Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра ИИТ

Лабораторная работа №6

По дисциплине: "СПП"

Выполнил:
Студент 3 курса
Группы ПО-3
Лущ М. Г.
Проверил:
Крощенко А. А.

Брест 2019

Цель работы: приобрести навыки применения паттернов проектирования при решении практических задач с использованием языка C#.

Вариант 14

Общее задание

• Прочитать задания, взятые из каждой группы.

• Определить паттерн проектирования, который может использоваться при реализации задания. Пояснить свой выбор.

• Реализовать фрагмент программной системы, используя выбранный паттерн. Реализовать все необходимые дополнительные классы.

Задание №1:

Проект «Туристическое бюро». Реализовать возможность выбора программы тура (проезд, проживание, питание, посещение музеев, выставок, экскурсии и т.д.). Должна формироваться итоговая стоимость заказа.

Для реализации задания был применён паттерн Fluent bulder.

Код программы:

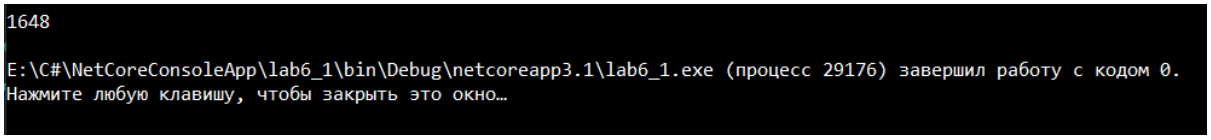
```
using System;

namespace lab6_1
{
    class Program
    {
        static void Main(string[] args)
        {
            Services services = Services.CreateBuilder().Travel("New-York", 1000).Accomodation("hotel", 100).
                Nutrition("all included", 345).MusiemsVisiting("all", 203);
            Console.WriteLine(services.SumCost());
        }


        public class Services
        {
            public Service Travel { get; set; }
            public Service Accomodation { get; set; }
            public Service Nutrition { get; set; }
            public Service MusiemsVisiting { get; set; }
            public Service ExhibitionsVisiting { get; set; }
```

```
    public Service ExcursionsVisiting { get; set; }

    public int SumCost()
    {
        int result = 0;
        result += Travel?.Cost ?? 0;
        result += Accomodation?.Cost ?? 0;
        result += Nutrition?.Cost ?? 0;
        result += MusiemsVisiting?.Cost ?? 0;
        result += ExhibitionsVisiting?.Cost ?? 0;
        result += ExcursionsVisiting?.Cost ?? 0;
        return result;
    }

    public static ServiceBuilder CreateBuilder()
    {
        return new ServiceBuilder();
    }
}

public class Service
{
    public string Name { get; set; }
    public int Cost { get; set; }

    public Service(string name, int cost)
    {
        Name = name;
        Cost = cost;
    }
}

public class ServiceBuilder
{
    private Services _services;
    public ServiceBuilder()
    {
```

```
    _services = new Services();
}
public ServiceBuilder Travel(string name, int cost)
{
    _services.Travel = new Service(name, cost);
    return this;
}
public ServiceBuilder Accomodation(string name, int cost)
{
    _services.Accomodation = new Service(name, cost);
    return this;
}
public ServiceBuilder Nutrition(string name, int cost)
{
    _services.Nutrition = new Service(name, cost);
    return this;
}
public ServiceBuilder MusiemsVisiting(string name, int cost)
{
    _services.MusiemsVisiting = new Service(name, cost);
    return this;
}
public ServiceBuilder ExhibitionsVisiting(string name, int cost)
{
    _services.ExhibitionsVisiting = new Service(name, cost);
    return this;
}
public ServiceBuilder ExcursionsVisiting(string name, int cost)
{
    _services.ExcursionsVisiting = new Service(name, cost);
    return this;
}

public static implicit operator Services(ServiceBuilder builder)
{
    return builder._services;
}
```

```
    }
  }
}
```

Результаты работы программы:



Задание №2:

Проект «Файловая система». Реализуйте модель работы файловой системы. Должна поддерживаться иерархичность ФС на уровне директорий и отдельных файлов. Файлы могут иметь все основные присущие им атрибуты (размер, расширение, дата создания и т.д.).

Для реализации задания был использован паттерн Composite.

Код программы:

```
using System;
using System.Collections.Generic;
namespace lab6_2
{
  class Program
  {
    static void Main(string[] args)
    {
      ExplorerComponent folder1 = new Directory("Folder1");
      ExplorerComponent folder2 = new Directory("folder2");
      ExplorerComponent folder3 = new Directory("folder3");

      ExplorerComponent text = new File("text", DateTime.Now + TimeSpan.FromTicks(TimeSpan.TicksPerDay), 1000,
"txt");
      ExplorerComponent exe = new File("pe", DateTime.Now + TimeSpan.FromTicks(TimeSpan.TicksPerDay * 2),
4356, "exe");
      ExplorerComponent png = new File("photo", DateTime.Now, 1000, "png");

      folder1.Add(folder2);
      folder1.Add(text);
      folder1.Add(exe);
```

```
        folder2.Add(folder3);
        folder2.Add(png);
        Console.WriteLine(folder1.GetInfo());
    }
}

abstract class ExplorerComponent
{
    public virtual void Add(ExplorerComponent component) { }
    public virtual void Remove(ExplorerComponent component) { }
    public virtual string GetInfo() { return string.Empty; }
}
class Directory : ExplorerComponent
{
    private List<ExplorerComponent> _items;

    public IReadOnlyCollection<ExplorerComponent> Items { get => _items; }
    public string Name { get; set; }

    public Directory() => _items = new List<ExplorerComponent>();
    public Directory(string name) : this() => Name = name;

    public override void Add(ExplorerComponent component) => _items.Add(component);
    public override void Remove(ExplorerComponent component) => _items.Remove(component);
    public override string GetInfo() => Name;
}
class File : ExplorerComponent
{
    public string Name { get; set; }
    public DateTime? CreationDate { get; set; }
    public uint Size { get; set; }
    public string Extension { get; set; }

    public File() { }
    public File(string name, DateTime date, uint size, string extension)
    {
```

```
            Name = name;
            CreationDate = date;
            Size = size;
            Extension = extension;
        }
        public override string GetInfo() => $"{Name} - {Extension} - {CreationDate} - {Size}";
    }
}
```

Результаты работы программы:

Имя	Значение	Тип
folder1	{lab6_2.Directory}	lab6_2.ExplorerComponent {lab6_2.Directory}
Name	"Folder1"	string
_explorer	Count = 3	System.Collections.Generic.List<lab6_2.ExplorerComponent>
[0]	{lab6_2.Directory}	lab6_2.ExplorerComponent {lab6_2.Directory}
Name	"folder2"	string
_explorer	Count = 2	System.Collections.Generic.List<lab6_2.ExplorerComponent>
[0]	{lab6_2.Directory}	lab6_2.ExplorerComponent {lab6_2.Directory}
Name	"folder3"	string
_explorer	Count = 0	System.Collections.Generic.List<lab6_2.ExplorerComponent>
[1]	{lab6_2.File}	lab6_2.ExplorerComponent {lab6_2.File}
CreationDate	{09.10.2020 20:18:23}	System.DateTime?
Extension	"png"	string
Name	"photo"	string
Size	1000	uint
Raw View		
[1]	{lab6_2.File}	lab6_2.ExplorerComponent {lab6_2.File}
CreationDate	{10.10.2020 20:18:23}	System.DateTime?
Extension	"txt"	string
Name	"text"	string
Size	1000	uint
[2]	{lab6_2.File}	lab6_2.ExplorerComponent {lab6_2.File}
CreationDate	{11.10.2020 20:18:23}	System.DateTime?
Extension	"exe"	string
Name	"pe"	string
Size	4356	uint
Raw View		

Задание №3:

Реализовать вывод ФС из 2-й группы заданий. Вывод файлов/директорий должен осуществляться в случайном порядке. Вывести основные атрибуты каждого файла/директории.

Для реализации задания был использован паттерн Visitor.

Код программы:

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

namespace lab6_3
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            IDisplayer displayer = new ConsoleRandomDisplayer();


            ExplorerComponent folder1 = new Directory("Folder1");
            ExplorerComponent folder2 = new Directory("folder2");
            ExplorerComponent folder3 = new Directory("folder3");


            ExplorerComponent text = new File("text", DateTime.Now +
TimeSpan.FromTicks(TimeSpan.TicksPerDay), 1000, "txt");
            ExplorerComponent exe = new File("pe", DateTime.Now +
TimeSpan.FromTicks(TimeSpan.TicksPerDay * 2), 4356, "exe");
            ExplorerComponent png = new File("photo", DateTime.Now, 1000, "png");


            folder1.Add(folder2);
            folder1.Add(text);
            folder1.Add(exe);


            folder2.Add(folder3);
            folder2.Add(png);


            folder1.Display(displayer);
        }
    }


    abstract class ExplorerComponent: IDisplayable
    {
        public virtual void Add(ExplorerComponent component) { }
        public virtual void Remove(ExplorerComponent component) { }
        public virtual string GetInfo() { return string.Empty; }


        public abstract void Display(IDisplayer directory);
    }
```

```
class Directory : ExplorerComponent
{
    private List<ExplorerComponent> _items;

    public IReadOnlyCollection<ExplorerComponent> Items { get => _items; }

    public string Name { get; set; }

    public Directory() => _items = new List<ExplorerComponent>();
    public Directory(string name) : this() => Name = name;

    public override void Add(ExplorerComponent component) => _items.Add(component);
    public override void Remove(ExplorerComponent component) => _items.Remove(component);
    public override string GetInfo() => Name;

    public override void Display(IDisplayer directory)
    {
        directory.DisplayItem(this);
    }
}

class File : ExplorerComponent
{
    public string Name { get; set; }
    public DateTime? CreationDate { get; set; }
    public uint Size { get; set; }
    public string Extension { get; set; }

    public File() { }
    public File(string name, DateTime date, uint size, string extension)
    {
        Name = name;
        CreationDate = date;
        Size = size;
```

```
        Extension = extension;
    }

    public override string GetInfo() => $"{Name} - {Extension} - {CreationDate} - {Size}";

    public override void Display(IDisplayer directory)
    {
        directory.DisplayItem(this);
    }
}

interface IDisplayable
{
    void Display(IDisplayer directory);
}

interface IDisplayer
{
    void DisplayItem(Directory directory);
    void DisplayItem(File directory);
}

class ConsoleRandomDisplayer : IDisplayer
{
    public void DisplayItem(Directory directory)
    {
        int[] indexes = new int[directory.Items.Count];
        for (int i = 0; i < directory.Items.Count; i++)
        {
            indexes[i] = i;
        }

        Random random = new Random(DateTime.Now.Millisecond);
        indexes = indexes.OrderBy(x => random.Next()).ToArray();
```

```
            Console.WriteLine(directory.GetInfo());
            foreach (int index in indexes)
            {
                directory.Items.ElementAt(index).Display(this);
            }
        }

        public void DisplayItem(File directory)
        {
            Console.WriteLine("\t" + directory.GetInfo());
        }
    }

}
```

Результаты работы программы:

```
Folder1
folder2
        photo - png - 10.10.2020 23:19:00 - 1000
folder3
        pe - exe - 12.10.2020 23:19:00 - 4356
        text - txt - 11.10.2020 23:19:00 - 1000

E:\C#\NetCoreConsoleApp\lab6_3\bin\Debug\netcoreapp3.1\lab6_3.exe (процесс 8324) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно…
```

```
Folder1
folder2
folder3
        photo - png - 10.10.2020 23:19:10 - 1000
        text - txt - 11.10.2020 23:19:10 - 1000
        pe - exe - 12.10.2020 23:19:10 - 4356

E:\C#\NetCoreConsoleApp\lab6_3\bin\Debug\netcoreapp3.1\lab6_3.exe (процесс 11092) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно…
```

```
Folder1
        text - txt - 11.10.2020 23:19:20 - 1000
folder2
        photo - png - 10.10.2020 23:19:20 - 1000
folder3
        pe - exe - 12.10.2020 23:19:20 - 4356

E:\C#\NetCoreConsoleApp\lab6_3\bin\Debug\netcoreapp3.1\lab6_3.exe (процесс 4280) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно…
```