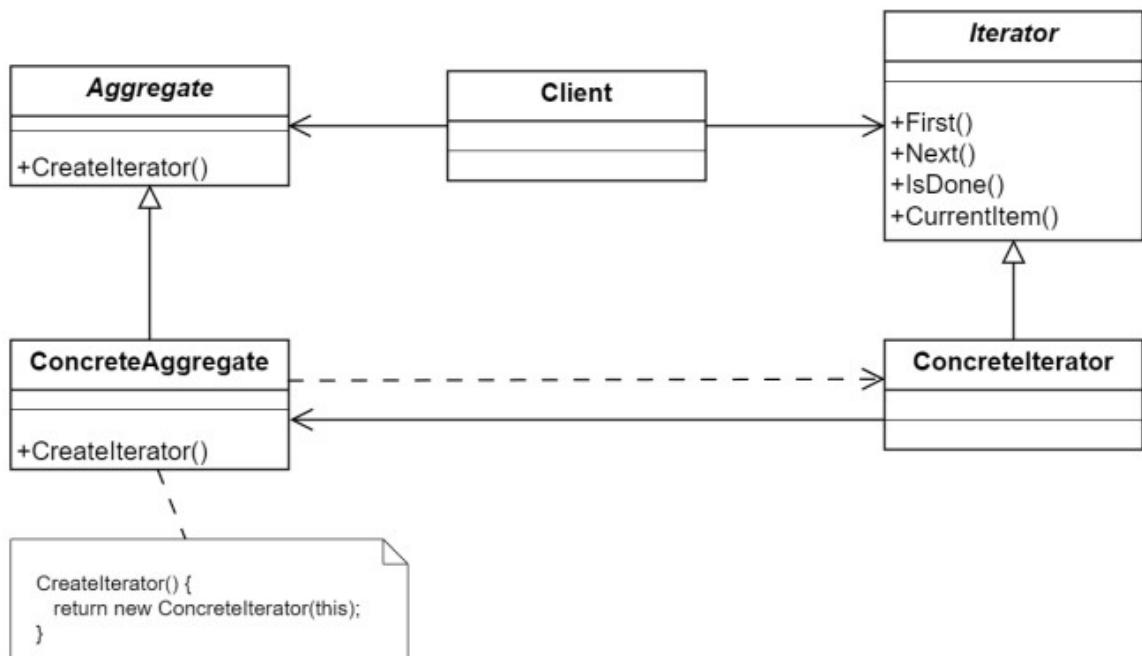


Рисунок 1.3. Структура патерну Ітератор

1.4. Шаблон «Proxy»

Призначення: «Proxy» (Проксі) – об'єкти є об'єктами-заглушками або двійниками/замінниками для об'єктів конкретного типу. Зазвичай, проксі об'єкти вносять додатковий функціонал або спрощують взаємодію з реальними об'єктами [5]. Проксі об'єкти використовувалися в більш ранніх версіях інтернетбраузерів, наприклад, для відображення картинки: поки картинка завантажується, користувачеві відображається «заглушка» картинки.

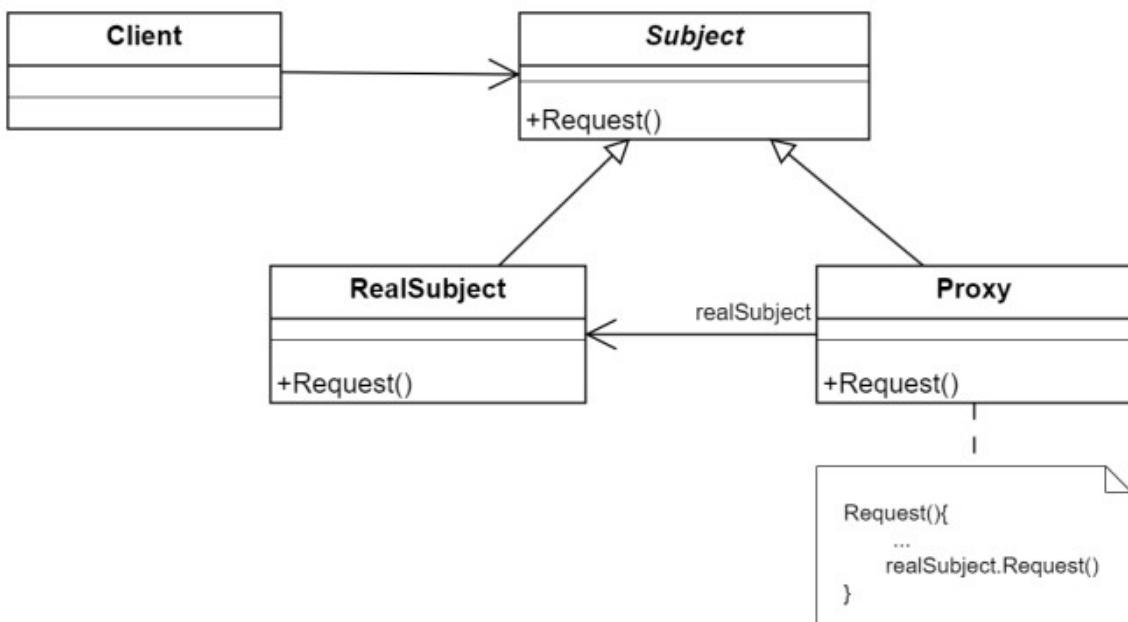


Рисунок 1.4. Структура патерну Proxy

1.5. Шаблон «State»

Призначення: Шаблон «State» (Стан) дозволяє змінювати логіку роботи об'єктів у випадку зміни їх внутрішнього стану [6]. Наприклад, відсоток нарахованих на картковий рахунок грошей залежить від стану картки: Visa Electron, Classic, Platinum і т.д. Або обсяг послуг, які надані хостинг компанією, змінюється в залежності від обраного тарифного плану (стану членства – бронзовий, срібний або золотий клієнт). Реалізація даного шаблону полягає в наступному: пов'язані зі станом поля, властивості, методи і дії виносяться в окремий загальний інтерфейс (State); кожен стан являє собою окремий клас (ConcreteStateA, ConcreteStateB), які реалізують загальний інтерфейс [6, 8].

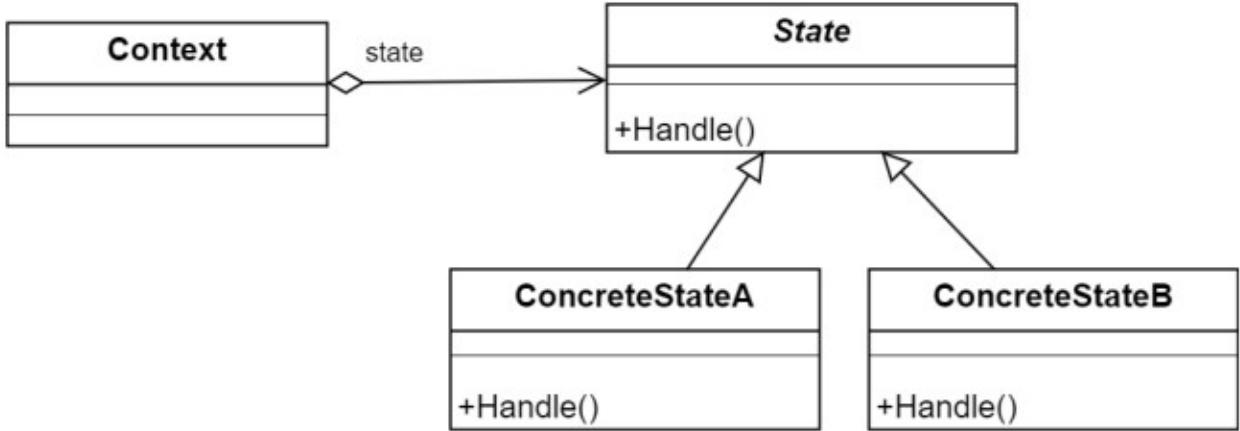


Рисунок 1.5. Структура патерну Стан

1.6. Шаблон «Strategy»

Призначення: Шаблон «Strategy» (Стратегія) дозволяє змінювати деякий алгоритм поведінки об'єкта іншим алгоритмом, що досягає ту ж мету іншим способом. Прикладом можуть служити алгоритми сортування: кожен алгоритм має власну реалізацію і визначений в окремому класі; вони можуть бути взаємозамінними в об'єкті, який їх використовує [6]. Даний шаблон дуже зручний у випадках, коли існують різні «політики» обробки даних. По суті, він дуже схожий на шаблон «State» (Стан), проте використовується в абсолютно інших цілях – незалежно від стану об'єкта відобразити різні можливі поведінки об'єкта (якими досягаються одні й ті самі або схожі цілі).

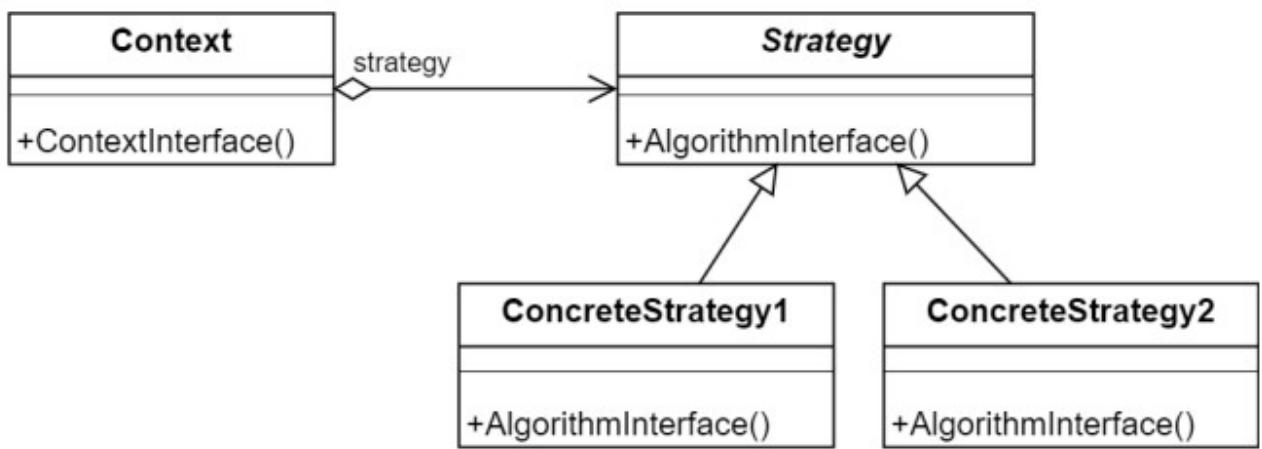


Рисунок 1.6. Структура патерну Стратегія.

2 Хід роботи

2.1. Використання патерну «Proxy» у Web crawler

Ідея

Система web crawler може працювати через зовнішній HTTP-проксі (для обходу обмежень, балансування навантаження, анонімності тощо). Замість того, щоб кожен раз відправляти запити напряму до HttpClient, застосовується проксі-об'єкт, який:

- може перемикати різні конкретні HTTP-клієнти;
- додавати логування, кешування або обмеження швидкості;
- прозоро для клієнтського коду реалізує одинаковий інтерфейс.

Основні класи

- HttpClient — інтерфейс базового HTTP-клієнта.
- RealHttpClient — конкретна реалізація HTTP-клієнта.
- HttpClientProxy — клас Proxy, який перехоплює виклики та делегує їх RealHttpClient, додаючи додаткову логіку (логування, використання проксі-серверів, обмеження запитів).
- CrawlerService — використовує HttpClient (через інтерфейс), не знаючи про наявність проксі.

2.2 Діаграма класів (фрагмент)

- HttpClient — інтерфейс: get(String url): HtmlPage.
- RealHttpClient реалізує HttpClient.
- HttpClientProxy реалізує HttpClient, має поле RealHttpClient realClient.
- CrawlerService має поле HttpClient httpClient.

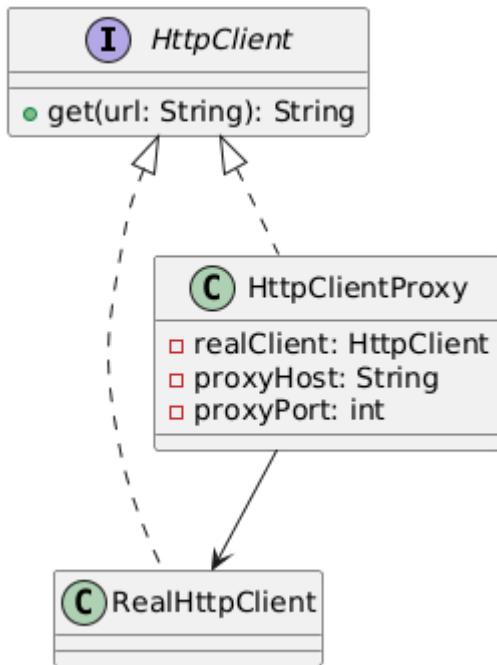


Рисунок 2.2 - Діаграма класів (фрагмент)

2.3. Фрагмент коду (Java)

```

public interface HttpClient {
    HtmlPage get(String url) throws IOException;
}

public class RealHttpClient implements HttpClient {
    @Override
    public HtmlPage get(String url) throws IOException {
        // Реальна логіка HTTP-запиту
        // Відповідь обгортається у HtmlPage
        return new HtmlPage(url, "<html>...</html>", 200, "text/html");
    }
}

public class HttpClientProxy implements HttpClient {
    private final HttpClient realClient;
    private final String proxyHost;
    private final int proxyPort;

    public HttpClientProxy(HttpClient realClient, String proxyHost, int proxyPort) {
        this.realClient = realClient;
        this.proxyHost = proxyHost;
    }
}

```

```
        this.proxyPort = proxyPort;
    }

@Override
public HtmlPage get(String url) throws IOException {
    // Додаткова логіка: логування, обмеження швидкості, використання прокси
    System.out.println("Request via proxy " + proxyHost + ":" + proxyPort + " → "
+ url);
    HtmlPage page = realClient.get(url);
    // Можливе кешування або додаткова обробка
    return page;
}

}

public class CrawlerService {
    private final HttpClient httpClient;

    public CrawlerService(HttpClient httpClient) {
        this.httpClient = httpClient;
    }

    public void crawl(String url) throws IOException {
        HtmlPage page = httpClient.get(url);
        // Подальша обробка сторінки
    }
}
```

Висновки

У процесі виконання роботи було вивчено основні патерни проєктування «Singleton», «Iterator», «Proxy», «State», «Strategy» та розглянуто можливості їх використання в системі web crawler. На практиці реалізовано патерн «Proxy» для організації доступу до веб-ресурсів через HTTP-проксі та інтегровано його в загальну структуру сканера. Це дозволяє відокремити логіку мережевої взаємодії від бізнес-логіки сканера та гнучко змінювати спосіб доступу до ресурсів.

Контрольні запитання

1. Що таке шаблон проектування?

Шаблон проектування — це типовий, перевірений часом спосіб розв'язання поширеної задачі проектування програмного забезпечення, описаний на рівні взаємодії класів і об'єктів.

2. Навіщо використовувати шаблони проектування?

Шаблони використовують, щоб спростити розробку, уникати типових помилок, підвищити гнучкість і розшируваність коду, а також полегшити спілкування між розробниками «спільною мовою» (назвами шаблонів).

3. Яке призначення шаблону «Стратегія»?

Шаблон «Стратегія» дозволяє визначити сімейство алгоритмів, інкапсулювати кожен з них у окремий клас і робити їх взаємозамінними під час виконання, не змінюючи код контексту.

4. Нарисуйте структуру шаблону «Стратегія».

1) Context — клас, який використовує стратегію.

2) Strategy — інтерфейс (або абстрактний клас) для всіх алгоритмів.

3) ConcreteStrategyA, ConcreteStrategyB, ... — конкретні реалізації різних алгоритмів.

5. Які класи входять в шаблон «Стратегія», та яка між ними взаємодія?

У шаблон входять:

- Strategy — оголошує спільний метод для алгоритму.
 - ConcreteStrategy — реалізує конкретний варіант алгоритму.
 - Context — містить посилання на Strategy і викликає її метод.
- Взаємодія: контекст делегує виконання алгоритму об'єкту стратегії, яку можна підмінити на іншу під час роботи програми.

6. Яке призначення шаблону «Стан»?

Шаблон «Стан» дозволяє об'єкту змінювати свою поведінку в залежності від внутрішнього стану, при цьому стан оформлюється як окремі класи.

7. Нарисуйте структуру шаблону «Стан».

Структура подібна до стратегії:

- Context — об'єкт, який має поточний стан.
- State — інтерфейс стану.

- ConcreteStateA, ConcreteStateB, ... — конкретні стани, які реалізують поведінку й можуть переводити контекст в інший стан.

8. Які класи входять в шаблон «Стан», та яка між ними взаємодія?

У шаблон входять:

- State — оголошув інтерфейс для дій у певному стані.
- ConcreteState — реалізує поведінку для конкретного стану та, за потреби, змінює стан контексту.
- Context — зберігає посилання на поточний State і делегує йому обробку запитів.

Взаємодія: контекст пересилає запити об'єкту стану, а той може змінювати стан контексту, підміняючи об'єкт State.

9. Яке призначення шаблону «Ітератор»?

Шаблон «Ітератор» надає уніфікований спосіб послідовного доступу до елементів колекції без розкриття її внутрішнього представлення.

10. Нарисуйте структуру шаблону «Ітератор».

1) Aggregate — інтерфейс або клас колекції, що створює ітератор.

2) ConcreteAggregate — конкретна колекція.

3) Iterator — інтерфейс ітератора (методи next(), hasNext() тощо).

4) ConcreteIterator — конкретний ітератор для відповідної колекції.

11. Які класи входять в шаблон «Ітератор», та яка між ними взаємодія?

У шаблон входять:

- Aggregate / ConcreteAggregate — колекція, яка створює ітератор.
 - Iterator / ConcreteIterator — об'єкт, що ходить по елементах колекції.
- Взаємодія: клієнт отримує від колекції ітератор і надалі працює з елементами через ітератор, не знаючи, як саме колекція зберігає дані

12. В чому полягає ідея шаблона «Одинак»?

«Одинак» гарантує, що певний клас має лише один екземпляр у системі, і надає глобальну точку доступу до цього екземпляра.

13. Чому шаблон «Одинак» вважають «анти-шаблоном»?

Його часто вважають анти-шаблоном, тому що:

- він створює приховану глобальну залежність;
- ускладнює тестування (важко підміняти екземпляр);

- може призводити до прихованіх побічних ефектів та порушення принципу єдиного обов'язку.

14. Яке призначення шаблону «Проксі»?

Шаблон «Проксі» надає сурогат або замінник іншого об'єкта, контролюючи доступ до нього (наприклад, для лінівої ініціалізації, кешування, логування, контролю доступу чи роботи по мережі).

15. Нарисуйте структуру шаблону «Проксі».

- 1)Subject — спільний інтерфейс для реального об'єкта і проксі.
- 2)RealSubject — реальний об'єкт, що виконує основну роботу.
- 3)Proxy — клас-проксі, який містить посилання на RealSubject і реалізує Subject, додаючи додаткову логіку до або після виклику.

16. Які класи входять в шаблон «Проксі», та яка між ними взаємодія?

У шаблон входять:

- Subject — інтерфейс, яким користується клієнт.
 - RealSubject — реальний об'єкт із основною функціональністю.
 - Proxy — обгортка над RealSubject, яка контролює доступ.
- Взаємодія: клієнт працює з об'єктом Proxy через інтерфейс Subject; проксі виконує додаткові дії (перевірки, кешування, логування тощо) і при необхідності делегує виклик до RealSubject.