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
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 «Adapter», «Builder», «Command», «Chain of responsibility», «Prototype» 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 «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). Проблема: Ви розробляєте редактор рівнів для 2D гри на основі спрайтів. В панелі інструментів ви маєте багато кнопок для різних елементів, які можна розташовувати на екрані, такі як сходи, стіни, підлога, оздоблення та інші. Ці елементи у вас об'єднані в ієрархію з базовим класом GameObject. Під кожен елемент можна зробити свій тип кнопки, але тоді ми отримаємо паралельну ієрархію кнопок і при додаванні

нового типу ігрового об’єкту потрібно буде додавати і новий тип кнопки. Рішення: Використовуючи патерн прототип, додаємо до базового об'єкта GameObject метод Clone(), а кнопки будуть зберігати посилання на об'єкт базового типу GameObject. При натисканні на кнопку об'єкт який потрібно добавити на ігровому полі отримуємо не створенням нового, а клонуванням прототипу, який прив'язаний до кнопки. Таким чином, коли ми будемо додавати нові типи ігрових об'єктів, то логіка роботи з кнопками не буде змінюватися, тому що не має прив'язки до конкретних типів. Також слід відзначити, що при копіюванні об'єкту, навіть приватні поля будуть скопійовані, тому що реалізація методу копіювання знаходиться в цьому класі, що копіюється і таким чином є доступ для копіювання і до відкритих і до закритих полів.

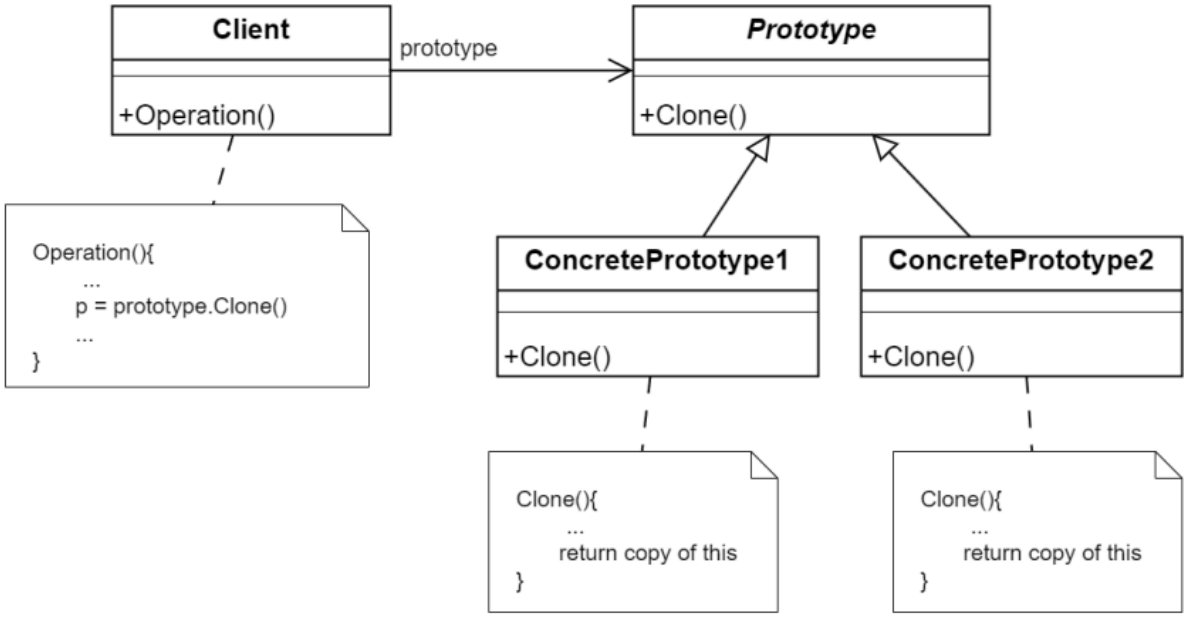


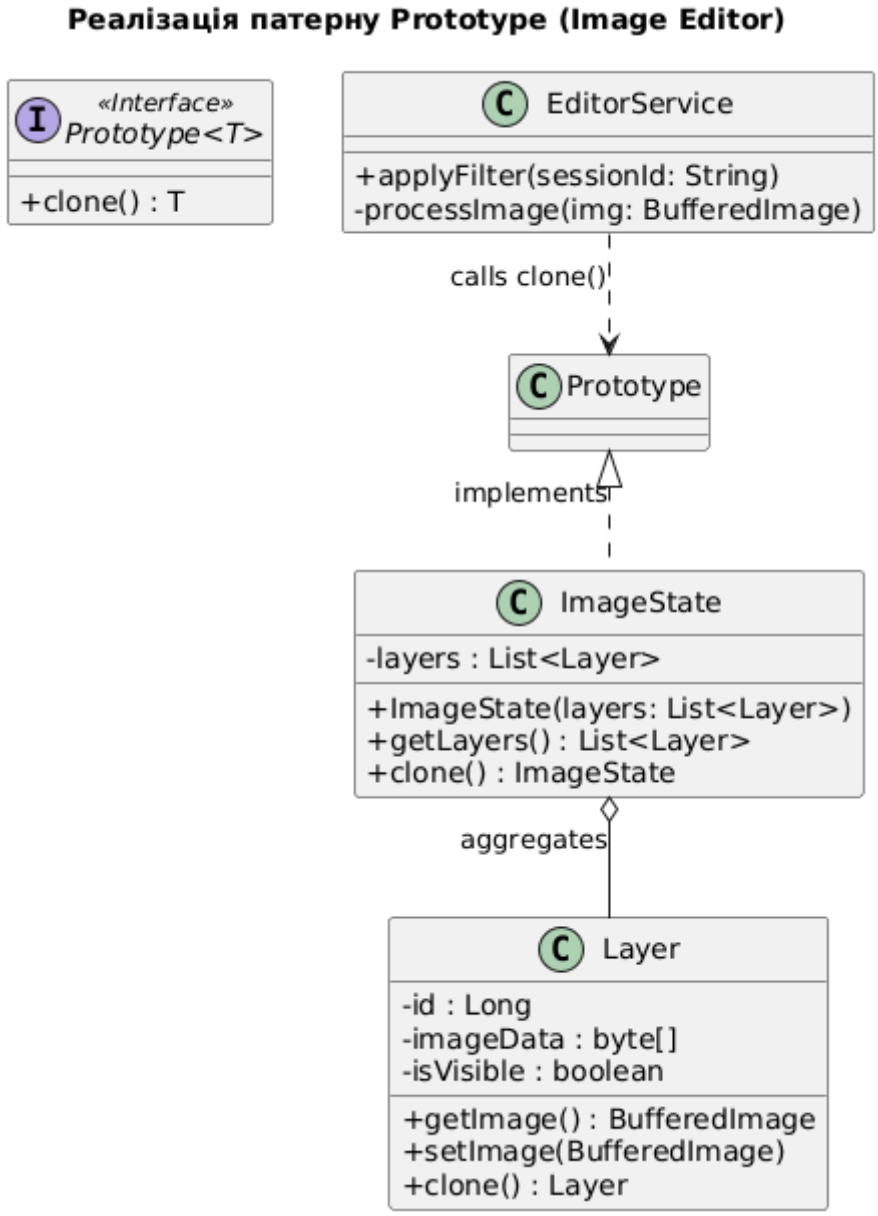
Рисунок 5.5. Структура патерну «Прототип»

Переваги та недоліки: + За рахунок клонування складних об’єктів замість їх створення, підвищується продуктивність. + Різні варіації об’єктів можна отримувати за рахунок клонування, а не розширення ієрархії класів. + Вища гнучкість, тому що клоновані об’єкти можна модифікувати незалежно, не впливаючи на об’єкт з якого була зроблена копія. - Реалізація глибокого клонування досить проблематична, коли об’єкт що клонується містить складну внутрішню структуру та посилання на інші об’єкти. - Надмірне використання

патерну Прототип може привести до ускладнення коду та проблем із супроводом такого коду.

Хід роботи:

Діаграма класів:



Ця діаграма ілюструє механізм створення знімка стану редактора:

1. Prototype<T> (Інтерфейс): Оголошує метод clone(). Це контракт, який гарантує, що об'єкт вміє створювати власну копію.
2. ImageState (Concrete Prototype): Реалізує інтерфейс Prototype.
   - Це "контейнер", який тримає поточний набір шарів.
   - У методі clone() він реалізує логіку глибокого копіювання (Deep Copy). Він не просто копіює посилання на список, а створює нові

екземпляри кожного шару (Layer). Це критично важливо, щоб малювання на копії не змінювало оригінал.

3. Layer: Допоміжний клас, який також бере участь у клонуванні (копіюється масив байтів imageData).
4. EditorService (Client): Перед тим як застосувати фільтр (наприклад, "Grayscale"), сервіс викликає currentState.clone(). Операції виконуються над клоном, а оригінал залишається незмінним (або зберігається в історію для функції Undo).

Опис реалізації патерну «Prototype»

У даному проекті патерн Prototype використовується для того, щоб безпечно створювати копії зображення перед його редагуванням. Головна ідея полягає в тому, що об'єкт сам вміє себе клонувати завдяки реалізації спеціального інтерфейсу з методом clone. Коли користувач хоче застосувати фільтр, програма не змінює оригінальну картинку, а створює її повний дублікат у пам'яті.

Особливістю реалізації є так зване глибоке копіювання. Це означає, що система створює не просто посилання на старий об'єкт, а повністю новий незалежний набір даних для кожного шару малюнка. Завдяки цьому будь-які зміни, зроблені на копії, ніяк не впливають на оригінал, що дозволяє уникнути помилок при обробці графіки та зберігати попередні стани проекту.

Фрагменти програмного коду:

Інтерфейс Prototype оголошує загальний інтерфейс для всіх об'єктів, що підтримують клонування. Це дозволяє створювати копії об'єктів, не прив'язуючись до їхніх конкретних класів.

Prototype.java

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


public interface Prototype<T> {
    T clone();
}
```

Клас ImageState реалізує інтерфейс Prototype. У методі clone() відбувається глибоке копіювання :створюється нова група шарів, і для кожного шару створюється новий екземпляр зображення. Це гарантує, що зміни в копії не вплинуть на оригінал.

ImageState.java

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

import ia32.eismont.image_editor_server.entity.Layer;
import ia32.eismont.image_editor_server.entity.LayerGroup;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;

public class ImageState implements Prototype<ImageState> {

    private LayerGroup layerGroup;

    public ImageState(LayerGroup layerGroup) {
        this.layerGroup = layerGroup;
    }

    public LayerGroup getLayerGroup() {
        return layerGroup;
    }

    @Override
    public ImageState clone() {
        LayerGroup newGroup = new LayerGroup();

        for (Layer layer : layerGroup.getLayers()) {
            BufferedImage original = layer.getImage();

            BufferedImage copy = new BufferedImage(
                original.getWidth(), original.getHeight(),
BufferedImage.TYPE_INT_ARGB
```

```
            );
            Graphics2D g = copy.createGraphics();
            g.drawImage(original, 0, 0, null);
            g.dispose();

            newGroup.addLayer(new Layer(copy));
        }

        return new ImageState(newGroup);
    }
}
```

У бізнес-логіці перед застосуванням фільтру ми викликаємо метод clone(). Усі маніпуляції з зображенням відбуваються вже над створеним клоном (nextState), що забезпечує безпеку даних.

Фрагмент коду **EditorService.java**

```
public String applyFilter(String sessionId, String filterType) {
    ImageState currentState = sessionStates.get(sessionId);
    if (currentState == null) return null;
    ImageState nextState = currentState.clone();

    BufferedImage bg = nextState.getLayerGroup().getLayer(0).getImage();
    processImage(bg, filterType);

    sessionStates.put(sessionId, nextState);
    return imageToBase64(Compositor.merge(nextState.getLayerGroup()));
}
```

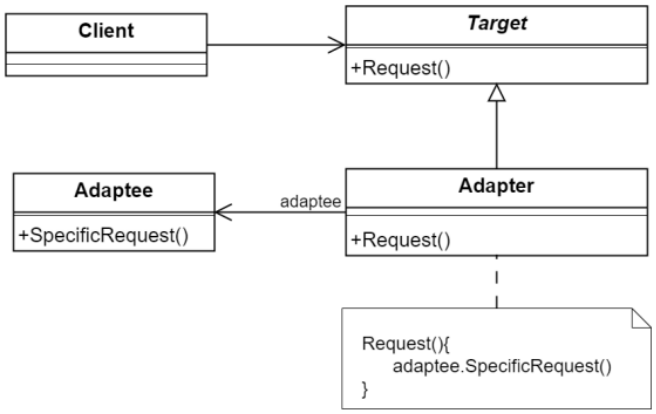
Питання до лабораторної роботи:

1. Яке призначення шаблону «Адаптер»?

Шаблон «Адаптер» використовується для адаптації інтерфейсу одного об'єкта до іншого. Цей патерн дозволяє привести різні бібліотеки з різними

інтерфейсами до одного уніфікованого інтерфейсу, що дає можливість у програмі звертатися до загального інтерфейсу замість написання різних сценаріїв роботи. Адаптери також називаються обгортками або «wrappers».

2. Нарисуйте структуру шаблону «Адаптер».



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

До складу шаблону входять класи Client, Target, Adaptee та Adapter . Взаємодія полягає в тому, що адаптер має посилання на об'єкт компонента (Adaptee) та реалізує інтерфейс (Target), викликаючи методи з компонента, який він адаптує. Таким чином, адаптер обгортає специфічний компонент, і весь клієнтський код працює з адаптерами через загальний інтерфейс.

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

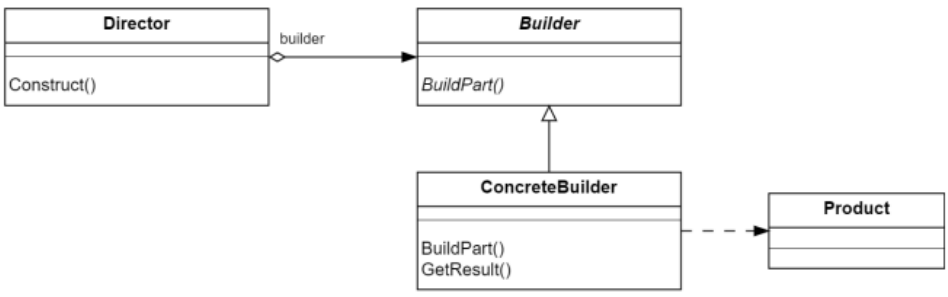
класів?

Адаптер об’єктів використовує компіляцію і зберігає посилання на екземпляр класу Adaptee і делегує йому виклики. Адаптер класів використовує спадкування.

5. Яке призначення шаблону «Будівельник»?

Шаблон «Будівельник» використовується для відділення процесу створення об'єкта від його представлення. Це доречно, коли об'єкт має складний процес створення або повинен мати декілька різних форм створення.

6. Нарисуйте структуру шаблону «Будівельник».



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

До шаблону входять класи Director, Builder, ConcreteBuilder та Product . Director керує процесом, викликаючи метод Construct, а ConcreteBuilder реалізує методи BuildPart для створення частин продукту та GetResult для повернення готового об'єкта.

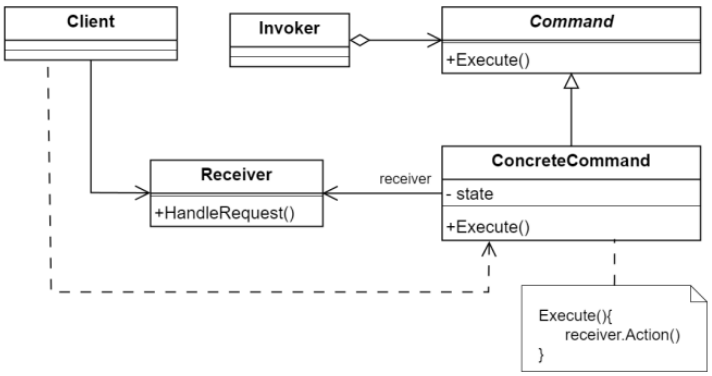
8. У яких випадках варто застосовувати шаблон «Будівельник»"?

Цей шаблон варто застосовувати у випадках, коли об'єкт має складний процес створення, наприклад, веб-сторінка як елемент повної відповіді веб-сервера. Також він доречний, коли об'єкт повинен мати декілька різних форм створення, наприклад, при конвертації тексту з одного формату в інший.

9. Яке призначення шаблону «Команда»?

Шаблон «Команда» перетворює звичайний виклик методу в клас, завдяки чому дії в системі стають повноправними об'єктами. Це дозволяє параметризувати клієнтів різними запитами та підтримувати скасування операцій.

10. Нарисуйте структуру шаблону «Команда».



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

У шаблон входять класи Client, Invoker, Command, Receiver та ConcreteCommand . Invoker викликає метод Execute у Command, а ConcreteCommand перенаправляє запит до Receiver, викликаючи метод Action .

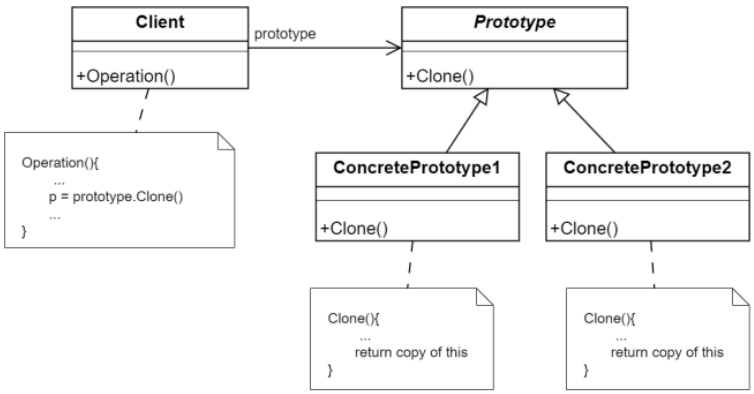
12. Розкажіть як працює шаблон «Команда».

Об'єкт команда сам по собі не виконує ніяких фактичних дій, окрім перенаправлення запиту одержувачеві. Команди можуть зберігати дані для підтримки додаткових функцій, таких як скасування або логування. Клієнт створює екземпляр команди та передає її ініціатору, який згодом викликає виконання команди.

13. Яке призначення шаблону «Прототип»?

Шаблон «Прототип» використовується для створення об'єктів за шаблоном шляхом копіювання існуючого об'єкта, який називається прототипом. Це дозволяє мінімізувати кількість змін до об'єкта та прибирає необхідність знати подробиці створення об'єкта.

14. Нарисуйте структуру шаблону «Прототип».



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

До шаблону входять класи Client, Prototype та конкретні реалізації ConcretePrototype1 і ConcretePrototype2 . Клієнт виконує операцію, викликаючи метод Clone у прототипу, який повертає копію самого себе.

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

Прикладом використання є процес підписання документа, який проходить від співробітника через менеджера до начальника.

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

Висновок:

Під час виконання лабораторної роботи №5 я ознайомився з призначенням та структурою патернів проектування «Adapter», «Builder», «Command», «Chain of Responsibility» та «Prototype». На практиці було реалізовано патерн Prototype (Прототип) для системи графічного редактора. Це дозволило створити механізм глибокого клонування (Deep Copy) стану проекту (клас ImageState та масиви байтів зображення), завдяки чому фільтри накладаються на копію зображення, зберігаючи оригінал незмінним. В результаті я набув навичок використання патернів для управління складними об'єктами та підвищення гнучкості архітектури програмного забезпечення.