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

з дисципліни: «Технології розроблення програмного забезпечення»

Виконав:
Тимчук Владислав
ІА-34
Перевірив:
Мягкий М. Ю.

Київ-2025

Тема: Вступ до паттернів проектування

Мета: Вивчити структуру шаблонів «Singleton», «Iterator», «Proxy», «State», «Strategy» та навчитися застосовувати їх в реалізації програмної системи.

Тема Лабораторного Практикуму:

Музичний програвач (iterator, command, memento, facade, visitor, clientserver)
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 аудіоформатів, еквалайзер.

Вступ

Метою цієї роботи є реалізація частини функціональності ПЗ та демонстрація використання одного з шаблонів проектування. Для вирішення задачі буде обрано шаблон **Strategy**. Шаблон зручно застосовувати в аудіоплеєрах, оскільки користувач може перемикати режими відтворення (звичайний, повтор, перемішування) без зміни бізнес-логіки самого плеєра.

У межах роботи також реалізовано основні класи предметної області.

Зміст

Хід роботи

1. ДІАГРАМА КЛАСІВ

Було реалізовано модуль відтворення плейлиста. Основна логіка:

- зберігання списку треків у плейлисті;
- клас, який керує відтворенням треків;
- можливість встановлювати різні режими вибору наступного треку;
- реалізація трьох стратегій:
    - **NormalPlaybackStrategy** — відтворення по порядку,
    - **RepeatAllPlaybackStrategy** — повтор усіх треків циклічно,
    - **ShufflePlaybackStrategy** — випадковий наступний трек.

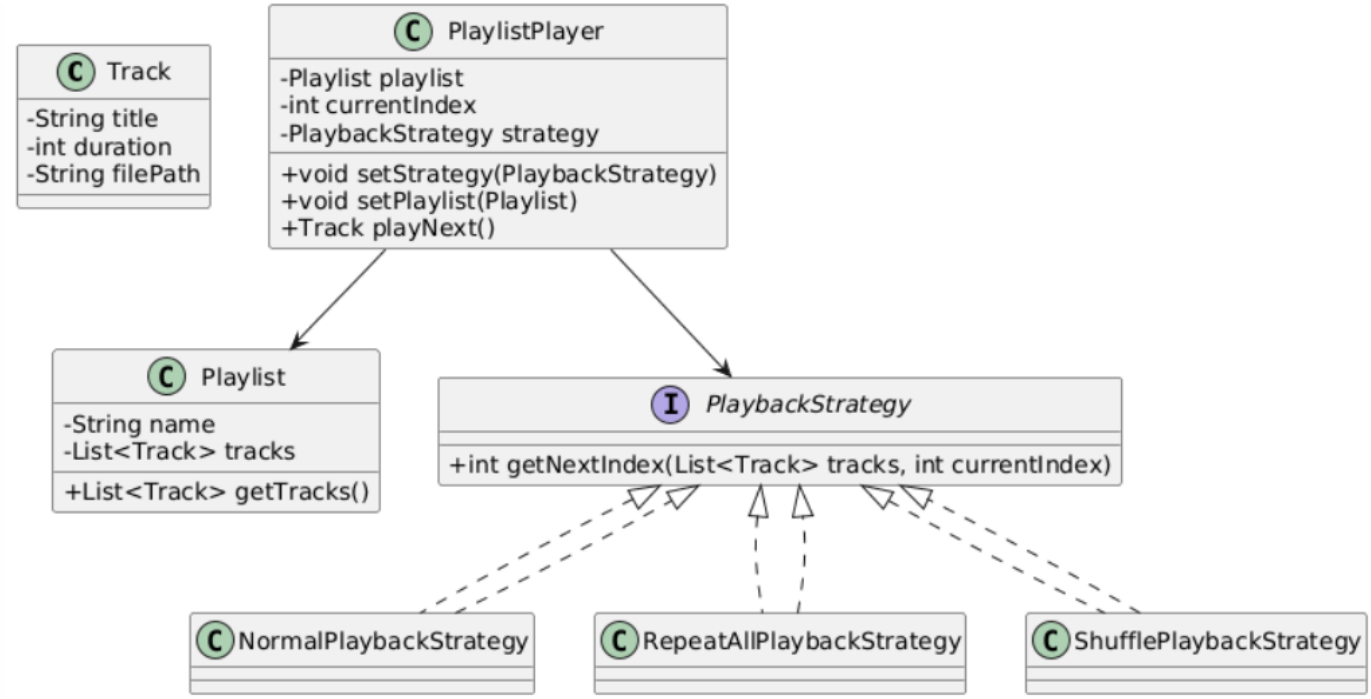


Рис. 1 – Діаграма класів

2. КОД

2.1. Шаблон Strategy

Шаблон **Strategy** дозволяє винести алгоритм у окремі класи. Наш клас не знає, як саме обирається наступний трек — він делегує це об’єкту-стратегії.

Сенс у тому, що поведінку можна змінювати на льоту:

```
player.setStrategy(new ShufflePlaybackStrategy());
```

Це дозволяє розширювати систему без зміни існуючого коду гравця.

2.2. Фрагменти коду реалізації шаблону

Увесь код знаходиться на віддаленому репозиторії: https://github.com/fromz67/TRPZ/tree/main/lab4

ВИСНОВОК

У ході лабораторної роботи було реалізовано частину функціональності системи музичного програвача — модуль управління відтворенням плейлиста. Реалізовано більше трьох класів, створено діаграму класів та наведено ключові фрагменти коду, що демонструють роботу шаблону.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке шаблон проєктування?

Шаблон проєктування — це готове, перевірене рішення типової задачі проєктування у програмуванні, яке описує взаємодію класів і об’єктів певним способом, не прив’язуючись до конкретної мови.

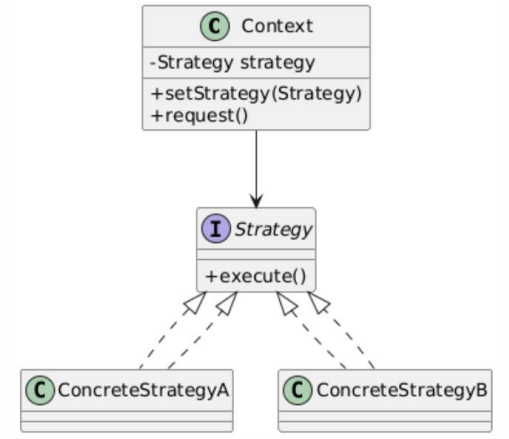
2. Навіщо використовувати шаблони проєктування?

Шаблони проєктування допомагають робити код гнучким, розширюваним і зрозумілим, уникати дублювання, спрощувати підтримку і повторне використання рішень у різних проєктах.

3. Яке призначення шаблону «Стратегія»?

«Стратегія» дозволяє вибирати один із кількох алгоритмів під час виконання програми, виносячи алгоритм у окремі класи та дозволяючи замінювати його без зміни основного коду.

4. Нарисуйте структуру шаблону «Стратегія».



5. Які класи входять в шаблон «Стратегія», та яка між ними взаємодія?

До шаблону входять: **Context** (використовує алгоритм), **Strategy** (інтерфейс алгоритму), **ConcreteStrategy** (конкретні реалізації). Контекст викликає метод стратегії через інтерфейс, не знаючи деталей реалізації.

6. Яке призначення шаблону «Стан»?

«Стан» дозволяє об’єкту змінювати свою поведінку при зміні внутрішнього стану, фактично поводячись так, ніби він належить до іншого класу.

7. Нарисуйте структуру шаблону «Стан».

Контекст → інтерфейс стану → конкретні стани. (Напр.: Context → State → StateA, StateB)

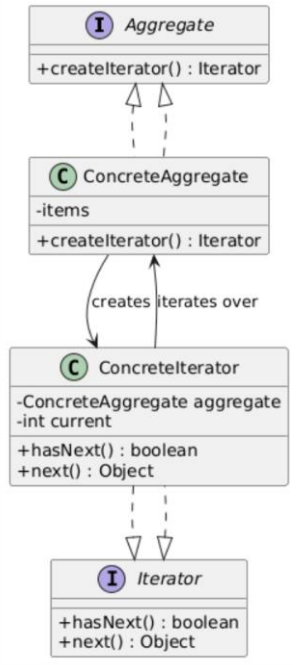
8. Які класи входять в шаблон «Стан», та яка між ними взаємодія?

Шаблон містить **Context** (зберігає поточний стан), **State** (інтерфейс поведінки), **ConcreteState** (реалізації). Контекст делегує операції активному стану, а стан може змінювати контекст на інший стан.

9. Яке призначення шаблону «Ітератор»?

Ітератор забезпечує зручний і уніфікований спосіб послідовного доступу до елементів колекції без розкриття її внутрішньої структури.

10. Нарисуйте структуру шаблону «Ітератор».



11. Які класи входять в шаблон «Ітератор», та яка між ними взаємодія?

Є **Aggregate/Collection** (джерело елементів), **Iterator** (інтерфейс для переходу між елементами), **ConcreteIterator** (реалізація переходу). Ітератор отримує доступ до елементів колекції й послідовно повертає їх.

12. В чому полягає ідея шаблона «Одинак»?

«Одинак» гарантує, що клас матиме лише один екземпляр у програмі та глобальну точку доступу до нього, зазвичай через статичний метод getInstance().

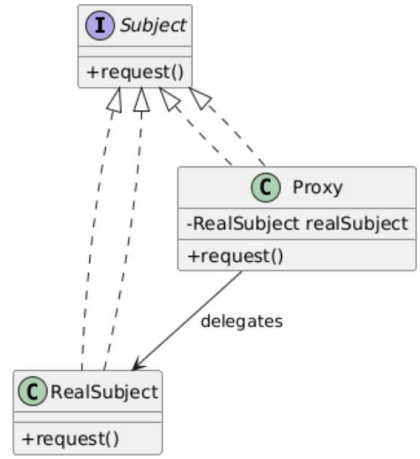
13. Чому шаблон «Одинак» вважають «анти-шаблоном»?

Його вважають антипатерном, бо він створює приховані залежності, ускладнює тестування, порушує принципи ООП і часто замінює нормальну ін’єкцію залежностей.

14. Яке призначення шаблону «Проксі»?

«Проксі» створює сурогатний об’єкт, який контролює доступ до реального об’єкта: додає кешування, ліниву ініціалізацію, безпеку, логування тощо.

15. Нарисуйте структуру шаблону «Проксі».



16. Які класи входять в шаблон «Проксі», та яка між ними взаємодія?

До шаблону входять: **Subject** (спільний інтерфейс), **RealSubject** (справжній об’єкт), **Proxy** (проксі-обгортка). Клієнт звертається до проксі, а той вирішує, коли і як викликати RealSubject.