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
«Патерни проектування.»

Виконав:
Студент групи ІА-32
Костінський Макарій

Перевірив:
Мягкий Михайло Юрійович

Зміст

Тема: Патерни проектування.

Мета: Вивчити структуру шаблонів «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator» та навчитися застосовувати їх в реалізації програмної системи.Вивчити структуру шаблонів «Adapter», «Builder», «Command», «Chain of responsibility», «Prototype» та навчитися застосовувати їх в реалізації програмної системи.

Теоретичні відомості

Абстрактна фабрика (Abstract Factory): Породжувальний патерн, що надає інтерфейс для створення сімейств взаємопов'язаних або залежних об'єктів без зазначення їхніх конкретних класів.

Фабричний метод (Factory Method): Породжувальний патерн, який визначає інтерфейс для створення об'єктів, але дозволяє підкласам (або методам) вирішувати, який клас інстанціювати. Це дозволяє делегувати створення об'єктів.

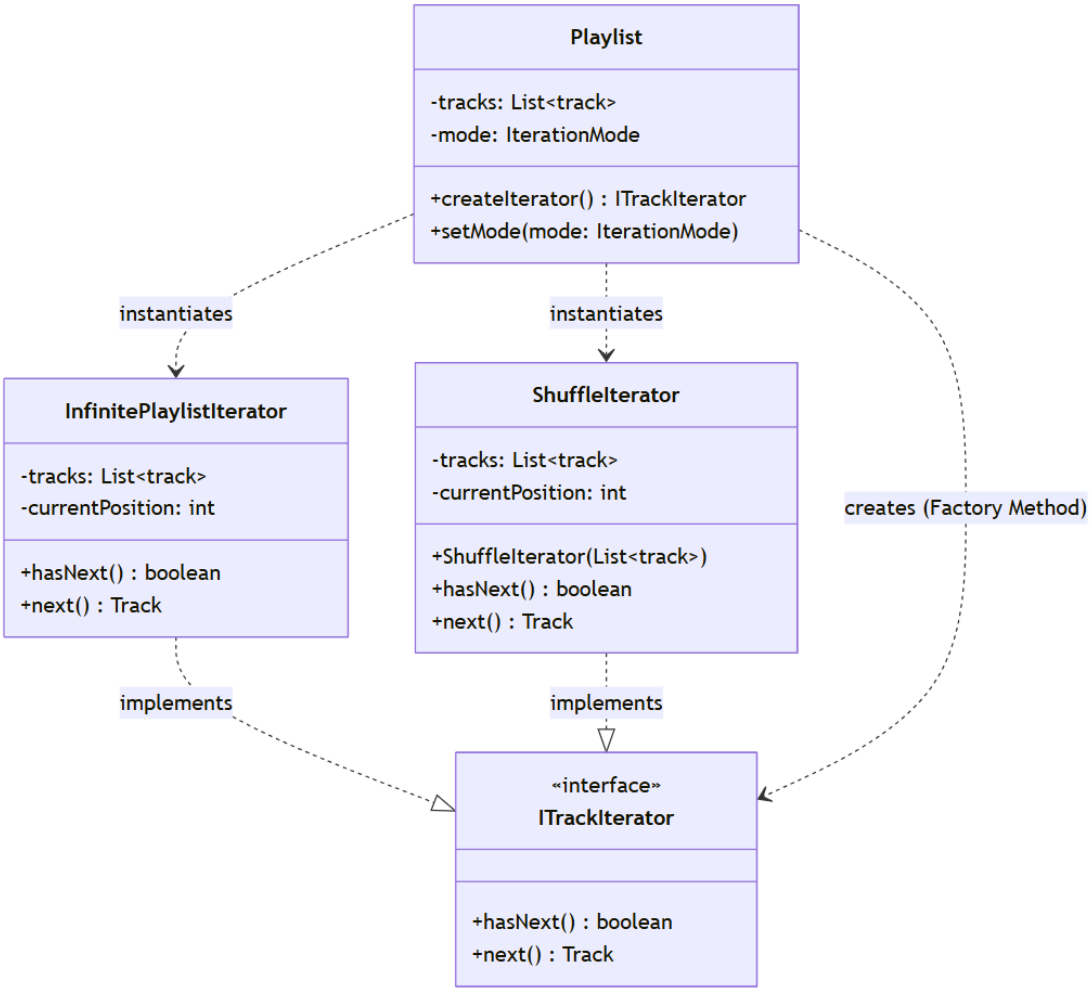
Знімок (Memento): Поведінковий патерн, що дозволяє зберігати та відновлювати попередній стан об'єкта, не порушуючи інкапсуляцію.

Спостерігач (Observer): Поведінковий патерн, який визначає залежність "один-до-багатьох" між об'єктами так, що при зміні стану одного об'єкта всі залежні від нього сповіщаються та оновлюються автоматично.

Декоратор (Decorator): Структурний патерн, який дозволяє динамічно додавати об'єктам нову функціональність, загортаючи їх у корисні "обгортки".

Хід роботи

Діаграма класів реалізації патерну «Factory Method»



Для реалізації функціоналу зміни режимів відтворення (послідовне, випадкове) у системі "Online Radio Station" було обрано патерн Factory Method. Роль "Творця" (Creator) виконує клас Playlist, який вирішує, який саме ітератор створити на основі обраного режиму.

Фрагменти коду по реалізації шаблону

1. ITrackIterator (Product Interface): Інтерфейс, який визначає загальні методи для обходу колекції треків.

```
package ua.kpi.iasa.onlineradio.models.iterators;

import ua.kpi.iasa.onlineradio.models.Track;
public interface ITrackIterator {
    boolean hasNext();
    Track next();
}
```

2. ShuffleIterator (Concrete Product): Нова реалізація ітератора для випадкового відтворення.

```
package ua.kpi.iasa.onlineradio.models.iterators;

import ua.kpi.iasa.onlineradio.models.Track;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class ShuffleIterator implements ITrackIterator {
    private final List<Track> tracks;
    private int currentPosition = 0;

    public ShuffleIterator(List<Track> originalTracks) {
        this.tracks = new ArrayList<>(originalTracks);
        Collections.shuffle(this.tracks);
    }

    @Override
    public boolean hasNext() {
        return currentPosition < tracks.size();
    }

    @Override
    public Track next() {
        if (!hasNext()) {
            return null;
        }
        Track track = tracks.get(currentPosition);
        currentPosition++;
        return track;
    }
}
```

3. Playlist (Creator): Клас, що містить фабричний метод createIterator(). Він створює відповідний об'єкт ітератора залежно від внутрішнього стану (mode).

```
public ITrackIterator createIterator() {
    switch (this.mode) {
        case SHUFFLE:
            System.out.println("Playlist: Створення Shuffle ітератора.");
            return new ShuffleIterator(this.tracks);
        case ONE_TIME:
            System.out.println("Playlist: Створення One-time ітератора (для
простоти насправді Infinite).");
            return new InfinitePlaylistIterator(this.tracks);
        case INFINITE:
        default:
            System.out.println("Playlist: Створення Infinite ітератора.");
            return new InfinitePlaylistIterator(this.tracks);
    }
}
```

Висновок:

У ході виконання лабораторної роботи було вивчено принципи роботи патернів проєктування: «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator».

На практиці для системи "Online Radio Station" було реалізовано патерн Factory Method. Це дозволило інкапсулювати логіку створення алгоритмів обходу плейлиста (ітераторів) всередині класу Playlist. Завдяки цьому клієнтський код (Streamer) залишається незалежним від конкретних класів ітераторів, що спрощує додавання нових режимів відтворення (наприклад, перемішування треків) у майбутньому без зміни основної логіки програми.