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
«Патерни проектування.»

Виконав:

Студент групи ІА-32

Костінський Макарій

Перевірив:

Мягкий Михайло Юрійович

Зміст

Тема: Патерни проектування.

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

Теоретичні відомості

Патерни проєктування — це типові рішення поширених проблем у розробці програмного забезпечення.

Адаптер (Adapter): Структурний патерн, який дозволяє об'єктам з несумісними інтерфейсами працювати разом. Він діє як "перекладач" між двома різними інтерфейсами.

Будівельник (Builder): Твірний патерн, який дозволяє створювати складні об'єкти покроково. Дає змогу використовувати один і той самий код будівництва для отримання різних представлень об'єкта.

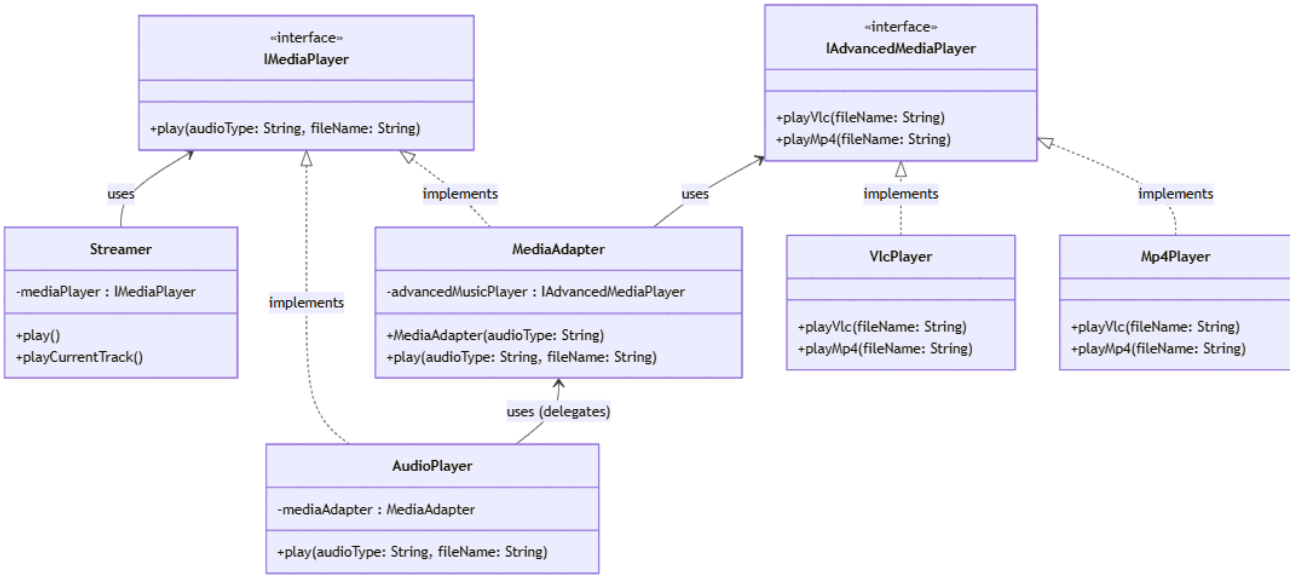
Команда (Command): Поведінковий патерн, який перетворює запит на об'єкт, дозволяючи параметризувати клієнтів чергами запитів, логувати їх та підтримувати скасування операцій.

Ланцюжок відповідальності (Chain of Responsibility): Поведінковий патерн, що дозволяє передавати запити послідовно ланцюжком обробників. Кожен обробник вирішує, чи може він обробити запит, чи слід передати його далі.

Прототип (Prototype): Твірний патерн, що дозволяє копіювати об'єкти, не вдаючись у подробиці їхньої реалізації (клонування замість створення з нуля).

Хід роботи

Діаграма класів реалізації патерну «Adapter»



На діаграмі зображено структуру класів, що реалізують патерн Adapter для забезпечення сумісності між стандартним аудіо-плеєром та новими форматами медіа-файлів.

Фрагменти коду по реалізації шаблону

IMediaPlayer (Target): Цільовий інтерфейс, з яким взаємодіє основна система (клас Streamer). Визначає метод play() для відтворення файлів.

```
public interface IMediaPlayer {  4 usages  2 implementations
    void play(String audioType, String fileName);  2 usages  2 implementations
}
```

AudioPlayer (Concrete Client): Клас, що реалізує стандартне відтворення (наприклад, mp3). У разі зустрічі з невідомим форматом (vlc, mp4), він використовує MediaAdapter для делегування задачі.

```
public class AudioPlayer implements IMediaPlayer {  2 usages
    private MediaAdapter mediaAdapter;  2 usages

    @Override  2 usages
    public void play(String audioType, String fileName) {
        // Вбудована підтримка mp3 (імітація старого коду)
        if (audioType.equalsIgnoreCase( anotherString: "mp3")) {
            System.out.println("Built-in Player: Програвання mp3 файлу. Ім'я: " + fileName);
        }
        // Використання адаптера для інших форматів
        else if (audioType.equalsIgnoreCase( anotherString: "vlc") || audioType.equalsIgnoreCase( anotherString: "mp4")) {
            mediaAdapter = new MediaAdapter(audioType);
            mediaAdapter.play(audioType, fileName);
        }
        else {
            System.out.println("Invalid media. " + audioType + " формат не підтримується.");
        }
    }
}
```

MediaAdapter (Adapter): Ключовий елемент патерну. Він реалізує інтерфейс IMediaPlayer, але всередині перетворює виклики методу play() у виклики специфічних методів інтерфейсу IAdvancedMediaPlayer.

```
public class MediaAdapter implements IMediaPlayer {  2 usages

    private IAdvancedMediaPlayer advancedMusicPlayer;  4 usages

    public MediaAdapter(String audioType) {  1 usage
        if (audioType.equalsIgnoreCase( anotherString: "vlc")) {
            advancedMusicPlayer = new VlcPlayer();
        } else if (audioType.equalsIgnoreCase( anotherString: "mp4")) {
            advancedMusicPlayer = new Mp4Player();
        }
    }

    @Override  2 usages
    public void play(String audioType, String fileName) {
        if (audioType.equalsIgnoreCase( anotherString: "vlc")) {
            advancedMusicPlayer.playVlc(fileName);
        } else if (audioType.equalsIgnoreCase( anotherString: "mp4")) {
            advancedMusicPlayer.playMp4(fileName);
        }
    }
}
```

IAdvancedMediaPlayer (Adaptee Interface): Інтерфейс для роботи з просунутішими форматами, який є несумісним зі стандартним IMediaPlayer.

```
public interface IAdvancedMediaPlayer {  3 usages  2 implementations
    void playVlc(String fileName);  1 usage  2 implementations
    void playMp4(String fileName);  1 usage  2 implementations
}
```

VlcPlayer та Mp4Player (Concrete Adaptees): Конкретні класи, що містять логіку відтворення специфічних форматів.

```
public class VlcPlayer implements IAdvancedMediaPlayer {  1 usage
    @Override  1 usage
    public void playVlc(String fileName) {
        System.out.println("Advanced Player: Програвання vlc файлу. Ім'я: " + fileName);
    }

    @Override  1 usage
    public void playMp4(String fileName) {
        // Нічого не робимо
    }
}
```

```java
public class Mp4Player implements IAdvancedMediaPlayer {  1 usage
    @Override  1 usage
    public void playVlc(String fileName) {
        // Нічого не робимо
    }

    @Override  1 usage
    public void playMp4(String fileName) {
        System.out.println("Advanced Player: Програвання mp4 файлу. Ім'я: " + fileName);
    }
}
```

Streamer: Клієнтський код радіостанції, який працює виключно з інтерфейсом IMediaPlayer, не знаючи про існування адаптерів та складних внутрішніх плеєрів.

```java
public void play() {  1 usage   Makarii-IA-32 *
    if (trackIterator == null || !trackIterator.hasNext()) {
        System.out.println("Помилка: плейлист не встановлено або він порожній.");
        return;
    }

    if (currentTrack == null) {
        nextTrack();
    } else {
        // Замість простого виводу в консоль, викликаємо наш плеєр
        playCurrentTrack();
    }
}
```

```java
// Метод для запуску програвання через Adapter
private void playCurrentTrack() {  2 usages  new *
    String filePath = currentTrack.getFilePath();
    // Проста логіка визначення формату за розширенням файлу
    String audioType = "mp3";

    if (filePath != null && filePath.contains(".")) {
        audioType = filePath.substring( beginIndex: filePath.lastIndexOf( str: ".") + 1);
    }

    System.out.println("Спроба відтворення: " + currentTrack.getArtist() + " - " + currentTrack.getTitle());
    // Викликаємо метод play нашого AudioPlayer
    mediaPlayer.play(audioType, filePath);
}
```

```
Спроба відтворення: Queen - Bohemian Rhapsody
Built-in Player: Програвання mp3 файлу. Ім'я: music/queen.mp3
Спроба відтворення: Nirvana - Smells Like Teen Spirit
Advanced Player: Програвання mp4 файлу. Ім'я: music/nirvana.mp4
```

Висновок:

У ході виконання лабораторної роботи засвоєно принципи використання структурних, твірних та поведінкових патернів проєктування: «Adapter», «Builder», «Command», «Chain of responsibility», «Prototype». Для розширення функціональних можливостей системи обрано та реалізовано патерн «Adapter». Це дозволило інтегрувати підтримку нових медіа-форматів (VLC, MP4) без зміни існуючого коду стрімера, забезпечивши сумісність між старим інтерфейсом програвача та новими бібліотеками відтворення.