Лабораторна робота №2

Тема: Породжувальні шаблони

Мета роботи: навчитися реалізовувати породжувальні шаблони проєктування

Хід роботи

Завдання 1: Фабричний метод. 1. Напишіть систему класів для реалізації функціоналу створення різних типів підписок для відео провайдера. 2. Кожна з підписок повинна мати щомісячну плату, мінімальний період підписки та список каналів й інших можливостей. 3. Види підписок: DomesticSubscription, EducationalSubscription, PremiumSubscription. 4. Придбати (тобто створити) підписку можна за допомогою трьох різних класів: WebSite, MobileApp, ManagerCall, кожен з них має реалізувати свою логіку створення підписок. 5. Покажіть правильність роботи свого коду запустивши його в головному методі програми. 6. Підготуйте діаграму створених у програмі класів та інтерфейсів за допомогою https://app.diagrams.net/, експортуйте та завантажте її до репозиторія.

Лістинг програми:

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

abstract class Subscription
{
    public double MonthlyFee { get; protected set; }
    public int MinimumSubscriptionPeriod { get; protected set; }
    public List<string> IncludedChannels { get; protected set; }

    public Subscription(double monthlyFee, int minimumSubscriptionPeriod, List<string>
includedChannels)
    {
        MonthlyFee = monthlyFee;
        MinimumSubscriptionPeriod = minimumSubscriptionPeriod;
        IncludedChannels = includedChannels;
    }
}

class DomesticSubscription : Subscription
{
    public DomesticSubscription() : base(10.99, 1, new List<string> { "News",
"Entertainment" })
    {
    }
}

class EducationalSubscription : Subscription
{
    public EducationalSubscription() : base(15.99, 6, new List<string> { "Documentary",
"Educational" })
    {
    }
}

class PremiumSubscription : Subscription
{
    public PremiumSubscription() : base(29.99, 12, new List<string> { "All Channels",
"On-demand" })
```

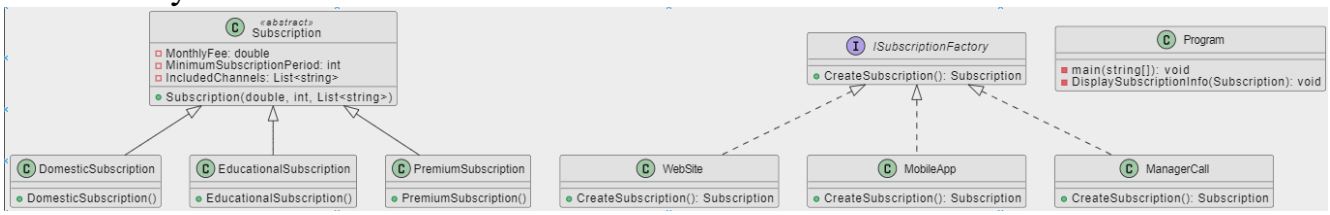
					ДУ «Житомирська політехніка».23.121.27.000 – Лр2		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Свистанюк Н.О.			Звіт з лабораторної роботи	Літ.	Арк. 1 / Аркушів 12
Перевір.		Фант М.О.					
Керівник						ФІКТ Гр. ІПЗ-22-3[2]	
Н. контр.							
Зав. каф.							

```
 {
    }
}

interface ISubscriptionFactory
{
    Subscription CreateSubscription();
}

class WebSite : ISubscriptionFactory
{
    public Subscription CreateSubscription()
    {
        return new DomesticSubscription();
    }
}

class MobileApp : ISubscriptionFactory
{
    public Subscription CreateSubscription()
    {
        return new EducationalSubscription();
    }
}

class ManagerCall : ISubscriptionFactory
{
    public Subscription CreateSubscription()
    {
        return new PremiumSubscription();
    }
}

class Program
{
    static void Main(string[] args)
    {
        ISubscriptionFactory websiteFactory = new WebSite();
        Subscription websiteSubscription = websiteFactory.CreateSubscription();

        ISubscriptionFactory appFactory = new MobileApp();
        Subscription appSubscription = appFactory.CreateSubscription();

        ISubscriptionFactory callFactory = new ManagerCall();
        Subscription callSubscription = callFactory.CreateSubscription();

        Console.WriteLine("Website Subscription:");
        DisplaySubscriptionInfo(websiteSubscription);

        Console.WriteLine("\nMobile App Subscription:");
        DisplaySubscriptionInfo(appSubscription);

        Console.WriteLine("\nManager Call Subscription:");
        DisplaySubscriptionInfo(callSubscription);
    }

    static void DisplaySubscriptionInfo(Subscription subscription)
    {
        Console.WriteLine($"Monthly Fee: {subscription.MonthlyFee}");
        Console.WriteLine($"Minimum Subscription Period:
{subscription.MinimumSubscriptionPeriod} months");
        Console.WriteLine("Included Channels:");
        foreach (var channel in subscription.IncludedChannels)
        {
            Console.WriteLine($"- {channel}");
```

```
        }
    }
}
```

```
Консоль отладки Microsoft Visual Studio

Website Subscription:
Monthly Fee: 10,99
Minimum Subscription Period: 1 months
Included Channels:
- News
- Entertainment

Mobile App Subscription:
Monthly Fee: 15,99
Minimum Subscription Period: 6 months
Included Channels:
- Documentary
- Educational

Manager Call Subscription:
Monthly Fee: 29,99
Minimum Subscription Period: 12 months
Included Channels:
- All Channels
- On-demand

C:\Users\Acer\source\repos\KPZ\Lab02_KPZ_SvistaniukNazar\FabrickMetho
) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите
томатически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:
```

Мал.1. Результат виконання



Мал.2.UML-Diagram

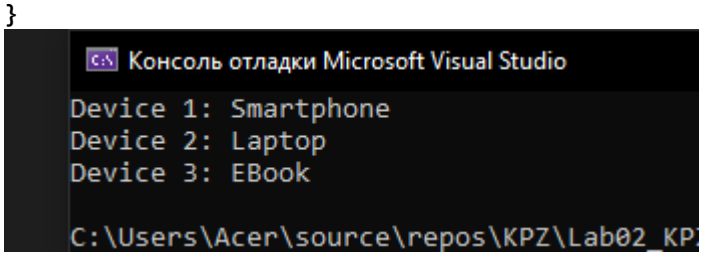
Завдання 2: Абстрактна фабрика. 1. Створіть фабрику виробництва техніки. 2. На фабриці мають створюватися різні девайси (наприклад, Laptop, Netbook, EBook, Smartphone) для різних брендів (IProne, Kiaomi, Balaxy). 3. Покажіть правильність роботи свого коду запустивши його в головному методі програми. 4. Підготуйте діаграму створених у програмі класів та інтерфейсів за допомогою https://app.diagrams.net/, експортуйте та завантажте її до репозиторія.

Лістинг програми:

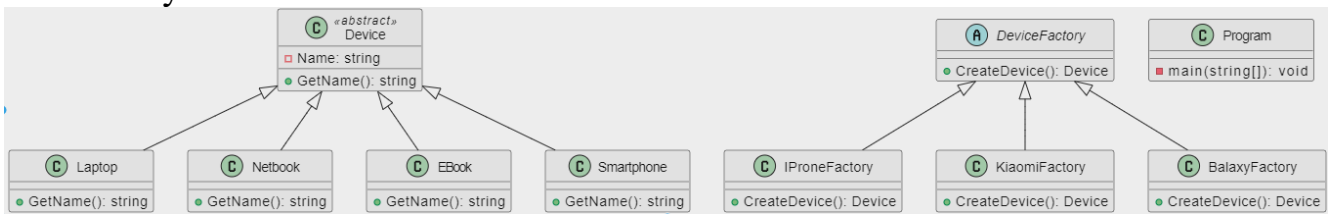
```
using System;

namespace FabrickAbstract {
abstract class Device
{
    public abstract string Name { get; }
}

class Laptop : Device
{
```

```
    public override string Name => "Laptop";
}

class Netbook : Device
{
    public override string Name => "Netbook";
}

class EBook : Device
{
    public override string Name => "EBook";
}

class Smartphone : Device
{
    public override string Name => "Smartphone";
}

abstract class DeviceFactory
{
    public abstract Device CreateDevice();
}

class IProneFactory : DeviceFactory
{
    public override Device CreateDevice()
    {
        return new Smartphone();
    }
}

class KiaomiFactory : DeviceFactory
{
    public override Device CreateDevice()
    {
        return new Laptop();
    }
}

class BalaxyFactory : DeviceFactory
{
    public override Device CreateDevice()
    {
        return new EBook();
    }
}

class Program
{
    static void Main(string[] args)
    {
        DeviceFactory factory1 = new IProneFactory();
        Device device1 = factory1.CreateDevice();

        DeviceFactory factory2 = new KiaomiFactory();
        Device device2 = factory2.CreateDevice();

        DeviceFactory factory3 = new BalaxyFactory();
        Device device3 = factory3.CreateDevice();

        Console.WriteLine($"Device 1: {device1.Name}");
        Console.WriteLine($"Device 2: {device2.Name}");
        Console.WriteLine($"Device 3: {device3.Name}");
    }
}
```

```
}
```



Мал.3.Результат виконання



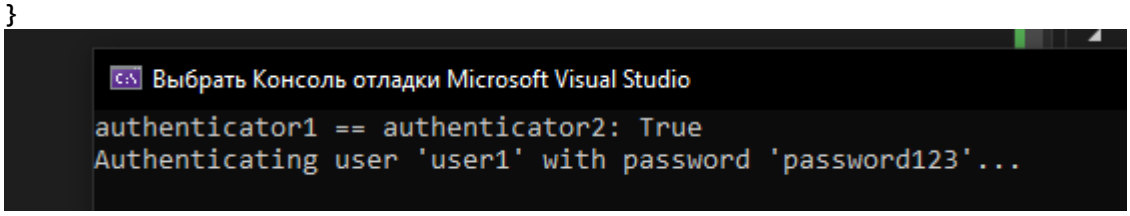
Мал.4. UML-Diagram

Завдання 3: Одинак. 1. Створіть клас Authenticator таким чином, щоб бути впевненим, що цей клас може створити лише один екземпляр, незалежно від кількості потоків і класів, що його наслідують. 2. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

Лістинг програми:

```
using System;

namespace AuthenticationSystem
{
    public class Authenticator
    {
        private static readonly object lockObject = new object();
        private static Authenticator instance;

        private Authenticator() { }

        public static Authenticator Instance
        {
            get
            {
                lock (lockObject)
                {
                    if (instance == null)
                    {
                        instance = new Authenticator();
                    }
                    return instance;
                }
            }
        }

        public void Authenticate(string username, string password)
        {
            Console.WriteLine($"Authenticating user '{username}' with password
'{password}'...");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Authenticator authenticator1 = Authenticator.Instance;
            Authenticator authenticator2 = Authenticator.Instance;
```

```
            Console.WriteLine($"authenticator1 == authenticator2: {authenticator1 ==
authenticator2}");

            authenticator1.Authenticate("user1", "password123");
        }
    }
}
```



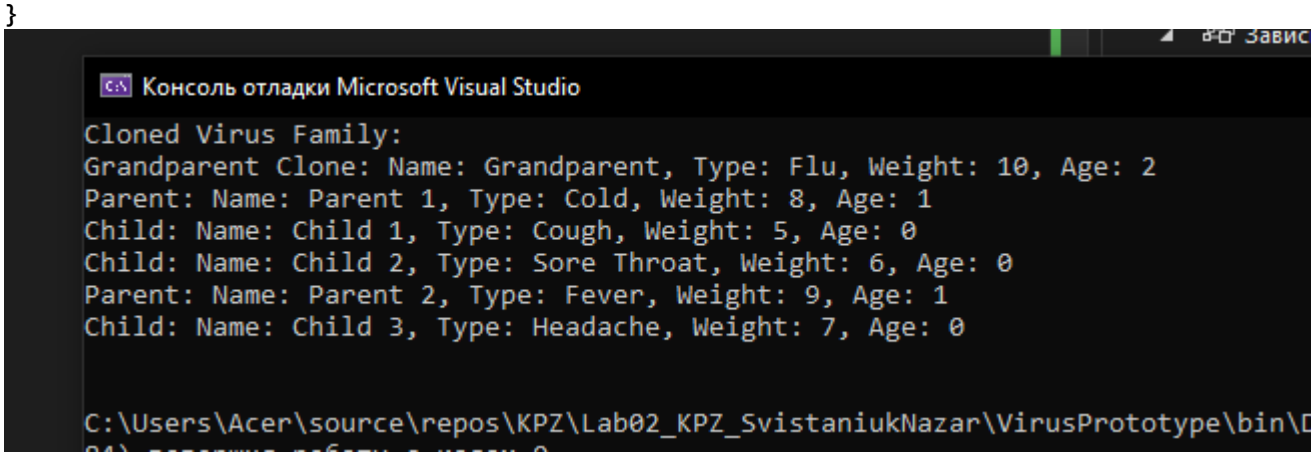
Мал.5.Результат виконання

Завдання 4: Прототип. 1. Створіть клас Virus. Він повинен містити вагу, вік, ім'я, вид і масив дітей, екземплярів Virus. 2. Створіть екземпляри для цілого "сімейства" вірусів (мінімум три покоління). 3. За допомогою шаблону Прототип реалізуйте можливість клонування наявних вірусів. 4. При клонуванні віруса-батька повинні клонуватися всі його діти. 5. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

Лістинг програми:

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

namespace VirusPrototype
{
    // Клас Virus являє собою прототип для клонування
    class Virus : ICloneable
    {
        public int Weight { get; set; }
        public int Age { get; set; }
        public string Name { get; set; }
        public string Type { get; set; }
        public List<Virus> Children { get; set; }

        public Virus(int weight, int age, string name, string type)
        {
            Weight = weight;
            Age = age;
            Name = name;
            Type = type;
            Children = new List<Virus>();
        }

        // Метод для додавання дитини
        public void AddChild(Virus child)
        {
            Children.Add(child);
        }

        // Метод для клонування віруса та всіх його дітей
        public object Clone()
        {
            Virus clone = new Virus(Weight, Age, Name, Type);
            foreach (var child in Children)
            {
                clone.AddChild((Virus)child.Clone());
            }
            return clone;
        }
```

```
        // Перевизначення методу ToString для зручного виводу
        public override string ToString()
        {
            return $"Name: {Name}, Type: {Type}, Weight: {Weight}, Age: {Age}";
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            // Створення родини вірусів
            Virus grandparent = new Virus(10, 2, "Grandparent", "Flu");
            Virus parent1 = new Virus(8, 1, "Parent 1", "Cold");
            Virus parent2 = new Virus(9, 1, "Parent 2", "Fever");

            Virus child1 = new Virus(5, 0, "Child 1", "Cough");
            Virus child2 = new Virus(6, 0, "Child 2", "Sore Throat");
            Virus child3 = new Virus(7, 0, "Child 3", "Headache");

            grandparent.AddChild(parent1);
            grandparent.AddChild(parent2);
            parent1.AddChild(child1);
            parent1.AddChild(child2);
            parent2.AddChild(child3);

            // Клонування вірусів
            Virus grandparentClone = (Virus)grandparent.Clone();

            // Виведення інформації про клон та його дітей
            Console.WriteLine("Cloned Virus Family:");
            Console.WriteLine("Grandparent Clone: " + grandparentClone.ToString());
            foreach (var parent in grandparentClone.Children)
            {
                Console.WriteLine("Parent: " + parent.ToString());
                foreach (var child in parent.Children)
                {
                    Console.WriteLine("Child: " + child.ToString());
                }
            }

            // Затримка консолі
            Console.ReadLine();
        }
    }
}
```



Мал.6.Результат виконання

Завдання 5: Будівельник. 1. Створіть клас HeroBuilder, який буде створювати персонажа гри, поступово додаючи до нього різні ознаки, наприклад зріст, статуру, колір волосся, очей, одяг, інвентар тощо (можете включити фантазію). 2. Створіть клас EnemyBuilder, який буде реалізовувати єдиний інтерфейс з HeroBuilder. Відмінністю між ними можуть бути спеціальні методи для творення добра або зла, а також списки добрих і злих справ відповідно. 3. За допомогою свого білдера і класу-директора створіть героя (або героїню) своєї мрії ☺, а також свого найзапеклішого ворога. 4. Зверніть увагу, що Ваші білдери повинні реалізовувати текучий интерфейс (fluent interface). 5. Покажіть правильність роботи свого коду запустивши його в головному методі програми. 6. Підготуйте діаграму створених у програмі класів та інтерфейсів за допомогою https://app.diagrams.net/, експортуйте та завантажте її до репозиторія.

Лістинг програми:

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

// Інтерфейс Builder
public interface ICharacterBuilder
{
    ICharacterBuilder SetHeight(int height);
    ICharacterBuilder SetBodyType(string bodyType);
    ICharacterBuilder SetHairColor(string hairColor);
    ICharacterBuilder SetEyeColor(string eyeColor);
    ICharacterBuilder SetClothing(string clothing);
    ICharacterBuilder SetInventory(List<string> inventory);
    Character Build();
}

// Клас Character
public class Character
{
    public int Height { get; set; }
    public string BodyType { get; set; }
    public string HairColor { get; set; }
    public string EyeColor { get; set; }
    public string Clothing { get; set; }
    public List<string> Inventory { get; set; }
    public List<string> Deeds { get; set; }
    public string Alignment { get; set; }

    public override string ToString()
    {
        return $"Height: {Height}\n" +
               $"BodyType: {BodyType}\n" +
               $"HairColor: {HairColor}\n" +
               $"EyeColor: {EyeColor}\n" +
               $"Clothing: {Clothing}\n" +
               $"Inventory: {string.Join(", ", Inventory)}\n" +
               $"Alignment: {Alignment}\n" +
               $"Deeds: {string.Join(", ", Deeds)}\n";
    }
}

// Клас HeroBuilder
public class HeroBuilder : ICharacterBuilder
{
    private Character _character;

    public HeroBuilder()
```

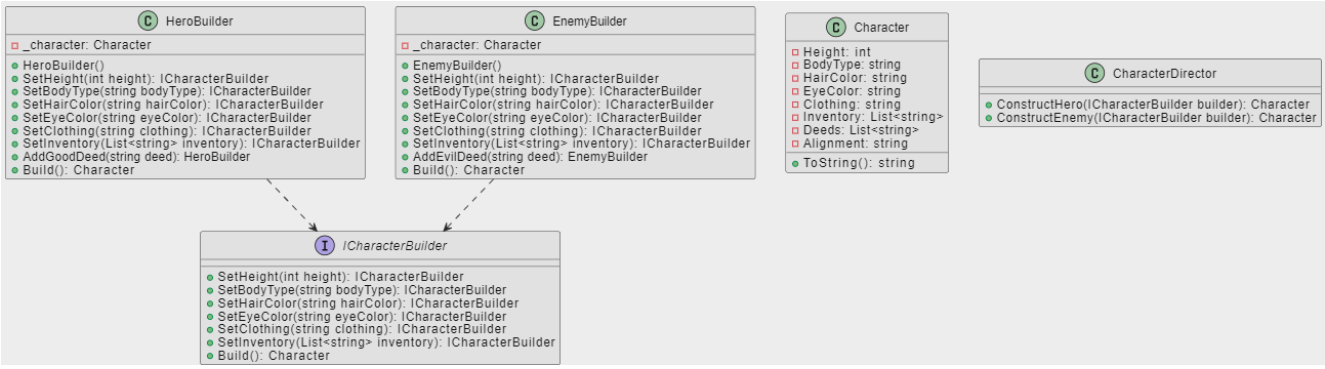
```
    {
        _character = new Character { Deeds = new List<string>(), Alignment = "Good" };
    }

    public ICharacterBuilder SetHeight(int height)
    {
        _character.Height = height;
        return this;
    }

    public ICharacterBuilder SetBodyType(string bodyType)
    {
        _character.BodyType = bodyType;
        return this;
    }

    public ICharacterBuilder SetHairColor(string hairColor)
    {
        _character.HairColor = hairColor;
        return this;
    }

    public ICharacterBuilder SetEyeColor(string eyeColor)
    {
        _character.EyeColor = eyeColor;
        return this;
    }

    public ICharacterBuilder SetClothing(string clothing)
    {
        _character.Clothing = clothing;
        return this;
    }

    public ICharacterBuilder SetInventory(List<string> inventory)
    {
        _character.Inventory = inventory;
        return this;
    }

    public HeroBuilder AddGoodDeed(string deed)
    {
        _character.Deeds.Add(deed);
        return this;
    }

    public Character Build()
    {
        return _character;
    }
}

// Клас EnemyBuilder
public class EnemyBuilder : ICharacterBuilder
{
    private Character _character;

    public EnemyBuilder()
    {
        _character = new Character { Deeds = new List<string>(), Alignment = "Evil" };
    }

    public ICharacterBuilder SetHeight(int height)
    {
        _character.Height = height;
```

```
        return this;
    }

    public ICharacterBuilder SetBodyType(string bodyType)
    {
        _character.BodyType = bodyType;
        return this;
    }

    public ICharacterBuilder SetHairColor(string hairColor)
    {
        _character.HairColor = hairColor;
        return this;
    }

    public ICharacterBuilder SetEyeColor(string eyeColor)
    {
        _character.EyeColor = eyeColor;
        return this;
    }

    public ICharacterBuilder SetClothing(string clothing)
    {
        _character.Clothing = clothing;
        return this;
    }

    public ICharacterBuilder SetInventory(List<string> inventory)
    {
        _character.Inventory = inventory;
        return this;
    }

    public EnemyBuilder AddEvilDeed(string deed)
    {
        _character.Deeds.Add(deed);
        return this;
    }

    public Character Build()
    {
        return _character;
    }
}

// Клас Director
public class CharacterDirector
{
    public Character ConstructHero(ICharacterBuilder builder)
    {
        return builder
            .SetHeight(180)
            .SetBodyType("Muscular")
            .SetHairColor("Blonde")
            .SetEyeColor("Blue")
            .SetClothing("Armor")
            .SetInventory(new List<string> { "Sword", "Shield" })
            .Build();
    }

    public Character ConstructEnemy(ICharacterBuilder builder)
    {
        return builder
            .SetHeight(175)
            .SetBodyType("Lean")
```

```
            .SetHairColor("Black")
            .SetEyeColor("Red")
            .SetClothing("Dark Robes")
            .SetInventory(new List<string> { "Dagger", "Potion" })
            .Build();
    }
}

class Program
{
    static void Main(string[] args)
    {
        var heroBuilder = new HeroBuilder();
        var enemyBuilder = new EnemyBuilder();
        var director = new CharacterDirector();

        var hero = director.ConstructHero(heroBuilder);
        heroBuilder.AddGoodDeed("Saved a village").AddGoodDeed("Defeated a dragon");

        var enemy = director.ConstructEnemy(enemyBuilder);
        enemyBuilder.AddEvilDeed("Destroyed a town").AddEvilDeed("Stole a treasure");

        Console.WriteLine("Hero:");
        Console.WriteLine(hero);

        Console.WriteLine("Enemy:");
        Console.WriteLine(enemy);
    }
}
```

```
Консоль отладки Microsoft Visual Studio
Hero:
Height: 180
BodyType: Muscular
HairColor: Blonde
EyeColor: Blue
Clothing: Armor
Inventory: Sword, Shield
Alignment: Good
Deeds: Saved a village, Defeated a dragon

Enemy:
Height: 175
BodyType: Lean
HairColor: Black
EyeColor: Red
Clothing: Dark Robes
Inventory: Dagger, Potion
Alignment: Evil
Deeds: Destroyed a town, Stole a treasure


C:\Users\Acer\source\repos\KPZ\Lab02_KPZ_SvistaniukNazar\Builde
```

Мал.7.Результат виконання



Мал.8.UML-Diagram