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 розробки програмного забезпечення

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

Виконала
студентка групи ІА-24
Тильна Марія Сергіївна

Перевірив:
Мягкий М.Ю.

Київ 2024

Зміст

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

Короткі теоретичні відомості

Adapter (Адаптер) – це структурний шаблон, який дозволяє об'єктам з різними інтерфейсами працювати разом. Адаптер перетворює інтерфейс одного класу на інтерфейс, який очікує інший клас. Це дозволяє інтегрувати різні компоненти в систему, не змінюючи їх внутрішню реалізацію. Адаптери часто використовуються для сумісності з існуючими бібліотеками чи старими системами.

Builder (Будівельник) – це створювальний шаблон, який розділяє процес створення складних об'єктів на кілька етапів. Це дозволяє створювати різні варіанти об'єкта за допомогою однакових кроків, але з різними параметрами. Шаблон Builder корисний для побудови об'єктів, які мають багато складових, і дозволяє керувати їх створенням поетапно.

Command (Команда) – це поведінковий шаблон, який інкапсулює запит як об'єкт, що дозволяє параметризувати методи виклику, ставити їх у чергу або записувати для подальшого виконання. Шаблон Command дозволяє розділяти запити та їх виконання, а також легко реалізовувати операції скасування (undo) чи повторення (redo). Це дозволяє зручно працювати з виконанням команд у додатках.

Chain of Responsibility (Цепочка відповідальності) – це поведінковий шаблон, який дозволяє передавати запит по ланцюгу обробників. Кожен обробник може або обробити запит, або передати його наступному обробнику в ланцюгу. Це дає змогу динамічно змінювати порядок обробки запитів, а також зменшує зв'язність між клієнтом і обробниками.

Prototype (Прототип) – це створювальний шаблон, який дозволяє копіювати існуючі об'єкти замість їх створення з нуля. За допомогою шаблону Prototype об'єкт може створити нові екземпляри на основі себе, зменшуючи накладні витрати на створення складних об'єктів. Цей шаблон корисний, коли потрібно створювати багато подібних об'єктів, але зі змінами, які важко досягти через звичайне конструювання.

Хід роботи

..11 Web crawler (proxy, chain of responsibility, memento, template method, composite, p2p)

Веб-сканер повинен вміти розпізнавати структуру сторінок сайту, переходити за посиланнями, збирати необхідну інформацію про зазначений термін, видаляти не семантичні одиниці (рекламу, об'єкти javascript і т.д.), зберігати знайдені дані у вигляді структурованого набору html файлів вести статистику відвіданих сайтів і метадані.

Основні принципи та застосування Chain of Responsibility:

1. **Передача запитів між обробниками:** Шаблон дозволяє передавати запити від одного обробника до наступного, що дає можливість централізовано управляти обробкою запитів без прив'язки до конкретного обробника.
2. **Зниження зв'язності:** Завдяки використанню ланцюга обробників, клієнт не має прямого доступу до обробників, що знижує зв'язність між клієнтським кодом і конкретною реалізацією обробників.
3. **Гнучкість у додаванні нових обробників:** Ланцюг обробників легко розширюється новими елементами без зміни існуючої логіки. Це дозволяє легко змінювати або додавати нові правила обробки запитів.
4. **Обробка запитів за умовами:** Кожен обробник може перевіряти чи відповідний запит може бути оброблений саме ним. Якщо ні — передає запит наступному обробнику. Це дає змогу реалізовувати різні типи перевірок, наприклад, для обробки помилок або авторизації.
5. **Використання в обробці подій:** Шаблон часто використовується для обробки подій в GUI-додатках або в системах обробки запитів, де запит повинен пройти через кілька етапів, наприклад, валідацію, авторизацію та виконання.

Реалізація шаблону проєктування для майбутньої системи

Шаблон **Chain of Responsibility (Ланцюг відповідальності)** є поведінковим шаблоном проєктування, який дозволяє передавати запит по ланцюгу обробників. Кожен обробник може або обробити запит, або передати його наступному обробнику в ланцюгу. Це дозволяє динамічно змінювати порядок обробки запитів і зменшує зв'язність між клієнтом і обробниками.

Переваги Chain of Responsibility у веб-сканері:

- Модульність: кожен обробник виконує одну чітко визначену задачу.
- Гнучкість: легко додавати або видаляти обробники.
- Повторне використання: обробники можна використовувати в інших частинах застосунку.

```
6 usages   3 implementations
public interface PageProcessor {
    3 implementations
    void process(Document page) throws Exception;
}
```

Рис. 1 – Код інтерфейсу PageProcessor

Це інтерфейс, в якому метод process(Document page) обробляє передану сторінку й може передати її далі в ланцюжку.

```java
public class AvailabilityChecker implements PageProcessor {

    @Override
    public void process(Document page) throws Exception {
        if (page == null) {
            throw new Exception("Page is not accessible or does not exist.");
        }
        System.out.println("Page is available.");
    }
}
```

Рис. 2 – Код класу AvailabilityChecker

Цей клас перевіряє, чи сторінка доступна. Якщо ні — кидає виняток.

```java
public class ContentFilterProcessor implements PageProcessor {

    1 usage
    private final ContentFilter contentFilter = new ContentFilter();

    @Override
    public void process(Document page) {
        contentFilter.removeNonSemanticContent(page);
        System.out.println("Non-semantic content removed.");
    }
}
```

Рис. 3 – Код класу ContentFilterProcessor

Видаляє рекламу, скрипти, стилі та інший несемантичний контент.

```
2 usages
public class SemanticAnalyzer implements PageProcessor {

    1 usage
    private final ContentFilter contentFilter = new ContentFilter();

    @Override
    public void process(Document page) {
        boolean isSemantic = contentFilter.isSemanticContent(page);
        System.out.println("Semantic analysis completed. Is semantic: " + isSemantic);
    }
}
```

Рис. 4 – Код класу SemanticAnalyzer

Аналізує текст сторінки й визначає, чи вона семантична.

```
3 usages
public class PageProcessingChain {

    2 usages
    private final List<PageProcessor> processors = new ArrayList<>();

    3 usages
    public void addProcessor(PageProcessor processor) {
        processors.add(processor);
    }

    public void process(Document page) throws Exception {
        for (PageProcessor processor : processors) {
            processor.process(page); // Передає сторінку кожному процесору
        }
    }
}
```

Рис. 5 – Код классу PageProccesorChain

Зберігає список обробників та викликає їх по черзі.

Як це працює:

1) Створюється PageProcessingChain.

2) До ланцюжка додаються обробники:

AvailabilityChecker,

ContentFilterProcessor,

SemanticAnalyzer.

3) Ланцюжок передає сторінку кожному обробнику в послідовності.

4) Обробники виконують свої дії й передають сторінку далі.

Зображення структури шаблону

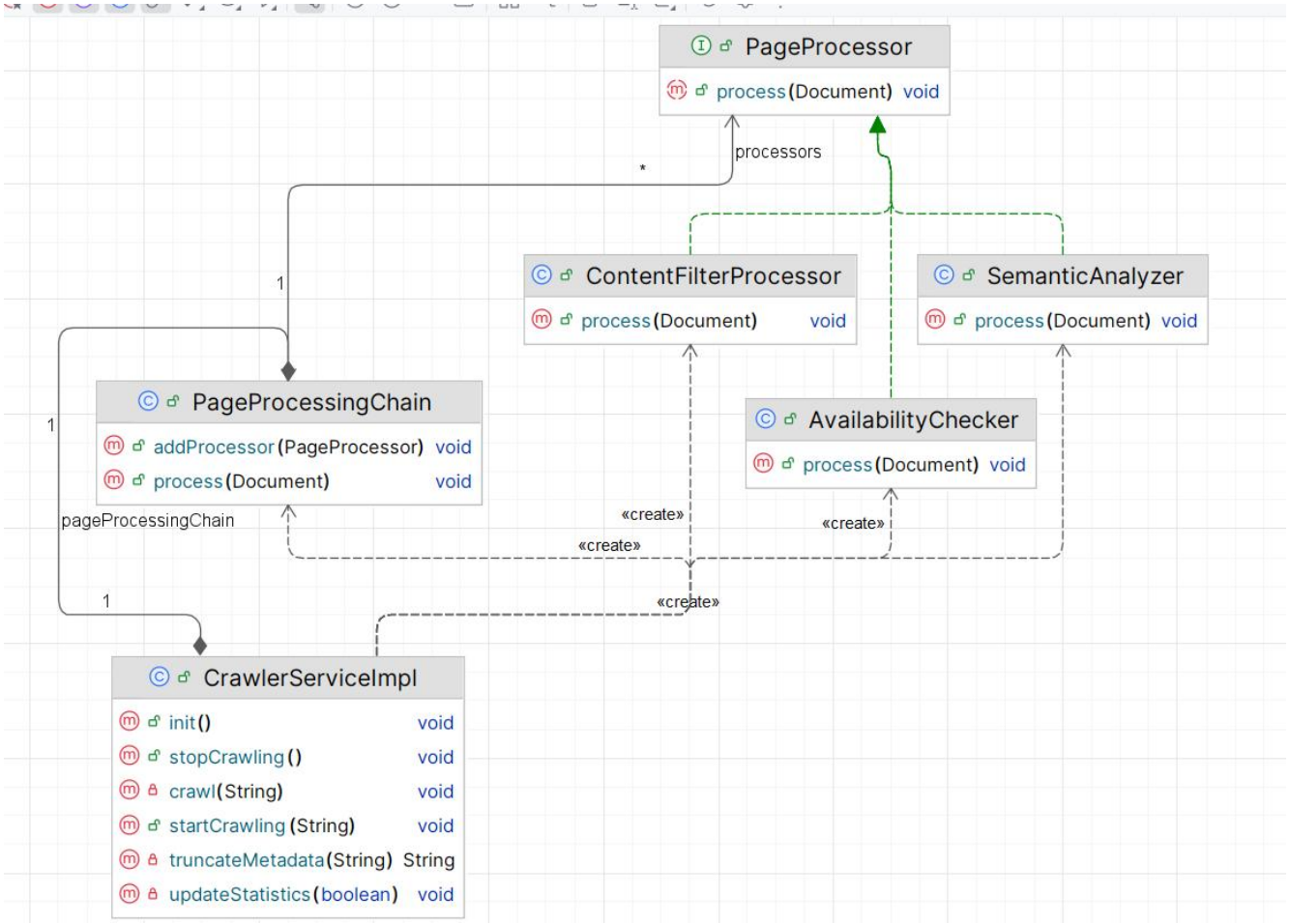

Рис. 6 – Структура шаблону

Посилання на репозиторій:

https://github.com/mashunchik/Webcrawler

Висновок: Шаблон Chain of Responsibility був успішно реалізований у проєкті для обробки та фільтрації вмісту сторінок перед їх подальшою обробкою й збереженням. Його використання дозволило спростити й структурувати логіку, а також забезпечило гнучкість і розширюваність процесу обробки.

Кожен обробник у ланцюжку відповідає за конкретний крок у процесі фільтрації та обробки сторінок (видалення несемантичного контенту, перевірка на

семантичність, перевірка доступності сторінки тощо). Це покращило читабельність і підтримку коду.

Застосування Chain of Responsibility значно покращило архітектуру проєкту, зробивши її гнучкою, масштабованою та зрозумілою. Це дозволяє легко додавати нові обробники або змінювати порядок їх виконання без модифікації основного коду.

Таким чином, використання ланцюжка обов'язків у веб-сканері відповідає кращим практикам проєктування програмного забезпечення та сприяє підвищенню якості коду.

.