

# Лабораторна робота №3

## Тема: Структурні шаблони

**Мета роботи:** навчитися реалізовувати структурні шаблони проєктування  
Адаптер, Декоратор, Міст, Компонувальник, Проксі, Легковаговик

### Хід Роботи

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

#### Завдання 1: Адаптер.

1. Створіть клас `Logger`, який буде мати методи `Log()`, `Error()`, `Warn()`, які виводять повідомлення в консоль різними кольорами (зеленим, червоним і оранжевим відповідно).
2. Створіть клас `FileWriter` з методами `Write()`, `WriteLine()`.
3. За допомогою шаблону Адаптер створіть файловий логер.
4. Покажіть правильність роботи свого коду запустивши його в головному методі програми

#### Код:

```
using System;
public interface ILogger
{
    void Log(string message);
    void Error(string message);
    void Warn(string message);
}
public class Logger : ILogger
{
    public void Log(string message)
    {
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine($"Log: {message}");
        Console.ResetColor();
    }
    public void Error(string message)
    {
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine($"Error: {message}");
        Console.ResetColor();
    }
    public void Warn(string message)
    {
        Console.ForegroundColor = ConsoleColor.Yellow;
        Console.WriteLine($"Warning: {message}");
        Console.ResetColor();
    }
}
```

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

```

public interface IFileWriter
{
    void Write(string text);
    void WriteLine(string text);
}

public class FileWriter : IFileWriter
{
    public void Write(string text)
    {
        Console.WriteLine($"Writing to file: {text}");
    }
    public void WriteLine(string text)
    {
        Console.WriteLine($"Writing line to file: {text}");
    }
}

public class FileLoggerAdapter : ILogger
{
    private readonly IFileWriter _fileWriter;

    public FileLoggerAdapter(IFileWriter fileWriter)
    {
        _fileWriter = fileWriter;
    }

    public void Log(string message)
    {
        _fileWriter.WriteLine($"Log: {message}");
    }

    public void Error(string message)
    {
        _fileWriter.WriteLine($"Error: {message}");
    }

    public void Warn(string message)
    {
        _fileWriter.WriteLine($"Warning: {message}");
    }
}

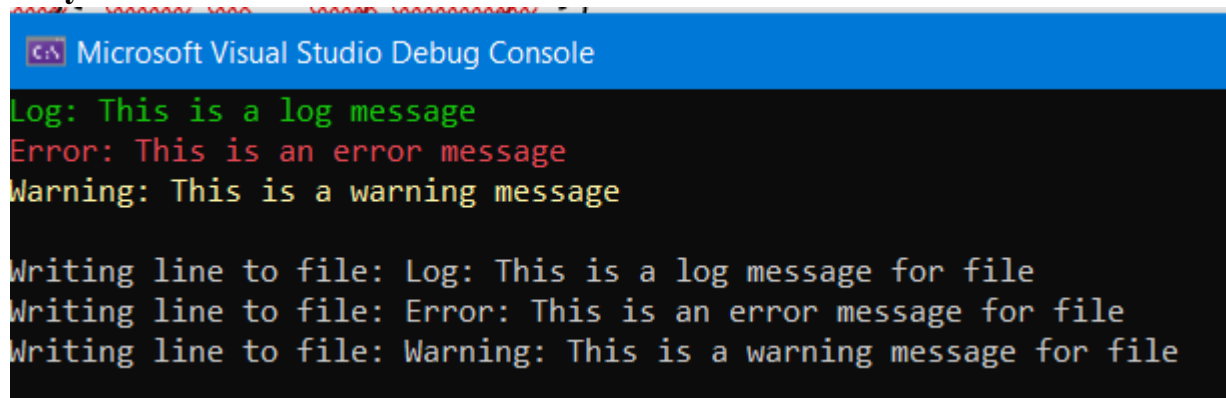
class Program
{
    static void Main(string[] args)
    {
        ILogger logger = new Logger();
        logger.Log("This is a log message");
        logger.Error("This is an error message");
        logger.Warn("This is a warning message");

        Console.WriteLine();

        IFileWriter fileWriter = new FileWriter();
        ILogger fileLogger = new FileLoggerAdapter(fileWriter);
        fileLogger.Log("This is a log message for file");
        fileLogger.Error("This is an error message for file");
        fileLogger.Warn("This is a warning message for file");
    }
}

```

## Результат:



```
Microsoft Visual Studio Debug Console

Log: This is a log message
Error: This is an error message
Warning: This is a warning message

Writing line to file: Log: This is a log message for file
Writing line to file: Error: This is an error message for file
Writing line to file: Warning: This is a warning message for file
```

## Завдання 2: Декоратор.

1. Ви розробляєте РПГ гру. Створіть класи героїв Warrior, Mage, Palladin.
2. Для героїв створіть інвентар (одяг, зброю, артефакти), який може підходити будь-якому типу героїв, у вигляді декораторів.
3. Важливою вимогою є можливість використання декількох екземплярів інвентаря на герої одночасно.
4. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

## Код:

```
using System;
using System.Collections.Generic;

public interface IInventory
{
    void Equip();
}

public abstract class Hero
{
    protected string Name;
    protected List<IInventory> Inventories = new List<IInventory>();

    public Hero(string name)
    {
        Name = name;
    }

    public void AddInventory(IInventory inventory)
    {
        Inventories.Add(inventory);
    }

    public virtual void ShowInventories()
    {
        Console.WriteLine($"Inventories for {Name}:");
        foreach (var inventory in Inventories)
```

```

        {
            inventory.Equip();
        }
    }
}

// Клас Warrior
public class Warrior : Hero
{
    public Warrior(string name) : base(name)
    {
    }

    public override void ShowInventories()
    {
        Console.WriteLine("\nWarrior Inventories:");
        base.ShowInventories();
    }
}

// Клас Mage
public class Mage : Hero
{
    public Mage(string name) : base(name)
    {
    }

    public override void ShowInventories()
    {
        Console.WriteLine("\nMage Inventories:");
        base.ShowInventories();
    }
}

// Клас Palladin
public class Palladin : Hero
{
    public Palladin(string name) : base(name)
    {
    }

    public override void ShowInventories()
    {
        Console.WriteLine("\nPalladin Inventories:");
        base.ShowInventories();
    }
}

public abstract class InventoryDecorator : IInventory
{
    protected IInventory Inventory;

    protected InventoryDecorator(IInventory inventory)
    {
        Inventory = inventory;
    }

    public virtual void Equip()
    {
        Inventory?.Equip();
    }
}

public class WeaponDecorator : InventoryDecorator

```

```

{
    private readonly string _weaponName;

    public WeaponDecorator(IInventory inventory, string weaponName) : base(inventory)
    {
        _weaponName = weaponName;
    }

    public override void Equip()
    {
        base.Equip();
        Console.WriteLine($"Weapon: {_weaponName}");
    }
}

public class ArmorDecorator : InventoryDecorator
{
    private readonly string _armorName;

    public ArmorDecorator(IInventory inventory, string armorName) : base(inventory)
    {
        _armorName = armorName;
    }

    public override void Equip()
    {
        base.Equip();
        Console.WriteLine($"Armor: {_armorName}");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Warrior warrior = new Warrior("Warrior1");
        Mage mage = new Mage("Mage1");
        Palladin palladin = new Palladin("Palladin1");

        warrior.AddInventory(new WeaponDecorator(null, "Sword"));
        warrior.AddInventory(new ArmorDecorator(null, "Chainmail"));
        warrior.AddInventory(new WeaponDecorator(new ArmorDecorator(null, "Leather
Armor"), "Axe"));
        mage.AddInventory(new WeaponDecorator(null, "Staff"));
        mage.AddInventory(new ArmorDecorator(null, "Robe"));
        mage.AddInventory(new WeaponDecorator(new ArmorDecorator(null, "Cloth Armor"),
"Wand"));
        palladin.AddInventory(new WeaponDecorator(null, "Mace"));
        palladin.AddInventory(new ArmorDecorator(null, "Plate Armor"));
        palladin.AddInventory(new WeaponDecorator(new ArmorDecorator(null, "Plate
Armor"), "Shield"));

        warrior.ShowInventories();
        mage.ShowInventories();
        palladin.ShowInventories();
    }
}

```

## Результат:

```
Microsoft Visual Studio Debug Console  
Warrior Inventories:  
Inventories for Warrior1:  
Weapon: Sword  
Armor: Chainmail  
Armor: Leather Armor  
Weapon: Axe  
  
Mage Inventories:  
Inventories for Mage1:  
Weapon: Staff  
Armor: Robe  
Armor: Cloth Armor  
Weapon: Wand  
  
Palladin Inventories:  
Inventories for Palladin1:  
Weapon: Mace  
Armor: Plate Armor  
Armor: Plate Armor  
Weapon: Shield
```

### Завдання 3: Міст.

1. Ви працюєте над графічним редактором. Створіть базовий клас Shape.

2. Створіть дочірні до Shape класи, Circle, Square, Triangle.

3. За допомогою шаблону Міст додайте можливість рендерингу кожної з фігур як векторної або растрової графіки (вивівши відповідне повідомлення у консоль, наприклад "Drawing Triangle as pixels").

4. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

## Код:

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.15.000 – Лр.3	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

```

using System;

public abstract class Shape
{
    protected IRenderer Renderer;

    protected Shape(IRenderer renderer)
    {
        Renderer = renderer;
    }

    public abstract void Draw();
}

public class Circle : Shape
{
    public Circle(IRenderer renderer) : base(renderer)
    {
    }

    public override void Draw()
    {
        Renderer.RenderCircle();
    }
}

public class Square : Shape
{
    public Square(IRenderer renderer) : base(renderer)
    {
    }

    public override void Draw()
    {
        Renderer.RenderSquare();
    }
}

public class Triangle : Shape
{
    public Triangle(IRenderer renderer) : base(renderer)
    {
    }

    public override void Draw()
    {
        Renderer.RenderTriangle();
    }
}

public interface IRenderer
{
    void RenderCircle();
    void RenderSquare();
    void RenderTriangle();
}

public class RasterRenderer : IRenderer
{
    public void RenderCircle()
    {
        Console.WriteLine("Drawing Circle as pixels");
    }
}

```

```

public void RenderSquare()
{
    Console.WriteLine("Drawing Square as pixels");
}

public void RenderTriangle()
{
    Console.WriteLine("Drawing Triangle as pixels");
}
}

public class VectorRenderer : IRenderer
{
    public void RenderCircle()
    {
        Console.WriteLine("Drawing Circle as vectors");
    }

    public void RenderSquare()
    {
        Console.WriteLine("Drawing Square as vectors");
    }

    public void RenderTriangle()
    {
        Console.WriteLine("Drawing Triangle as vectors");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Rendering shapes in raster format:");
        var rasterCircle = new Circle(new RasterRenderer());
        rasterCircle.Draw();

        var rasterSquare = new Square(new RasterRenderer());
        rasterSquare.Draw();

        var rasterTriangle = new Triangle(new RasterRenderer());
        rasterTriangle.Draw();

        Console.WriteLine("\nRendering shapes in vector format:");
        var vectorCircle = new Circle(new VectorRenderer());
        vectorCircle.Draw();

        var vectorSquare = new Square(new VectorRenderer());
        vectorSquare.Draw();

        var vectorTriangle = new Triangle(new VectorRenderer());
        vectorTriangle.Draw();
    }
}

```

**Результат:**

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.15.000 – Лр.3	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		



```

Rendering shapes in raster format:
Drawing Circle as pixels
Drawing Square as pixels
Drawing Triangle as pixels

Rendering shapes in vector format:
Drawing Circle as vectors
Drawing Square as vectors
Drawing Triangle as vectors

```

#### Завдання 4: Проксі.

1. Створіть клас SmartTextReader, який вмє читати вміст текстового файлу і перетворювати його на двовірний масив якому зовнішній масив відповідає рядкам тексту, а вкладені масиви відповідають символам у відповідному рядку.
2. Створіть проксі для SmartTextReader з логуванням SmartTextChecker, який буде виводити інформацію про успішне відкриття, прочитання і закриття файлу, а також буде виводити загальну кількість рядків і символів у прочитаному тексті.
3. Створіть проксі для SmartTextReader з обмеженням доступу до певних файлів SmartTextReaderLocker. Цей клас в конструкторі приймає регулярний вираз, по якому лімітується доступ до певної групи файлів. Якщо клієнт викликати метод для прочитання такого лімітованого файлу, замість прочитання файлу в консоль має виводитися повідомлення "Access denied!".
4. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

**Код:** `using System;`  
`using System.IO;`  
`using System.Text.RegularExpressions;`

```

public interface ITextReader
{
    string[, ] ReadTextFile(string filePath);
}

public class SmartTextReader : ITextReader
{
    public string[, ] ReadTextFile(string filePath)
    {
        string[] lines = File.ReadAllLines(filePath);
        int maxLength = lines.Length > 0 ? lines[0].Length : 0;
        string[, ] result = new string[lines.Length, maxLength];
        for (int i = 0; i < lines.Length; i++)

```

```

        {
            for (int j = 0; j < lines[i].Length; j++)
            {
                result[i, j] = lines[i][j].ToString();
            }
        }
        return result;
    }
}

public class SmartTextChecker : ITextReader
{
    private readonly ITextReader _reader;

    public SmartTextChecker(ITextReader reader)
    {
        _reader = reader;
    }

    public string[,] ReadTextFile(string filePath)
    {
        Console.WriteLine($"Opening file: {filePath}");
        string[,] result = _reader.ReadTextFile(filePath);
        Console.WriteLine($"Successfully read file: {filePath}");
        Console.WriteLine($"Number of lines: {result.GetLength(0)}");
        Console.WriteLine($"Number of characters: {result.Length}");
        Console.WriteLine($"Closing file: {filePath}");
        return result;
    }
}

public class SmartTextReaderLocker : ITextReader
{
    private readonly ITextReader _reader;
    private readonly Regex _regex;

    public SmartTextReaderLocker(ITextReader reader, string pattern)
    {
        _reader = reader;
        _regex = new Regex(pattern);
    }

    public string[,] ReadTextFile(string filePath)
    {
        if (_regex.IsMatch(filePath))
        {
            Console.WriteLine("Access denied!");
            return null;
        }
        return _reader.ReadTextFile(filePath);
    }
}

class Program
{
    static void Main(string[] args)
    {
        ITextReader textReader = new SmartTextReader();
        ITextReader smartReaderWithLogging = new SmartTextChecker(textReader);
        string[,] fileContent = smartReaderWithLogging.ReadTextFile("example.txt");

        ITextReader smartReaderWithLock = new SmartTextReaderLocker(textReader,
"restricted.*");
    }
}

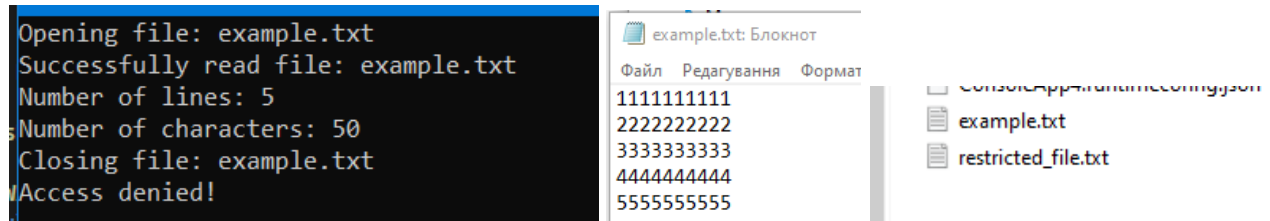
```

```

        string[,] restrictedContent =
smartReaderWithLock.ReadTextFile("restricted_file.txt");
    }
}

```

## Результат:



## Завдання 5: Компонувальник.

1. Вам потрібно створити власну мову розмітки LightHTML.
2. Кожен елемент розмітки має наслідувати клас LightNode.
3. Створіть два дочірніх класи від LightNode: LightElementNode, LightTextNode.
4. LightTextNode може містити лише текст.
5. LightElementNode може містити будь-які LightNode. LightElementNode повинен мати інформацію про назву тега, його тип відображення (блочний чи рядковий), тип закриття (одиночний тег, як `<img/>` чи з закриваючим тегом) список CSS класів,

кількість дочірніх елементів, а також має бути можливість виводити на екран його outerHTML і innerHTML.

6. За допомогою своєї мови розмітки виведіть в консоль елемент сторінки на Ваш вибір (наприклад якусь таблицю, список тощо).
7. 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 class LightTextNode : LightNode
{
    private readonly string _text;

    public LightTextNode(string text)
    {
        _text = text;
    }

    public override string OuterHTML => _text;
    public override string InnerHTML => _text;
}

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;

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

    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();
}
}
}

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);

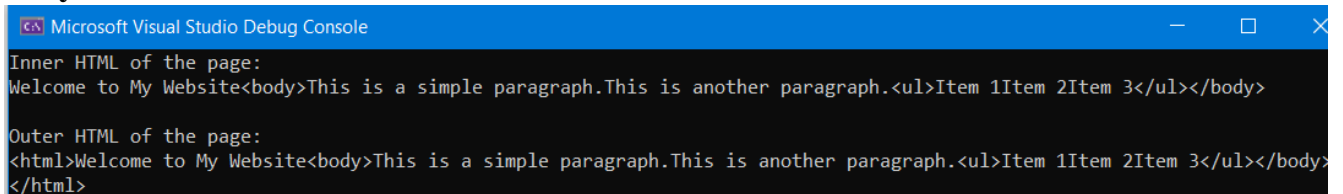
        var bodyChildren = new List<LightNode> { paragraph1, paragraph2, unorderedList };
        var body = new LightElementNode("body", true, false, new List<string>(),
bodyChildren);

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

        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>

```

**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 class LightTextNode : LightNode
{
    private readonly string _text;

    public LightTextNode(string text)
    {

```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.15.000 – Лр.3	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

```

        _text = text;
    }

    public override string OuterHTML => _text;
    public override string InnerHTML => _text;
}

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;

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

    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 class LightweightNode : LightNode
{
    private readonly string _content;

    public LightweightNode(string content)
    {
        _content = content;
    }

    public override string OuterHTML => _content;
    public override string InnerHTML => _content;
}

class Program
{
    static void Main(string[] args)
    {
        var h1 = new LightweightNode("<h1>Chapter 1: Introduction</h1>");
        var h2 = new LightweightNode("<h2>Section 1.1: Overview</h2>");
        var p1 = new LightweightNode("<p>This is the first paragraph of the
introduction.</p>");
        var p2 = new LightweightNode("<p>This is the second paragraph of the
introduction.</p>");
        var blockquote = new LightweightNode("<blockquote>This is a
blockquote.</blockquote>");

        Console.WriteLine("Outer HTML:");
        Console.WriteLine(h1.OuterHTML);
        Console.WriteLine(h2.OuterHTML);
        Console.WriteLine(p1.OuterHTML);
        Console.WriteLine(p2.OuterHTML);
        Console.WriteLine(blockquote.OuterHTML);
        Console.WriteLine();

        Console.WriteLine("Inner HTML:");
        Console.WriteLine(h1.InnerHTML);
        Console.WriteLine(h2.InnerHTML);
        Console.WriteLine(p1.InnerHTML);
        Console.WriteLine(p2.InnerHTML);
        Console.WriteLine(blockquote.InnerHTML);
    }
}

```

## Результат:

```

Outer HTML:
<h1>Chapter 1: Introduction</h1>
<h2>Section 1.1: Overview</h2>
<p>This is the first paragraph of the introduction.</p>
<p>This is the second paragraph of the introduction.</p>
<blockquote>This is a blockquote.</blockquote>

Inner HTML:
<h1>Chapter 1: Introduction</h1>
<h2>Section 1.1: Overview</h2>
<p>This is the first paragraph of the introduction.</p>
<p>This is the second paragraph of the introduction.</p>
<blockquote>This is a blockquote.</blockquote>

```

**Висновок:** У ході виконання лабораторної роботи я навчився реалізовувати структурні шаблони проектування Адаптер, Декоратор, Міст, Компонувальник, Проксі, Легковаговик.