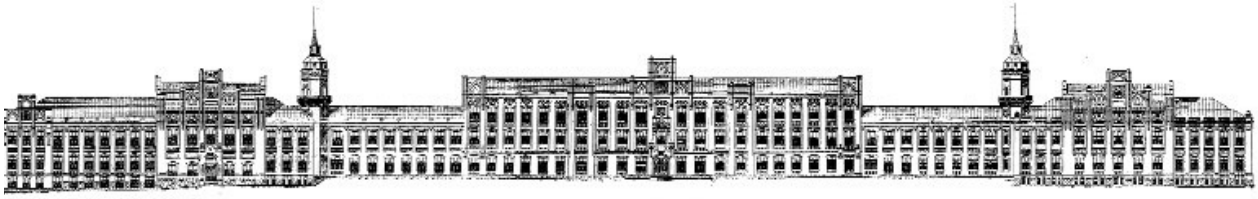

Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №5

з дисципліни «Технології розроблення

програмного забезпечення»

Тема: «Патерни проектування»

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

Зміст

Вступ

Темою роботи є «Вступ до патернів проєктування». Мета цієї лабораторної це вивчити структуру шаблонів «Adapter», «Builder», «Command», «Chain of Responsibility», «Prototype» 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. Шаблон «Adapter»

Призначення патерну: Шаблон "Adapter" (Адаптер) використовується для адаптації інтерфейсу одного об'єкту до іншого [6]. Наприклад, існує декілька бібліотек для роботи з принтерами, проте кожна має різний інтерфейс (хоча однакові можливості і призначення). Має сенс розробити уніфікований інтерфейс (сканування, асинхронне сканування, двостороннє сканування, потокове сканування і тому подібне), і реалізувати відповідні адаптери для приведення бібліотек до уніфікованого інтерфейсу. Це дозволить в програмі звертатися до загального інтерфейсу, а не приводити різні сценарії роботи залежно від способу реалізації бібліотеки. Адаптери також називаються "wrappers" (обгортками).

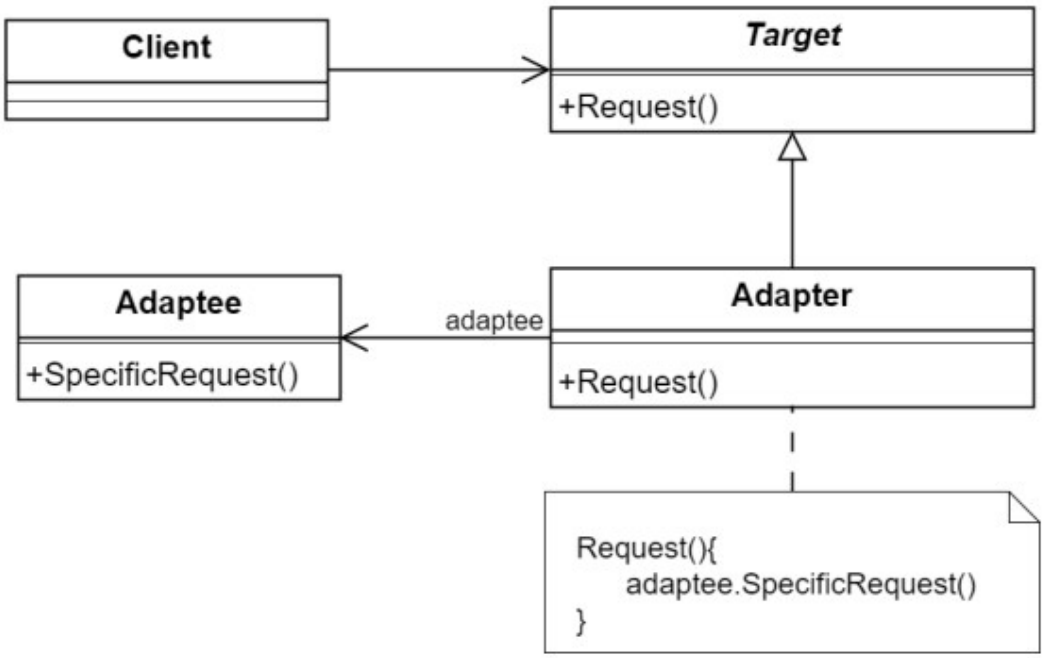

Рисунок 1.1. Структура патерну Адаптер на рівні об’єктів

1.2. Шаблон «Builder»

Призначення патерну: Шаблон «Builder» (Будівельник) використовується для відділення процесу створення об'єкту від його представлення [6]. Це доречно у випадках, коли об'єкт має складний процес створення (наприклад, Webсторінка як елемент повної відповіді web- сервера) або коли об'єкт повинен мати декілька різних форм створення (наприклад, при конвертації тексту з формату у формат).

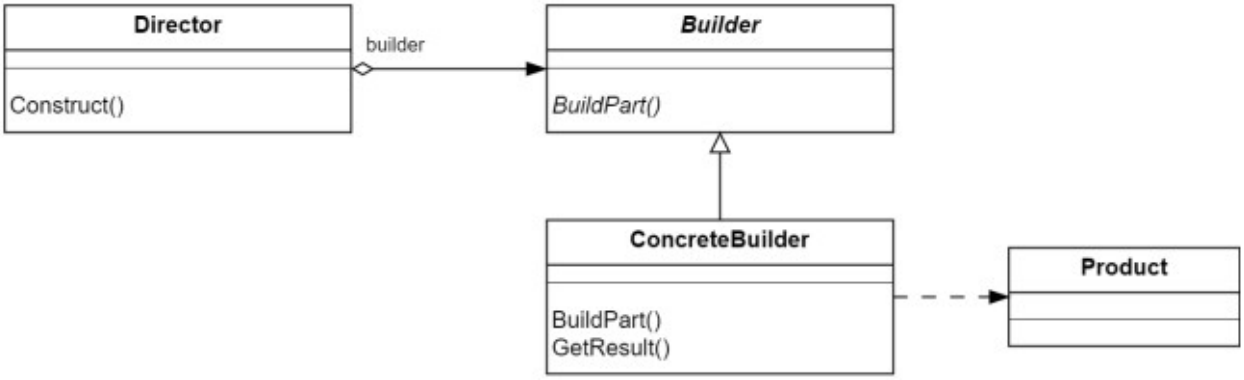

Рисунок 1.2. Структура патерну Builder

1.3. Шаблон «Command»

Призначення патерну: Шаблон "command" (команда) перетворить звичайний виклик методу в клас [6]. Таким чином дії в системі стають повноправними об'єктами.

Це зручно в наступних випадках:

• Коли потрібна розвинена система команд – відомо, що команди будуть добавлятися;

• Коли потрібна гнучка система команд – коли з'являється необхідність додавати командам можливість відміни, логування і інш.;

• Коли потрібна можливість складання ланцюжків команд або виклику команд в певний час.

Об'єкт команда сама по собі не виконує ніяких фактичних дій окрім перенаправлення запиту одержувачеві (тобто команди все ж виконуються одержувачем), однак ці об'єкти можуть зберігати дані для підтримки додаткових функцій відміни, логування і інш. Наприклад, команда вставки символу може запам'ятовувати символ, і при виклику відміни викликати відповідну функцію витирання символу. Можна також визначити параметр «застосовності» команди (наприклад, на картинці писати не можна) – і використати цей аттрибут для засвічування піктограми в меню. Такий підхід до команд дозволяє побудувати дуже гнучку систему команд, що настроюється. У більшості додатків це буде зайвим (використовується спрощений варіант), проте життєво важливий в додатках з великою кількістю команд (редактори).

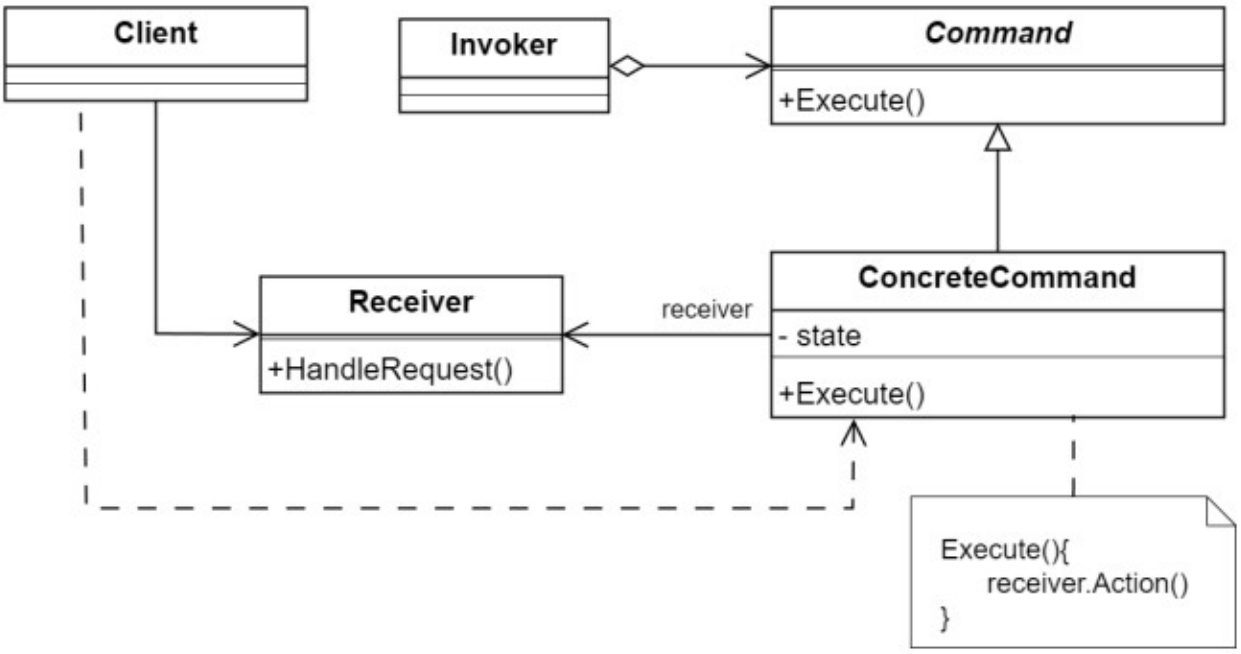

Рисунок 1.3. Структура патерну Ітератор

1.4. Шаблон «Chain of Responsibility»

Призначення патерну: Шаблон «Chain of responsibility» (Ланцюжок відповідальності) частково можна спостерігати в житті, коли підписання відповідного документу проходить від його складання у одного із співробітників компанії через менеджера і начальника до головного начальника, який ставить свій підпис [5].

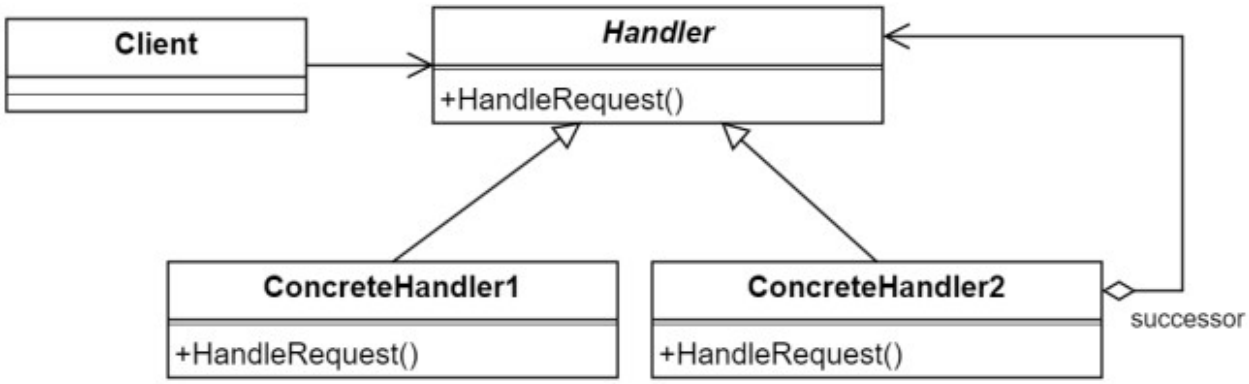

Рисунок 1.4. Структура патерна Ланцюжок відповідальності

1.5. Шаблон «Prototype»

Призначення патерну: Шаблон «Prototype» (Прототип) використовується для створення об'єктів за «шаблоном» (чи «кресленням», «ескізом») шляхом копіювання шаблонного об'єкту, який називається прототипом. Для цього визначається метод «клонувати» в об'єктах цього класу [6]. Цей шаблон зручно використати, коли заздалегідь відомо як виглядатиме кінцевий об'єкт (мінімізується кількість змін до об'єкту шляхом створення шаблону), а також для видалення необхідності створення об'єкту – створення відбувається за рахунок клонування, і самій програмі немає необхідності знати, як створювати об'єкт.

Також, це дозволяє маніпулювати об'єктами під час виконання програми шляхом настроювання відповідних прототипів; значно зменшується ієрархія наслідування (оскільки в іншому випадку це були б не прототипи, а вкладені класи, що наслідують).

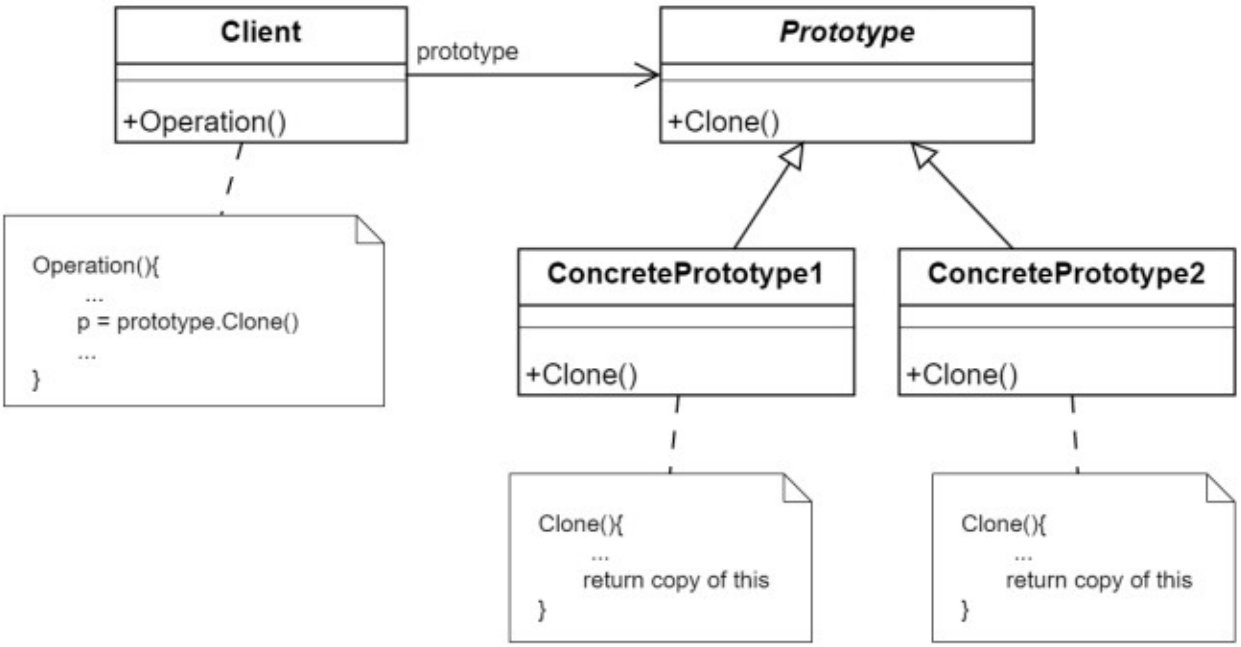

Рисунок 1.5. Структура патерну «Прототип»

2 Хід роботи

2.1. Використання «Chain of Responsibility» для обробки сторінки

Ідея

При обробці HTML-сторінки web crawler виконує кілька кроків:

1. Видалення скриптів та стилів.
2. Видалення рекламних блоків.
3. Витяг текстового контенту.
4. Пошук ключових слів.
5. Фільтрація за додатковими правилами.

Щоб не зосереджувати усю логіку в одному класі, застосовується ланцюжок обробників, де кожен відповідає за один крок.

Основні класи

- PageProcessingContext
    - Містить вхідну HTML-сторінку та проміжні результати (очищений HTML, текст, знайдені ключові слова тощо).
- PageHandler (абстрактний клас / інтерфейс)
    - Метод handle(PageProcessingContext ctx).
    - Посилання на наступний обробник next.
- Конкретні обробники:

    - RemoveScriptsHandler
    - RemoveAdsHandler
    - ExtractTextHandler
    - KeywordSearchHandler

- ContentProcessor — конфігурує ланцюжок обробників і запускає обробку.

2.2 Діаграма класів (фрагмент)

- PageHandler — базовий абстрактний клас.
- RemoveScriptsHandler → наслідує PageHandler.
- RemoveAdsHandler → наслідує PageHandler.
- ExtractTextHandler → наслідує PageHandler.
- KeywordSearchHandler → наслідує PageHandler.
- ContentProcessor має посилання на перший елемент ланцюжка.

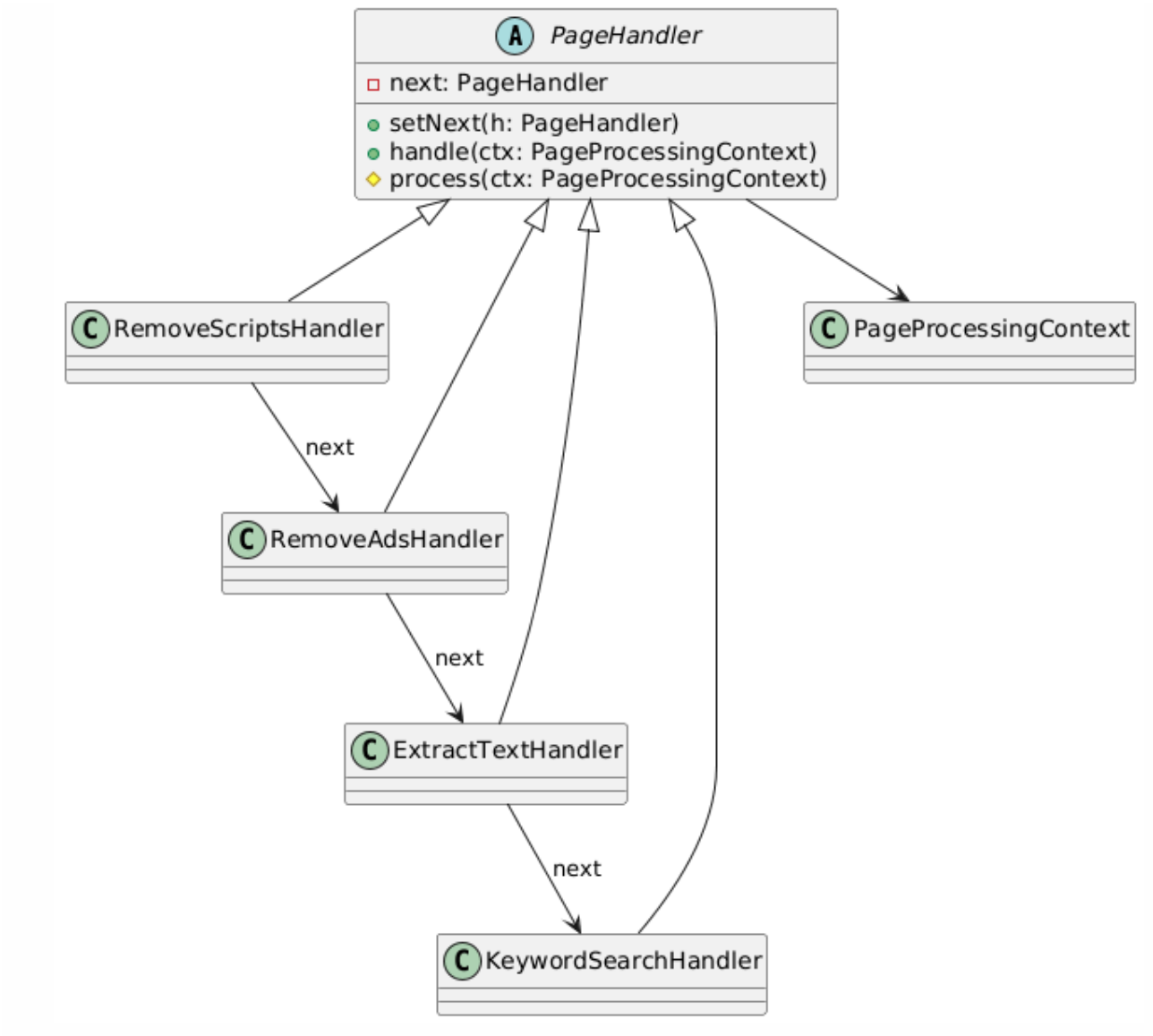

Рисунок 2.2 - Діаграма класів (фрагмент)

2.3. Фрагмент коду (Java)

```
public class PageProcessingContext {
    private String rawHtml;
    private String cleanedHtml;
    private String plainText;
    private Map<String, Integer> keywordCounts = new HashMap<>();

    public PageProcessingContext(String rawHtml) {
        this.rawHtml = rawHtml;
    }

    // getters and setters
```

```
}

public abstract class PageHandler {
    protected PageHandler next;

    public PageHandler setNext(PageHandler next) {
        this.next = next;
        return next;
    }

    public void handle(PageProcessingContext ctx) {
        process(ctx);
        if (next != null) {
            next.handle(ctx);
        }
    }

    protected abstract void process(PageProcessingContext ctx);
}

public class RemoveScriptsHandler extends PageHandler {
    @Override
    protected void process(PageProcessingContext ctx) {
        String html = ctx.getRawHtml();
        // Спрощена логіка: видалення <script>...</script>
        String cleaned = html.replaceAll("(?s)<script.*?>.*?</script>", "");
        ctx.setCleanedHtml(cleaned);
    }
}

public class RemoveAdsHandler extends PageHandler {
    @Override
    protected void process(PageProcessingContext ctx) {
        String html = ctx.getCleanedHtml();
        // Спрощена логіка: видалення блоків з певними CSS-класами
        String cleaned = html.replaceAll("(?s)<div class=\"ad\".*?>.*?</div>", "");
        ctx.setCleanedHtml(cleaned);
    }
}

public class ExtractTextHandler extends PageHandler {
```

```
    @Override
    protected void process(PageProcessingContext ctx) {
        String html = ctx.getCleanedHtml();
        // Спрощений варіант витягу тексту без тегів
        String text = html.replaceAll("<[^>]+>", " ");
        ctx.setPlainText(text);
    }
}

public class KeywordSearchHandler extends PageHandler {
    private final List<String> keywords;

    public KeywordSearchHandler(List<String> keywords) {
        this.keywords = keywords;
    }

    @Override
    protected void process(PageProcessingContext ctx) {
        String text = ctx.getPlainText().toLowerCase();
        Map<String, Integer> counts = new HashMap<>();
        for (String keyword : keywords) {
            int count = text.split(keyword.toLowerCase(), -1).length - 1;
            counts.put(keyword, count);
        }
        ctx.setKeywordCounts(counts);
    }
}

public class ContentProcessor {
    private final PageHandler firstHandler;

    public ContentProcessor(List<String> keywords) {
        // Конфігурація ланцюжка
        PageHandler removeScripts = new RemoveScriptsHandler();
        PageHandler removeAds = new RemoveAdsHandler();
        PageHandler extractText = new ExtractTextHandler();
        PageHandler keywordSearch = new KeywordSearchHandler(keywords);

        removeScripts.setNext(removeAds)
                     .setNext(extractText)
                     .setNext(keywordSearch);
```

```
        this.firstHandler = removeScripts;
    }

    public PageProcessingContext process(String rawHtml) {
        PageProcessingContext ctx = new PageProcessingContext(rawHtml);
        firstHandler.handle(ctx);
        return ctx;
    }
}
```

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. Яке призначення шаблону «Адаптер»?

Шаблон «Адаптер» призначений для узгодження двох несумісних інтерфейсів, дозволяючи використовувати існуючий клас там, де очікується інший інтерфейс, без модифікації коду цього класу.

2. Нарисуйте структуру шаблону «Адаптер».

Структура включає такі елементи:

- Client – працює через інтерфейс Target;
- Target – інтерфейс, який очікує клієнт;
- Adapter – реалізує Target і містить об’єкт Adaptee;
- Adaptee – існуючий клас з несумісним інтерфейсом.
- Adapter приймає виклики Client через Target і перенаправляє їх до Adaptee.

3. Які класи входять в шаблон «Адаптер», та яка між ними взаємодія?

У шаблон входять:

- Target – інтерфейс, який очікує клієнт;
- Adaptee – існуючий клас, який має інший інтерфейс;
- Adapter – робить інтерфейси сумісними;
- Client – працює лише через Target.
  Взаємодія: Client викликає методи Target → Adapter приймає виклик → Adapter викликає відповідний метод Adaptee.

4. Яка різниця між реалізацією «Адаптера» на рівні об’єктів та на рівні класів?

- Адаптер на рівні об’єктів використовує композицію: Adapter містить посилання на Adaptee. Це гнучкіше, можна підміняти Adaptee у runtime.
- Адаптер на рівні класів використовує наслідування: Adapter наслідує Adaptee та Target. Менш гнучкий, потребує множинного наслідування.

5. Яке призначення шаблону «Будівельник»?

Шаблон «Будівельник» призначений для поетапного створення складних об’єктів, відокремлюючи процес конструювання від кінцевої структури, дозволяючи створювати різні варіанти одного й того ж продукту.

6. Нарисуйте структуру шаблону «Будівельник».

Структура включає:

- Director – керує порядком виконання кроків;
- Builder – інтерфейс кроків побудови;
- ConcreteBuilder – реалізує кроки побудови і формує продукт;
- Product – кінцевий об’єкт, що створюється.

7. Які класи входять в шаблон «Будівельник», та яка між ними взаємодія?

- Builder оголошує методи побудови частин продукту.
- ConcreteBuilder реалізує ці методи та створює продукт.
- Director задає порядок виклику методів Builder.
- Product – результат роботи ConcreteBuilder.
  Взаємодія: Director → Builder → ConcreteBuilder → Product.

8. У яких випадках варто застосовувати шаблон «Будівельник»?

Будівельник потрібний, коли:

- об’єкт складається з багатьох частин;
- треба мати різні конфігурації одного продукту;
- конструктор має надто багато параметрів;
- треба приховати деталізований процес створення об’єкта.

9. Яке призначення шаблону «Команда»?

Шаблон «Команда» інкапсулює запит у вигляді об’єкта, дозволяючи ставити команди в чергу, передавати, зберігати, логувати та реалізовувати undo/redo.

10. Нарисуйте структуру шаблону «Команда».

- Command – інтерфейс з методом execute().
- ConcreteCommand – реалізація команди, що містить Receiver.
- Receiver – виконує реальні дії.
- Invoker – зберігає команду і викликає її.
- Client – створює команди та налаштовує взаємодію.

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

Client створює ConcreteCommand і задає Receiver.
Invoker отримує команду та викликає її execute().

ConcreteCommand делегує виконання Receiver.
Receiver виконує реальну операцію.

12. Розкажіть як працює шаблон «Команда».

Клієнт створює команду, яка інкапсулює операцію та параметри.
Invoker зберігає команду і викликає execute(), коли потрібно.
Команда викликає необхідні методи Receiver.
Таким чином відправник (Invoker) не знає, що саме виконується лише викликає execute().

13. Яке призначення шаблону «Прототип»?

Прототип дозволяє створювати нові об’єкти шляхом копіювання існуючих екземплярів, замість створення їх через new. Це корисно, коли створення об’єкта дороге або складне.

14. Нарисуйте структуру шаблону «Прототип».

- Prototype – інтерфейс/абстракція з методом clone().
- ConcretePrototype – реалізує клонування.
- Client – зберігає та копіює прототипи через clone().

15. Які класи входять в шаблон «Прототип», та яка між ними взаємодія?

Prototype визначає метод clone().
ConcretePrototype створює копію свого стану.
Client викликає clone() для створення нових об’єктів, не знаючи деталей реалізації.

16. Які можна привести приклади використання шаблону «Ланцюжок відповідальності»?

Приклади:

- обробка HTTP-запитів через middleware (автентифікація, логування, фільтрація);
- обробка подій у графічних інтерфейсах;
- фільтрація або трансформація даних поетапно;
- обробка тексту або веб-сторінок у декілька кроків - як у веб-сканері: видалення реклами, скриптів, очищення контенту, пошук ключових слів.