

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.15.000 – Лр.3		
Змн.	Арк.	№ докум.	Підпис	Дата	SupportHandler		
Розроб.	Нагаль О.С.				Літ.	Арк.	Аркуші
Перевір.						1	9
Керівник					ФІКТ, гр. ВТ-22-1		
Н. контр.							
Затверд.							

```

{
    public override void HandleRequest(string request)
    {
        if (request == "level1")
        {
            Console.WriteLine("Your issue is being resolved at Level One Support.");
        }
        else
        {
            base.HandleRequest(request);
        }
    }
}

public class LevelTwoSupport : SupportHandler
{
    public override void HandleRequest(string request)
    {
        if (request == "level2")
        {
            Console.WriteLine("Your issue is being resolved at Level Two Support.");
        }
        else
        {
            base.HandleRequest(request);
        }
    }
}

public class LevelThreeSupport : SupportHandler
{
    public override void HandleRequest(string request)
    {
        if (request == "level3")
        {
            Console.WriteLine("Your issue is being resolved at Level Three Support.");
        }
        else
        {
            base.HandleRequest(request);
        }
    }
}

public class LevelFourSupport : SupportHandler
{
    public override void HandleRequest(string request)
    {
        if (request == "level4")
        {
            Console.WriteLine("Your issue is being resolved at Level One Support.");
        }
        else
        {
            base.HandleRequest(request);
        }
    }
}

class Program
{
    static void Main(string[] args)
    {
        SupportHandler level1 = new LevelOneSupport();
        SupportHandler level2 = new LevelTwoSupport();
        SupportHandler level3 = new LevelThreeSupport();
    }
}

```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.15.000 – Лр.3	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		2

```

SupportHandler level4 = new LevelFourSupport();
level1.SetNextHandler(level2).SetNextHandler(level3).SetNextHandler(level4);

string request = "";

while (request != "exit")
{
    Console.WriteLine("Please enter your support request (level1, level2, level3, level4), or 'exit' to quit:");
    request = Console.ReadLine();
    level1.HandleRequest(request);
}
}

```

Результат:

```

Please enter your support request (level1, level2, level3, level4), or 'exit' to quit:
level1
Your issue is being resolved at Level One Support.
Please enter your support request (level1, level2, level3, level4), or 'exit' to quit:
level2
Your issue is being resolved at Level Two Support.
Please enter your support request (level1, level2, level3, level4), or 'exit' to quit:
level3
Your issue is being resolved at Level Three Support.
Please enter your support request (level1, level2, level3, level4), or 'exit' to quit:
level4
Your issue is being resolved at Level One Support.
Please enter your support request (level1, level2, level3, level4), or 'exit' to quit:

```

Завдання 2: Посередник.

1. Відрефакторте код продемонстрований на лекції за допомогою використання шаблону Посередник.
2. В результаті рефакторингу Aircraft не повинен “знати” про Runway і навпаки. Обидві сутності повинні “знати” лише про CommandCentre.
3. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

Код:

```

using System;
using System.Collections.Generic;

namespace DesignPatterns.Mediator
{
    class CommandCentre
    {
        private List<Runway> _runways = new List<Runway>();
        private List<Aircraft> _aircrafts = new List<Aircraft>();

        public CommandCentre()
        {
        }

        public void AddRunway(Runway runway)
        {
        }
    }
}

```

```

    {
        _runways.Add(runway);
    }

    public void AddAircraft(Aircraft aircraft)
    {
        _aircrafts.Add(aircraft);
    }

    public void RequestLanding(Aircraft aircraft)
    {
        foreach (var runway in _runways)
        {
            if (!runway.CheckIsActive())
            {
                aircraft.Land(runway);
                return;
            }
        }
        Console.WriteLine($"All runways are busy. {aircraft.Name} cannot land at the
moment.");
    }

    public void RequestTakeoff(Aircraft aircraft)
    {
        foreach (var runway in _runways)
        {
            if (runway.IsBusyWithAircraft == aircraft)
            {
                aircraft.TakeOff(runway);
                return;
            }
        }
        Console.WriteLine($" {aircraft.Name} cannot take off. It is not on any
runway.");
    }
}

class Runway
{
    public readonly Guid Id = Guid.NewGuid();
    public Aircraft? IsBusyWithAircraft;

    public bool CheckIsActive()
    {
        return IsBusyWithAircraft == null;
    }

    public void HighLightRed()
    {
        Console.WriteLine($"Runway {Id} is busy!");
    }

    public void HighLightGreen()
    {
        Console.WriteLine($"Runway {Id} is free!");
    }
}

class Aircraft
{
    public string Name { get; }

    public Aircraft(string name)

```

```

    {
        Name = name;
    }

    public void Land(Runway runway)
    {
        Console.WriteLine($"Aircraft {Name} is landing.");
        Console.WriteLine($"Checking runway.");
        if (runway.IsBusyWithAircraft == null)
        {
            Console.WriteLine($"Aircraft {Name} has landed.");
            runway.IsBusyWithAircraft = this;
            runway.HighLightRed();
        }
        else
        {
            Console.WriteLine($"Could not land, the runway is busy.");
        }
    }

    public void TakeOff(Runway runway)
    {
        Console.WriteLine($"Aircraft {Name} is taking off.");
        runway.IsBusyWithAircraft = null;
        runway.HighLightGreen();
        Console.WriteLine($"Aircraft {Name} has taken off.");
    }
}

class Program
{
    static void Main(string[] args)
    {
        CommandCentre commandCentre = new CommandCentre();

        Runway runway1 = new Runway();
        Runway runway2 = new Runway();
        Runway runway3 = new Runway();

        commandCentre.AddRunway(runway1);
        commandCentre.AddRunway(runway2);
        commandCentre.AddRunway(runway3);

        Aircraft aircraft1 = new Aircraft("Boeing 747");
        Aircraft aircraft2 = new Aircraft("Airbus A320");

        commandCentre.AddAircraft(aircraft1);
        commandCentre.AddAircraft(aircraft2);

        commandCentre.RequestLanding(aircraft1);
        commandCentre.RequestLanding(aircraft2);

        commandCentre.RequestTakeoff(aircraft1);
        commandCentre.RequestTakeoff(aircraft2);
    }
}

```

Результат:

```
All runways are busy. Boeing 747 cannot land at the moment.  
All runways are busy. Airbus A320 cannot land at the moment.  
Boeing 747 cannot take off. It is not on any runway.  
Airbus A320 cannot take off. It is not on any runway.
```

Завдання 3: Спостерігач.

1. Додайте до Вашого LightHTML з завдання 5 ЛР №3 можливість додавання EventListener до Ваших HTML елементів.
2. Додайте можливість підписки на різні івенти ("click", "mouseover" тощо).
3. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

Код:

```
using System;  
using System.Collections.Generic;  
using System.Text;  
  
public abstract class LightNode  
{  
    public abstract string OuterHTML { get; }  
    public abstract string InnerHTML { get; }  
    public abstract void AddEventListener(string eventType, EventHandler eventHandler);  
    public abstract void RemoveEventListener(string eventType, EventHandler eventHandler);  
}  
  
public class LightTextNode : LightNode  
{  
    private readonly string _text;  
  
    public LightTextNode(string text)  
    {  
        _text = text;  
    }  
  
    public override string OuterHTML => _text;  
    public override string InnerHTML => _text;  
  
    public override void AddEventListener(string eventType, EventHandler eventHandler)  
    {  
        // Текстовий вузол не підтримує події  
    }  
  
    public override void RemoveEventListener(string eventType, EventHandler eventHandler)  
    {  
        // Текстовий вузол не підтримує події  
    }  
}  
  
public class LightElementNode : LightNode  
{  
    private readonly string _tag;
```

```

private readonly bool _blockType;
private readonly bool _selfClosing;
private readonly List<string> _classes;
private readonly List<LightNode> _children;
private readonly Dictionary<string, List<EventHandler>> _eventListeners;

public LightElementNode(string tag, bool blockType, bool selfClosing, List<string>
classes, List<LightNode> children)
{
    _tag = tag;
    _blockType = blockType;
    _selfClosing = selfClosing;
    _classes = classes;
    _children = children;
    _eventListeners = new Dictionary<string, List<EventHandler>>();
}

public override string OuterHTML
{
    get
    {
        StringBuilder builder = new StringBuilder();
        builder.Append($"<{_tag}>");
        if (_classes.Count > 0)
        {
            builder.Append(" class=\"");
            builder.Append(string.Join(" ", _classes));
            builder.Append("\"");
        }
        builder.Append(">");
        if (!_selfClosing)
        {
            foreach (var child in _children)
            {
                builder.Append(child.OuterHTML);
            }
            builder.Append($"</{_tag}>");
        }
        return builder.ToString();
    }
}

public override string InnerHTML
{
    get
    {
        StringBuilder builder = new StringBuilder();
        foreach (var child in _children)
        {
            builder.Append(child.OuterHTML);
        }
        return builder.ToString();
    }
}

public override void AddEventListener(string eventType, EventHandler eventHandler)
{
    if (!_eventListeners.ContainsKey(eventType))
    {
        _eventListeners[eventType] = new List<EventHandler>();
    }
    _eventListeners[eventType].Add(eventHandler);
}

```

```

public override void RemoveEventListener(string eventType, EventHandler eventHandler)
{
    if (_eventListeners.ContainsKey(eventType))
    {
        _eventListeners[eventType].Remove(eventHandler);
    }
}

class Program
{
    static void Main(string[] args)
    {
        var title = new LightTextNode("Welcome to My Website");
        var paragraph1 = new LightTextNode("This is a simple paragraph.");
        var paragraph2 = new LightTextNode("This is another paragraph.");
        var listItems = new List<LightNode>
        {
            new LightTextNode("Item 1"),
            new LightTextNode("Item 2"),
            new LightTextNode("Item 3")
        };
        var unorderedList = new LightElementNode("ul", true, false, new List<string>(),
listItems);

        var bodyChildren = new List<LightNode> { paragraph1, paragraph2, unorderedList };
        var body = new LightElementNode("body", true, false, new List<string>(),
bodyChildren);

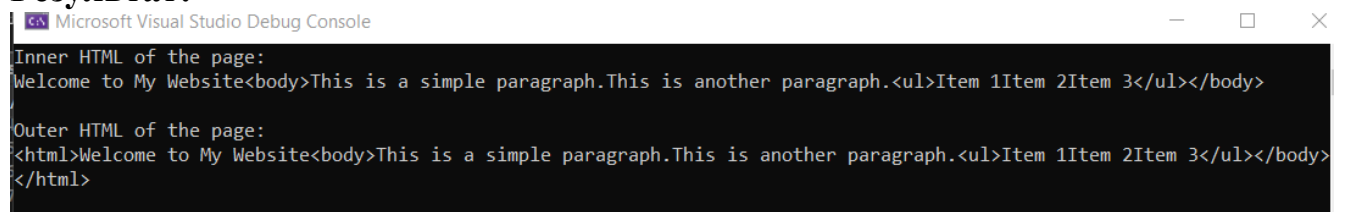
        var htmlChildren = new List<LightNode> { title, body };
        var html = new LightElementNode("html", true, false, new List<string>(),
htmlChildren);

        // Додамо подію click для заголовка
        title.AddEventListener("click", (sender, e) => Console.WriteLine("Title
clicked!"));

        Console.WriteLine("Inner HTML of the page:");
        Console.WriteLine(html.InnerHTML);
        Console.WriteLine();
        Console.WriteLine("Outer HTML of the page:");
        Console.WriteLine(html.OuterHTML);
    }
}

```

Результат:



```

Microsoft Visual Studio Debug Console
Inner HTML of the page:
Welcome to My Website<body>This is a simple paragraph.This is another paragraph.<ul>Item 1Item 2Item 3</ul></body>

Outer HTML of the page:
<html>Welcome to My Website<body>This is a simple paragraph.This is another paragraph.<ul>Item 1Item 2Item 3</ul></body>
</html>

```


Завдання 4: Стратегія.

1. Додайте новий елемент Image, який за допомогою стратегії в залежності від переданого href буде завантажувати картинку або з файлової системи або з мережі.
2. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

Код:

```
using System;

public interface IImageStrategy
{
    void LoadImage(string url);
}

public class FileSystemImageStrategy : IImageStrategy
{
    public void LoadImage(string url)
    {
        Console.WriteLine($"Loading image from file system: {url}");
    }
}

public class NetworkImageStrategy : IImageStrategy
{
    public void LoadImage(string url)
    {
        Console.WriteLine($"Loading image from network: {url}");
    }
}

public class Image
{
    private readonly string _url;
    private readonly IImageStrategy _strategy;

    public Image(string url, IImageStrategy strategy)
    {
        _url = url;
        _strategy = strategy;
    }

    public void Load()
    {
        _strategy.LoadImage(_url);
    }
}

class Program
{
    static void Main(string[] args)
    {
        Image fileSystemImage = new Image("path/to/image.jpg", new
        FileSystemImageStrategy());
        fileSystemImage.Load();

        Image networkImage = new Image("http://example.com/image.jpg", new
        NetworkImageStrategy());
        networkImage.Load();
    }
}
```

```
}  
}
```

Результат:

```
Loading image from file system: path/to/image.jpg  
Loading image from network: http://example.com/image.jpg
```

Завдання 5: Мементо.

1. Уявіть, що ви створюєте програму текстового редактора.
2. Для цього завдання буде достатньо РОС-версії: клас **TextEditor**, який містить в собі поточну версію текстового документу, який представлений спеціальним класом **TextDocument**. Назви класам можете давати власні.
3. Реалізуйте функцію збереження і скасування, яка дозволить користувачам відмінити зміни, зроблені в документі. Завдання полягає в тому, щоб зберегти стан документа в різні моменти часу та відновити його, коли це необхідно, не розкриваючи внутрішню реалізацію документа чи текстового редактору.

Код:

```
using System;  
using System.Collections.Generic;  
public class TextDocument  
{  
    public string Content { get; set; }  
  
    public TextDocument(string content)  
    {  
        Content = content;  
    }  
  
    public TextDocumentMemento CreateMemento()  
    {  
        return new TextDocumentMemento(Content);  
    }  
    public void RestoreMemento(TextDocumentMemento memento)  
    {  
        Content = memento.Content;  
    }  
}  
public class TextDocumentMemento  
{  
    public string Content { get; }  
    public TextDocumentMemento(string content)  
    {  
        Content = content;  
    }  
}  
public class TextEditor  
{  
    private readonly Stack<TextDocumentMemento> _mementos = new  
Stack<TextDocumentMemento>();  
    private readonly TextDocument _document;  
    public TextEditor(TextDocument document)
```

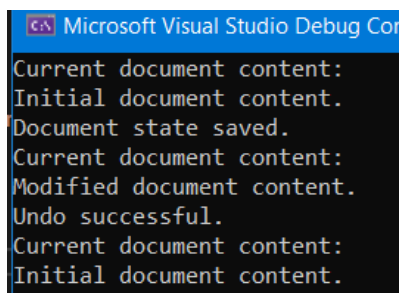
					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.15.000 – Лр.3	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		10

```

    {
        _document = document;
    }
    public void Save()
    {
        _mementos.Push(_document.CreateMemento());
        Console.WriteLine("Document state saved.");
    }
    public void Undo()
    {
        if (_mementos.Count > 0)
        {
            var memento = _mementos.Pop();
            _document.RestoreMemento(memento);
            Console.WriteLine("Undo successful.");
        }
        else
        {
            Console.WriteLine("No previous states to undo.");
        }
    }
    public void DisplayDocument()
    {
        Console.WriteLine("Current document content:");
        Console.WriteLine(_document.Content);
    }
}
class Program
{
    static void Main(string[] args)
    {
        var document = new TextDocument("Initial document content.");
        var editor = new TextEditor(document);
        editor.DisplayDocument();
        editor.Save();
        document.Content = "Modified document content.";
        editor.DisplayDocument();
        editor.Undo();
        editor.DisplayDocument();
    }
}

```

Результат:



```

Microsoft Visual Studio Debug Console
Current document content:
Initial document content.
Document state saved.
Current document content:
Modified document content.
Undo successful.
Current document content:
Initial document content.

```

Висновок: У ході виконання лабораторної роботи я навчився реалізовувати структурні шаблони проектування Ланцюжок відповідальностей, Посередник, Спостерігач, Стратегія, Мементо