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 №5

із дисципліни «Технології розроблення програмного забезпечення»

Тема: «ШАБЛОНИ «ADAPTER», «BUILDER», «COMMAND», «CHAIN OF RESPONSIBILITY», «PROTOTYPE»

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.

1. Ознайомитися з короткими теоретичними відомостями.

2. Реалізувати частину функціоналу робочої програми у вигляді класів та їх взаємодій для досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми.

Виконання:

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

Інтерфейс Prototype<T>:

Цей інтерфейс визначає метод clone(), який дозволяє створювати копії об'єктів. Це основа для реалізації патерну Prototype, що допомагає уникати повторного створення складних об'єктів, натомість створюючи їх копії.

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

Клас Element:

Абстрактний клас для графічних елементів, таких як фігури (прямокутники, кола). Він реалізує інтерфейс Prototype та має базову структуру для збереження координат, кольору та абстрактного методу для малювання елементу.

Реалізовано метод clone(), який дозволяє створювати нові об'єкти типу Element на основі вже існуючого.

```
public abstract class Element implements Prototype<Element> {
    protected int x;
    protected int y;
    protected Color color;

    public Element(int x, int y, Color color) {
        this.x = x;
        this.y = y;
        this.color = color;
    }

    public abstract void draw(Graphics g);

    @Override
    public abstract Element clone();

    public int getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }

    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }

    public Color getColor() {
        return color;
    }

    public void setColor(Color color) {
        this.color = color;
    }
}
```

Клас GraphicElement:

Цей клас є конкретною реалізацією класу Element. Він визначає типи графічних фігур і реалізує методи для їх малювання та копіювання.

Методи малювання працюють через об'єкт Graphics, що дозволяє на екрані відображати відповідні фігури, а метод clone() дозволяє створювати копії цих елементів.

```
public class GraphicElement extends Element {
    private String figure;

    public GraphicElement(int x, int y, String figure, Color color) {
        super(x, y, color);
        this.figure = figure;
    }

    @Override
    public void draw(Graphics g) {
        g.setColor(color);
        switch (figure) {
            case "rectangle":
```

```
                g.fillRect(x, y, 100, 50);
                break;
            case "circle":
                g.fillOval(x, y, 50, 50);
                break;
        }
    }

    @Override
    public GraphicElement clone() {
        return new GraphicElement(x, y, figure, color);
    }

    public String getFigure() {
        return figure;
    }

    public void setFigure(String figure) {
        this.figure = figure;
    }
}
```

Клас Layer:

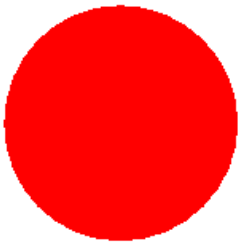
Клас Layer представляє шар зображення, що містить графічний контент і може бути видимим чи прихованим.

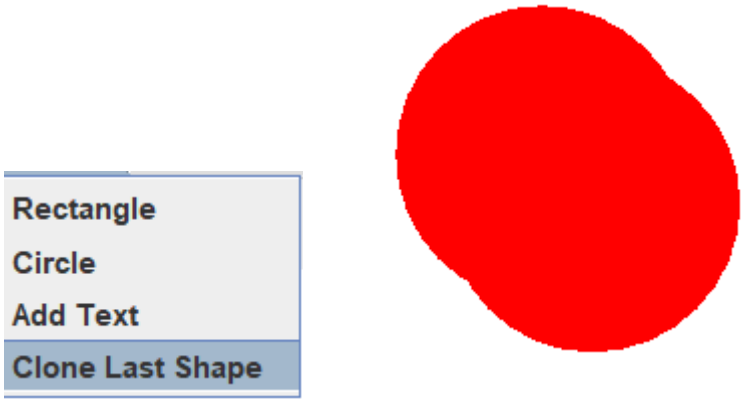
Шар також реалізує метод clone(), що дозволяє створювати його копії з усім вмістом. Це дозволяє користувачам працювати з декількома копіями шарів, що корисно при роботі з різними етапами редагування.

```
public class Layer implements Cloneable {
    private long id;
    private String name;
    private BufferedImage content;
    private boolean visible;

    public Layer(long id) {
        this.id = id;
        this.name = "Layer " + id;
        this.visible = true;
        this.content = new BufferedImage(800, 600, BufferedImage.TYPE_INT_ARGB);
    }

    public long getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public BufferedImage getContent() {
        return content;
    }

    public void setContent(BufferedImage content) {
```

```
        this.content = content;
    }

    public boolean isVisible() {
        return visible;
    }

    public void setVisible(boolean visible) {
        this.visible = visible;
    }

    @Override
    public Layer clone() {
        try {
            Layer clone = (Layer) super.clone();
            clone.id = System.currentTimeMillis();
            clone.content = new BufferedImage(content.getWidth(), content.getHeight(),
content.getType());
            Graphics g = clone.content.getGraphics();
            g.drawImage(content, 0, 0, null);
            g.dispose();
            return clone;
        } catch (CloneNotSupportedException e) {
            throw new RuntimeException("Layer cloning failed", e);
        }
    }
}
```

Результат:






```
Layer saved: 1733320499926
Layer loaded: 1733320499926
Layer saved: 1733320506488
```

Висновок: У процесі виконання лабораторної роботи було реалізовано використання шаблону проектування Prototype для створення графічного редактора, що дозволяє ефективно керувати об'єктами графічних елементів і шарів. Цей підхід дає змогу створювати копії об'єктів без необхідності їх повного відтворення, що особливо корисно для швидкої маніпуляції елементами редактора, таких як графічні фігури та шари.

Патерн Prototype дозволяє користувачам легко дублювати об'єкти (наприклад, графічні елементи та шари) без необхідності створювати їх з нуля. Це важливо, оскільки дає змогу ефективно керувати складними графічними об'єктами, не витрачаючи зайвих ресурсів на їх повторне створення. Клас Layer з методами копіювання шарів надає гнучкість у роботі з різними варіантами зображень, які можуть бути редаговані та змінювані.