МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № 3

з дисципліни “Програмування. Об'єктно-орієнтоване програмування та шаблони проєктування”
тема “Структурні шаблони.”

Виконала
студентка ІІ курсу
групи КП-02
Красношапка Анастасія Андріївна

Перевірила
“____” “_____” 20___ р.
викладач
Заболотня Тетяна Миколаївна

Варіант 5

Київ 2021

Мета роботи

Ознайомитися зі структурними шаблонами, навчитися визначати, де саме потрібно використовувати кожен з розглянутих варінтів, а також розробляти програмне забезпечення на мові C# з їх використанням.

Постановка задачі

1. За допомогою шаблону проєктування створити структуру класів, до якої будуть входити класи, які відповідають таким об’єктам як Лікар, Відділення та Поліклініка. Кожний такий об’єкт має своє ім’я. Поліклініка та відділення характеризуються кількістю лікарів та хворих, а також мають власних завідувачів. У програмі передбачити реалізацію спеціального методу – «Пройти медичний огляд» у одного, декількох або всіх лікарів.

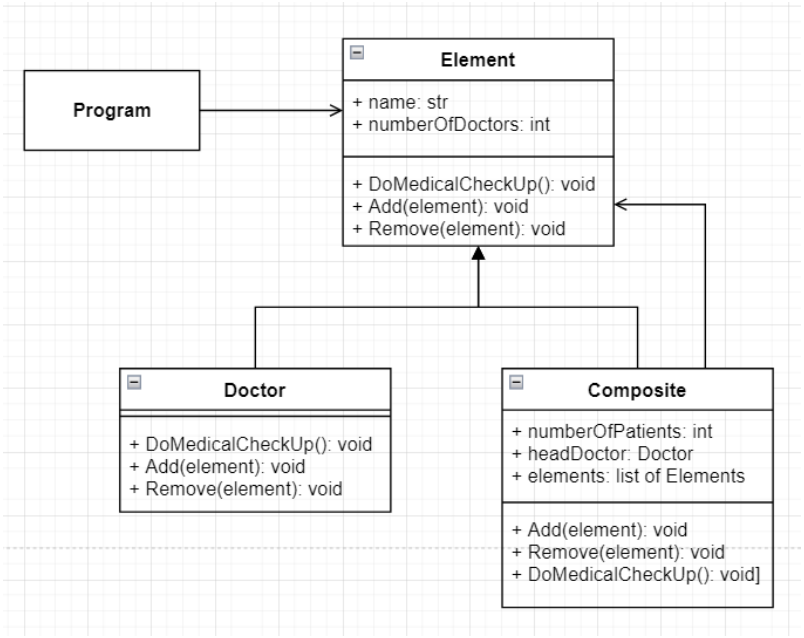
2. У старій системі реєстрації користувача необхідно було обов’язково вказувати ім’я, прізвище, місто проживання, вік, адресу електронної пошти, ключове слово, а також логін та пароль. Для прискорення процесу реєстрації необхідно вдосконалити систему: зробити таку надбудову над системою, щоб користувач повинен був вводити тільки ім’я, прізвище, логін, пароль та ключове слово. Всі інші незаповнені поля повинні отримати значення за замовчанням від системи.

Обгрунтування вибору шаблонів

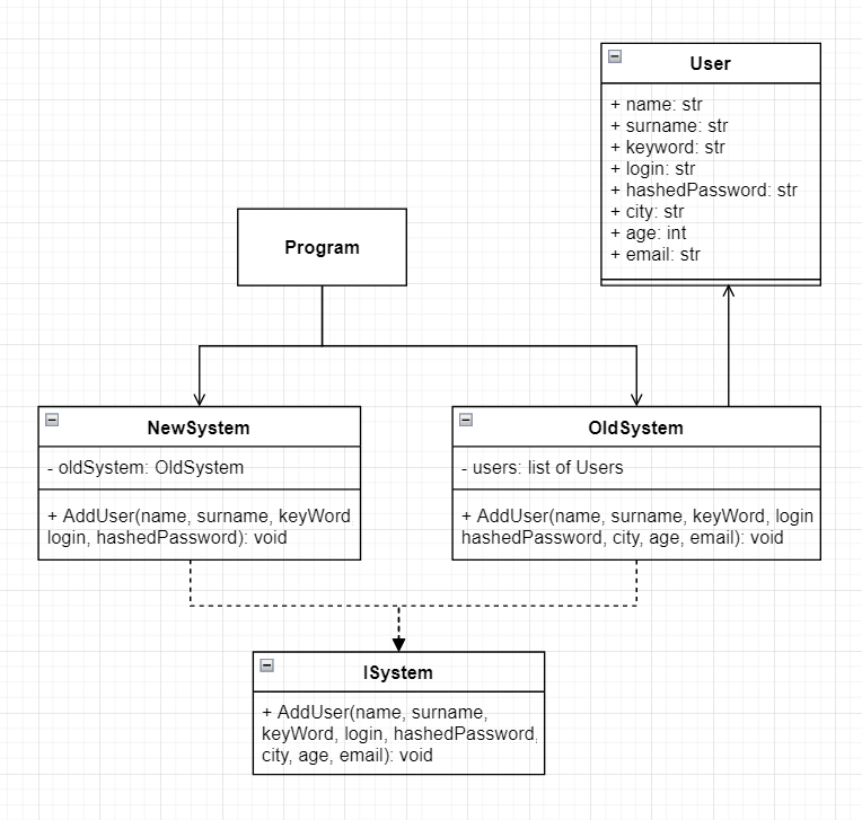
1. Для першого завдання був вибраний шаблон компонувальник (Composite), оскільки цей шаблон дозволяє групувати декілька об’єктів у деревоподібну структуру, а потім працювати з нею, так начебто це єдиний об’єкт. А в завданні якраз присутня подібна структура (поліклініка, відділення, лікар), з якою потрібно працювати таким чином.
2. Для другого завдання був обраний шаблон замісник (Proxy), оскільки він дозволяє перехопити виклик до іншого об’єкта, виконати певні дії з даними, а вже потім відправити їх далі до об’єкта. Таким чином, можна змінити взаємодію зі старою системою, не чіпаючи її, що саме і потрібно було зробити у завданні.

UML діаграми класів

Завдання 1



Завдання 2



Текст програми

Завдання 1

Program.cs

```
using System;

namespace task1
{
    class Program
    {
        static void Main()
        {
            Composite hospital = new Composite("Abraham", 10, new Doctor("Alice"));

            Composite department1 = new Composite("Dentists", 5, new Doctor("Michael"));
            department1.Add(new Doctor("Daniel"));
            department1.Add(new Doctor("Kyle"));
            department1.Add(new Doctor("Synthia"));

            hospital.Add(department1);
            hospital.Add(new Doctor("Abraam"));

            hospital.DoMedicalCheckUp();
        }
    }
}
```

Element.cs

```
namespace task1
{
    public abstract class Element
    {
        public string name;
        public int numberOfDoctors;

        public Element(string name)
        {
            this.name = name;
        }

        public abstract void Add(Element element);
        public abstract void Remove(Element element);
        public abstract void DoMedicalCheckUp();
    }
}
```

Composite.cs

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

namespace task1
{
    public class Composite: Element
    {
        private int numberOfPatients;
```

```
        private Doctor headDoctor;
        public List<Element> elements = new List<Element>();

        public Composite(string name) : base(name)
        {
            this.name = name;
        }

        public Composite(string name, int numberOfPatients, Doctor headDoctor) : base(name)
        {
            this.name = name;
            this.numberOfDoctors = 1;
            this.numberOfPatients = numberOfPatients;
            this.elements.Add(headDoctor);
            this.headDoctor = headDoctor;
        }

        public override void Add(Element element)
        {
            this.elements.Add(element);
            this.numberOfDoctors += element.numberOfDoctors;
        }

        public override void Remove(Element element)
        {
            if(this.elements.Remove(element))
            {
                this.numberOfDoctors -= element.numberOfDoctors;
            }
            else
            {
                Console.WriteLine($"Doctor {element.name} is not a part of {this.name}.");
            }
        }

        public override void DoMedicalCheckUp()
        {
            foreach(Element element in this.elements)
            {
                element.DoMedicalCheckUp();
            }
        }
    }
}
```

Doctor.cs

```
using System;

namespace task1
{
    public class Doctor: Element
    {
        public Doctor(string name) : base(name)
        {
            this.name = name;
            this.numberOfDoctors = 1;
        }

        public override void Add(Element element)
        {
            Console.WriteLine("Impossible operation.");
        }

        public override void DoMedicalCheckUp()
        {
```

```
            Console.WriteLine($"Doctor {this.name} visited.");
        }

        public override void Remove(Element element)
        {
            Console.WriteLine("Impossible operation.");
        }
    }
}
```

Завдання 2

Program.cs

```
using System;

namespace task2
{
    class Program
    {
        static void Main()
        {
            OldSystem oldSystem = new OldSystem();
            NewSystem newSystem = new NewSystem(oldSystem);

            newSystem.AddUser("Margo", "Goody", "key", "qwerty", "ew34hfj5545k45t4o34");

            oldSystem.PrintUsers();
        }
    }
}
```

ISystem.cs

```
namespace task2
{
    public interface ISystem
    {
        public void AddUser(string name, string surname, string keyWord, string login,
            string hashedPassword,string city, int age, string email);
    }
}
```

OldSystem.cs

```
using System.Collections.Generic;

namespace task2
{
    public class OldSystem: ISystem
    {
        private List<User> users = new List<User>();

        public void AddUser(string name, string surname, string keyWord, string login,
            string hashedPassword, string city, int age, string email)
        {
            this.users.Add(new User(name, surname, keyWord, login, hashedPassword, city, age, email));
```

```
        }

        // Methods to work with users

        public void PrintUsers()
        {
            foreach(User user in this.users)
            {
                System.Console.WriteLine(user + "\r\n");
            }
        }
    }
}
```

NewSystem.cs

```
namespace task2
{
    public class NewSystem: ISystem
    {
        private OldSystem oldSystem;

        public NewSystem(OldSystem oldSystem)
        {
            this.oldSystem = oldSystem;
        }

        public void AddUser(string name, string surname, string keyWord, string login,
            string hashedPassword, string city = "-", int age = 18, string email = "system@gmail.com")
        {
            this.oldSystem.AddUser(name, surname, keyWord, login, hashedPassword, city, age, email);
        }
    }
}
```

User.cs

```
namespace task2
{
    public class User
    {
        public string name;
        public string surname;
        public string keyWord;
        public string login;
        public string hashedPassword;
        public string city;
        public int age;
        public string email;

        public User(string name, string surname, string keyWord, string login,
            string hashedPassword, string city, int age, string email)
        {
            this.name = name;
            this.surname = surname;
            this.keyWord = keyWord;
            this.login = login;
            this.hashedPassword = hashedPassword;
            this.city = city;
            this.age = age;
            this.email = email;
        }
```

```
        public override string ToString()
        {
            return $"{this.name} {this.surname}:\r\n    Login: {this.login}\r\n    City: {this.city}" +
                $"\r\n    Age: {this.age} years old\r\n    Email: {this.email}";
        }
    }
}
```

Приклади результатів

Завдання 1

```
PS D:\KPI\SEMESTER 3\Progbase\lab3\task1> dotnet run
Doctor Alice visited.
Doctor Michael visited.
Doctor Daniel visited.
Doctor Kyle visited.
Doctor Synthia visited.
Doctor Abraam visited.
```

Завдання 2

```
PS D:\KPI\SEMESTER 3\Progbase\lab3\task2> dotnet run
Margo Goody:
    Login: qwerty
    City: -
    Age: 18 years old
    Email: system@gmail.com
```

Висновки

Під час лабораторної роботи я ознайомилася зі структурними шаблонами, дізналась про особливості їх використання. Навчилась відрізняти, де саме який шаблон потрібно використовувати. А також попрактикувалась у розробленні програмного забезпечення на мові C# з використанням шаблонів Конпонувальник та Замісник.