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*

із дисципліни ***«Технології розробки програмного забезпечення»***

Тема: ***«Вступ до паттернів проектування»***

Виконав:

Студент групи ІА-34

Ястремський Богдан

Перевірив:

асистент кафедри ІСТ

Мягкий Михайло Юрійович

Київ — 2025

Тема: Вступ до паттернів проектування.

Мета: Вивчити структуру шаблонів «Singleton», «Iterator», «Proxy», «State», «Strategy» та навчитися застосовувати їх в реалізації програмної системи.

Застосунок (№6): Web-browser (proxy, chain of responsibility, factory method, template method, visitor, p2p).

Веб-браузер повинен мати можливість зробити наступне: мати адресний рядок для введення адреси сайту, переміщатися і відображати структур html документа, переглядати підключений javascript та css файли, перегляд всіх підключених ресурсів (зображень), коректна обробка відповідей з сервера (коди відповідей HTTP) – переходи при перенаправленнях, відображення сторінок 404 і 502/503.

Короткі теоретичні відомості:

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.

Відповідне використання патернів проєктування дає розробнику ряд незаперечних переваг. 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. Застосування патернів проєктування підвищує стійкість системи до зміни вимог та спрощує неминуче подальше доопрацювання системи.

«Singleton» (Одинак) являє собою клас в термінах ООП, який може мати не більше одного об'єкта. Даний об'єкт найчастіше зберігається як статичне поле в самому класі.

«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.

«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.

Шаблон «State» (Стан) дозволяє змінювати логіку роботи об'єктів у випадку зміни їх внутрішнього стану. Наприклад, відсоток нарахованих на картковий рахунок грошей залежить від стану картки: Visa Electron, Classic, Platinum і т.д. 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.

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.

Хід роботи:

1. Повна діаграма класів (без шаблону).

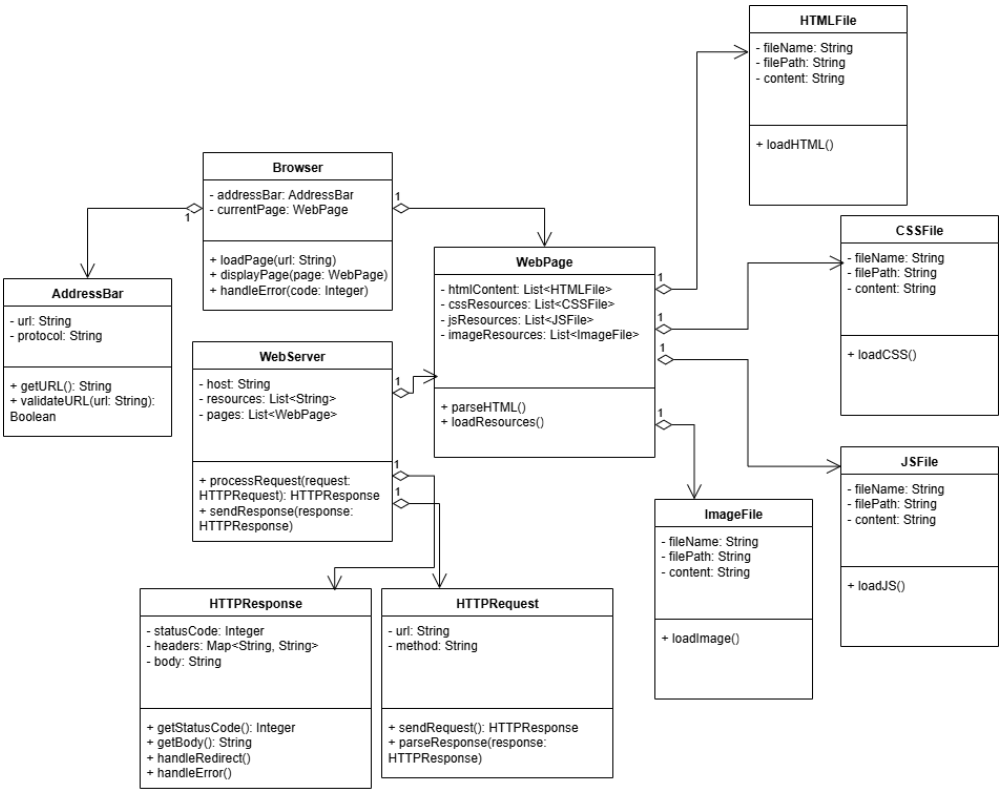


Рис. 4.1 – Діаграма класів системи

2. Діаграма класів з використанням шаблону «Proxy»:

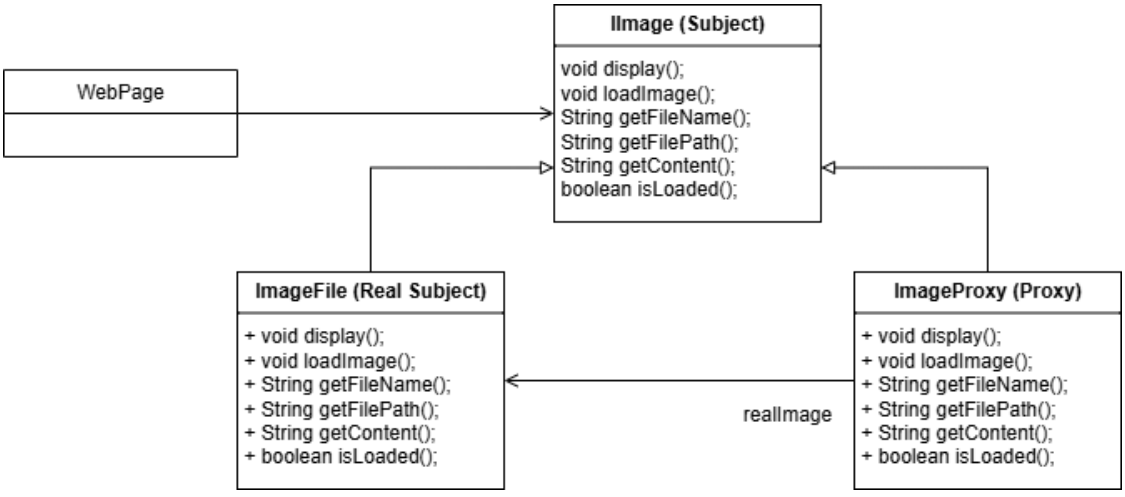


Рис. 4.2 – Діаграма класів системи з використанням шаблону «Proxy»

Я визначаю інтерфейс IImage, який визначає в собі методи для відображення картинки, завантаження, отримання імені, шляху і вмісту і перевірку завантаженості вмісту.

Далі я пов’язую його із давно створеним класом о створеним класом ImageFile, і реалізую цей інтерфейс в цьому класі – клас, який оброблює реальні зображення.

Далі я створюю клас ImageProxy, який визначає заглушку і реалізує той самий інтерфейс.

По суті я розділив ці картинки, і я можу не задавати заглушки в класі, а винести їх в окремі класи і обробити їх відповідно.

3. Фрагменти коду:

3.1. Subject:

```
package org.example.webbrowser;

public interface IImage {
   void display();
   void loadImage();
   String getFileName();
   String getFilePath();
   String getContent();
   boolean isLoaded();
}
```

3.2. Real Subject:

```
package org.example.webbrowser;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Base64;

public class ImageFile implements IImage {
```

```
private String fileName;
private String filePath;
private String content; // Base64 encoded content
private boolean loaded;

public ImageFile(String fileName, String filePath) {
    this.fileName = fileName;
    this.filePath = filePath;
    this.content = "";
    this.loaded = false;
}

@Override
public String getFileName() {
    return fileName;
}

public void setFileName(String fileName) {
    this.fileName = fileName;
}

@Override
public String getFilePath() {
    return filePath;
}

public void setFilePath(String filePath) {
    this.filePath = filePath;
}
```

```
    @Override
    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    @Override
    public boolean isLoaded() {
        return loaded;
    }

    @Override
    public void loadImage() {
        System.out.println("Loading real image: " + fileName + "...");

        // Simulating load for testing
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }

        try {
            byte[] fileContent = Files.readAllBytes(Paths.get(filePath));
            this.content                                                    =
Base64.getEncoder().encodeToString(fileContent);
            this.loaded = true;
```

```
            System.out.println("Real image loaded: " + fileName);
        } catch (IOException e) {
            System.err.println("Error loading image file: " + fileName);
            this.loaded = false;
        }
    }

    @Override
    public void display() {
        if (!loaded) {
            loadImage();
        }
        System.out.println("Displaying real image: " + fileName + " (size:
" + content.length() + " chars)");
    }
}
```

3.3. Proxy:

```
package org.example.webbrowser;

import java.util.Base64;

public class ImageProxy implements IImage {
    private ImageFile realImage;
    private String fileName;
    private String filePath;
    private String placeholderContent; // Заглушка
    private boolean isRealImageLoaded;

    public ImageProxy(String fileName, String filePath) {
```

```
    this.fileName = fileName;
    this.filePath = filePath;
    this.isRealImageLoaded = false;
    this.placeholderContent = createPlaceholder();
}

@Override
public String getFileName() {
    return fileName;
}

@Override
public String getFilePath() {
    return filePath;
}

@Override
public String getContent() {
    if (isRealImageLoaded && realImage != null) {
        return realImage.getContent();
    }
    return placeholderContent;
}

@Override
public boolean isLoaded() {
    return isRealImageLoaded;
}

@Override
```

```
    public void display() {
        if (!isRealImageLoaded) {
            System.out.println("Displaying placeholder for: " + fileName);
            System.out.println("   (Real image will be loaded on demand)");
        }

        loadImage();

        if (realImage != null) {
            realImage.display();
        }
    }

    @Override
    public void loadImage() {
        if (!isRealImageLoaded) {
            System.out.println("Proxy initiating real image loading...");

            realImage = new ImageFile(fileName, filePath);
            realImage.loadImage();

            isRealImageLoaded = true;
            System.out.println("Proxy: Real image is now loaded and cached");
        } else {
            System.out.println("Proxy: Using cached image (already loaded)");
        }
    }
```

```
    private String createPlaceholder() {
        String svg = String.format("""
            <svg width='300' height='200' xmlns='http://www.w3.org/2000/svg'>
                <defs>
                    <linearGradient id='grad' x1='0%%' y1='0%%' x2='100%%' y2='100%%'>
                        <stop offset='0%%' style='stop-color:#e0e0e0;stop-opacity:1' />
                        <stop offset='100%%' style='stop-color:#f5f5f5;stop-opacity:1' />
                    </linearGradient>
                </defs>
                <rect width='300' height='200' fill='url(#grad)' stroke='#cccccc' stroke-width='2'/>
                <circle cx='150' cy='80' r='25' fill='#999999' opacity='0.5'/>
                <circle cx='150' cy='80' r='15' fill='#cccccc'/>
                <polygon points='140,75 145,85 155,85 160,75' fill='#999999' opacity='0.5'/>
                <text x='150' y='140' text-anchor='middle' font-family='Arial' font-size='14' fill='#666666'>
                    Loading...
                </text>
                <text x='150' y='160' text-anchor='middle' font-family='Arial' font-size='12' fill='#999999'>
                    %s
                </text>
            </svg>
            """, fileName);
```

```
        return "" +
    Base64.getEncoder().encodeToString(svg.getBytes());
    }
}
```

Висновок: на даному лабораторному занятті я познайомився з поняттям патернів проектування, які є їх види, та їхні плюси і мінуси. Шаблон «Singleton» хоча і є прикладом глобальних даних, проте дуже важко підтримувати код при використанні цього патерну через складність роботи з глобальними змінними. Шаблон «Iterator» відповідає за доступ до елементів колекції і проходу по ним і є гарним прикладом розділення відповідальності, але є накладним, якщо можна використати звичайний цикл. Шаблон «Proxy» хоча і дозволяє легко керувати життєвим циклом об’єкту, але є ризик втрати швидкості роботи системи. Шаблон «State» дозволяє системі бути гнучкою, в той же час треба ускладнювати контекстний клас для перемикання між станами. Шаблон «Strategy» дещо схожий на «State», але відображає поведінку об’єкта незалежно від стану.

Відповіді на контрольні питання:

1. Що таке шаблон проєктування?

   Формалізований опис, який часто зустрічається в завданнях проєктування, вдале рішення даної задачі, а також рекомендації по застосуванню цього рішення в різних ситуаціях.

2. Навіщо використовувати шаблони проєктування?

   Модель системи, побудована в межах патернів проєктування, фактично є структурованим виокремленням тих елементів і зв'язків, які значимі при вирішенні поставленого завдання.

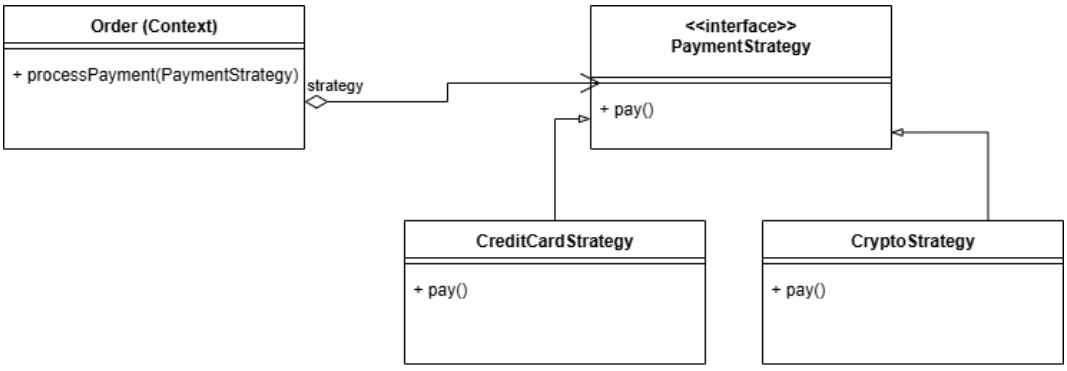
   Крім цього, модель, побудована з використанням патернів проєктування, більш проста і наочна у вивченні, ніж стандартна модель.

3. Яке призначення шаблону «Стратегія»?

   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.

4. Нарисуйте структуру шаблону «Стратегія».



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

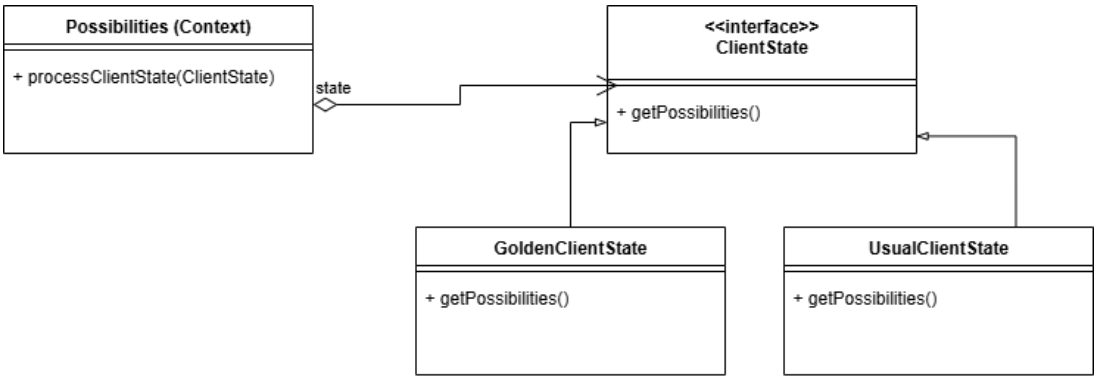
   Context – визначає метод, в якому один з параметрів – стратегія

   Strategy – інтерфейс або абстрактний клас, який містить спільний метод для всіх інших класів-стратегій, які реалізують цей інтерфейс.

6. Яке призначення шаблону «Стан»?

Шаблон «State» (Стан) дозволяє змінювати логіку роботи об'єктів у випадку зміни їх внутрішнього стану.

7. Нарисуйте структуру шаблону «Стан».



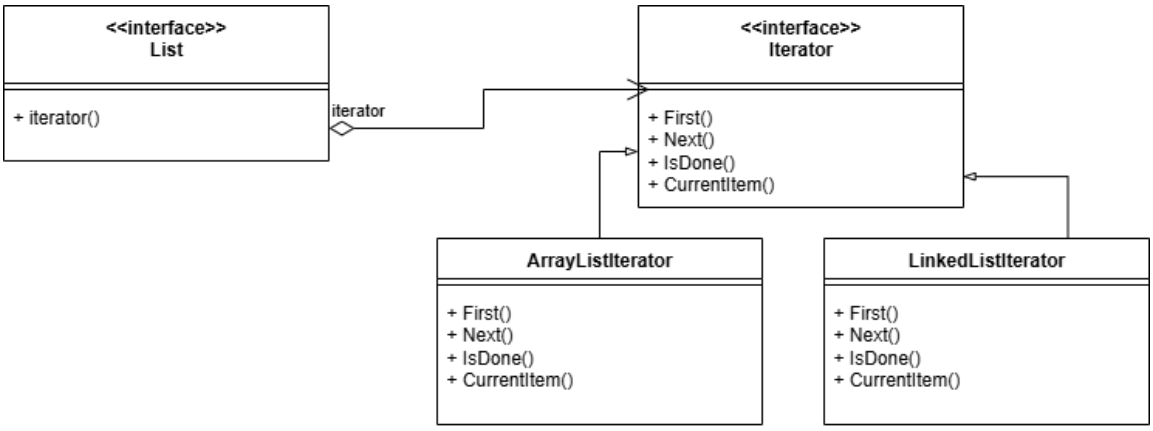
8. Які класи входять в шаблон «Стан», та яка між ними взаємодія?

   Об’єкти, що мають стан (Context), при зміні стану просто записують новий об'єкт в поле state, що призводить до повної зміни поведінки об'єкта, а 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.

9. Яке призначення шаблону «Ітератор»?

   «Iterator» (Ітератор) являє собою шаблон реалізації об'єкта доступу до набору (колекції, агрегату) елементів без розкриття внутрішніх механізмів реалізації.

10. Нарисуйте структуру шаблону «Ітератор».



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

    Шаблонний ітератор містить:

- First() – установка покажчика перебору на перший елемент колекції;
- Next() – установка покажчика перебору на наступний елемент колекції;
- IsDone – булевське поле, яке встановлюється як true коли покажчик перебору досяг кінця колекції;
- CurrentItem – поточний об'єкт колекції.

12. В чому полягає ідея шаблона «Одинак»?

    «Singleton» (Одинак) являє собою клас в термінах ООП, який може мати не більше одного об'єкта. Даний об'єкт найчастіше зберігається як статичне поле в самому класі.

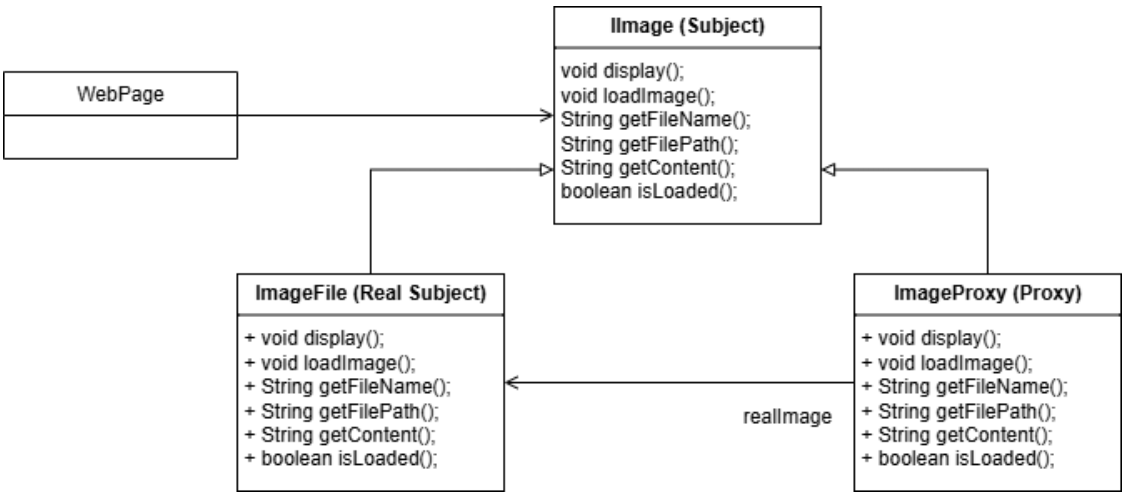
13. Чому шаблон «Одинак» вважають «анти-шаблоном»?

    Це пов'язано з тим, що «одинаки» представляють собою глобальні дані (як глобальна змінна), що мають стан. Стан глобальних об'єктів важко відслідковувати і підтримувати коректно.

14. Яке призначення шаблону «Проксі»?

    «Proxy» (Проксі) – об'єкти є об'єктами-заглушками або двійниками/замінниками для об'єктів конкретного типу.

15. Нарисуйте структуру шаблону «Проксі».



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

    Зовнішня система (WebPage) – приймає картинку з предмету інтерфейсу (IImage), а вже справжній предмет (ImageFile – звичайна картинка) та проксі (ImageProxy) реалізовують його.