МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Лабораторна робота №5

Тема**: «**Патерн проектування Міст»

Дисципліна: «Ефективність та якість архітектурних рішень інформаційних систем»

Виконав: студент групи ІКМ-М224а
Нестеренко Владислав Валентинович

Перевірив: Асистент кафедри
Хорошун Андрій Сергійович

Харків - 2024р.

Мета: Здобути навички з реалізацією паттерна проектування Міст.

Завдання:

1. Ознайомитися з патерном Міст.
2. Виконати всі пункти лабораторної роботи.
3. Надати звіт про виконану роботу

Хід роботи:

1. Є система у якій є декілька типів сторінок(**Page**) та декілька типів подання(**Renderer**) цих сторінок.
2. Типи сторінок:
   - Проста сторінка (**SimplePage**)
   - Сторінка товару (**ProductPage**)
3. Типи подання:
   - HTML (**HTMLRenderer**)
   - JSON (**JsonRenderer**)
   - XML (**XmlRenderer**)
4. Проста сторінка (SimplePage) складається з заголовку сторінки там контенту.
5. Сторінка товару (ProductPage) має містити назву товару, опис товару, зображення та id товару. Вся ця информація представлена у обїекті класса **Product**

Необхідно створити структуру класів та методів яка буде демонструвати реалізацію патерну Міст і буде вирішувати описанне завдання.

Також необхідно навести приклад клієнтського коду де буде реалізовано умовний рендерінг обох типів сторінок у кількох типах.

У рамках виконання лабораторної роботи не потрібно описувати деталі реалізації самих методів! Достатньо вказати сам метод та параметри який він приймає та повертає.

Виконання:

Міст — це структурний патерн проектування, який розділяє один або кілька класів на дві окремі ієрархії — абстракцію та реалізацію, дозволяючи змінювати код в одній гілці класів, незалежно від іншої.

Реалізація на Java:

Абстрактний клас Page -представляє сторінку.

```
abstract class Page {
    protected Renderer renderer;

    public Page(Renderer renderer) {
        this.renderer = renderer;
    }

     public abstract void render();
}
```

Клас SimplePage - представляє просту сторінку з заголовком та контентом.

```
class SimplePage extends Page {
    private String title;
    private String content;

    // Конструктор, який приймає Renderer, заголовок і контент
    public SimplePage(Renderer renderer, String title, String content) {
        super(renderer);
        this.title = title;
        this.content = content;
    }

    // Реалізація методу render для простої сторінки
    @Override
    public void render() {
        renderer.renderPage(this);
    }

    // Геттери для заголовка і контенту
    public String getTitle() {
        return title;
    }

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

Клас ProductPage - представляє сторінку товару з інформацією про продукт (назва, опис, зображення та ID).

```
// Конкретний клас, що представляє сторінку товару
class ProductPage extends Page {
    private Product product;

    // Конструктор, який приймає Renderer і продукт
    public ProductPage(Renderer renderer, Product product) {
        super(renderer);
        this.product = product;
    }

    // Реалізація методу render для сторінки товару
    @Override
    public void render() {
        renderer.renderPage(this);
    }

    // Геттер для продукту
    public Product getProduct() {
        return product;
    }
}
```

Клас Product: - представляє продукт.

```
class Product {
    private String name;
    private String description;
    private String image;
    private int id;

    // Конструктор для ініціалізації полів продукту
    public Product(String name, String description, String image, int id) {
        this.name = name;
        this.description = description;
        this.image = image;
```

```
        this.id = id;
    }

    // Геттери для полів продукту
    public String getName() {
        return name;
    }

    public String getDescription() {
        return description;
    }

    public String getImage() {
        return image;
    }

    public int getId() {
        return id;
    }
}
```

Інтерфейс Renderer - визначає методи для рендерингу сторінок.

```
interface Renderer {
    // Метод для рендерингу простої сторінки
    void renderPage(SimplePage simplePage);

    // Метод для рендерингу сторінки товару
    void renderPage(ProductPage productPage);
}
```

Клас HTMLRenderer - реалізує інтерфейс Renderer для рендерингу сторінок у форматі HTML.

```
class HTMLRenderer implements Renderer {
    @Override
    public void renderPage(SimplePage simplePage) {
        // реалізація рендерінгу простої сторінки у форматі HTML
        System.out.println("<html>");
        System.out.println("<head><title>" + simplePage.getTitle() +
"</title></head>");
        System.out.println("<body>");
        System.out.println("<h1>" + simplePage.getTitle() + "</h1>");
        System.out.println("<p>" + simplePage.getContent() + "</p>");
        System.out.println("</body>");
        System.out.println("</html>");
    }

    @Override
    public void renderPage(ProductPage productPage) {
        // реалізація рендерінгу сторінки товару у форматі HTML
        Product product = productPage.getProduct();
        System.out.println("<html>");
        System.out.println("<head><title>" + product.getName() + "</title></head>");
        System.out.println("<body>");
        System.out.println("<h1>" + product.getName() + "</h1>");
        System.out.println("<p>ID: " + product.getId() + "</p>");
        System.out.println("<p>" + product.getDescription() + "</p>");
        System.out.println("<img src='" + product.getImage() + "' />");
        System.out.println("</body>");
        System.out.println("</html>");
    }
}
```

Клас JsonRenderer - реалізує інтерфейс Renderer для рендерингу сторінок у форматі JSON.

```
class JsonRenderer implements Renderer {
    @Override
    public void renderPage(SimplePage simplePage) {
        // реалізація рендерінгу простої сторінки у форматі JSON
        System.out.println("{");
        System.out.println("  \"title\": \"" + simplePage.getTitle() + "\",");
        System.out.println("  \"content\": \"" + simplePage.getContent() + "\"");
        System.out.println("}");
    }

    @Override
    public void renderPage(ProductPage productPage) {
        // реалізація рендерінгу сторінки товару у форматі JSON
        Product product = productPage.getProduct();
        System.out.println("{");
        System.out.println("  \"name\": \"" + product.getName() + "\",");
        System.out.println("  \"id\": " + product.getId() + ",");
        System.out.println("  \"description\": \"" + product.getDescription() + "\",");
        System.out.println("  \"image\": \"" + product.getImage() + "\"");
        System.out.println("}");
    }
}
```

Клас XmlRenderer - реалізує інтерфейс Renderer для рендерингу сторінок у форматі XML.

```
class XmlRenderer implements Renderer {
    @Override
    public void renderPage(SimplePage simplePage) {
        // реалізація рендерінгу простої сторінки у форматі XML
        System.out.println("<page>");
        System.out.println("  <title>" + simplePage.getTitle() + "</title>");
        System.out.println("  <content>" + simplePage.getContent() + "</content>");
        System.out.println("</page>");
    }

    @Override
    public void renderPage(ProductPage productPage) {
        // реалізація рендерінгу сторінки товару у форматі XML
        Product product = productPage.getProduct();
        System.out.println("<product>");
        System.out.println("  <name>" + product.getName() + "</name>");
        System.out.println("  <id>" + product.getId() + "</id>");
        System.out.println("  <description>" + product.getDescription() +
"</description>");
        System.out.println("  <image>" + product.getImage() + "</image>");
        System.out.println("</product>");
    }
}
```

Клієнтський код який демонструє використання патерну «Міст». Створює та рендерить сторінки у різних форматах.

```
public class Main {
        public static void main(String[] args) {
            // Створюємо різні рендерери
            Renderer htmlRenderer = new HTMLRenderer();
            Renderer jsonRenderer = new JsonRenderer();
            Renderer xmlRenderer = new XmlRenderer();

            // Створюємо сторінки з різними рендерерами
            Page simplePage = new SimplePage(htmlRenderer, "Welcome", "This is the
```

```
welcome page.");
            Page productPage = new ProductPage(jsonRenderer, new Product("Laptop", "A
powerful laptop.", "laptop.jpg", 101));

            // Рендеримо сторінки
            simplePage.render();
            productPage.render();

            // Змінюємо Renderer для тих самих сторінок
            simplePage = new SimplePage(xmlRenderer, "Welcome", "This is the welcome
page.");
            productPage = new ProductPage(htmlRenderer, new Product("Laptop", "A
powerful laptop.", "laptop.jpg", 101));

            // Рендеримо сторінки з новими рендерерами
            simplePage.render();
            productPage.render();
        }
    }
```

Висновки

Під час виконання даної лабораторної роботи я здобув навички з реалізації патерна проектування «Міст».

Я створив структуру класів та методів, яка демонструє застосування цього патерна для відокремлення абстракцій сторінок від їх реалізацій рендерингу. Зокрема, я реалізував два типи сторінок: просту сторінку (SimplePage) та сторінку товару (ProductPage). Також я розробив різні рендерери (HTMLRenderer, JsonRenderer, XmlRenderer), які можуть рендерити ці сторінки у різних форматах (HTML, JSON, XML).

В результаті, структура коду дозволяє легко змінювати або додавати нові типи рендерингу без необхідності зміни класів сторінок, що підтверджує гнучкість та ефективність патерна «Міст».