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 №6

Технології розроблення програмного забезпечення

Тема: «Патерни проектування»

Виконав:

Студент групи ІА-31

Калиновський В.О

Перевірив:

Мягкий Михайло Юрійович

Київ-2025

Мета: Вивчити структуру шаблонів «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator» та навчитися застосовувати їх в реалізації програмної системи.

Тема роботи:

Графічний редактор (proxy, prototype, decorator, bridge, flyweight, SOA)

Графічний редактор повинен вміти створювати / редагувати растрові (або векторні на розсуд студента) зображення в 2-3 основних популярних форматах (bmp, png, jpg), мати панель інструментів для створення графічних примітивів, вибору кольорів, нанесення тексту, додавання найпростіших візуальних ефектів (ч/б растр, інфрачервоний растр, 2-3 на вибір учня), роботи з шарами.

Теоретичні відомості

Abstract-Factory

Дозволяє створювати сімейства взаємопов'язаних об'єктів без вказівки їх конкретних класів. Наприклад, у генерації кімнат для гри створюються стіни, двері, підлога і меблі одного стилю. Переваги: об'єкти одного стилю узгоджені, код структурований, легко додавати нові стилі. Недоліки: складність коду, додавання нового типу продукту потребує змін у багатьох місцях.

Factory-Method

Визначає інтерфейс для створення об'єктів базового типу, дозволяючи підставляти власні підкласи без зміни базової поведінки. Наприклад, створення TcpPacket та UdpPacket через відповідні фабрики. Переваги: відсутність прив'язки до конкретних класів, централізоване створення продуктів, легке додавання нових продуктів. Недоліки: може створювати велику кількість ієрархій класів.

Memento

Дозволяє зберігати і відновлювати стан об'єкта без порушення інкапсуляції. Часто використовується разом з патерном Command для реалізації undo. Переваги: не порушує інкапсуляцію, спрощує структуру об'єкта. Недоліки: потребує багато пам'яті при частому створенні знімків, необхідно контролювати звільнення застарілих станів.

Observer

Призначення: Шаблон визначає залежність «один-до-багатьох» таким чином, що коли один об'єкт змінює власний стан, усі інші об'єкти отримують про це сповіщення і мають можливість змінити власний стан також.

Decorator

Дозволяє динамічно додавати функціональні можливості об'єкту без зміни його коду, обертаючи його в новий клас. Наприклад, додавання смуги прокрутки до візуальних елементів UI. Переваги: декомпозиція на дрібні об'єкти, динамічне додавання обов'язків, гнучкість. Недоліки: велика кількість дрібних класів, складно конфігурувати об'єкти з кількома обгортками одночасно.

Хід роботи

1. Діаграма класів для реалізації шаблону Декоратор

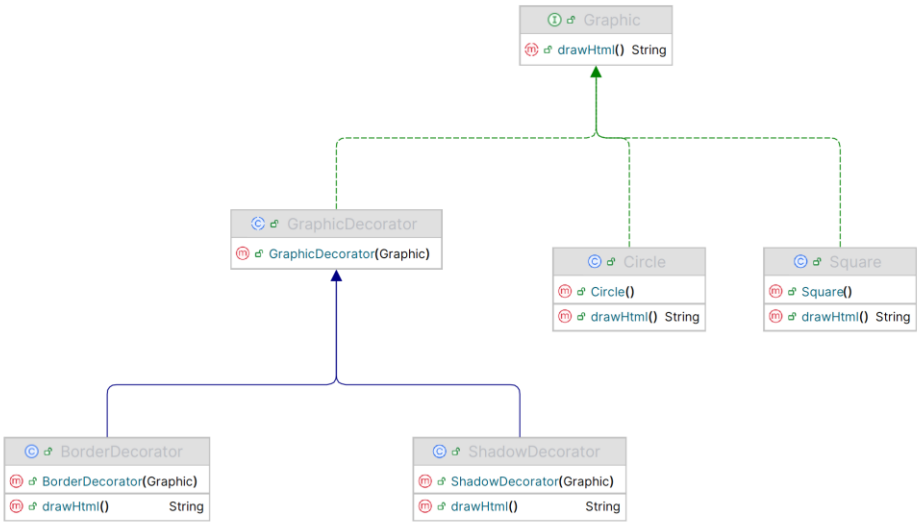


Рисунок 1 – Діаграма класів

Опис діаграми класів

Клас Graphic є базовим компонентом, який визначає спільний інтерфейс для всіх графічних елементів у редакторі. Він містить метод відображення, який реалізують як звичайні об’єкти, так і декоратори.

Класи Circle та Square виступають конкретними компонентами та відповідають за створення HTML-представлення відповідно кола та квадрата. Вони реалізують основну поведінку і можуть відображатися самостійно або бути розширеними декораторами.

Абстрактний клас GraphicDecorator виступає базою для всіх декораторів і містить посилання на інший об’єкт типу Graphic. Він зберігає інтерфейс компонента, але дозволяє додавати додаткову поведінку, обгортаючи початковий об’єкт.

Конкретні декоратори BorderDecorator і ShadowDecorator розширюють можливості базових фігур. BorderDecorator додає до фігури рамку, а ShadowDecorator — тінь. Обидва декоратори формують HTML-обгортку навколо початкового компонента, зберігаючи його основні властивості.

Приклад реалізації шаблону:

1. Користувач вибирає фігуру (circle або square) і декоратор через HTML-кнопки.
2. Браузер відправляє HTTP-запит на контролер /decorator з параметрами shape та decorator.
3. Контролер створює базовий об’єкт фігури (Circle або Square).

4. Якщо обрано декоратор, контролер обгортає об’єкт у відповідні декоратори (BorderDecorator, ShadowDecorator або обидва).
5. Об’єкт (базовий або обгорнутий декоратором) додається у список graphics.
6. Список передається у HTML-шаблон через модель.
7. Для кожного об’єкта викликається метод draw(), який формує HTML/CSS-код для відображення фігури разом із доданими ефектами.

```java
@Controller
public class GraphicController {

    private final List<Graphic> graphics = new ArrayList<>();  2 usages

    @GetMapping({"/", "/decorator"})
    public String index(Model model,
                        @RequestParam(required = false) String shape,
                        @RequestParam(required = false) String decorator) {
        if (shape != null) {
            Graphic g;
            if ("circle".equals(shape)) g = new Circle();
            else g = new Square();

            if ("border".equals(decorator)) g = new BorderDecorator(g);
            if ("shadow".equals(decorator)) g = new ShadowDecorator(g);
            if ("both".equals(decorator)) g = new BorderDecorator(new ShadowDecorator(g));

            graphics.add(g);
        }

        model.addAttribute( attributeName: "graphics", graphics);
        return "index";
    }

}
```

Рисунок 2 – код контролеру

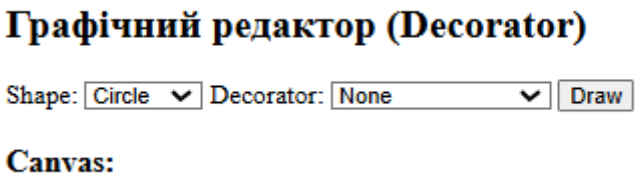


Рисунок 3 – вигляд сторінки у браузері

Результат виконання програми:

Рисунок 4 – приклад роботи шаблону декоратор

Переваги використання шаблону Декоратор для графічного редактора:

1. Динамічно додає нові ефекти до графічних об’єктів (наприклад, рамки, тіні, підсвічування) без зміни базового класу фігури.
2. Гнучкість: можна комбінувати декоратори для окремих об’єктів незалежно від інших.
3. Зменшує потребу у великій ієрархії класів через спадкування.
4. Кожен декоратор можна повторно застосовувати до різних графічних об’єктів..

Недоліки використання шаблону Декоратор для графічного редактора:

1. Кількість об’єктів у системі зростає через обгортання кожного об’єкта декораторами.
2. Може ускладнювати відстеження стану об’єкта через кілька рівнів декорацій.
3. Складніше відлагоджувати або розуміти послідовність застосованих ефектів при великій кількості комбінацій декораторів.

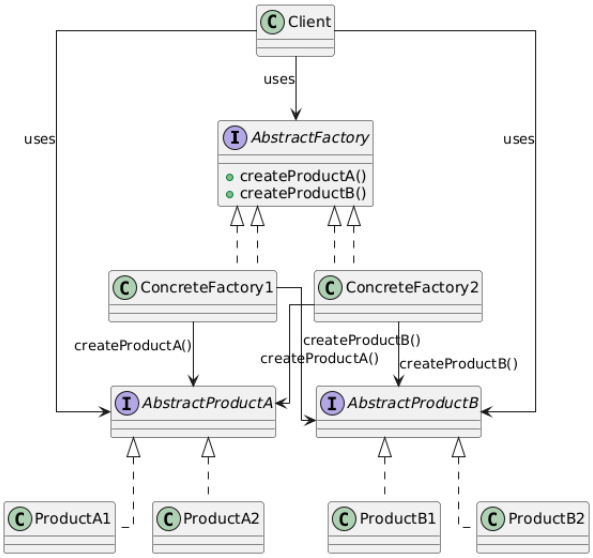
Висновок: У роботі реалізовано шаблон Decorator для динамічного додавання ефектів до графічних об’єктів у редакторі, таких як рамки та тіні. Це дозволяє змінювати поведінку окремих фігур без модифікації їх базових класів, комбінувати декоратори та повторно використовувати їх для різних об’єктів. Основною перевагою є гнучкість і масштабованість при додаванні нових ефектів, а недоліком — збільшення кількості об’єктів та ускладнення відстеження стану об’єктів при використанні багатьох декораторів одночасно.

Відповіді на контрольні питання

1. Яке призначення шаблону «Абстрактна фабрика»?

Дозволяє створювати сімейства взаємопов’язаних об’єктів без вказівки їх конкретних класів. Забезпечує узгодженість об’єктів одного стилю та відокремлює процес їх створення від використання.

2. Нарисуйте структуру шаблону «Абстрактна фабрика».



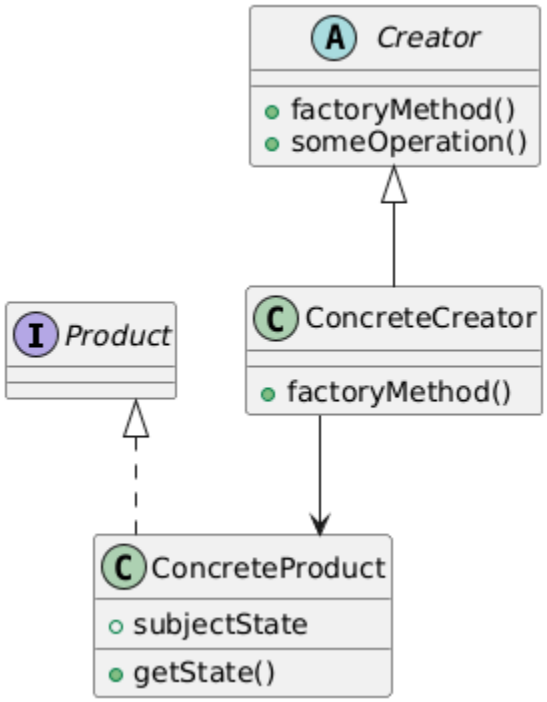
3. Які класи входять в шаблон «Абстрактна фабрика», та яка між ними взаємодія?

- AbstractFactory — інтерфейс для створення продуктів.
- ConcreteFactory — реалізація конкретної фабрики, створює конкретні продукти.
- AbstractProduct — інтерфейс продукту.
- ConcreteProduct — реалізація конкретного продукту.
  Взаємодія: клієнт використовує фабрику через AbstractFactory для створення продуктів одного стилю.

4. Яке призначення шаблону «Фабричний метод»?

Визначає інтерфейс для створення об’єктів певного базового типу і дозволяє підставляти підкласи без зміни коду клієнта.

5. Нарисуйте структуру шаблону «Фабричний метод».



6. Які класи входять в шаблон «Фабричний метод», та яка між ними взаємодія?

- Creator — базовий клас, що визначає фабричний метод.
- ConcreteCreator — реалізація фабричного методу, створює конкретний продукт.
- Product — базовий інтерфейс продукту.
- ConcreteProduct — конкретна реалізація продукту.
  Взаємодія: клієнт використовує Creator для створення об’єкта через фабричний метод, не знаючи конкретного підкласу.

7. Чим відрізняється шаблон «Абстрактна фабрика» від «Фабричний метод»?

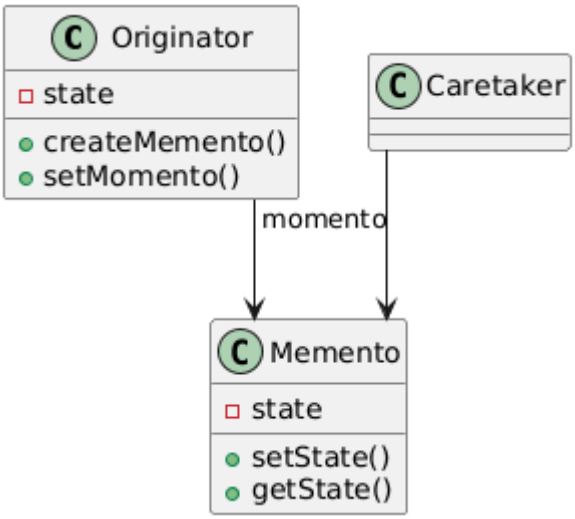
Abstract Factory створює сімейства взаємопов’язаних об’єктів (кілька типів продуктів), узгоджених між собою.

Factory Method створює один тип об’єкта і дозволяє підставляти підкласи без зміни коду клієнта.

8. Яке призначення шаблону «Знімок»?

Дозволяє зберігати та відновлювати стан об’єкта без порушення інкапсуляції. Використовується для реалізації undo/redo.

9. Нарисуйте структуру шаблону «Знімок».



10. Які класи входять в шаблон «Знімок», та яка між ними взаємодія?

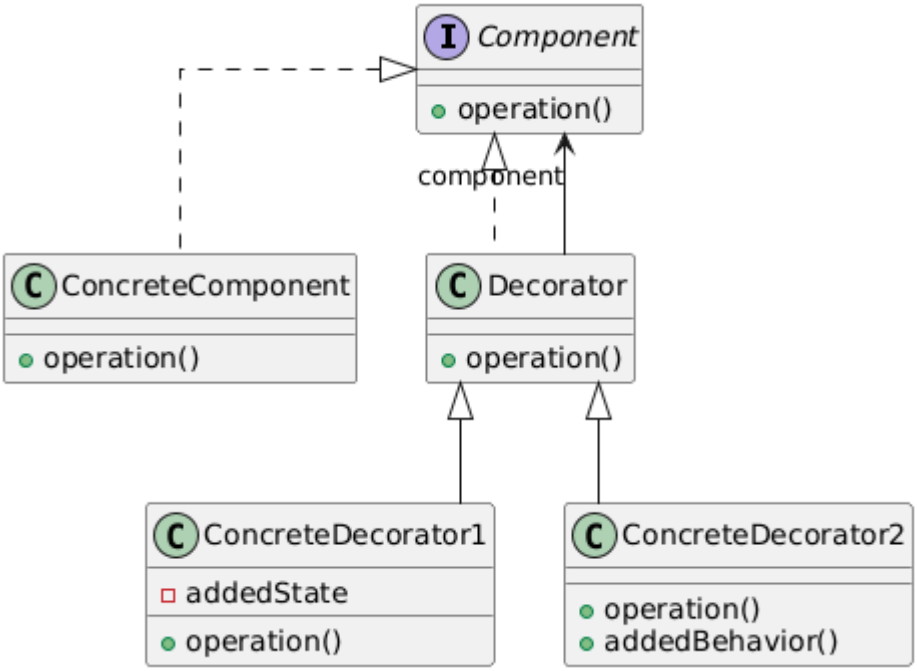
- Originator — об’єкт, стан якого потрібно зберегти.
- Memento — об’єкт для зберігання стану.
- Caretaker — управляє збереженими станами.

Взаємодія: Originator створює Memento, Caretaker зберігає його, а пізніше передає назад для відновлення стану.

11. Яке призначення шаблону «Декоратор»?

Дозволяє динамічно додавати функціональні можливості об’єкту без зміни його коду, обертаючи його в новий клас.

12. Нарисуйте структуру шаблону «Декоратор».



13. Які класи входять в шаблон «Декоратор», та яка між ними взаємодія?

- Component — базовий інтерфейс або клас об’єкта.
- ConcreteComponent — конкретний об’єкт, який можна декорувати.
- Decorator — базовий клас для декораторів, містить посилання на Component.
- ConcreteDecorator — додає нові обов’язки до об’єкта.
  Взаємодія: Decorator обертає Component і виконує додаткові дії під час виклику operation().

14. Які є обмеження використання шаблону «декоратор»?

Велика кількість дрібних класів ускладнює систему.

Складно конфігурувати об’єкти, які обгорнуті в декілька обгорток одночасно.