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 № 8

з дисципліни «Технології розроблення програмного забезпечення»

Тема: “ШАБЛОНИ «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR»”

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

Шаблони роботи з базами даних (БД) при розробці корпоративних додатків застосовуються для масштабних систем з великою кількістю користувачів і пов'язаних програм. Вони забезпечують ефективну організацію доступу до даних при збільшенні обсягу інформації. Шаблон Active Record – один об'єкт керує даними і поведінкою, зберігаючи логіку доступу до БД у сутності. Він обгортає один рядок з БД і включає доступ до даних та логіку обробки. Використовується в простих системах, наприклад, Ruby on Rails, але при збільшенні запитів логіку часто переносять в окремий об'єкт.

Шаблон Table Data Gateway – окремий клас для кожного типу даних взаємодіє з БД, містячи всю логіку запитів, включаючи збереження та видалення. Це забезпечує гнучкість і тестованість, розділяючи дані та взаємодію з БД. Код шлюзу часто повторюється, тому створюють базовий клас для спільної логіки. Цей шаблон також називають репозиторієм.

Шаблон Data Mapping використовується для перетворення об'єкта даних у формат, прийнятний для БД або передачі по мережі. Маппер містить інформацію про відповідність колонок таблиці та властивостей об'єкта даних, вирішуючи невідповідності типів даних між ними. Він забезпечує зручне відображення об'єктів у реляційні таблиці БД.

Шаблон «Composite» використовується для складання об'єктів в деревоподібну структуру для подання ієрархій типу «частина цілого». Він дозволяє уніфіковано обробляти як поодинокі об'єкти, так і об'єкти з вкладеністю. Простий приклад: форма, яка може містити дочірні елементи (поля для введення тексту, цифр, написи тощо), і ці елементи, в свою чергу, можуть містити інші елементи. Наприклад, операція розтягування форми застосовується до всієї ієрархії рекурсивно. Цей шаблон зручний для обробки ієрархій об'єктів. Проблема виникає, коли модель програми структурована у вигляді дерева, наприклад, як у замовленнях, що містять продукти та коробки з вкладеними рівнями. Підрахунок вартості всього замовлення

вручну може бути складним через невідомі структури. Рішення: компонувальник пропонує використовувати єдиний інтерфейс для об'єктів із методом отримання вартості.

Шаблон «Flyweight» використовується для зменшення кількості об'єктів у додатку шляхом поділу об'єктів між ділянками програми. Наприклад, у тексті здається, ніби кожна літера є окремим об'єктом, але насправді існує лише один об'єкт із безліччю посилань на нього. Важливим є поділ станів на внутрішній (характерний поділюваному об'єкту, наприклад, код букви) та зовнішній (дані про застосування об'єкта, наприклад, позиція в тексті). Внутрішній стан зберігається у спільному об'єкті, а зовнішній — у контексті. Цей шаблон підходить для роботи з великою кількістю однакових об'єктів. Проблема виникає, коли програма не може обробляти безліч об'єктів, як у грі з реалістичною системою частинок. Рішення: виділення спільних даних (наприклад, колір і спрайт) у загальний об'єкт, а унікальних даних (координати, вектор руху) — у контекст.

Шаблон «Interpreter» використовується для подання граматики та інтерпретатора для вибраної мови. Граматика описується через термінальні та нетермінальні символи, кожен із яких інтерпретується в контексті. Клієнт передає контекст і сформовану пропозицію у вигляді абстрактного синтаксичного дерева. Батьківські вирази інтерпретують дочірні рекурсивно, а потім виконують свої операції. Шаблон зручний для невеликих граматик і простих контекстів. Проблема: необхідність обробки частих змін, як у пошуку рядків за зразком. Рішення: створення інтерпретатора, що визначає граматику, клієнт будує дерево і викликає метод розбору в контексті.

Шаблон «Visitor» дозволяє визначати операції над елементами без зміни їх структури. Це спрощує додавання нових операцій, але ускладнює додавання нових елементів, оскільки необхідно змінювати всіх відвідувачів. Проблема: експорт графа з геоданими до XML, коли класи вузлів не можна змінювати. Рішення: додати нову поведінку в окремий клас-відвідувач, який виконує операцію над переданими об'єктами.

2. Шаблон Composite

2.1 Структура та обґрунтування вибору

Ознайомившись з теоретичними матеріалами, було прийнято рішення реалізувати шаблон Composite (Компонувальник). Компонувальник — це структурний патерн проєктування, що дає змогу згрупувати декілька об’єктів у деревоподібну структуру, а потім працювати з нею так, ніби це одиничний об’єкт.

Буде доречно застосувати цей шаблон до desktop застосунку, де елементи графічного інтерфейсу користувача можуть бути «вкладені» один в одного. Наприклад, головне вікно за визначенням має всередині себе всі інші елементи застосунку. Також можна навести приклад кнопки меню, яка при натисканні відкриває список підпунктів. Загальну структуру графічно описано на Рисунку 1.

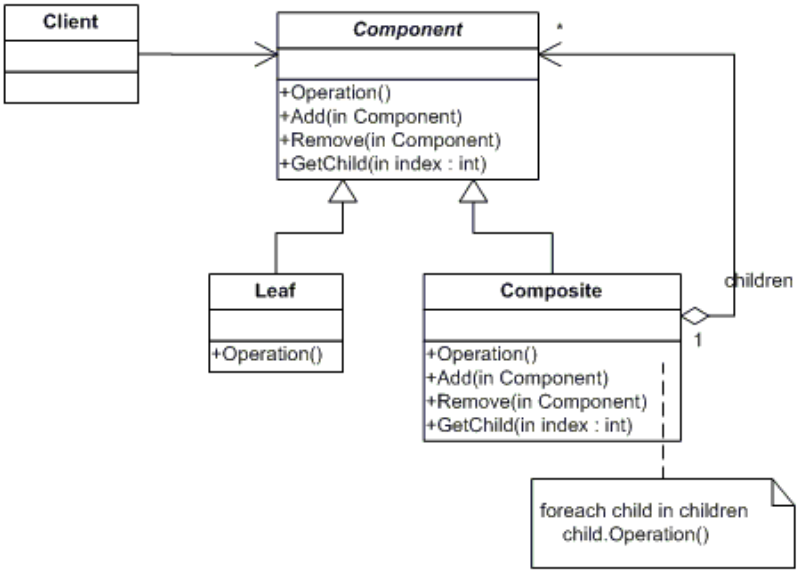

Рисунок 1. Загальна структура шаблону Composite

Компонент описує загальний інтерфейс для простих і складових компонентів дерева. Лист — це простий компонент дерева, який не має відгалужень. Класи листя міститимуть більшу частину корисного коду, тому що їм нікому передавати його виконання. Контейнер (або композит) — це складовий компонент дерева. Він містить набір дочірніх компонентів, але нічого не знає про їхні типи. Клієнт працює з деревом через загальний інтерфейс компонентів.

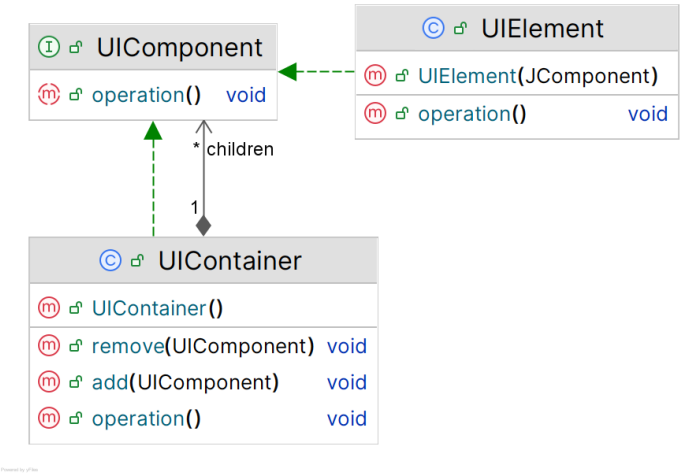

Рисунок 2. Структура реалізованого шаблону Composite

На Рисунку 2 зображено діаграму класів для шаблону Composite. UIComponent (інтерфейс) описує загальний контракт для всіх компонентів інтерфейсу користувача. Метод operation() виконує дію, специфічну для компонента.

UIContainer (клас Composite) реалізує контейнер для вкладених компонентів. Методи add() і remove() додають або видаляють дочірні компоненти. Поле children це список компонентів, які можуть бути контейнерами або елементами. Виклик operation() для кожного дочірнього компонента.

UIElement (клас Leaf) реалізує окремий елемент інтерфейсу користувача. Поле component зберігає Swing-компонент. Метод operation() виконує дію над компонентом, викликаючи його repaint() для оновлення UI.

2.2 Реалізація функціоналу у коді з використанням шаблону

Interface UIComponent

```
package org.example.composite;

public interface UIComponent {
    void operation();
}
```

Class UIContainer

```
package org.example.composite;

import java.util.ArrayList;
import java.util.List;
```

```
public class UIContainer implements UIComponent {
    private List<UIComponent> children = new ArrayList<>();

    public void add(UIComponent component) {
        children.add(component);
    }

    public void remove(UIComponent component) {
        children.remove(component);
    }

    @Override
    public void operation() {
        for (UIComponent child : children) {
            child.operation();
        }
    }
}
```

Class UIElement

```
package org.example.composite;

import javax.swing.*;

public class UIElement implements UIComponent {
    private JComponent component;

    public UIElement(JComponent component) {
        this.component = component;
    }

    @Override
    public void operation() {
        component.repaint();
    }
}
```

Class AudioEditorUI (використання шаблону)

```
package org.example.swing;

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

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);

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

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

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

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

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

        Logger logger = new Logger();

        UIContainer mainContainer = new UIContainer();

        UIContainer buttonPanel = new UIContainer();
        buttonPanel.add(new UIElement(loadFileButton));
        buttonPanel.add(new UIElement(convertToMp3Button));
        buttonPanel.add(new UIElement(convertToOggButton));
        buttonPanel.add(new UIElement(convertToFlacButton));

        mainContainer.add(buttonPanel);
        mainContainer.add(new UIElement(fileLabel));
        mainContainer.add(new UIElement(waveformPanel));

        frame.add(loadFileButton);
        frame.add(fileLabel);
        frame.add(convertToMp3Button);
        frame.add(convertToOggButton);
        frame.add(convertToFlacButton);
        frame.add(waveformPanel);

        AudioEditorMediator mediator = new AudioEditorMediator(
                loadFileButton, convertToMp3Button, convertToOggButton,
convertToFlacButton,
                fileLabel, waveformPanel, logger);

        mainContainer.operation();

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

Висновки: При виконанні цієї лабораторної роботи я розглянула шаблони проєктування Composite, Flyweight, Interpreter та Visitor. Composite використовується для роботи з ієрархічними структурами об'єктів, Flyweight — для зменшення витрат пам'яті через спільне використання об'єктів, Interpreter — для обробки виразів та реалізації мов, а Visitor — для додавання нових операцій до об'єктів без зміни їхньої структури.

Посилання на репозиторій з кодом проєкту:

https://github.com/KatiaArkhyp/AudioEditor