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 №5

Технології розроблення програмного забезпечення

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

Варіант №5

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

Київ 2023

Тема: Шаблони *«Singleton», «Iterator», «Proxy», «State», «Strategy»*

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).

Було прийнято рішення використати для розробки звстосунку мову програмування Python (в минулих лабораторних зазначалося, що використовуватиметься мова програмування Java або C#). Вирішив змінити мову програмування тому, що в Python є деякі інструменти, з якими буде легше реалізовувати завдання. Також на вибір Python вплинуло те, що писати на ньому застосунок виявилось легше і ефективніше, як на мене.

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.

Adapter

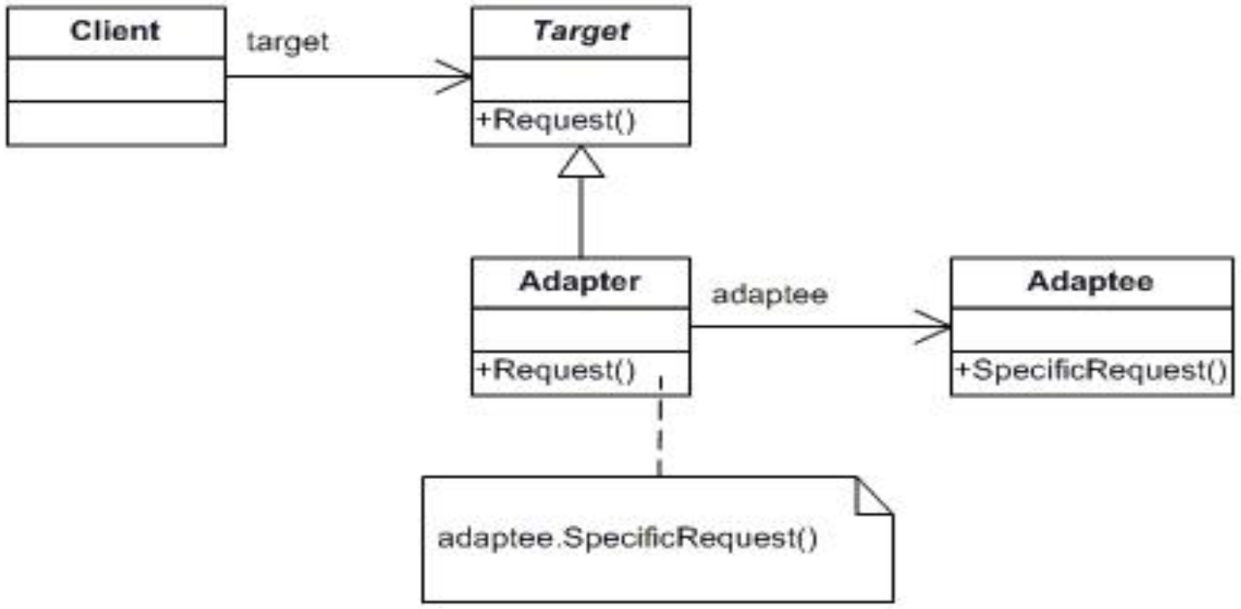

Рис. 1 - Структура Adapter

Даний паттерн використовується для адаптації інтерфейсу одного об'єкту до іншого.

Adapter - це об’єкт-перекладач, який трансформує інтерфейс або дані одного об’єкта таким чином, щоб він став зрозумілим іншому об’єкту. Адаптер загортає один з об’єктів так, що інший об’єкт навіть не підозрює про існування першого.

Переваги та недоліки:

+ Відокремлює та приховує від клієнта подробиці перетворення різних інтерфейсів.

- Ускладнює код програми внаслідок введення додаткових класів.

Builder

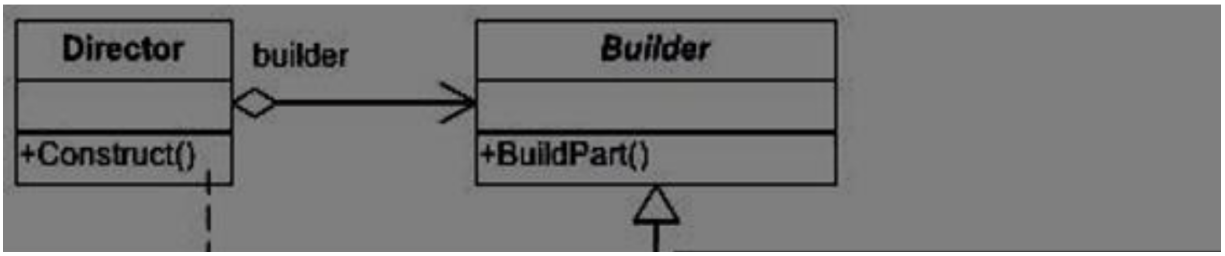

Рис. 2 - Структура паттерна Builder

Builder використовується для відділення процесу створення об'єкту від його представлення. Це доречно у випадках, коли об'єкт має складний процес створення або коли об'єкт повинен мати декілька різних форм створення (наприклад, при конвертації тексту з формату у формат).

Переваги та недоліки:

+ Дозволяє використовувати один і той самий код для створення різноманітних продуктів.

- Клієнт буде прив’язаний до конкретних класів будівельників, тому що в інтерфейсі будівельника може не бути методу отримання результату.

Command

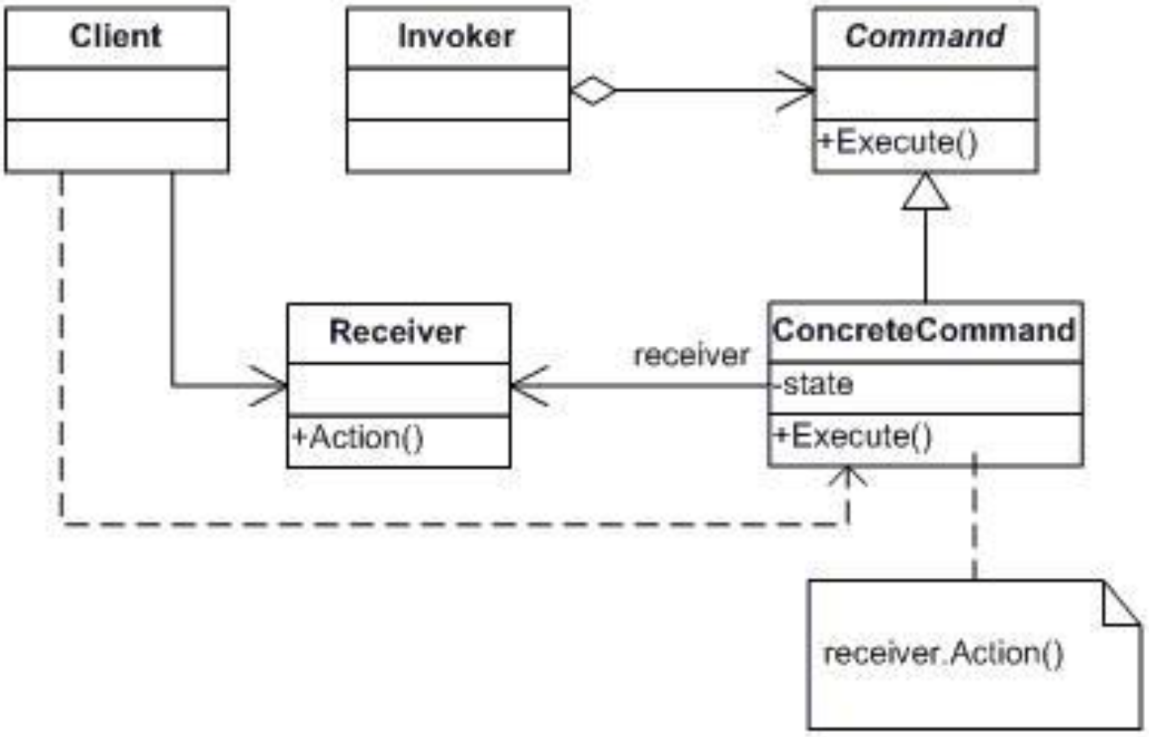

Рис. 3 - Структура паттерна Command

Цей шаблон перетворить звичайний виклик методу в клас. Таким чином дії в системі стають повноправними об'єктами. Це зручно в наступних випадках:

- Коли потрібна розвинена система команд - відомо, що команди будуть добавлятися;

-Коли потрібна гнучка система команд - коли з'являється необхідність додавати командам можливість відміни, логування і інш.;

- Коли потрібна можливість складання ланцюжків команд або виклику команд в певний час;

Об'єкт команда сама по собі не виконує ніяких фактичних дій окрім перенаправлення запиту одержувачеві (тобто команди все ж виконуються одержувачем), однак ці об'єкти можуть зберігати дані для підтримки додаткових функцій відміни, логування і інш.

Переваги та недоліки:

+ Прибирає пряму залежність між об’єктами, що викликають операції, та об’єктами, які їх

безпосередньо виконують.

+ Дозволяє реалізувати просте скасування і повтор операцій.

+ Дозволяє реалізувати відкладений запуск операцій.

+ Реалізує принцип відкритості/закритості.

- Ускладнює код програми внаслідок введення великої кількості додаткових класів.

Chain of Responsibility

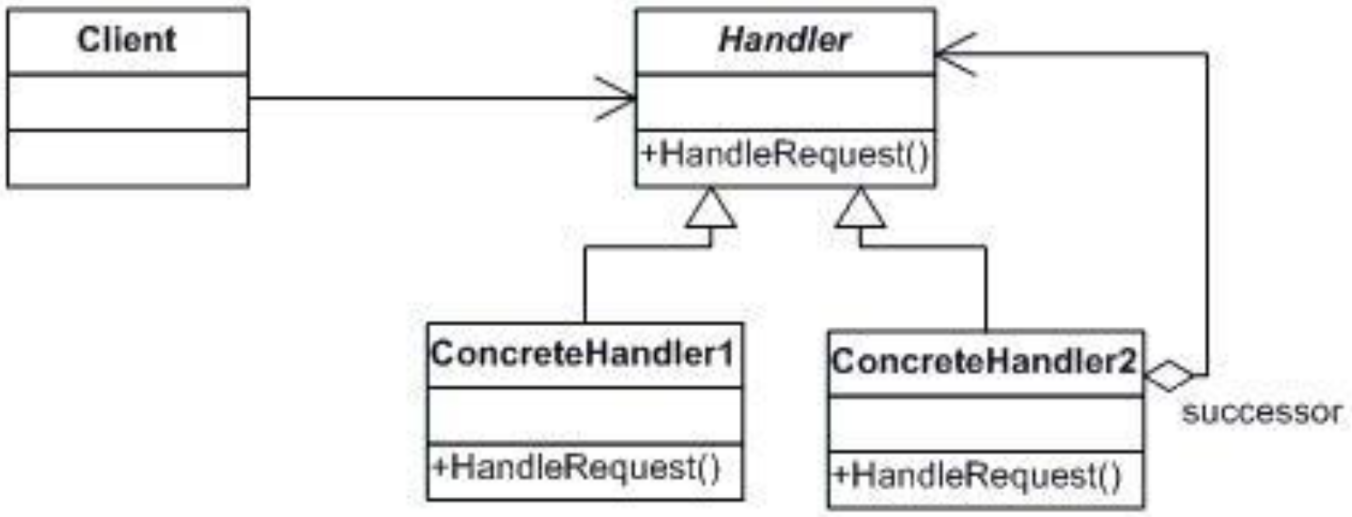

Рис. 4 - Структура паттерна Chain of Responsibility

Цей шаблон частково можна спостерігати в житті, коли підписання відповідного документу проходить від його складання у одного із співробітників компанії через менеджера і начальника до головного начальника, який ставить свій підпис.

Як і багато інших поведінкових патернів, ланцюжок обов'язків базується на тому, щоб перетворити окремі поведінки на об'єкти.

Патерн пропонує зв'язати всі об'єкти обробників в один ланцюжок. Кожен обробник міститиме посилання на наступного обробника в ланцюзі. Таким чином, після отримання запиту обробник зможе не тільки опрацювати його самостійно, але й передати обробку наступному об'єкту в ланцюжку. Обробник не обов'язково повинен передавати запит далі.

Переваги та недоліки:

+ Зменшує залежність між клієнтом та обробниками.

+ Реалізує принцип єдиного обов'язку.

+ Реалізує принцип відкритості/закритості.

- Запит може залишитися ніким не опрацьованим.

Prototype

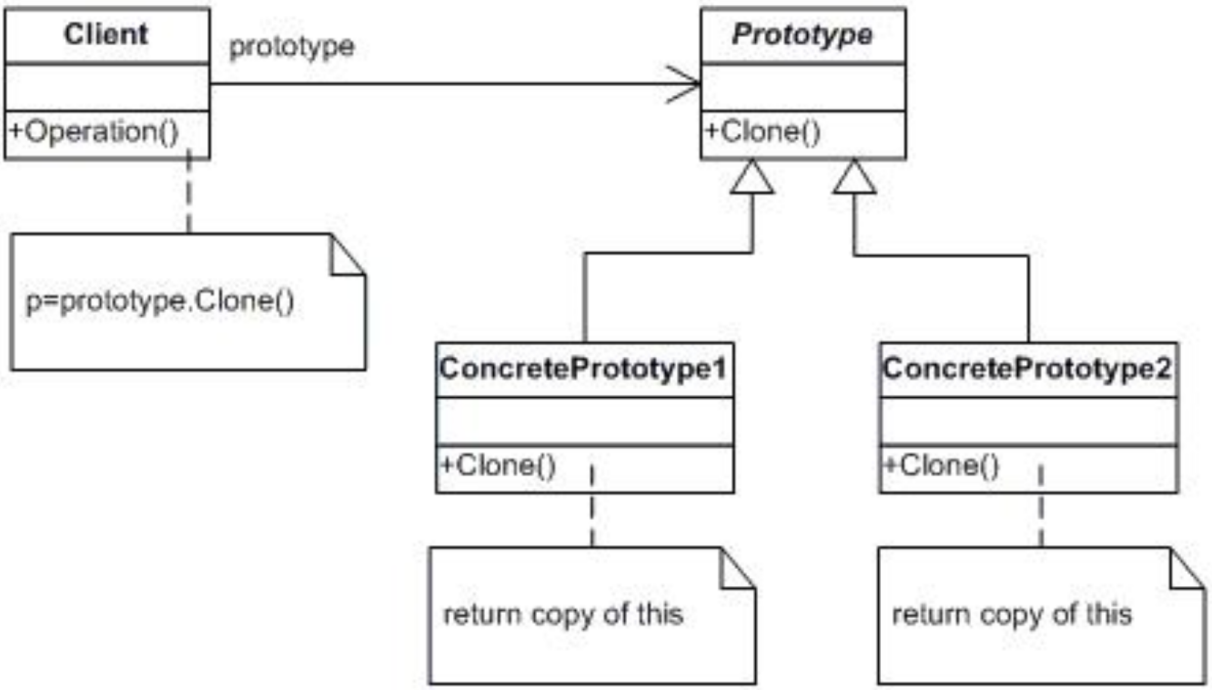

Рис. 5 - Структура паттерна Prototype

Шаблон «Prototype» використовується для створення об'єктів за "шаблоном" (чи "кресленню", "ескізу") шляхом копіювання шаблонного об'єкту. Для цього визначається метод "клонувати" в об'єктах цього класу.

Цей шаблон зручно використати, коли заздалегідь відомо як виглядатиме кінцевий об'єкт (мінімізується кількість змін до об'єкту шляхом створення шаблону), а також для видалення необхідності створення об'єкту - створення відбувається за рахунок клонування, і зухвалій програмі абсолютно немає необхідності знати, як створювати об'єкт.

Також, це дозволяє маніпулювати об'єктами під час виконання програми шляхом настроювання відповідних шаблонів; значно зменшується ієрархія спадкоємства (оскільки в іншому випадку це були б не шаблони, а вкладені класи, що наслідують).

Переваги та недоліки:

+ Дозволяє клонувати об’єкти без прив’язки до їхніх конкретних класів.

+ Менша кількість повторювань коду ініціалізації об’єктів.

+ Прискорює створення об’єктів.

+ Альтернатива створенню підкласів під час конструювання складних об’єктів.

- Складно клонувати складові об’єкти, що мають посилання на інші об’єкти.

2. Застосування одного з розглянутих шаблонів при реалізації програми.

Як зазначено в завданні, необхідно реалізувати шаблон Adapter.

Як вже було зазначено в теорії даний паттерн використовується для адаптації інтерфейсу одного об'єкту до іншого.

Рис. 6 - Структура програми

У файлі audio_adapter.py знаходиться такий функціонал: методі завантаження аудіо та збереження аудіо, які в майбутньому будуть використовуватися сервером.

Також було використано бібліотеку pydub, що містить функціонал для маніпулювання аудіо.

```python
from pydub import AudioSegment

class AudioAdapter:
    def load_audio(self, file_path, format):
        audio = AudioSegment.from_file(file_path, format)
        return audio

    def save_audio(self, audio, output_path, format):
        audio.export(output_path, format)
```

Рис. 7 - Вміст файлу audio_adapter.py

Висновки: Під час виконання даної лабораторної роботи ми вивчили інформацію про *шаблони «adapter», «builder», «command», «chain of responsibility», «prototype»* та реалізували шаблон Adapter у нашому застосунку.