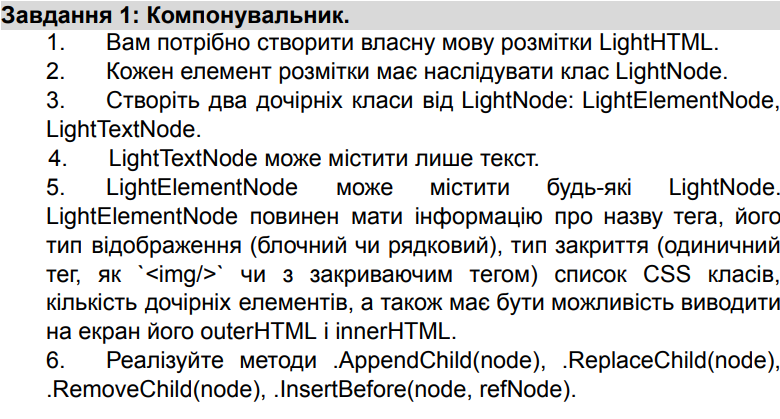
**Лабораторна робота № 5**

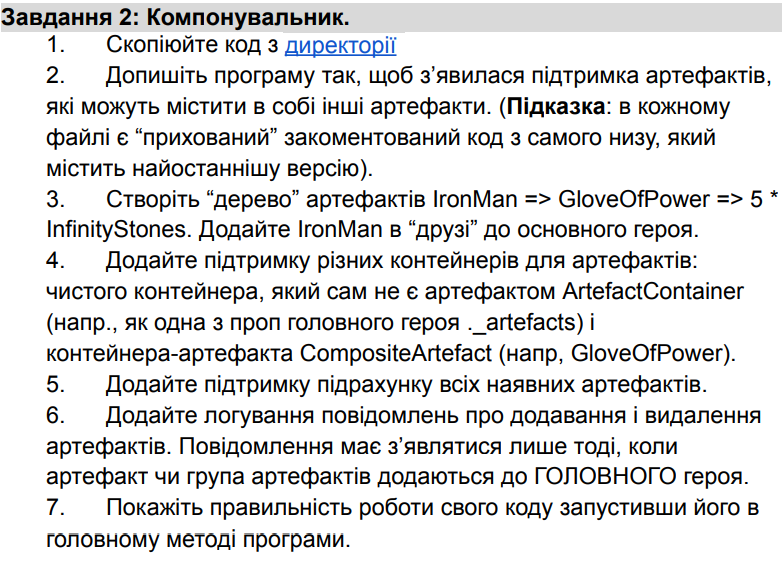
**СТРУКТУРНІ ШАБЛОНИ**

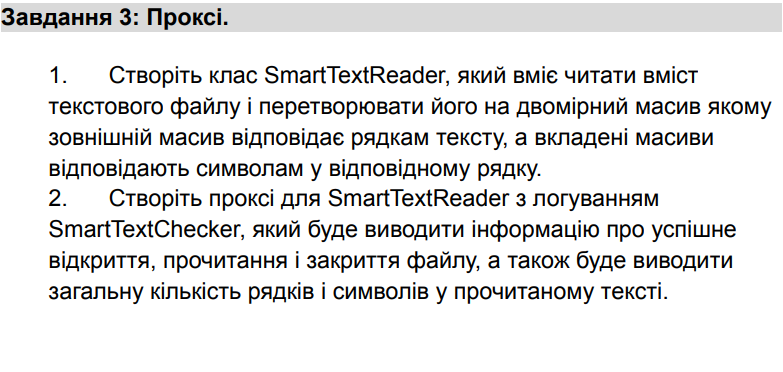
**Мета:** навчитися реалізовувати структурні шоблони проєктування Компонувальник, Проксі, Легковаговик, Посередник

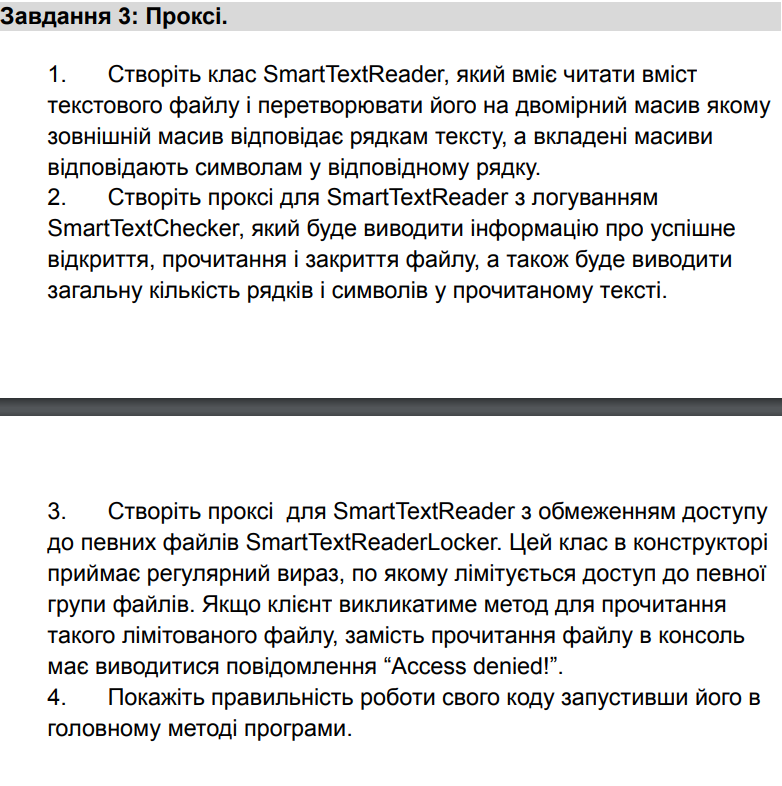
**Хід роботи:**

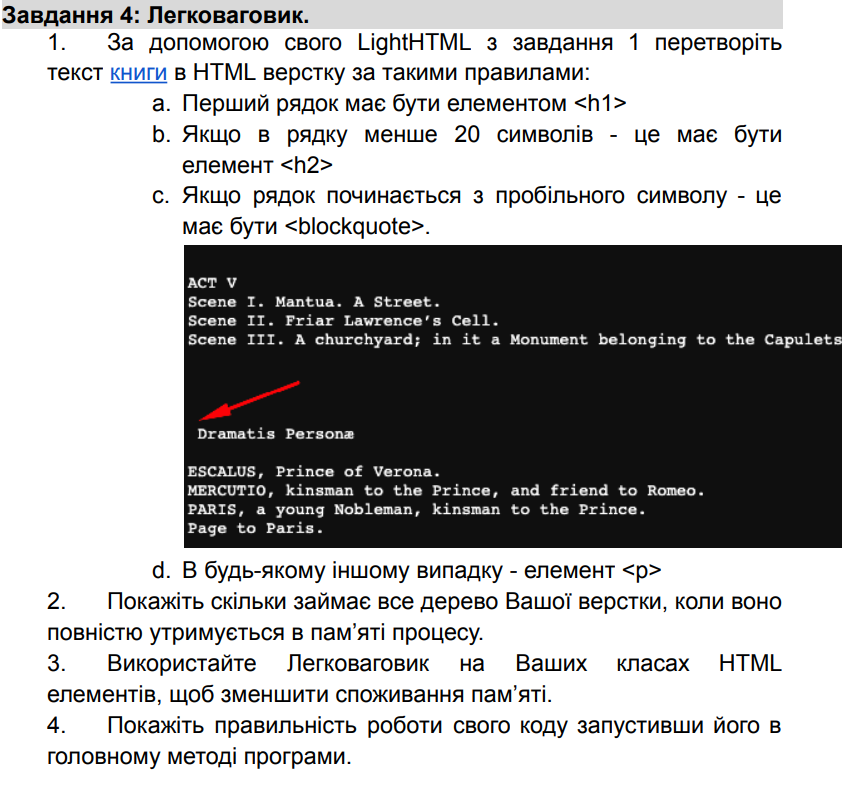
****

# 



****

****

****

**Завдання 1:** Компонувальник.

Лістинг коду класу **LightNode:**

namespace lab5.Composite.Interfaces

{

internal interface ILightNode

{

public string Display();

public ViewType ViewType { get; }

public LightElementNode Parent { get;set;}

}

}

Лістинг коду класу **LightTextNode:**

Лістинг коду:

namespace lab5.Composite.Clasess

{

internal class LightTextNode : BaseNode

{

protected string Text;

public override ViewType ViewType { get; }

public int Ident { get; set; } = 0;

public LightTextNode(string text)

{

Text = text;

ViewType = ViewType.String;

}

public override string Display()

{

var s = new StringBuilder();

if (Parent != null)

{

for (int i = 0; i < Parent.Ident; i++)

{

s.Append("\t");

}

}

return s.Append(Text).ToString();

}

}

}

Лістинг коду класу **LightElementNode:**

Лістинг коду:

namespace lab5.Composite.Clasess

{

public enum ViewType

{

Block,

String

}

enum ClosureType

{

Single,

Patrial

}

internal class LightElementNode : BaseNode, INodeEditor, ICloneable

{

public Dictionary<string, string> Attributes { get; protected set; }

public string Name { get; protected set; }

public ClosureType ClosureType { get; protected set; }

public List<ILightNode> Nodes { get; protected set; }

public override ViewType ViewType {get;}

public int Ident { get; set; } = 0;

public void SetAttributes(Dictionary<string, string> attributes)

{

Attributes = attributes;

}

public LightElementNode(string name, ClosureType closure, ViewType view)

{

Name = name;

ClosureType = closure;

ViewType = view;

Nodes = new List<ILightNode>();

Attributes = new Dictionary<string, string>();

}

public LightElementNode()

{

Nodes = new List<ILightNode>();

Attributes = new Dictionary<string, string>();

}

public LightElementNode(LightElementNode prototype)

{

ClosureType = prototype.ClosureType;

Nodes = prototype.Nodes;

Attributes = prototype.Attributes;

ViewType = prototype.ViewType;

Name = prototype.Name;

Parent = prototype.Parent;

}

public void AppendChild(ILightNode node)

{

if (!this.IsAppendable())

{

throw new Exception($"The tag type is not support appending");

}

int index = Nodes.IndexOf(node);

if (index == -1)

{

Nodes.Add(node);

node.Parent = this;

}

}

public override string Display()

{

var \_tagBuilder = new LightELementBuilder();

if (ClosureType == ClosureType.Single)

{

return \_tagBuilder.SetNode(this)

.SetIdent()

.SetOpenBracket()

.SetAttributes()

.SetSlash()

.SetCloseBracket()

.TagToString();

}

return \_tagBuilder.SetNode(this)

.SetIdent()

.SetOpenBracket()

.SetAttributes()

.SetCloseBracket()

.SetChilds()

.SetIdent()

.SetCloseBrackets()

.TagToString();

}

public string InnerHtml()

{

var builder = new StringBuilder();

foreach (var node in Nodes)

{

builder.Append(node.Display());

}

return builder.ToString();

}

public string? OuterHtml()

{

return Display();

}

public void RemoveChild(ILightNode node)

{

Nodes.Remove(node);

}

public void ReplaceChild(ILightNode node, ILightNode replaceNode)

{

int index = Nodes.IndexOf(node);

if (index != -1)

{

Nodes[index] = replaceNode;

}

}

public void InsertBefore(ILightNode refNode, ILightNode node)

{

int index = Nodes.IndexOf(refNode);

if (index != -1)

{

Nodes.Insert(index + 1, node);

}

}

public object Clone()

{

return new LightElementNode(this);

}

}

}

# Результат виконання програми:

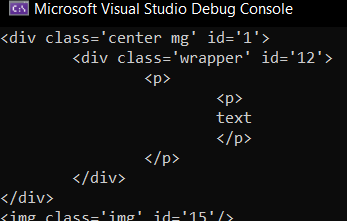


Рис.5.1. Реалізація завдання.

**Завдання 2:** Компонувальник.

Лістинг коду**:**

using System;

using lab5.HeroComposite.Abstract;

using lab5.HeroComposite.Interfaces;

namespace lab5.HeroComposite

{

enum HeroType

{

EmptyHero,Mage, Warrior, Rogue, Hunter,

Paladin, Druid, Warlock, Shaman,

}

enum Nature

{

Kind, Evil, Good, Neutral

}

internal class MarvelHero<T> : ContainableEntitetBase<T> where T:Entitet

{

public HeroType Type { get; protected set; }

public Nature Nature { get; protected set; }

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

public MarvelHero()

{

\_entitets = new List<T>();

}

public void SetNature(Nature nature)

{

Nature=nature;

}

public void SetHeroType(HeroType type)

{

Type = type;

}

}

}

namespace lab5.HeroComposite

{

internal class Artefact:Entitet

{

public void CalculatePowerBuf(MarvelHero<Entitet> hero)

{

System.Console.WriteLine("Calculation...");

}

}

}

namespace lab5.HeroComposite

{

internal class CompositeArtefact

{

private List<Entitet> \_children;

public CompositeArtefact()

{

\_children = new List<Entitet>();

}

public void AddChild(Entitet entitet)

{

\_children.Add(entitet);

}

public void RemoveChild(Entitet entitet)

{

\_children.Add(entitet);

}

public int GetWeight()

{

return \_children.Aggregate(0, (current, entitet) => current += entitet.GetWeight());

}

public int GetPower()

{

return \_children.Aggregate(0, (current, entitet) => current += entitet.GetPower());

}

public int GetCount()

{

return \_children.Aggregate(0, (current, entitet) => current += entitet.GetCount());

}

}

}

Результат виконання програми:

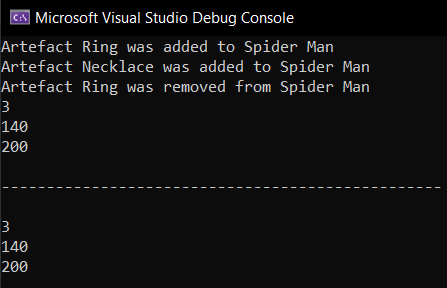


Рис.5.2. Реалізація завдання.

**Завдання 3:** Проксі.

Лістинг коду класу **SmartTextReader:**

Лістинг коду:

using System;

using lab5.Proxy.Interfaces;

namespace lab5.Proxy.Classes

{

public class SmartTextReader : ISmartTextReader

{

private string[,] arr = { };

public string[,] Read(string path)

{

using (StreamReader sr = new StreamReader(path))

{

int i = 0;

while (!sr.EndOfStream)

{

string? line = sr.ReadLine();

if(line!=null){

arr = (string[,])ResizeArray(arr, new int[] { i + 1, 1 });

arr[i, 0] = line;

i++;

}

}

}

return arr;

}

private static Array ResizeArray(Array arr, int[] newSizes)

{

if (newSizes.Length != arr.Rank)

throw new ArgumentException("arr must have the same number of dimensions " +

"as there are elements in newSizes", "newSizes");

var temp = Array.CreateInstance(arr.GetType().GetElementType(), newSizes);

int length = arr.Length <= temp.Length ? arr.Length : temp.Length;

Array.ConstrainedCopy(arr, 0, temp, 0, length);

return temp;

}

}

}

Клас **SmartTextChecker**

Лістинг коду:

using System;

using lab5.Proxy.Interfaces;

namespace lab5.Proxy.Proxies

{

internal class SmartTextChecker : ISmartTextReader

{

private string[,] \_resultArray = { };

private ISmartTextReader \_smartTextReader;

private int \_symbolCount = 0;

public SmartTextChecker(ISmartTextReader smartTextReader)

{

\_smartTextReader = smartTextReader;

}

private int GetSymbolsCount()

{

foreach (var item in \_resultArray)

{

\_symbolCount += item.Length;

}

return \_symbolCount;

}

public string[,] Read(string fileName)

{

try

{

var res = \_smartTextReader.Read(fileName);

\_resultArray = new string[res.Length - 1, 0];

System.Console.WriteLine("File opened");

\_resultArray = res;

}

catch (Exception e)

{

Console.WriteLine("File openning failed with error: " + e.Message);

}

finally

{

System.Console.WriteLine("File closed");

System.Console.WriteLine($"Lines read: {\_resultArray.Length}, symbols count: {GetSymbolsCount()}");

}

return \_resultArray;

}

}

}

Результат виконання програми:

Зображення, що містить текст, знімок екрана

Автоматично згенерований опис

Рис.5.3. Реалізація завдання.

**Завдання 4:** Легковаговик.

Лістинг коду:

class MarvelHero

{

private string name;

private int level;

private List<Artefact> artefacts;

public MarvelHero(string name, int level)

{

this.name = name;

this.level = level;

artefacts = new List<Artefact>();

}

public virtual void Strike()

{

Console.WriteLine($"Герой {name} б'є з силою {level}");

}

public void AddArtefact(Artefact artefact)

{

artefacts.Add(artefact);

Console.WriteLine($"Додано артефакт: {artefact.GetName()} до героя: {name}");

}

public void CountArtefacts()

{

Console.WriteLine($"Загальна кількість артефактів у героя {name}: {artefacts.Count}");

}

public void CalculateArtefactsWeight()

{

int totalWeight = 0;

foreach (Artefact artefact in artefacts)

{

totalWeight += artefact.GetWeight();

}

Console.WriteLine($"Загальна вага артефактів у героя {name}: {totalWeight}");

}

}

class Artefact

{

private string name;

private int level;

private int weight;

public Artefact(string name, int level, int weight)

{

this.name = name;

this.level = level;

this.weight = weight;

}

public string GetName()

{

return name;

}

public int GetWeight()

{

return weight;

}

}

class ArtefactContainer : Artefact

{

private List<Artefact> artefacts;

public ArtefactContainer(string name, int level, int weight) : base(name, level, weight)

{

artefacts = new List<Artefact>();

}

public void AddArtefact(Artefact artefact)

{

artefacts.Add(artefact);

Console.WriteLine($"Додано артефакт: {artefact.GetName()} до контейнера: {GetName()}");

}

}

class CompositeArtefact : Artefact

{

private List<Artefact> artefacts;

public CompositeArtefact(string name, int level, int weight) : base(name, level, weight)

{

artefacts = new List<Artefact>();

}

public void AddArtefact(Artefact artefact)

{

artefacts.Add(artefact);

Console.WriteLine($"Додано артефакт: {artefact.GetName()} до артефакту: {GetName()}");

}

public void CountArtefacts()

{

Console.WriteLine($"Загальна кількість артефактів у артефакту {GetName()}: {artefacts.Count}");

}

public void CalculateArtefactsWeight()

{

int totalWeight = GetWeight();

foreach (Artefact artefact in artefacts)

{

totalWeight += artefact.GetWeight();

}

Console.WriteLine($"Загальна вага артефактів у артефакті {GetName()}: {totalWeight}");

}

}

Результат виконання програми:

Зображення, що містить текст, знімок екрана

Автоматично згенерований опис

Рис.5.4. Реалізація завдання.

**Висновок:** на лабораторній роботі було отримано навички з реалізовування структурних шаблонів Компонувальник, Проксі, Легковаговик, Посередник.