Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота №6

із дисципліни «Технології розроблення програмного забезпечення»

Тема: «ШАБЛОНИ «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»»

Перевірив:
Мягкий М.Ю

Виконала:
Студентка групи ІА-24
Ганжа Х.М

Варіант:

..4 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.

Завдання.

• Ознайомитися з короткими теоретичними відомостями.

• Реалізувати частину функціонала робочої програми у вигляді класів і їх взаємодій для досягнення конкретних функціональних можливостей.

• Застосування одного з даних шаблонів при реалізації програми.

Виконання:

https://github.com/Hrystynkkaa/trpz/tree/main/%D0%BA%D0%BE%D0%B4/GraphicEditor

Інтерфейс ImageEffect:

Інтерфейс для всіх ефектів, що можуть бути застосовані до зображень. Він визначає метод applyEffect(BufferedImage image), що приймає зображення і повертає його після застосування ефекту.

```
public interface ImageEffect {
    BufferedImage applyEffect(BufferedImage image);
}
```

InfraredEffect: клас для застосування інфрачервоного ефекту до зображення. Ефект збільшує інтенсивність червоного кольору, зменшує зелену і синю компоненти, що дає унікальний візуальний ефект.

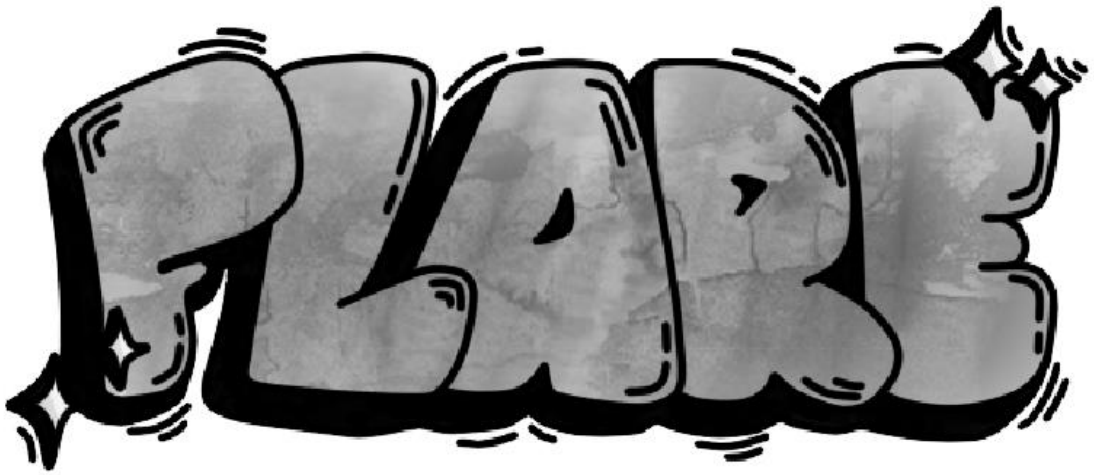
```
public class InfraredEffect implements ImageEffect {
    @Override
    public BufferedImage applyEffect(BufferedImage image) {
        int width = image.getWidth();
        int height = image.getHeight();

        for (int y = 0; y < height; y++) {
            for (int x = 0; x < width; x++) {
```

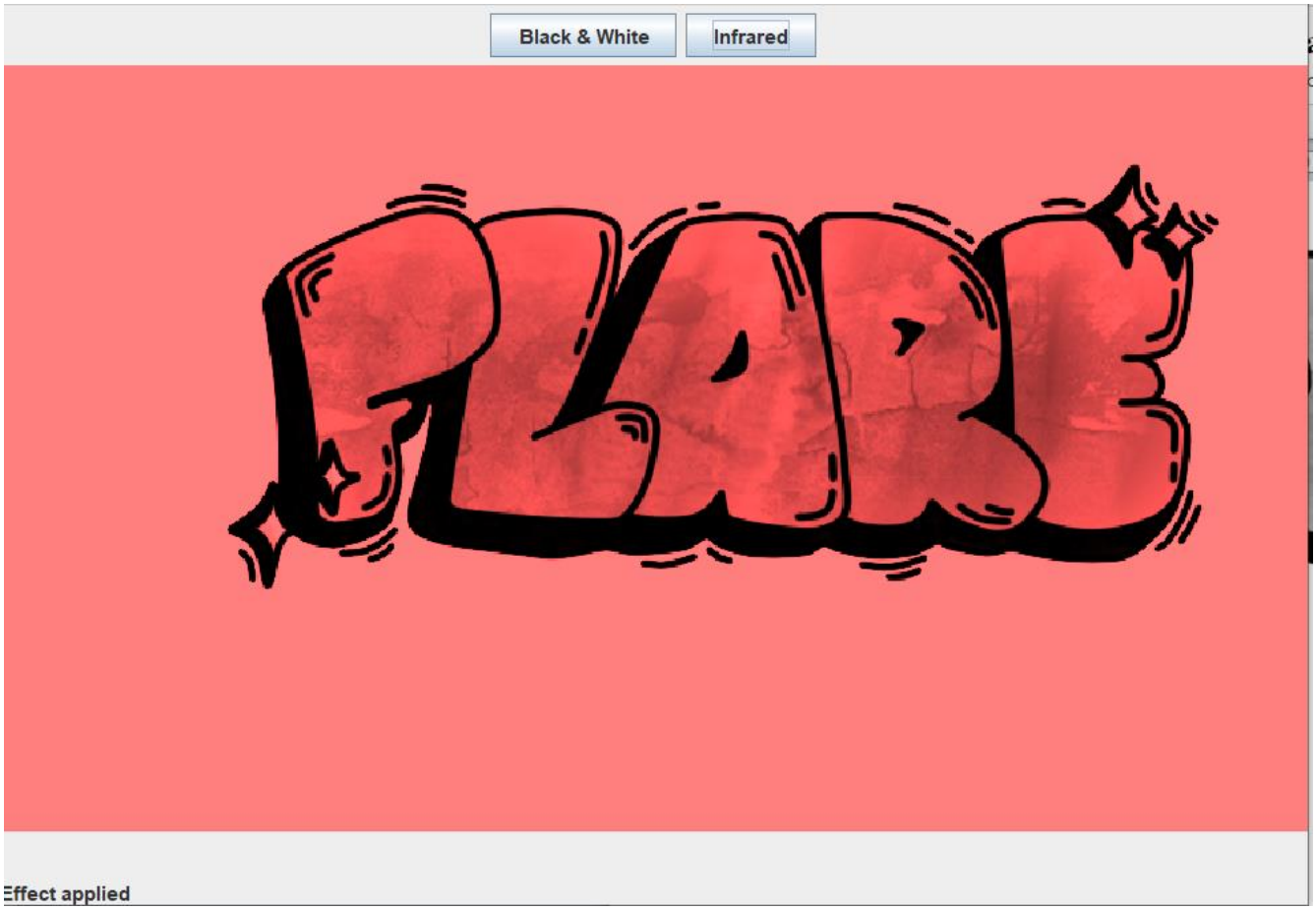
```
                // Get the RGB value of the pixel
                int rgb = image.getRGB(x, y);

                int red = (rgb >> 16) & 0xFF;
                int green = (rgb >> 8) & 0xFF;
                int blue = rgb & 0xFF;

                red = Math.min((int)(red * 1.5), 255);
                green = Math.max((int)(green * 0.5), 0);
                blue = Math.max((int)(blue * 0.5), 0);

                int newRgb = (red << 16) | (green << 8) | blue;

                image.setRGB(x, y, newRgb);
            }
        }
        return image;
    }
}
```

BlackAndWhiteEffect: клас для перетворення зображення в чорно-біле. Ефект обчислює яскравість кожного пікселя і замінює кольорові значення на відповідний відтінок сірого.

```
public class BlackAndWhiteEffect implements ImageEffect {
    @Override
    public BufferedImage applyEffect(BufferedImage image) {
        int width = image.getWidth();
        int height = image.getHeight();

        for (int y = 0; y < height; y++) {
            for (int x = 0; x < width; x++) {
                // Get the RGB value of the pixel
                int rgb = image.getRGB(x, y);

                int red = (rgb >> 16) & 0xFF;
                int green = (rgb >> 8) & 0xFF;
                int blue = rgb & 0xFF;

                int luminance = (int)(0.3 * red + 0.59 * green + 0.11 * blue);

                int gray = (luminance << 16) | (luminance << 8) | luminance;

                image.setRGB(x, y, gray);
            }
        }
        return image;
    }
}
```

Результат:





Висновок: Було додано можливість динамічного застосування різноманітних візуальних ефектів до зображень без зміни їхньої базової структури. Це досягається завдяки створенню декораторів для ефектів, що дозволяє додавати нову поведінку до об'єктів зображень, не змінюючи їх основної логіки.

Кожен ефект (наприклад, чорно-білий або інфрачервоний) реалізує інтерфейс ImageEffect, що дозволяє гнучко додавати нові ефекти до існуючих зображень. Це

дозволяє комбінувати кілька ефектів без необхідності модифікації базових класів або створення нових класів для кожного можливого поєднання.