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 №8

Теорія розробки програмного забезпечення

ШАБЛОН «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD».

Тема: «Музичний програвач»

Виконав
студент групи ІА–14:
Галаган Є.В.

Київ 2023

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 аудіоформатів, еквалайзер.

Завдання.

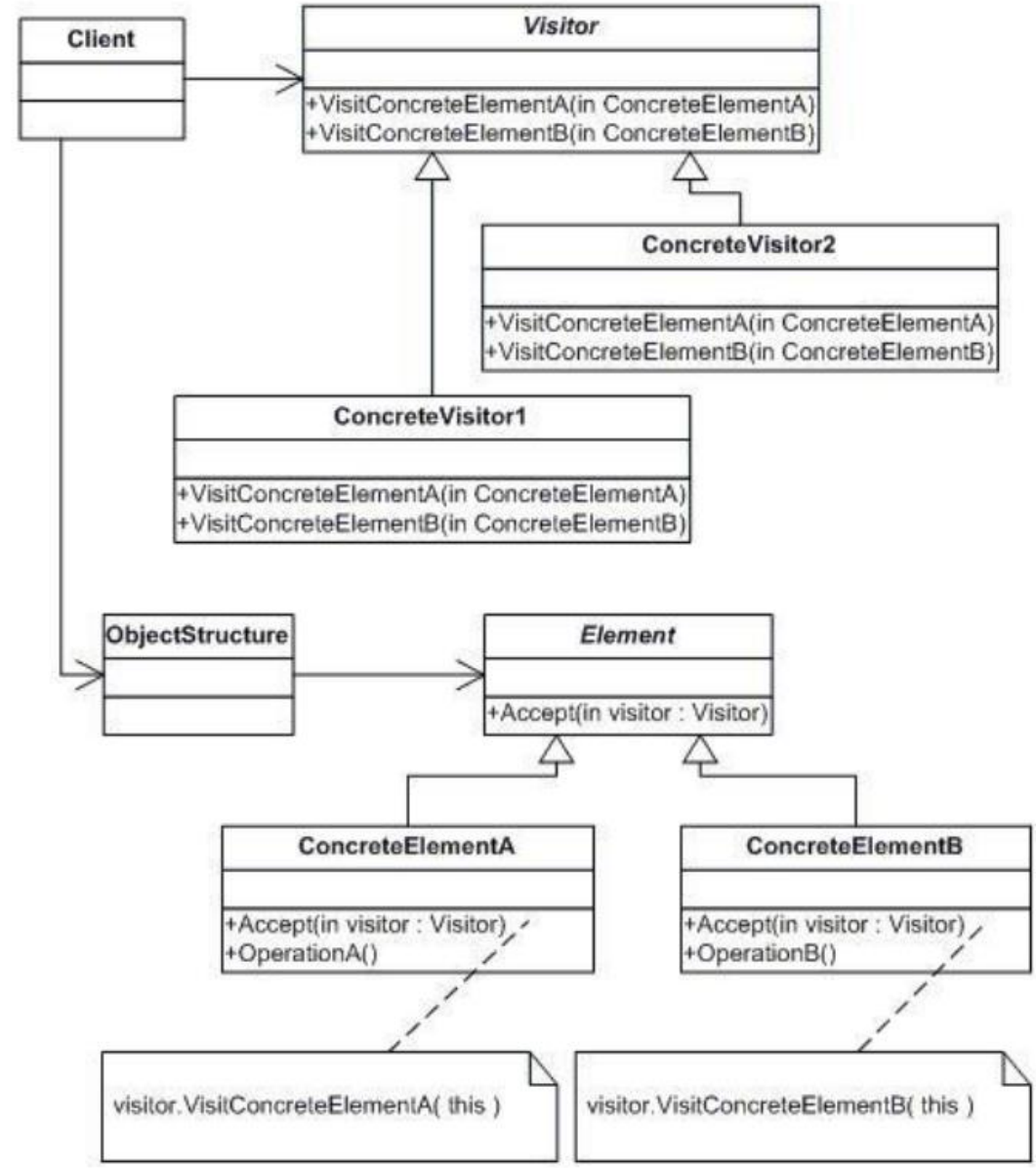
• Ознайомитися з короткими теоретичними відомостями.

• Реалізувати частину функціонала робочої програми у вигляді класів і їх взаємодій для досягнення конкретних функціональних можливостей.

• Застосування одного з даних шаблонів при реалізації програми.

Шаблон «Visitor»

Структура:



В даному випадку, для мого завдання було вирішено зробити вісітор для перегляду даних кожного типу об’єкту (пісня, еквалайзер, плейліст).

```java
package visitor;

import entity.Equalizer;
import entity.Playlist;
import entity.Song;


1 usage  1 implementation  new *
public interface IMediaPlayerVisitor {
    1 usage  1 implementation  new *
    public void visitPlaylist(Playlist playlist);
    2 usages  1 implementation  new *
    public void visitSong(Song song);
    1 usage  1 implementation  new *
    public void visitEqualizer(Equalizer equalizer);
}
```

Інтерфейс класу вісітор

```java
public class MediaPlayerVisitor implements IMediaPlayerVisitor {
    1 usage
    EqualizerService equalizerService;
    2 usages
    PlaylistService playlistService;
    3 usages
    SongService songService;

    1 usage  new *
    public MediaPlayerVisitor(EqualizerService equalizerService, PlaylistService playlistService, SongService songService) {
        this.equalizerService = equalizerService;
        this.playlistService = playlistService;
        this.songService = songService;
    }

    1 usage  new *
    @Override
    public void visitPlaylist(Playlist playlist) {
        System.out.println(playlist.getName());
        System.out.println("Пісні плейліста:");
        SongIterator songIterator = new SongIterator(playlist.getSongs());
        PlaylistShuffleCommand playlistShuffleCommand = new PlaylistShuffleCommand(playlistService);
        while (songIterator.hasNext()) {
            visitSong(songIterator.getNext());

        }
    }

    2 usages  new *
    @Override
    public void visitSong(Song song) {
        System.out.println(song.getName() + " " + song.getGenre());
        SongPlayCommand songPlayCommand = new SongPlayCommand(song, songService);
        SongDeleteCommand songDeleteCommand = new SongDeleteCommand(song, songService);
    }

    1 usage  new *
    @Override
    public void visitEqualizer(Equalizer equalizer) {
        System.out.println(equalizer.getName());
        System.out.println(equalizer.getBassBooster());
        System.out.println(equalizer.getBassBooster());
    }
}
```

Реалізація (консольний вивід даних + для пісні створюються кнопки, як зазвичай у медіаплеєрах, пісня + кнопка її прослуховування)

Увесь цей функціонал застосовується у фасаді (Лаб7)

```
no usages  new *
@Override
public void openPlaylist(long id) throws SQLException {
    playlistService.openPlaylist(id);
    mediaPlayerVisitor.visitPlaylist(playlistService.getOpenedPlaylist());
}

no usages  new *
@Override
public void openSong(long id) {
    mediaPlayerVisitor.visitSong(playlistService.openSongFromPlaylist(id));
}

no usages  new *
@Override
public void openEqualizer(long id) throws SQLException {
    equalizerService.chooseEqualizer(id);
    mediaPlayerVisitor.visitEqualizer(equalizerService.getChosenEqualizer());
}
```