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 №4
ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»

Варіант 1

Виконав
студент групи ІА – 13:
Вознюк Максим

Перевірив:
Мягкий М. Ю

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

Варіант:

..1 Музичний програвач (iterator, command, memento, facade, visitor, client-server)

Музичний програвач становить собою програму для програвання музичних файлів або відтворення потокової музики з можливістю створення, запам'ятовування і редагування списків програвання, перемішування/повторення (shuffle/repeat), розпізнавання різних аудіо-форматів, еквалайзер.

Хід роботи

Паттерн Ітератор(Iterator)

Ітератор — це поведінковий патерн проектування, що дає змогу послідовно обходити елементи складових об'єктів, не розкриваючи їхньої внутрішньої організації.

```python
class PlaylistIterator:
    def __init__(self, tracks):
        self.tracks = tracks
        self.index = 0

    def __iter__(self):
        return self

    def __next__(self):
        if self.index < len(self.tracks):
            current_track = self.tracks[self.index]
            self.index += 1
            return current_track
        else:
            raise StopIteration
```

```python
1 usage
def play_in_thread(self, client_socket):
    client_socket.sendall(f"Playing........".encode('utf-8'))
    if not self.current_playlist_id:
        client_socket.sendall("No playlist selected. Create or select a playlist.".encode('utf-8'))
        return
    tracks = self.db_manager.get_tracks_for_playlist(self.current_playlist_id)

    if not tracks:
        client_socket.sendall("Current playlist is empty. Add some songs.".encode('utf-8'))
        return

    playlist_iterator = PlaylistIterator(tracks)
    for track in playlist_iterator:
        pygame.mixer.music.load(track[1])
        pygame.mixer.music.play()
        self.playing = True

        while pygame.mixer.music.get_busy() and self.playing:
            time.sleep(1)

        if not self.playing:
            break
```

В данному випадку паттерн Iterator використовується для ітерації через список треків у плейлисті. Паттерн Iterator дозволяє послідовно обходити елементи колекції, не розкриваючи її внутрішньої структури. У цьому випадку, PlaylistIterator визначає два методи: __iter__ та __next__, щоб об'єкт можна було використовувати у циклах for.

__init__(self, tracks): Конструктор класу приймає список tracks і ініціалізує атрибути tracks та index. tracks - це список треків, які будуть ітеровані, а index служить для відстеження поточного положення в ітерації.

__iter__(self): Метод повертає об'єкт ітератора, який у цьому випадку є самим об'єктом класу PlaylistIterator. Це дозволяє використовувати об'єкт у циклі for.

__next__(self): Метод визначає логіку ітерації. Якщо індекс менший за кількість треків у плейлисті, метод повертає поточний трек і збільшує індекс. Якщо індекс вже дорівнює або перевищує кількість треків, генерується виключення StopIteration, що сигналізує про завершення ітерації.

Цей паттерн дозволяє ефективно ітерувати через треки у плейлисті без необхідності розкривати внутрішню структуру класу PlaylistIterator. Код, який використовує ітератор, може просто використовувати цикл for, не знаючи деталей реалізації обходу.