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 № 7

з дисципліни «Технології розроблення програмного забезпечення»

Тема: “ШАБЛОН «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD»”

Варіант №5

Виконала:
студентка групи ІА-23
Архип’юк Катерина

Перевірив:
Мягкий Михайло Юрійович

Київ 2024

Зміст

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.

Тема (Варіант №5)

..5 Аудіо редактор (singleton, adapter, observer, mediator, composite, client-server)

Аудіо редактор повинен володіти наступним функціоналом: представлення аудіо даних будь-якого формату в WAVE-формі, вибір і подальші операції копіювання / вставки / вирізання / деформації по сегменту аудіозапису, можливість роботи з декількома звуковими доріжками, кодування в найбільш поширених форматах (ogg, flac, mp3).

Хід роботи

1. Короткі теоретичні відомості

Принципи проєктування:

Don't Repeat Yourself (DRY) наголошує на важливості уникати повторень в коді. Повторюваний код ускладнює підтримку, оскільки будь-яка зміна потребує коригування в кількох місцях, що підвищує ймовірність помилок. Найкраще уникати дублювання, використовуючи методи, класи або інтерфейси для спільних функцій.

Keep it simple, stupid! (KISS) зводиться до того, що система повинна бути простою. Система, побудована з невеликих простих компонентів, є більш надійною, зрозумілою та зручною для підтримки. Простота дозволяє уникнути непотрібних складнощів, що спрощує розробку та тестування.

You only load it once! (YOLO) рекомендує ініціалізувати та конфігурувати змінні один раз при запуску програми, щоб уникнути надмірного навантаження на систему через повторні зчитування даних. Це сприяє покращенню продуктивності та зменшенню витрат часу на обробку даних.

Принцип Парето вказує на те, що в багатьох випадках 80% результату досягається завдяки лише 20% зусиль. Це допомагає фокусуватися на важливих аспектах, таких як оптимізація найбільш значущих частин програми, що дає найбільший ефект при мінімальних витратах часу та ресурсів.

You ain’t gonna need it наголошує на важливості уникати надмірної складності та створення загальних рішень для випадків, які, ймовірно, не виникнуть. Система має реалізовувати тільки ту функціональність, яка справді необхідна, без додаткової гнучкості, яка може уповільнити процес розробки та заплутати код.

Шаблон Mediator використовується для зменшення складності системи шляхом централізації взаємодії між об'єктами. Замість того, щоб об'єкти безпосередньо взаємодіяли один з одним, всі комунікації проходять через медіатора. Це дозволяє зменшити зв'язність (coupling) між об'єктами і полегшує підтримку та розширення системи.

Шаблон Facade надає спрощений інтерфейс для роботи з підсистемами, приховуючи складність внутрішньої реалізації. Це дозволяє знизити кількість взаємодій між користувачем і підсистемами, полегшуючи використання складних систем або наборів класів.

Шаблон Bridge дозволяє розділити абстракцію від її реалізації, що дозволяє їх розвивати незалежно один від одного. Цей шаблон використовується для зменшення кількості класів і підвищення гнучкості системи, особливо коли необхідно розширювати абстракцію або реалізацію без впливу на іншу частину.

Шаблон Template Method визначає структуру алгоритму в методі батьківського класу, дозволяючи підкласам реалізовувати певні кроки алгоритму. Це дозволяє спільно використовувати структуру алгоритму, але залишати гнучкість для зміни конкретних кроків підкласами, забезпечуючи збереження основної логіки в базовому класі.

2. Шаблон Mediator

2.1 Структура та обгрунтування вибору

Ознайомившись з теоретичними матеріалами, було прийнято рішення реалізувати шаблон Mediator. Застосунок складається з багатьох елементів графічного інтерфейсу, робити між ними залежності всередині один одного – нераціонально. Mediator дає змогу зменшити зв’язність елементів між собою, отже значно облегшить розробку. Основна ідея шаблону — централізувати управління взаємодією між компонентами через один об'єкт-посередник. Загальну структуру графічно описано на Рисунку 1.

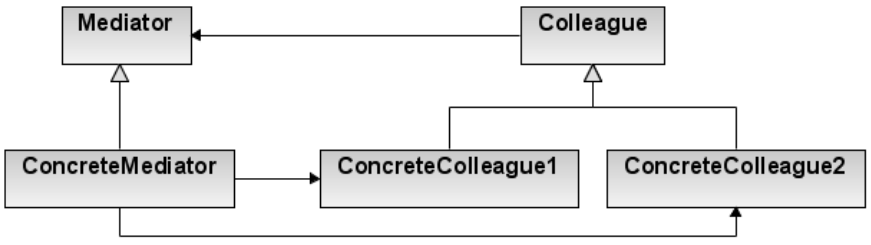

Рисунок 1. Загальна структура шаблону Mediator

Компоненти — це різнорідні об’єкти, що містять бізнес-логіку програми. Посередник визначає інтерфейс для обміну інформацією з компонентами. Конкретний посередник містить код взаємодії кількох компонентів між собою. Компоненти не повинні спілкуватися один з одним безпосередньо. Якщо в компоненті відбувається важлива подія, він повинен повідомити свого посередника, а той сам вирішить, чи стосується подія інших компонентів, і чи треба їх сповістити.

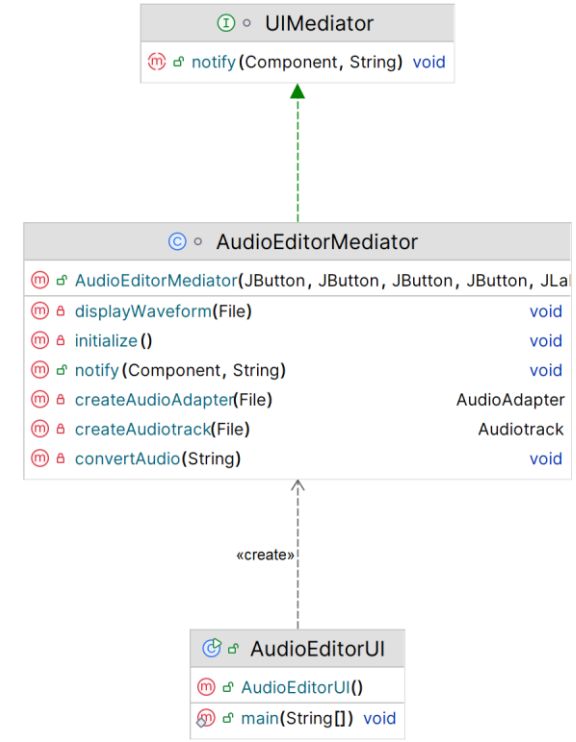

Рисунок 2. Структура реалізованого шаблону Mediator

Інтерфейс UIMediator визначає метод notify(Component sender, String event) для зв'язку між компонентами.

Конкретний Посередник AudioEditorMediator реалізує інтерфейс UIMediator та містить логіку координації між компонентами інтерфейсу.

Головний клас AudioEditorUI створює графічний інтерфейс і ініціалізує AudioEditorMediator.

У цьому випадку компонентами інтерфейсу є: JButton (loadFileButton, convertToMp3Button, тощо), JLabel (fileLabel), JPanel (waveformPanel),Logger.

2.2 Реалізація функціоналу у коді з використанням шаблону

Interface UIMediator

```
package org.example.swing;

import java.awt.*;

interface UIMediator {
    void notify(Component sender, String event);
}
```

Class AudioEditorUI

```
package org.example.swing;

import org.example.database.DatabaseInitializer;
import org.example.logs.Logger;

import javax.swing.*;
import java.awt.*;

public class AudioEditorUI {
    public static void main(String[] args) {
        DatabaseInitializer.initializeDatabase();

        JFrame frame = new JFrame("Audio Editor");
        frame.setSize(600, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(null);

        JButton loadFileButton = new JButton("Load Audio File");
        loadFileButton.setBounds(50, 50, 180, 30);
        frame.add(loadFileButton);

        JLabel fileLabel = new JLabel("No file selected");
        fileLabel.setBounds(250, 50, 300, 30);
        frame.add(fileLabel);

        JButton convertToMp3Button = new JButton("Convert to MP3");
        convertToMp3Button.setBounds(50, 100, 180, 30);
        frame.add(convertToMp3Button);

        JButton convertToOggButton = new JButton("Convert to OGG");
        convertToOggButton.setBounds(50, 150, 180, 30);
        frame.add(convertToOggButton);

        JButton convertToFlacButton = new JButton("Convert to FLAC");
        convertToFlacButton.setBounds(50, 200, 180, 30);
        frame.add(convertToFlacButton);

        JPanel waveformPanel = new JPanel();
        waveformPanel.setBounds(50, 250, 500, 100);
        waveformPanel.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        frame.add(waveformPanel);

        Logger logger = new Logger();

        AudioEditorMediator mediator = new AudioEditorMediator(
                loadFileButton, convertToMp3Button, convertToOggButton,
convertToFlacButton,
```

```
                fileLabel, waveformPanel, logger);

        frame.setVisible(true);
    }
}
```

Class AudioEditorMediator

```
class AudioEditorMediator implements UIMediator {
    private JButton loadFileButton;
    private JButton convertToMp3Button;
    private JButton convertToOggButton;
    private JButton convertToFlacButton;
    private JLabel fileLabel;
    private JPanel waveformPanel;
    private Logger logger;
    private File selectedFile;

    public AudioEditorMediator(JButton loadFileButton, JButton convertToMp3Button,
JButton convertToOggButton, JButton convertToFlacButton,
                               JLabel fileLabel, JPanel waveformPanel, Logger logger)
{
        this.loadFileButton = loadFileButton;
        this.convertToMp3Button = convertToMp3Button;
        this.convertToOggButton = convertToOggButton;
        this.convertToFlacButton = convertToFlacButton;
        this.fileLabel = fileLabel;
        this.waveformPanel = waveformPanel;
        this.logger = logger;

        initialize();
    }
    private void initialize() {
        loadFileButton.addActionListener(e -> notify(loadFileButton, "loadFile"));
        convertToMp3Button.addActionListener(e -> notify(convertToMp3Button,
"convertToMp3"));
        convertToOggButton.addActionListener(e -> notify(convertToOggButton,
"convertToOgg"));
        convertToFlacButton.addActionListener(e -> notify(convertToFlacButton,
"convertToFlac"));
    }

    @Override
    public void notify(Component sender, String event) {
        switch (event) {
            case "loadFile":
                JFileChooser fileChooser = new JFileChooser();
                int returnValue = fileChooser.showOpenDialog(null);
                if (returnValue == JFileChooser.APPROVE_OPTION) {
                    selectedFile = fileChooser.getSelectedFile();
                    fileLabel.setText("Selected: " + selectedFile.getName());
                    displayWaveform(selectedFile);
                    logger.fileOpen();
                }
                break;

            case "convertToMp3":
                convertAudio("mp3");
                break;
```

```
            case "convertToOgg":
                convertAudio("ogg");
                break;

            case "convertToFlac":
                convertAudio("flac");
                break;

            default:
                throw new IllegalArgumentException("Unknown event: " + event);
        }
    }

    private AudioAdapter createAudioAdapter(File file) {
        Audiotrack audiotrack = createAudiotrack(file);
        return new AudioAdapter(audiotrack);
    }

    private Audiotrack createAudiotrack(File file) {
        String fileName = file.getName().toLowerCase();
        if (fileName.endsWith(".mp3")) {
            return new Mp3(file.getAbsolutePath());
        } else if (fileName.endsWith(".ogg")) {
            return new Ogg(file.getAbsolutePath());
        } else if (fileName.endsWith(".flac")) {
            return new Flac(file.getAbsolutePath());
        } else {
            throw new IllegalArgumentException("Unsupported file format: " +
fileName);
        }
    }
}
```

Висновки: При виконанні цієї лабораторної роботи я закріпила навички застосування шаблонів проєктування. Шаблон Mediator використовується для централізації взаємодії між об'єктами, що зменшує їхню зв'язність. Facade спрощує взаємодію з складною системою, приховуючи її складність за простим інтерфейсом. Bridge дозволяє незалежно розвивати абстракції та їх реалізації, що підвищує гнучкість системи. Template Method визначає структуру алгоритму, залишаючи підкласам реалізацію певних кроків, що забезпечує гнучкість та повторне використання коду.

Посилання на репозиторій з кодом проєкту:

https://github.com/KatiaArkhyp/AudioEditor