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 №4

із дисципліни «Технології розроблення програмного забезпечення»

Тема: «ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»

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 їхньої взаємодії 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

Клас RealImage для роботи з реальним зображенням :

RealImage відповідає за фактичне завантаження, збереження та відображення зображення.

```
public class RealImage extends Image {
    private String filePath;
    private BufferedImage bufferedImage;
    private BufferedImage originalBufferedImage; // Для збереження початкового стану

    public RealImage(long id, String format, String filePath) {
        super(id, format);
        this.filePath = filePath;
        if (!filePath.isEmpty()) {
            loadImage();
        }
    }

    // Конструктор для створення порожнього зображення
    public RealImage(long id, String format, int width, int height) {
        super(id, format);
        bufferedImage = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
```

```
        Graphics2D g2d = bufferedImage.createGraphics();
        g2d.fillRect(0, 0, width, height);
        g2d.dispose();
        originalBufferedImage = deepCopy(bufferedImage); // Зберігаємо початковий стан
    }

    private void loadImage() {
        try {
            File file = new File(filePath);
            if (!file.exists()) {
                System.err.println("Файл не знайдений: " + filePath);
                return;
            }
            bufferedImage = ImageIO.read(file);
            if (bufferedImage == null) {
                System.err.println("Не вдалося завантажити зображення: " + filePath);
            } else {
                System.out.println("Зображення завантажене успішно: " + filePath);
                originalBufferedImage = deepCopy(bufferedImage); // Зберігаємо початковий
стан
            }
        } catch (IOException e) {
            System.err.println("Помилка завантаження: " + e.getMessage());
        }
    }

    public void saveImage(String path) {
        try {
            if (bufferedImage == null) {
                throw new IllegalArgumentException("Зображення порожнє");
            }
            File outputFile = new File(path);
            ImageIO.write(bufferedImage, "PNG", outputFile);
            System.out.println("Зображення збережено: " + path);
        } catch (IOException | IllegalArgumentException e) {
            System.err.println("Помилка збереження: " + e.getMessage());
        }
    }

    public BufferedImage getBufferedImage() {
        return bufferedImage;
    }

    public void setBufferedImage(BufferedImage bufferedImage) {
        this.bufferedImage = bufferedImage;
    }

    @Override
    public void display() {
        System.out.println("Displaying Real Image ID: " + getId());
        if (bufferedImage == null) {
            System.err.println("Зображення не завантажено або порожнє.");
            return;
        }

        JFrame frame = new JFrame("Display Image ID: " + getId());
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setSize(bufferedImage.getWidth(), bufferedImage.getHeight());
        JLabel imageLabel = new JLabel(new ImageIcon(bufferedImage));
        frame.add(imageLabel);
        frame.pack();
        frame.setVisible(true);
    }

    public void reload() {
        if (originalBufferedImage != null) {
            bufferedImage = deepCopy(originalBufferedImage); // Відновлюємо з оригіналу
            System.out.println("Зображення відновлено до початкового стану.");
```

```
        } else {
            System.err.println("Початковий стан зображення відсутній.");
        }
    }

    // Метод для створення копії BufferedImage
    private BufferedImage deepCopy(BufferedImage source) {
        BufferedImage copy = new BufferedImage(source.getWidth(), source.getHeight(),
source.getType());
        Graphics2D g = copy.createGraphics();
        g.drawImage(source, 0, 0, null);
        g.dispose();
        return copy;
    }
}
```

Клас ImageProxy для лінивого завантаження

ImageProxy забезпечує завантаження реального зображення лише за потреби

```
package com.graphiceditor.model;

public class ImageProxy extends Image {

    private RealImage realImage;
    private String filePath;

    public ImageProxy(long id, String format, String filePath) {
        super(id, format);
        this.filePath = filePath;
    }

    @Override
    public void display() {
        if (realImage == null) {
            realImage = new RealImage(getId(), getFormat(), filePath);
        }
        realImage.display();
    }
}
```

ImageService (інтерфейс):

Використовується для реалізації операцій із зображеннями.

```
public interface ImageService {
    void saveImage(Image image);
    Image loadImage(long id);
    void deleteImage(long id);
}
```

ImageServiceImpl:

Основна реалізація сервісу для бази даних.

```
public class ImageServiceImpl implements ImageService {

    private Map<Long, Image> imageDatabase = new HashMap<>();

    @Override
    public void saveImage(Image image) {
        imageDatabase.put(image.getId(), image);
        System.out.println("Image saved to the database: " + image.getId());
```

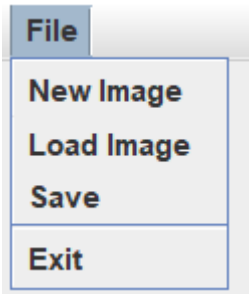
```
    }

    @Override
    public Image loadImage(long id) {
        if (imageDatabase.containsKey(id)) {
            return imageDatabase.get(id);
        }
        System.out.println("Image not found in the database: " + id);
        return null;
    }

    @Override
    public void deleteImage(long id) {
        imageDatabase.remove(id);
        System.out.println("Image deleted from the database: " + id);
    }
}
```

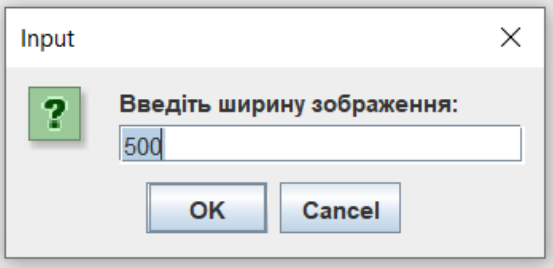
ImageStorageProxy:

Проксі-клас, який перевіряє файли в системі перед викликом методів ImageServiceImpl

```
public class ImageStorageProxy implements ImageService {

    private ImageServiceImpl imageServiceImpl;
    private String storagePath;

    public ImageStorageProxy(String storagePath) {
        this.imageServiceImpl = new ImageServiceImpl();
        this.storagePath = storagePath;
    }

    @Override
    public void saveImage(Image image) {
        if (image instanceof RealImage) {
            RealImage realImage = (RealImage) image;
            String imagePath = storagePath + "/" + image.getId() + "." +
image.getFormat().toLowerCase();

            try {
                // Перевірка на наявність BufferedImage перед збереженням
                if (realImage.getBufferedImage() != null) {
                    // Збереження зображення на диск
                    File outputFile = new File(imagePath);
                    ImageIO.write(realImage.getBufferedImage(),
image.getFormat().toLowerCase(), outputFile);
                    System.out.println("Image saved to file system: " + imagePath);

                    // Після збереження викликаємо saveImage в ImageServiceImpl
                    imageServiceImpl.saveImage(image);
                } else {
                    System.err.println("Cannot save: BufferedImage is null.");
                }
            } catch (IOException e) {
                System.err.println("Error saving image: " + e.getMessage());
            }
        }
    }




    @Override
    public Image loadImage(long id) {
        for (String ext : new String[]{"png", "jpg", "bmp"}) {
            String imagePath = storagePath + "/" + id + "." + ext;
```

```
            File imageFile = new File(imagePath);

            // Логування шляху до файлу
            System.out.println("Looking for image at path: " + imagePath);

            if (imageFile.exists()) {
                System.out.println("Image found: " + imagePath);
                return new RealImage(id, ext.toUpperCase(), imagePath);
            }
        }
        System.out.println("Image not found in file system for ID: " + id);
        return null;
    }

    @Override
    public void deleteImage(long id) {
        String imagePath = storagePath + "/" + id + ".png";
        File imageFile = new File(imagePath);

        if (imageFile.exists()) {
            imageFile.delete();
            System.out.println("Image deleted from file system: " + imagePath);
        } else {
            System.out.println("Image not found in file system for deletion: " +
imagePath);
        }
        imageServiceImpl.deleteImage(id);
    }
}
```





Зображення завантажено: D:\flare\ФЛЕЙР.png

```
Зображення завантажене успішно: D:\flare\ФЛЕЙР.png
Image saved to file system: D://1733319209462.png
Image saved to the database: 1733319209462
```

```
Зображення збережено: D:\kj
```

Висновок: У результаті виконання лабораторної роботи було реалізовано графічний редактор з використанням шаблону проектування Proxy. Головною метою було забезпечення лінивого завантаження зображень, що дозволяє зберігати ресурси та завантажувати зображення лише за необхідності. В межах роботи було розроблено кілька основних компонентів, які взаємодіють між собою для досягнення зазначених функціональних можливостей.

Клас RealImage: Цей клас відповідає за фактичне завантаження, збереження та відображення зображень. Він містить методи для завантаження зображень з файлу, збереження на диск та відображення у графічному інтерфейсі.

Клас ImageProxy: Цей клас реалізує патерн Proxy. Він забезпечує ліниве завантаження реального зображення лише при необхідності (коли викликається метод display()).Це

дозволяє зменшити витрати пам'яті та покращити ефективність програми, особливо коли працюємо з великими або численними зображеннями.

Таким чином, зображення не завантажується відразу, а лише тоді, коли це дійсно потрібно, що дозволяє заощаджувати ресурси і покращити ефективність роботи програми.

Інтерфейс ImageService та реалізація ImageServiceImpl:Цей компонент надає інтерфейс для роботи з зображеннями, зокрема для їх збереження, завантаження та видалення. Реалізація ImageServiceImpl використовує колекцію для зберігання зображень та забезпечує доступ до них.

Проксі-клас ImageStorageProxy:Цей клас реалізує додаткову функціональність для зберігання та завантаження зображень з файлової системи. Проксі перевіряє наявність зображень у файловій системі перед виконанням операцій з ними, забезпечуючи додаткову оптимізацію та контроль доступу до зображень.