

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій

Кафедра комп'ютерної інженерії та інформаційних систем

## **Лабораторна робота № 5**

з дисципліни «Об'єктно-орієнтовані технології програмування»

на тему:

**«Поведінкові патерни»**

**Виконав:**

студент 1 курсу, групи КІ2м-23-2

\_\_\_\_\_ Мариняк В.М.

**Перевірив:**

\_\_\_\_\_ Лисенко С.М.

## Хід роботи

Розглянемо варіант реалізації патернів Ітератор та Стан на прикладі застосунку для відтворення музики, де перший буде використано для почергового програвання пісень, а другий для зміни їхнього стану (програвання, пауза тощо) відповідно.

### Технічне завдання:

1. Визначити інтерфейси `IIterator` та `IAggregate` згідно принципів патерну `Iterator`.
2. Створити конкретні класи `Playlist`, `PlaylistIterator`, що реалізують вказані інтерфейси, та клас `Track`, який використовуватиметься в плейлисті.
3. Визначити інтерфейс `IState` згідно принципів патерну `State`.
4. Створити конкретні класи станів музичного програвача: `Play`, `Pause`, `Idle`, що реалізовуватимуть вказаний інтерфейс.
5. Створити клас `MusicPlayer`, що працюватиме з плейлистами, треками та станом програвання музики.

### Реалізація:

1. Визначення інтерфейсів `IIterator` та `IAggregate`

```
1 namespace IteratorAndStateExample.Maryniak.Models;
2
3 public interface IAggregate<T>
4 {
5     IIterator<T> GetIterator();
6 }
```

```
1 namespace IteratorAndStateExample.Maryniak.Models;
2
3 public interface IIterator<T>
4 {
5     T? Current { get; }
6     bool HasNext { get; }
7     T? Next();
8
9     void Restart();
10 }
```

2. Реалізація класів Playlist, PlaylistIterator, що реалізують відповідні інтерфейси, та решти суміжних класів

```
1 namespace IteratorAndStateExample.Maryniak.Models;
2
3 public class Playlist(string title) : IAggregate<Track>
4 {
5     public string Title { get; set; } = title;
6
7     private readonly List<Track> tracks = [];
8
9     public void AddTrack(Track track) => tracks.Add(track);
10
11     public IIterator<Track> GetIterator() => new PlaylistIterator(tracks);
12 }
```

```
1 namespace IteratorAndStateExample.Maryniak.Models;
2
3 public class PlaylistIterator(List<Track> tracks) : IIterator<Track>
4 {
5     private int position = 0;
6
7     public Track? Current => tracks.ElementAtOrDefault(position);
8
9     public bool HasNext => position < tracks.Count;
10
11     public Track? Next()
12         => HasNext ? tracks[position++] : null;
13
14     public void Restart() => position = 0;
15 }
```

```
1 namespace IteratorAndStateExample.Maryniak.Models;
2
3 public class Track(string title)
4 {
5     public string Title { get; set; } = title;
6 }
```

### 3. Визначення інтерфейсу IState

```
1 namespace IteratorAndStateExample.Maryniak.Models;
2
3 public interface IState
4 {
5     void Play(MusicPlayer context);
6     void Pause(MusicPlayer context);
7     void NextTrack(MusicPlayer context);
8 }
9
```

## 4. Реалізація конкретних станів музичного програвача

```
1 namespace IteratorAndStateExample.Maryniak.Models;
2
3 public class IdleState : IState
4 {
5     public void NextTrack(MusicPlayer context)
6     {
7         context.CurrentTrack = context.CurrentPlaylistIterator?.Next();
8
9         if (context.CurrentTrack is null)
10        {
11            Console.WriteLine($"Starting the next track... The playlist is over :(");
12            context.SetState(new IdleState());
13        }
14        else
15        {
16            Console.WriteLine($"Starting the {context.CurrentTrack?.Title ?? "Default Track"}");
17            context.SetState(new PlayingState());
18        }
19    }
20
21    public void Pause(MusicPlayer context)
22    {
23        Console.WriteLine($"There is nothing to pause in the Idle state.");
24    }
25
26    public void Play(MusicPlayer context)
27    {
28        context.CurrentTrack = context.CurrentPlaylistIterator?.Next();
29
30        if (context.CurrentTrack is null)
31        {
32            Console.WriteLine($"Starting the track... The playlist is over :(");
33            context.SetState(new IdleState());
34        }
35        else
36        {
37            Console.WriteLine($"Starting the track - {context.CurrentTrack?.Title ?? "Default Track"}");
38            context.SetState(new PlayingState());
39        }
40    }
41 }
```

```
1 namespace IteratorAndStateExample.Maryniak.Models;
2
3 public class PlayingState : IState
4 {
5     public void NextTrack(MusicPlayer context)
6     {
7         context.CurrentTrack = context.CurrentPlaylistIterator?.Next();
8
9         if (context.CurrentTrack is null)
10        {
11            Console.WriteLine($"Skipping to the next track... The playlist is over :(");
12            context.SetState(new IdleState());
13        }
14        else
15        {
16            Console.WriteLine($"Skipping to the {context.CurrentTrack?.Title ?? "Default Track"}");
17            context.SetState(new PlayingState());
18        }
19    }
```

```

21     public void Pause(MusicPlayer context)
22     {
23         Console.WriteLine($"Pausing the {context.CurrentTrack?.Title ?? "Default Track"}.");
24         context.SetState(new PausedState());
25     }
26
27     public void Play(MusicPlayer context)
28     => Console.WriteLine($"The {context.CurrentTrack?.Title ?? "Default Track"} is already playing.");
29 }

1  namespace IteratorAndStateExample.Maryniak.Models;
2
3  public class PausedState : IState
4  {
5      public void Play(MusicPlayer context)
6      {
7          Console.WriteLine($"Resuming the {context.CurrentTrack?.Title ?? "Default Track"}.");
8          context.SetState(new PlayingState());
9      }
10
11     public void Pause(MusicPlayer context)
12     {
13         Console.WriteLine($"The {context.CurrentTrack?.Title ?? "Default Track"} is already paused.");
14     }
15
16     public void NextTrack(MusicPlayer context)
17     {
18         context.CurrentTrack = context.CurrentPlaylistIterator?.Next();
19
20         if (context.CurrentTrack is null)
21         {
22             Console.WriteLine($"Skipping to the next track... The playlist is over :(");
23             context.SetState(new IdleState());
24         }
25         else
26         {
27             Console.WriteLine($"Skipping to the next track {context.CurrentTrack?.Title ?? "Default Track"}");
28             context.SetState(new PlayingState());
29         }
30     }
31 }

```

## 5. Реалізація конкретного класу MusicPlayer

```

1  namespace IteratorAndStateExample.Maryniak.Models;
2
3  public class MusicPlayer
4  {
5      private readonly Dictionary<string, Playlist> playlists = [];
6
7      public IIterator<Track>? CurrentPlaylistIterator { get; private set; }
8      public Track? CurrentTrack { get; set; }
9
10     public void SetCurrentPlaylist(string name)
11     => CurrentPlaylistIterator = playlists[name].GetIterator();

```

```

13     private IState state = new IdleState();
14
15     public void SetState(IState state) => this.state = state;
16
17     public void Play() => state.Play(this);
18
19     public void Pause() => state.Pause(this);
20
21     public void NextTrack() => state.NextTrack(this);
22
23     public void AddPlaylist(string name, Playlist playlist)
24         => playlists.Add(name, playlist);
25 }

```

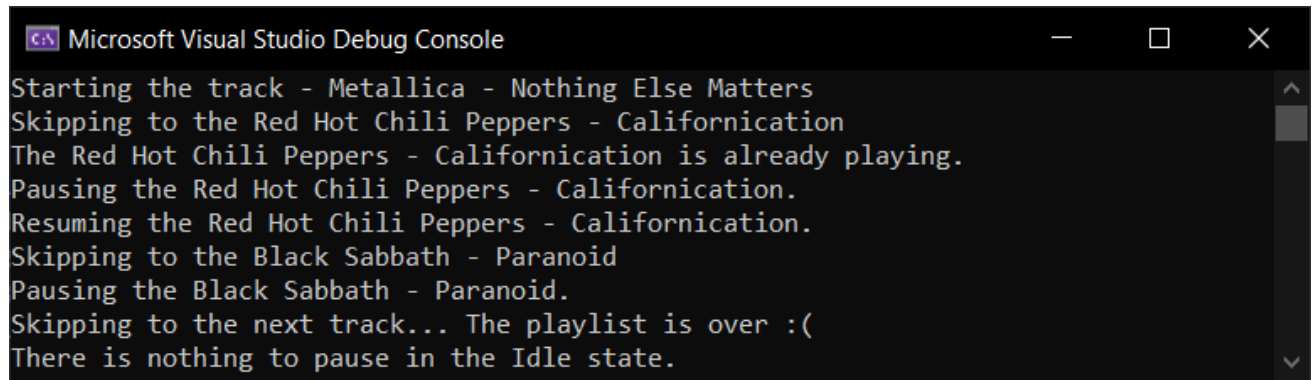
## 6. Реалізація клієнтського коду

```

1  using IteratorAndStateExample.Maryniak.Models;
2
3  namespace IteratorAndStateExample.Maryniak;
4
5  internal class Program
6  {
7      static void Main()
8      {
9          var player = new MusicPlayer();
10
11          var playlist = new Playlist(title: "Classic");
12          playlist.AddTrack(new Track("Metallica - Nothing Else Matters"));
13          playlist.AddTrack(new Track("Red Hot Chili Peppers - Californication"));
14          playlist.AddTrack(new Track("Black Sabbath - Paranoid"));
15          player.AddPlaylist(playlist.Title, playlist);
16
17          player.SetCurrentPlaylist(playlist.Title);
18
19          player.Play();
20          player.NextTrack();
21          player.Play();
22          player.Pause();
23          player.Play();
24          player.NextTrack();
25          player.Pause();
26          player.NextTrack();
27          player.Pause();
28      }
29 }

```

## 7. Результат виконання програми



```
Microsoft Visual Studio Debug Console
Starting the track - Metallica - Nothing Else Matters
Skipping to the Red Hot Chili Peppers - Californication
The Red Hot Chili Peppers - Californication is already playing.
Pausing the Red Hot Chili Peppers - Californication.
Resuming the Red Hot Chili Peppers - Californication.
Skipping to the Black Sabbath - Paranoid
Pausing the Black Sabbath - Paranoid.
Skipping to the next track... The playlist is over :(
There is nothing to pause in the Idle state.
```

**Висновок.** Під час лабораторної роботи я навчився реалізовувати патерни Ітератор і Стан та створив консольний додаток, який демонструє принципи їхньої роботи.