



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**

з дисципліни: «Технології розроблення програмного забезпечення»

Виконав:

Тимчук Владислав

ІА-34

Перевірив:

Мягкий М. Ю.

**Тема:** Патерни проектування

**Мета:** Вивчити структуру шаблонів «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator» 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 лабораторної роботи є реалізація частини функціональності програмного забезпечення «Музичний програвач» із використанням шаблону проектування Observer, тобто «спостерігач». У межах роботи буде реалізовано: суб'єкт ObservablePlaylistPlayer, інтерфейс спостерігача PlayerObserver та два конкретні спостерігачі: NowPlayingDisplay (оновлює інформацію про поточний трек) і PlaybackLogger (логує події відтворення).

## Хід роботи

### 1 Опис реалізованої функціональності

Реалізована функціональність моделює оновлення інтерфейсу при зміні стану програвача:

Клас ObservablePlaylistPlayer відповідає за:

- вибір поточного треку;
- зміну стану відтворення
- сповіщення всіх підписаних спостерігачів про зміну стану.

Інтерфейс PlayerObserver описує, як спостерігачі отримують оновлення (update(currentTrack, state)).

Класи NowPlayingDisplay та PlaybackLogger реалізують різні реакції на ті самі події:

- NowPlayingDisplay виводить у консоль/умовний UI назву поточного треку та стан;
- PlaybackLogger логуює зміни стану (для історії/діагностики).

## 1.1 Діаграма класів

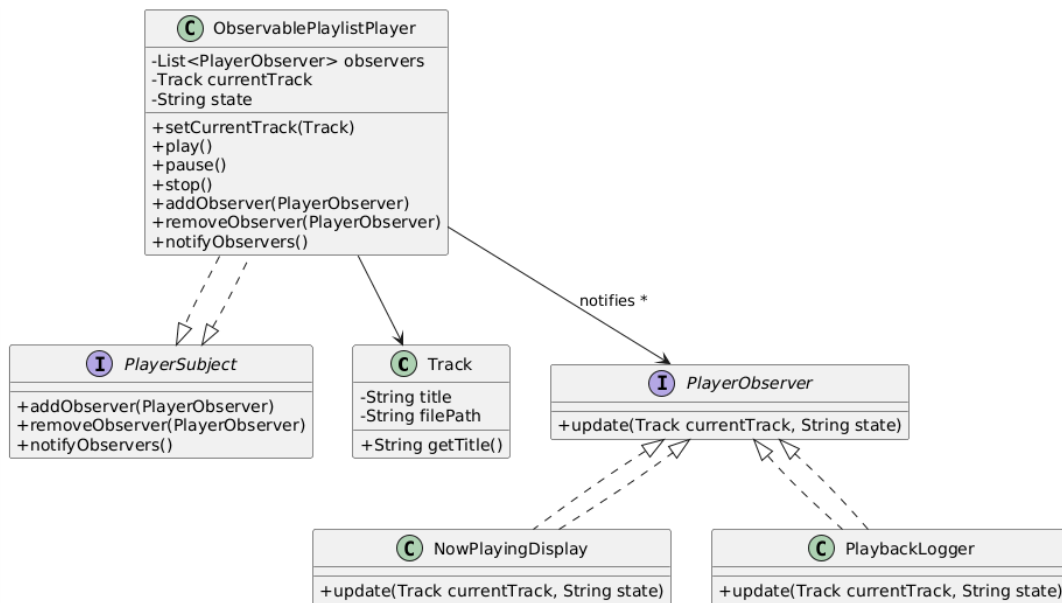


Рис. 1 – Діаграма класів

## 2 Фрагменти коду реалізації шаблону

Увесь код представлений для перегляду на віддаленому репозиторії:

<https://github.com/fromz67/TRPZ/tree/main/lab6>

## ВИСНОВОК

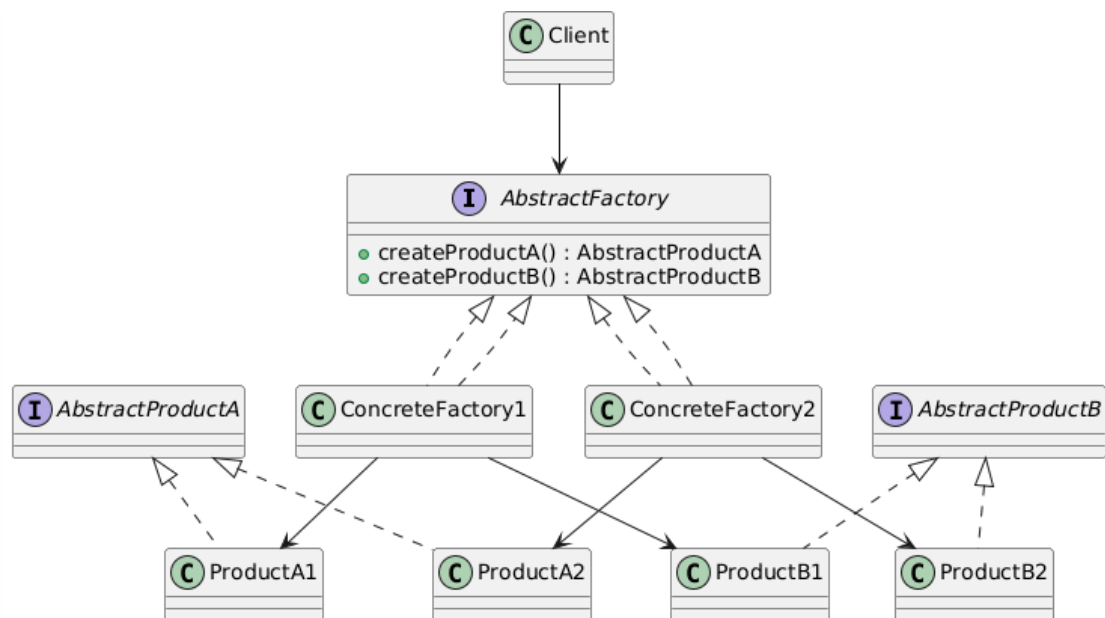
Під час виконання лабораторної роботи було реалізовано частину функціональності музичного програвача з використанням шаблону проєктування Observer. Було створено суб'єкт який підтримує підписку спостерігачів і сповіщає їх про зміну стану відтворення, та два конкретних спостерігачі. Такий підхід дозволяє розв'язати зв'язок між логікою відтворення і відображенням/логуванням стану.

## КОНТРОЛЬНІ ЗАПИТАННЯ

### 1. Яке призначення шаблону «Абстрактна фабрика»?

Шаблон «Абстрактна фабрика» призначений для створення груп взаємопов'язаних об'єктів, які повинні використовуватися разом, без прив'язки до конкретних класів; він дозволяє перемикати цілі сімейства продуктів, не змінюючи клієнтський код.

### 2. Нарисуйте структуру шаблону «Абстрактна фабрика».



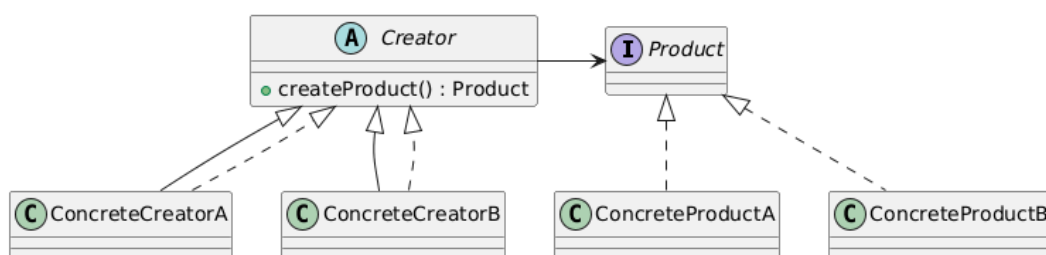
**3. Які класи входять в шаблон «Абстрактна фабрика», та яка між ними взаємодія?**

Шаблон включає AbstractFactory (інтерфейс для створення продуктів), ConcreteFactory (конкретні фабрики, що створюють узгоджені сімейства продуктів), AbstractProduct (інтерфейси продуктів), ConcreteProduct (їх реалізації) та Client, який працює лише з абстракціями; клієнт викликає метод фабрики, а конкретна фабрика створює відповідні продукти, не розкриваючи деталей створення.

**4. Яке призначення шаблону «Фабричний метод»?**

«Фабричний метод» дозволяє делегувати створення об'єктів підкласам, визначаючи спільний інтерфейс для створення, але дозволяючи нащадкам вирішувати, яке саме конкретне втілення створювати, забезпечуючи розширюваність і заміну типів продуктів.

**5. Нарисуйте структуру шаблону «Фабричний метод».**



**6. Які класи входять в шаблон «Фабричний метод», та яка між ними взаємодія?**

До «Фабричного методу» входять: Creator (оголошує фабричний метод), ConcreteCreator (перевизначає його і створює конкретні продукти), Product (спільний інтерфейс продукту), ConcreteProduct (конкретні реалізації); клієнт працює з Creator, а підкласи вирішують, який продукт створювати.

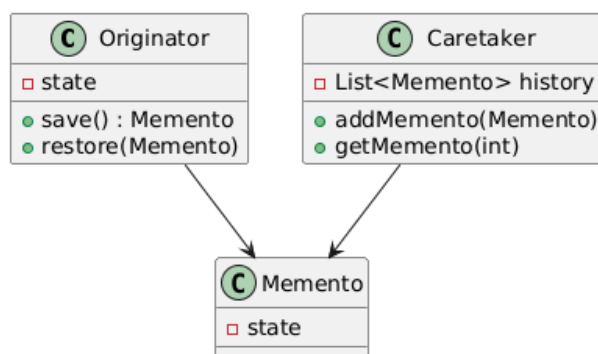
### 7. Чим відрізняється шаблон «Абстрактна фабрика» від «Фабричний метод»?

«Абстрактна фабрика» створює цілі сімейства продуктів, які повинні узгоджено працювати разом, тоді як «Фабричний метод» створює один продукт, і логіка вибору конкретного виду перекладається на підклас; Abstract Factory використовує кілька фабричних методів всередині себе.

### 8. Яке призначення шаблону «Знімок»?

«Знімок» призначений для збереження та відновлення попереднього стану об'єкта, не порушуючи інкапсуляції: він дозволяє створити snapshot стану та повернутися до нього пізніше (undo/redo).

### 9. Нарисуйте структуру шаблону «Знімок».



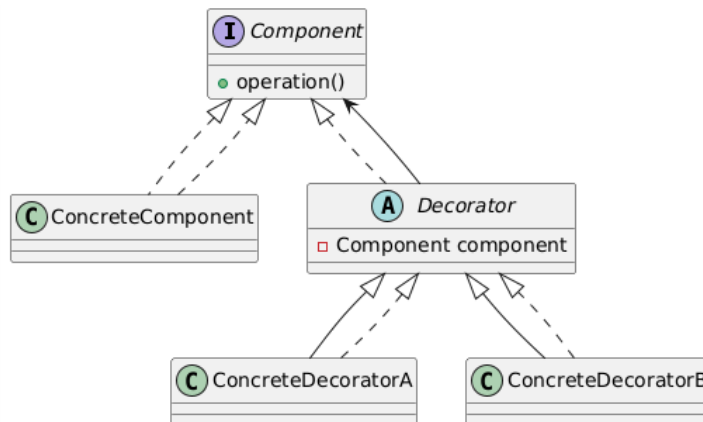
### 10. Які класи входять в шаблон «Знімок», та яка між ними взаємодія?

Шаблон містить: Originator (об'єкт, чий стан зберігається), Memento (знімок стану), Caretaker (керує збереженими знімками); Originator створює Memento з внутрішнім станом і може відновлюватися з нього, а Caretaker лише зберігає ці Memento, не знаючи їхнього вмісту.

### 11. Яке призначення шаблону «Декоратор»?

«Декоратор» дозволяє динамічно додавати нову поведінку або функціональність до об'єктів, не змінюючи їхнього класу, за допомогою обгортки, що реалізують той самий інтерфейс.

## 12. Нарисуйте структуру шаблону «Декоратор».



## 13. Які класи входять в шаблон «Декоратор», та яка між ними взаємодія?

Шаблон включає: **Component** (базовий інтерфейс), **ConcreteComponent** (реальний об'єкт), **Decorator** (абстрактна обгортка над компонентом), **ConcreteDecorator** (конкретні розширення); декоратор містить компонент і делегує йому базову поведінку, додаючи додаткові можливості до або після виклику.

## 14. Які є обмеження використання шаблону «Декоратор»?

Декоратор ускладнює структуру об'єктів через велику кількість дрібних класів, може бути складно відстежити реальний порядок викликів у глибокому ланцюжку обгортки, а також він не підходить, якщо потрібна глобальна зміна логіки — бо декоратори працюють лише на рівні конкретних екземплярів.

**Хід роботи**

**ВИСНОВОК**

**КОНТРОЛЬНІ ЗАПИТАННЯ**