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 №6

Технології розроблення програмного забезпечення

ШАБЛОН «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD»

Варіант №5

Виконав
студент групи ІА-13:
Засінець А. Є.

Київ 2023

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

Хід роботи:

Тема проєкту:

..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).

Завдання:

1. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.

Для початку визначимо та опишемо кожен із шаблонів, що розглядаються та застосовуються під час виконання даної лабораторної роботи.

Mediator

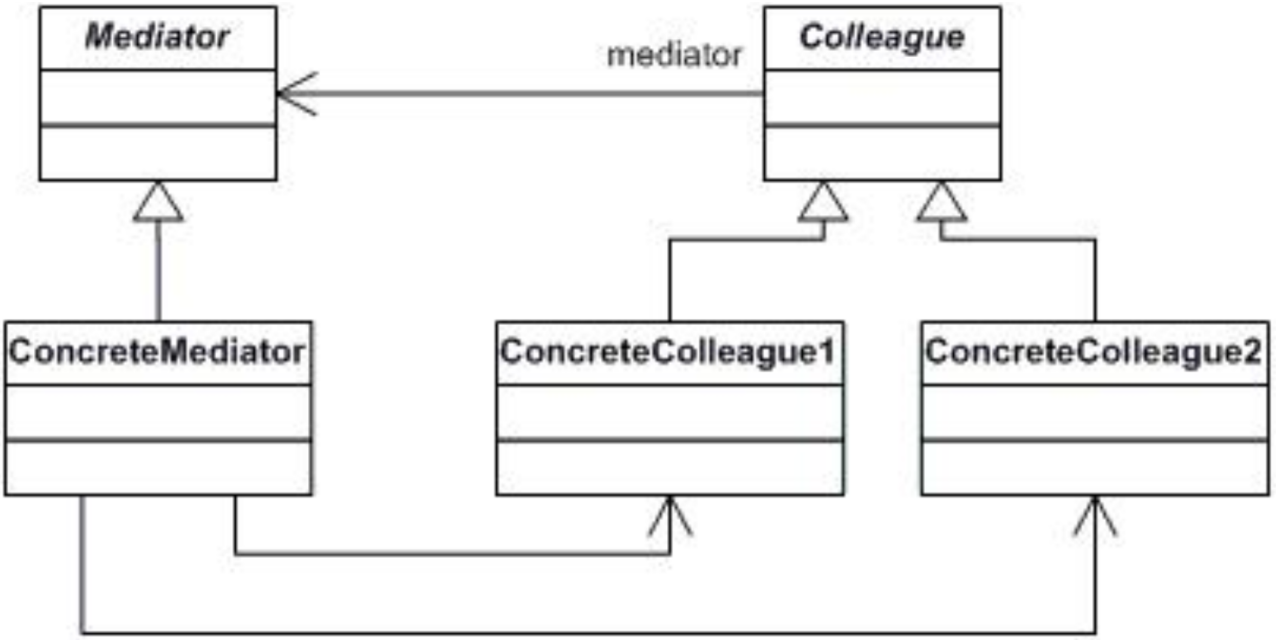

Рис. 1 - Структура Mediator

Шаблон «Mediator» (посередник) використовується для визначення взаємодії об'єктів за допомогою іншого об'єкта (замість зберігання посилань один на одного). Даний шаблон схожий на шаблон «команда», проте в даному випадку замість зберігання даних про конкретну дію, зберігаються дані про взаємодії між компонентами.

Його зручно застосовувати у випадках, коли безліч об'єктів взаємодіє між собою деяким структурованим чином, однак складним для розуміння. У такому випадку вся логіка взаємодії виноситься в окремий об'єкт.

Патерн «Mediator» змушує об'єкти спілкуватися не безпосередньо один з одним, а через окремий об'єкт-посередник, який знає, кому потрібно перенаправити той чи інший запит. Завдяки цьому, компоненти системи будуть залежати тільки від посередника, а не від десятків інших компонентів.

Переваги та недоліки:

+ Усуває залежності між компонентами, дозволяючи повторно їх використовувати.

+ Спрощує взаємодію між компонентами.

+ Централізує управління в одному місці.

- Посередник може сильно роздутися.

Facade

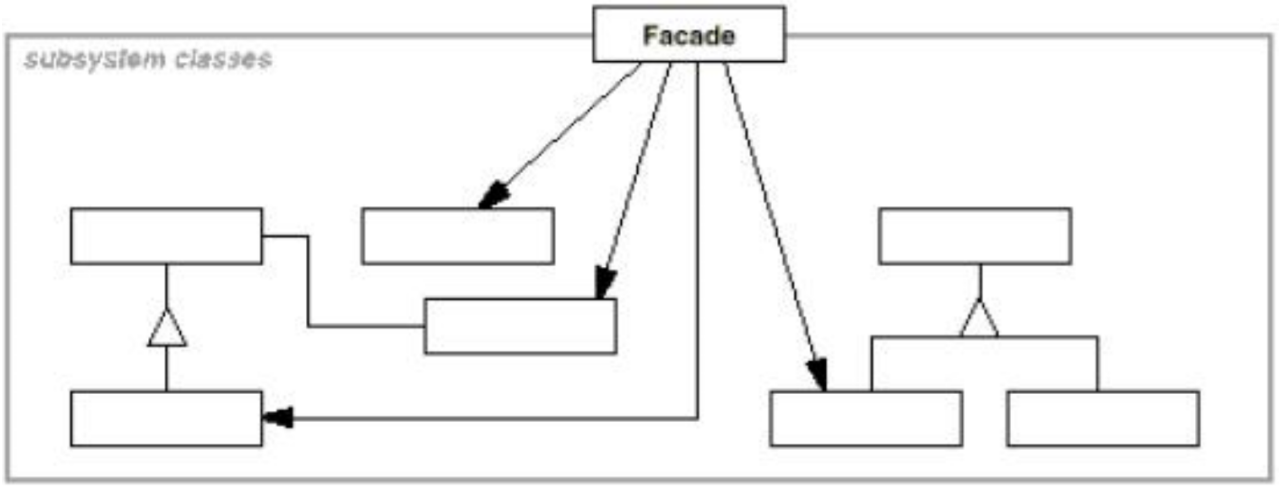

Рис. 2 - Структура паттерна Facade

Цей паттерн передбачає створення єдиного уніфікованого способу доступу до підсистеми без розкриття внутрішніх деталей підсистеми. Оскільки підсистема може складатися з безлічі класів, а кількість її функцій - не більше десяти, то щоб уникнути створення «спагеті-коду» (коли все тісно пов'язано між собою) виділяють один загальний інтерфейс доступу, здатний правильним чином звертатися до внутрішніх деталей.

Фасад - це простий інтерфейс для роботи зі складною підсистемою, що містить безліч класів. Фасад може мати урізаний інтерфейс, який не має 100% функціональності, якої можна досягти, використовуючи складну підсистему безпосередньо. Але він надає саме ті фічі, які потрібні клієнту, і приховує всі інші.

Фасад корисний, якщо ви використовуєте якусь складну бібліотеку з безліччю рухомих частин, але вам потрібна тільки частина її можливостей.

Переваги та недоліки:

+ Ізолює клієнтів від компонентів складної підсистеми.

- Фасад ризикує стати божественним об'єктом, прив'язаним до всіх класів програми.

Bridge

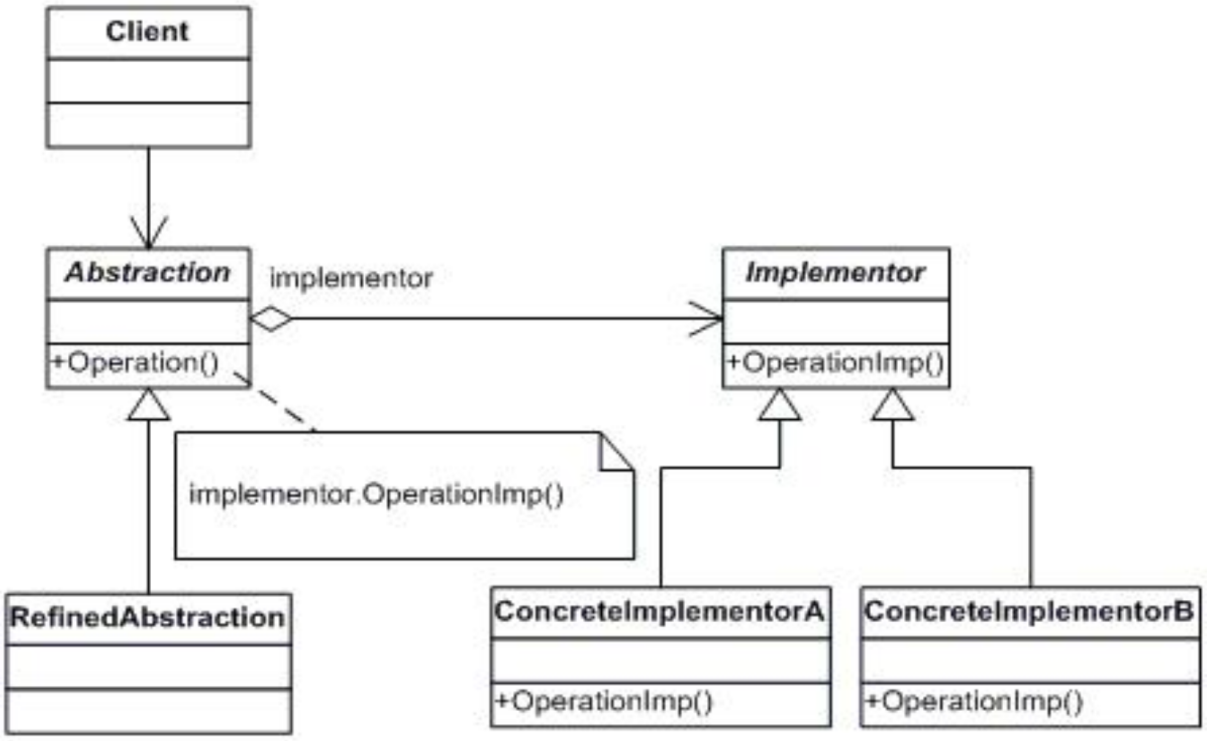

Рис. 3 - Структура паттерна Bridge

Цей шаблон використовується для поділу інтерфейсу і його реалізації. Це необхідно у випадках, коли може існувати кілька різних абстракцій, над якими можна проводити дії різними способами.

Патерн «Bridge» пропонує замінити спадкування агрегацією або композицією.

Переваги та недоліки:

+ Дозволяє будувати платформо-незалежні програми.

+ Приховує зайві або небезпечні деталі реалізації від клієнтського коду.

+ Реалізує принцип відкритості / закритості..

- Ускладнює код програми через введення додаткових класів.

Template Method

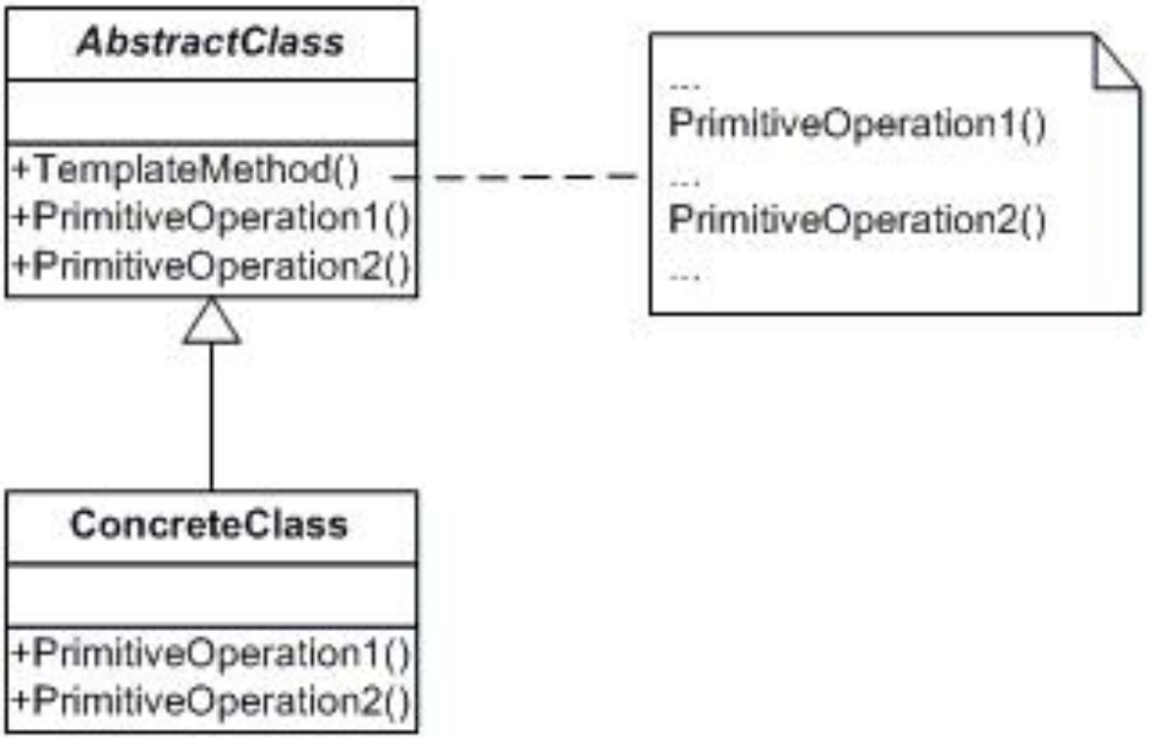

Рис. 4 - Структура паттерна Template Method

Цей шаблон дозволяє реалізувати покроково алгоритм в абстрактному класі, але залишити специфіку реалізації подклассам.

Патерн «Template Method» пропонує розбити алгоритм на послідовність кроків, описати ці кроки в окремих методах і викликати їх в одному шаблонному методі один за одним. Це дозволить підкласам перевизначити деякі кроки алгоритму, залишаючи без змін його структуру і інші кроки, які для цього підкласу не так важливі.

Переваги та недоліки:

+ Полегшує повторне використання коду.

- Ви жорстко обмежені скелетом існуючого алгоритму.

- Ви можете порушити принцип підстановки Барбари Лісков, змінюючи базову поведінку одного з кроків алгоритму через підклас.

- З ростом кількості кроків шаблонний метод стає занадто складно підтримувати.

2. Застосування одного з розглянутих шаблонів при реалізації програми.

Як зазначено в завданні, необхідно реалізувати шаблон Mediator.

Це такий паттерн, який використовується для визначення взаємодії об'єктів за допомогою іншого об'єкта.

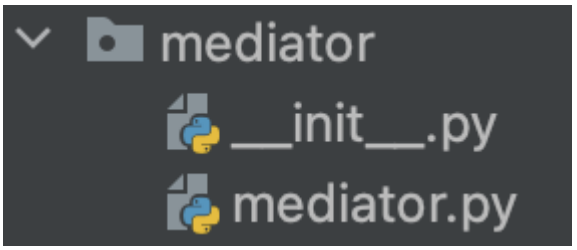

Рис. 6 - Структура програми

У файлі mediator.py реалізований метод обробки команд handle_command(), який буде використовуватись сервером.

```
class Mediator:
    def __init__(self, server, observer):
        self.server = server
        self.observer = observer

    def handle_command(self, command):
        response = self.server.handle_command(command)
        return response
```

Рис. 7 - Вміст файлу mediator.py

Висновки: Під час виконання даної лабораторної роботи ми вивчили інформацію про шаблони *«MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD»* та реалізували шаблон Mediator у нашому застосунку.