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
«Патерни проектування»

Тема: Редактор зображень

Виконала:

студентка групи ІА-32

Ейсмонт А.В.

Перевірив:

Мягкий М.Ю.

Київ 2025

Зміст

Тема: Патерни проектування.

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

Завдання:

- Ознайомитись з короткими теоретичними відомостями.
- Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
- Реалізувати один з розглянутих шаблонів за обраною темою.
- Реалізувати не менше 3-х класів відповідно до обраної теми.
- Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму класів, яка представляє використання шаблону в реалізації системи, навести фрагменти коду по реалізації цього шаблону.

Теоретичні відомості:

Шаблон «Memento»

Призначення: Шаблон використовується для збереження і відновлення стану об'єктів без порушення інкапсуляції [6]. Об'єкт «Memento» служить виключно для збереження змін над початковим об'єктом (Originator). Лише початковий об'єкт має можливість зберігати і отримувати стан об'єкту «Memento» для власних цілей, цей об'єкт є «порожнім» для кого-небудь ще. Об'єкт «Caretaker» використовується для передачі і зберігання ментo об'єктів в системі. Таким чином вдається досягти наступних цілей: • зберігання стану повністю відділяється від початкових об'єктів, що полегшує їх реалізацію; • передача об'єктів «Memento» лягає на плечі Caretaker об'єктів, що дозволяє гнучкіше управляти станами об'єктів і спростити дизайн класів початкових об'єктів збереження і відновлення стану реалізовані у вигляді двох простих методів і є

закритими для кого-небудь ще окрім початкових об'єктів, таким чином не порушуючи інкапсуляцію.

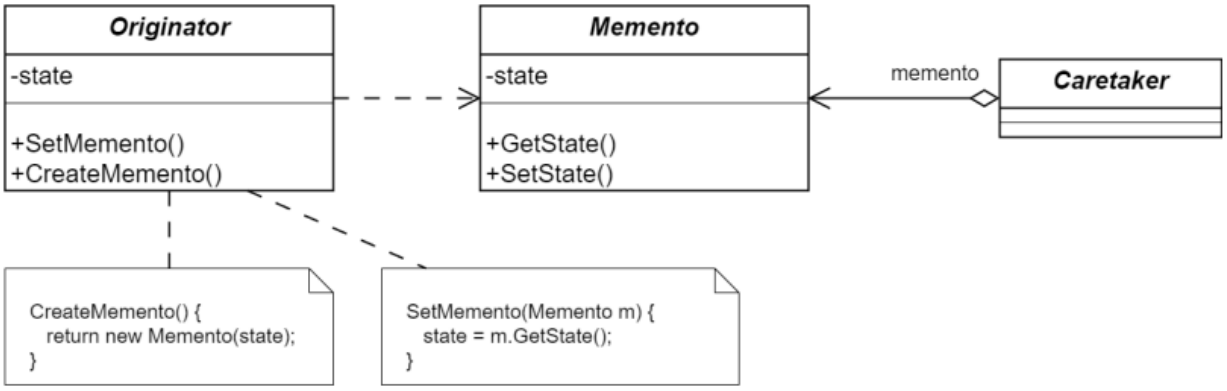
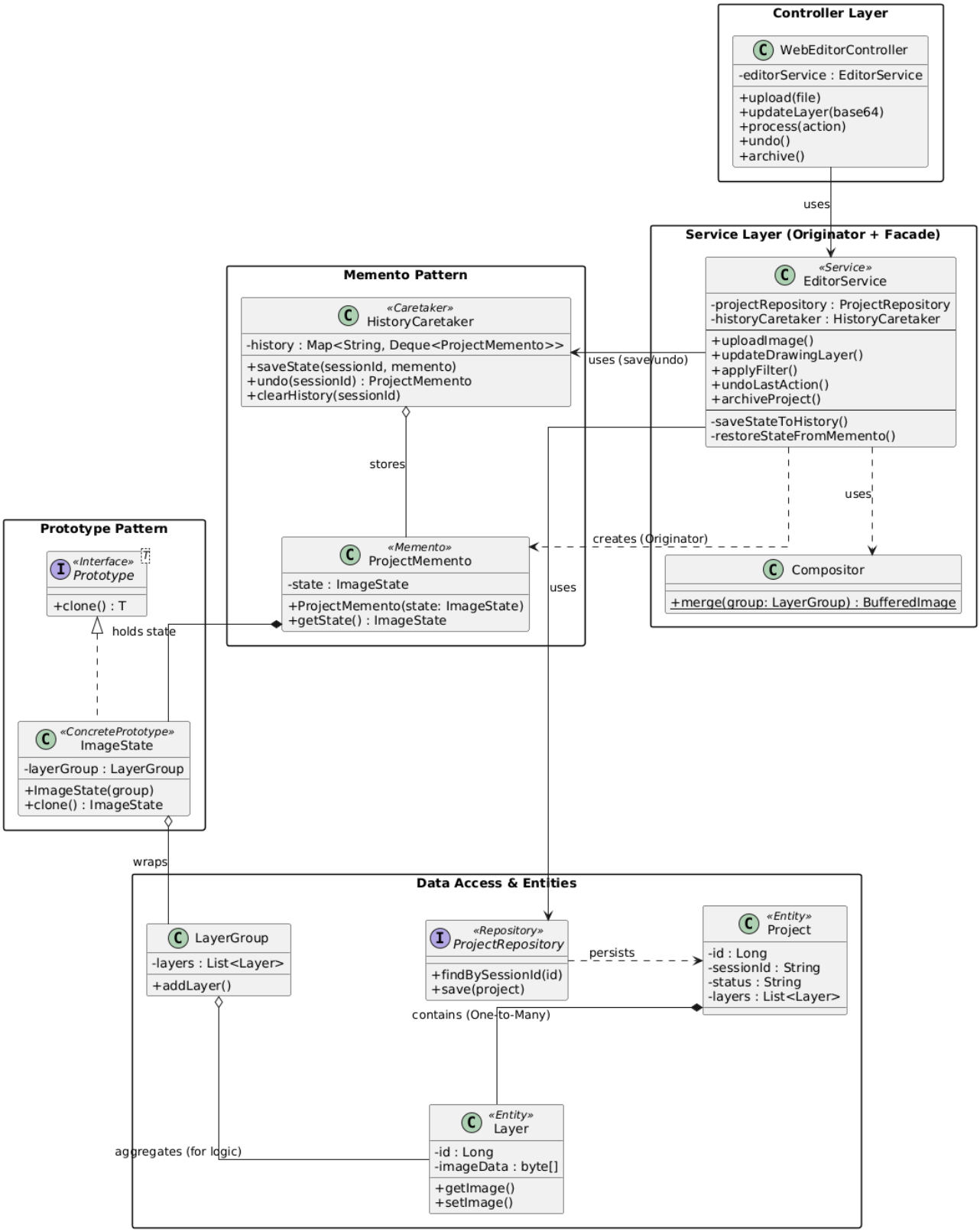


Рисунок 6.3. Структура шаблону «Знімок»

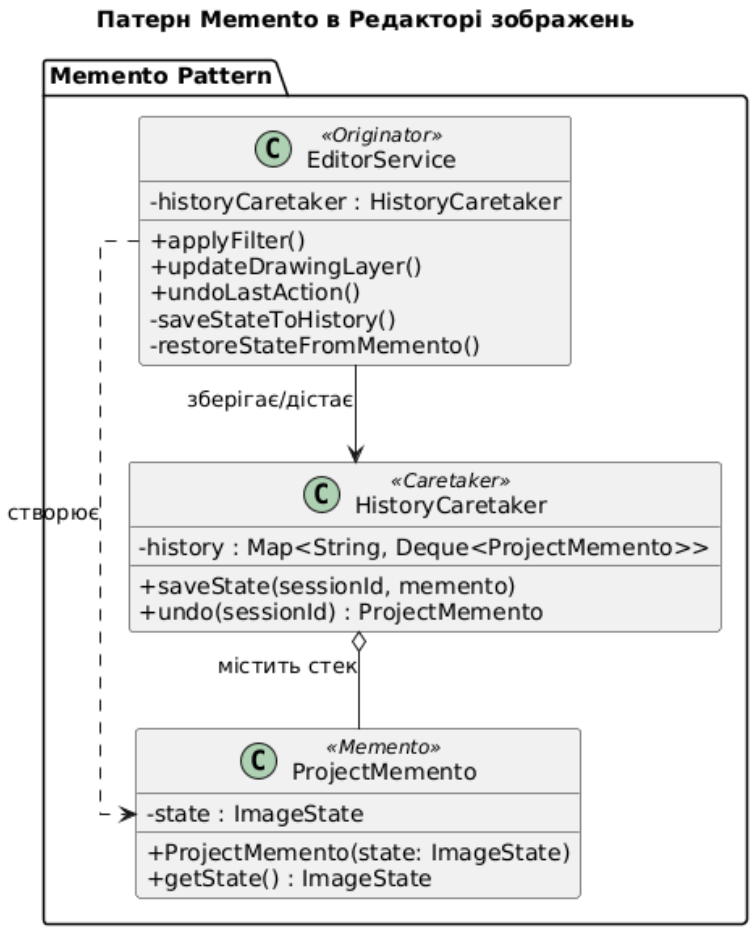
Шаблон «Мементо» дуже зручно використати разом з шаблоном «Команда» для реалізації «скасовних» дій – дані про дію зберігаються в мементо, а команда має можливість вважати і відновити початкове положення відповідних об'єктів. Переваги та недоліки: + Не порушує інкапсуляцію вихідного об’єкта. + Спрощує структуру вихідного об’єкта. Не потрібно зберігати історію версій свого стану. - Вимагає багато пам’яті, якщо клієнти дуже часто створюють знімки. - Може спричинити додаткові витрати пам’яті, якщо об’єкти, що зберігають історію, не звільняють ресурси, зайняті застарілими знімками.

Хід роботи:

Діаграма класів:



Діаграма класів, яка представляє використання шаблону Memento в реалізації системи



Детальна структура реалізації патерну Memento

Опис реалізації патерну «Мементо»

Для реалізації функції скасування дій у моєму проєкті було використано поведінковий патерн проектування Memento. Цей патерн дозволяє зберігати внутрішній стан об'єкта та відновлювати його пізніше, не порушуючи принцип інкапсуляції.

У розробленій системі графічного редактора патерн реалізовано через взаємодію трьох учасників:

1. Originator : Цю роль виконує сервісний клас EditorService. Він відповідає за створення знімка свого поточного стану перед внесенням будь-яких змін (малювання, застосування фільтрів) та за відновлення стану зі знімка.
2. Memento : Клас ProjectMemento. Він є незмінним контейнером, що зберігає стан проекту (набір шарів зображення). Для створення точної

копії даних усередині знімка використовується механізм глибокого клонування через патерн Prototype.

3. Caretaker: Клас HistoryCaretaker. Він відповідає за зберігання історії знімків. Опікун організований як стек, що дозволяє отримувати останній збережений стан за принципом LIFO. Caretaker не має доступу до вмісту знімка, лише зберігає його.

Така архітектура дозволяє легко реалізувати багаторівневе скасування дій для кожного користувача окремо, оскільки історія прив'язана до sessionId.

Фрагменти програмного коду:

ProjectMemento.java

```
package ia32.eismont.image_editor_server.patterns.memento;

import ia32.eismont.image_editor_server.patterns.prototype.ImageState;

public class ProjectMemento {

    private final ImageState state;

    public ProjectMemento(ImageState state) {
        this.state = state.clone();
    }

    public ImageState getState() {
        return state.clone();
    }
}
```

HistoryCaretaker.java

```
package ia32.eismont.image_editor_server.patterns.memento;

import org.springframework.stereotype.Component;
import java.util.ArrayDeque;
import java.util.Deque;
import java.util.HashMap;
import java.util.Map;

@Component
public class HistoryCaretaker {

    private final Map<String, Deque<ProjectMemento>> history = new HashMap<>();

    public void saveState(String sessionId, ProjectMemento memento) {
        history.computeIfAbsent(sessionId, k -> new ArrayDeque<>()).push(memento);

        if (history.get(sessionId).size() > 10) {
            history.get(sessionId).removeLast();
        }
    }

    public ProjectMemento undo(String sessionId) {
        Deque<ProjectMemento> stack = history.get(sessionId);
        if (stack != null && !stack.isEmpty()) {
            return stack.pop();
        }
        return null;
    }
```

```
    public void clearHistory(String sessionId) {
        if (history.containsKey(sessionId)) {
            history.get(sessionId).clear();
        }
    }
}
```

Фрагмент коду **EditorService.java**

```
@Service
public class EditorService {
    // ...

    public String applyFilter(String sessionId, String filterType) {
        Project project = getProjectOrThrow(sessionId);

        // 1. Створення Memento перед зміною
        saveStateToHistory(sessionId, project);

        // 2. Зміна стану
        BufferedImage bg = project.getLayer(0).getImage();
        processImage(bg, filterType);

        // ... збереження в БД
    }

    // Метод скасування дії
    public String undoLastAction(String sessionId) {
        // 1. Отримання Memento від Caretaker
        ProjectMemento memento = historyCaretaker.undo(sessionId);
        if (memento == null) throw new IllegalStateException("Історія порожня");
```

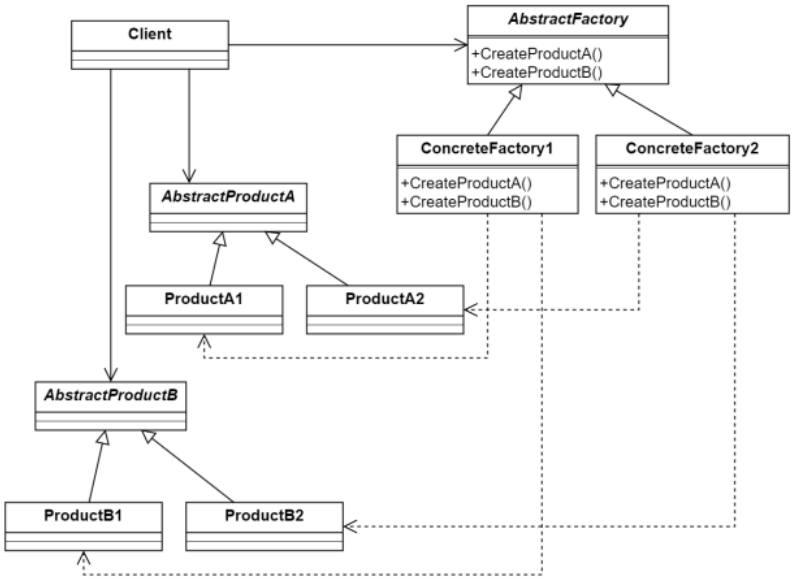
```
        // 2. Відновлення стану
        restoreStateFromMemento(project, memento);

        return imageToBase64(mergeProjectLayers(project));
    }
}
```

Питання до лабораторної роботи:

1. Яке призначення шаблону «Абстрактна фабрика»?

Шаблон «Абстрактна фабрика» використовується для створення сімейств об'єктів (продуктів) без вказівки їх конкретних класів. Він структурує знання про схожі об'єкти і створює можливість взаємозаміни різних сімейств продуктів.

2. Нарисуйте структуру шаблону «Абстрактна фабрика».



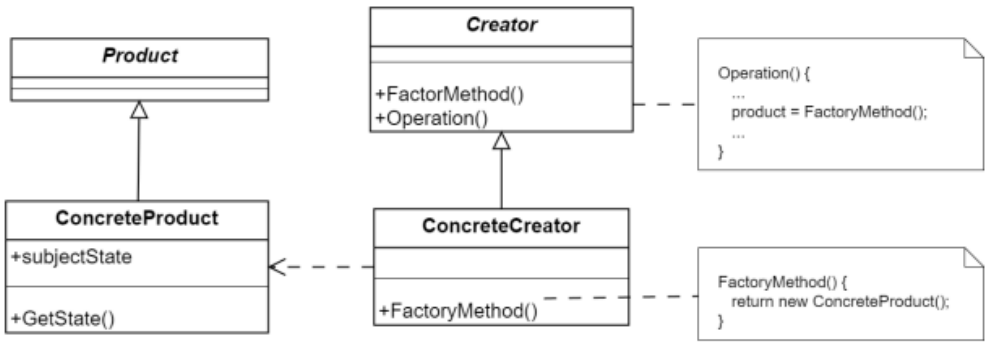
3. Які класи входять в шаблон «Абстрактна фабрика», та яка між ними взаємодія?

До складу шаблону входять: AbstractFactory (оголошує інтерфейс фабрики) , ConcreteFactory (реалізує інтерфейс фабрики для створення продуктів певного сімейства) , AbstractProduct (оголошує інтерфейс продукту) , Product (конкретна реалізація продукту) та Client (використовує фабрику та продукти). Client взаємодіє з AbstractFactory для отримання об'єктів, які реалізують інтерфейси AbstractProduct.

4. Яке призначення шаблону «Фабричний метод»?

Шаблон «Фабричний метод» призначений для визначення інтерфейсу для створення об'єктів певного базового типу, але він передає відповідальність за створення конкретного об'єкта його підкласам (нащадкам). Це дозволяє підставляти власні об'єкти у процес створення, зберігаючи при цьому базову функціональність.

5. Нарисуйте структуру шаблону «Фабричний метод».



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

Основними класами є: Product (базовий тип об'єкта) , ConcreteProduct (конкретний об'єкт, що створюється) , Creator (абстрактний клас, який оголошує FactoryMethod()) та ConcreteCreator (перевизначає FactoryMethod() для повернення конкретного продукту). ConcreteCreator успадковується від Creator і створює ConcreteProduct.

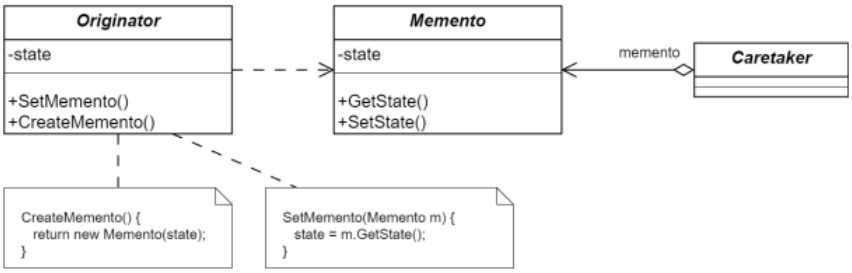
7. Чим відрізняється шаблон «Абстрактна фабрика» від «Фабричний метод»?

«Фабричний метод» вирішує проблему створення одного об'єкта і дозволяє підкласам змінити його тип, тоді як «Абстрактна фабрика» надає інтерфейс для створення сімейств пов'язаних продуктів (об'єктів кількох типів), забезпечуючи їх узгодженість.

8. Яке призначення шаблону «Знімок»?

Шаблон «Знімок» використовується для збереження та відновлення стану об'єктів без порушення їх інкапсуляції. Об'єкт Memento (знімок) є сховищем, а доступ до його стану має лише об'єкт-ініціатор Originator.

9. Нарисуйте структуру шаблону «Знімок».



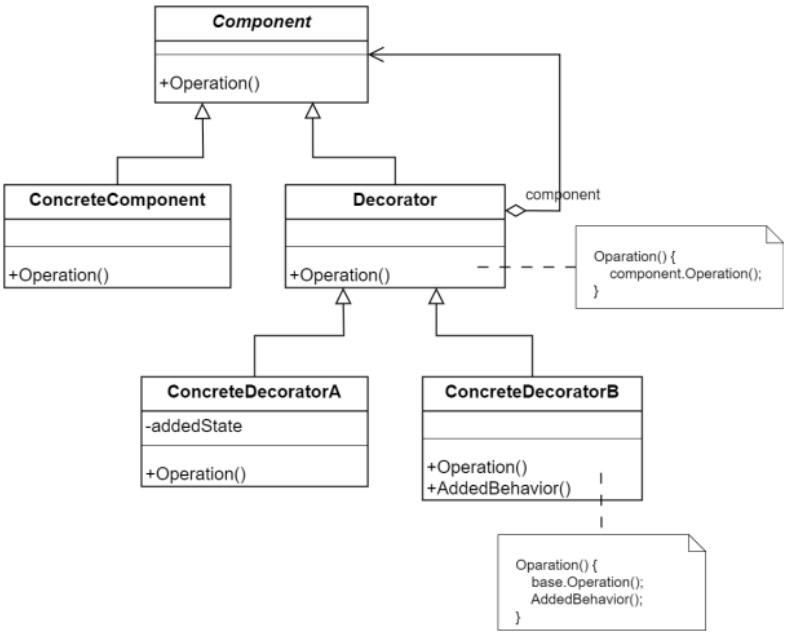
10. Які класи входять в шаблон «Знімок», та яка між ними взаємодія?

Класами, що входять до шаблону, є: Originator (об'єкт, стан якого зберігається, має методи CreateMemento() та SetMemento()) , Memento (об'єкт, який зберігає стан і має доступ до нього лише Originator) , та Caretaker (об'єкт, який передає та зберігає Memento в системі). Originator створює Memento і може відновлювати свій стан із нього, а Caretaker виступає посередником для зберігання та передачі знімків .

11. Яке призначення шаблону «Декоратор»?

Призначенням шаблону «Декоратор» є динамічне додавання функціональних можливостей (обов'язків) до об'єкта під час виконання програми. Він «обертає» початковий об'єкт (через агрегацію), зберігаючи його функції та додаючи нові дії. Це забезпечує більшу гнучкість, ніж просте спадкування.

12. Нарисуйте структуру шаблону «Декоратор».



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

Основні класи: Component (інтерфейс, що визначає загальні операції) , ConcreteComponent (конкретна реалізація компонента) , Decorator (абстрактний клас-обгортка, що містить посилання на Component і делегує виклики) , та ConcreteDecorator (конкретна реалізація, що додає обов'язки). Decorator містить посилання на Component і викликає його операцію, додаючи свою логіку .

14. Які є обмеження використання шаблону «декоратор»?

Обмеження використання шаблону «Декоратор» полягають у тому, що він може призвести до великої кількості крихітних класів та ускладнює конфігурування об'єктів, які загорнуті у декілька обгорток одночасно.

Вихідний код:
https://github.com/eismonta/TRPZ_2025/tree/195226649091bc889fa9b9f14edb4f38d71d66ec/image-editor

Висновок: У ході виконання лабораторної роботи я дослідила структуру шаблонів «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator». Також було успішно реалізовано патерн Memento для забезпечення можливості скасування дій користувача. Це дозволило відокремити механізм збереження історії від основної бізнес-логіки редактора.