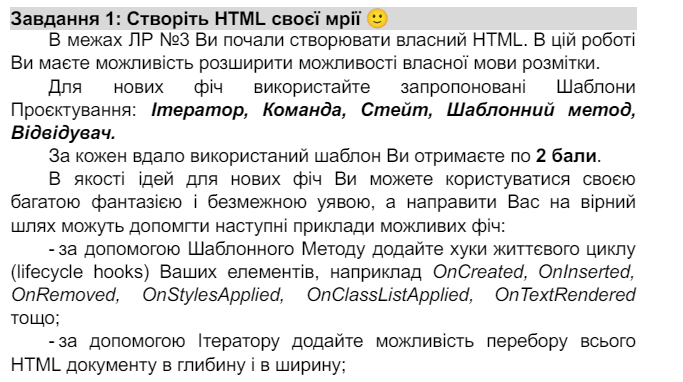
**Лабораторна робота №5**

**Тема: Поведінкові шаблони**

**Мета роботи:** навчитися реалізовувати структурні шаблони проєктування Ітератор, Команда, Стейт, Шаблонний метод, Відвідувач

**Хід Роботи**

**Репозиторій:** <https://github.com/Oleksandr-Nagal/KPZ>



Iterator:

Код:

using System;

using System.Collections;

using System.Collections.Generic;

public class HTMLElement

{

public string TagName { get; set; }

public List<HTMLElement> Children { get; set; } = new List<HTMLElement>();

public HTMLElement(string tagName)

{

TagName = tagName;

}

public void AddChild(HTMLElement child)

{

Children.Add(child);

}

}

public class ElementIterator : IEnumerator<HTMLElement>

{

private Stack<HTMLElement> stack = new Stack<HTMLElement>();

private HTMLElement current;

public ElementIterator(HTMLElement root)

{

stack.Push(root);

}

public bool MoveNext()

{

if (stack.Count == 0) return false;

current = stack.Pop();

for (int i = current.Children.Count - 1; i >= 0; i--)

{

stack.Push(current.Children[i]);

}

return true;

}

public void Reset()

{

stack.Clear();

}

public HTMLElement Current => current;

object IEnumerator.Current => Current;

public void Dispose() { }

}

public class Program

{

public static void Main(string[] args)

{

var root = new HTMLElement("root");

var child1 = new HTMLElement("div");

var child2 = new HTMLElement("p");

root.AddChild(child1);

root.AddChild(child2);

var iterator = new ElementIterator(root);

while (iterator.MoveNext())

{

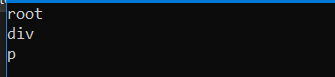
Console.WriteLine(iterator.Current.TagName);

}

}

}

Результат:



Comanda:

Код:

using System;

using System.Collections.Generic;

public abstract class Command

{

public abstract void Execute();

}

public class HTMLElement

{

public string TagName { get; set; }

public List<string> Classes { get; set; } = new List<string>();

public HTMLElement(string tagName)

{

TagName = tagName;

}

public void AddClass(string className)

{

Classes.Add(className);

}

}

public class AddClassCommand : Command

{

private HTMLElement \_element;

private string \_className;

public AddClassCommand(HTMLElement element, string className)

{

\_element = element;

\_className = className;

}

public override void Execute()

{

\_element.AddClass(\_className);

}

}

public class Program

{

public static void Main(string[] args)

{

var element = new HTMLElement("p");

var addClassCommand = new AddClassCommand(element, "highlight");

addClassCommand.Execute();

Console.WriteLine(string.Join(", ", element.Classes)); // Output: highlight

}

}

Результат:



Steyt:

Код:

using System;

public abstract class ElementState

{

protected HTMLElement \_element;

protected ElementState(HTMLElement element)

{

\_element = element;

}

public abstract void Enter();

public abstract void Exit();

}

public class VisibleState : ElementState

{

public VisibleState(HTMLElement element) : base(element) { }

public override void Enter()

{

\_element.Visible = true;

}

public override void Exit()

{

\_element.Visible = false;

}

}

public class HiddenState : ElementState

{

public HiddenState(HTMLElement element) : base(element) { }

public override void Enter()

{

\_element.Visible = false;

}

public override void Exit()

{

\_element.Visible = true;

}

}

public class HTMLElement

{

public string TagName { get; set; }

public bool Visible { get; set; } = true;

public HTMLElement(string tagName)

{

TagName = tagName;

}

}

public class Context

{

private ElementState \_currentState;

public void TransitionTo(ElementState state)

{

\_currentState?.Exit();

\_currentState = state;

\_currentState.Enter();

}

}

public class Program

{

public static void Main(string[] args)

{

// Usage

var element = new HTMLElement("p");

var context = new Context();

context.TransitionTo(new HiddenState(element));

Console.WriteLine(element.Visible); // Output: False

context.TransitionTo(new VisibleState(element));

Console.WriteLine(element.Visible); // Output: True

}

}

Результат:



Shablonuy metod:

Код:

using System;

public abstract class CustomElement

{

public CustomElement()

{

OnCreated();

}

public void Insert()

{

OnInserted();

}

public void Remove()

{

OnRemoved();

}

protected virtual void OnCreated() { }

protected virtual void OnInserted() { }

protected virtual void OnRemoved() { }

}

public class HTMLElement : CustomElement

{

public string TagName { get; set; }

public HTMLElement(string tagName)

{

TagName = tagName;

}

protected override void OnCreated()

{

Console.WriteLine($"{TagName} element created.");

}

protected override void OnInserted()

{

Console.WriteLine($"{TagName} element inserted.");

}

protected override void OnRemoved()

{

Console.WriteLine($"{TagName} element removed.");

}

}

public class Program

{

public static void Main(string[] args)

{

var element = new HTMLElement("div");

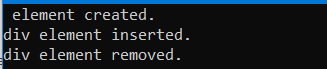
element.Insert();

element.Remove();

}

}

Результат:



Vidvidyvach:

Код:

using System;

using System.Collections.Generic;

public interface IVisitor

{

void Visit(HTMLElement element);

}

public class HighlightVisitor : IVisitor

{

public void Visit(HTMLElement element)

{

if (element.TagName == "p")

{

element.Highlighted = true;

}

}

}

public class HTMLElement

{

public string TagName { get; set; }

public bool Highlighted { get; set; } = false;

public List<HTMLElement> Children { get; set; } = new List<HTMLElement>();

public HTMLElement(string tagName)

{

TagName = tagName;

}

public void AddChild(HTMLElement child)

{

Children.Add(child);

}

public void Accept(IVisitor visitor)

{

visitor.Visit(this);

foreach (var child in Children)

{

child.Accept(visitor);

}

}

}

public class Program

{

public static void Main(string[] args)

{

var root = new HTMLElement("root");

var child1 = new HTMLElement("div");

var child2 = new HTMLElement("p");

root.AddChild(child1);

root.AddChild(child2);

var highlightVisitor = new HighlightVisitor();

root.Accept(highlightVisitor);

Console.WriteLine(child2.Highlighted); // Output: True

}

}

Результат:



**Висновок:** У ході виконання лабораторної роботи я навчися реалізовувати структурні шаблони проєктування Ітератор, Команда, Стейт, Шаблонний метод, Відвідувач.