Лабораторна робота №2

Тема: Породжувальні шаблони

Мета роботи: навчитися реалізовувати породжувальні шаблони проєктування

Хід Роботи

Репозиторій: https://github.com/Oleksandr-Nagal/KPZ

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.

Код:

```
using System;
using System.Collections.Generic;
public abstract class Subscription
{
    public decimal MonthlyFee { get; protected set; }
    public int MinPeriod { get; protected set; }
    public List<string> Channels { get; protected set; }
}
public class DomesticSubscription : Subscription
{
    public DomesticSubscription()
    {
        MonthlyFee = 10;
        MinPeriod = 1;
        Channels = new List<string> { "Domestic Channels" };
    }
}

public class EducationalSubscription : Subscription
{
    public EducationalSubscription()
    {
        MonthlyFee = 20;
        MinPeriod = 3;
        Channels = new List<string> { "Educational Channels" };
    }
}
public class PremiumSubscription : Subscription
{
```

```
    public PremiumSubscription()
    {
        MonthlyFee = 30;
        MinPeriod = 6;
        Channels = new List<string> { "Premium Channels" };
    }
}
public abstract class SubscriptionCreator
{
    public abstract Subscription CreateSubscription();
}
public class WebSite : SubscriptionCreator
{
    public override Subscription CreateSubscription()
    {
        Console.WriteLine("Creating subscription through website...");
        Console.WriteLine("Please choose the type of subscription:");
        Console.WriteLine("1. Domestic");
        Console.WriteLine("2. Educational");
        Console.WriteLine("3. Premium");
        int choice = int.Parse(Console.ReadLine());
        switch (choice)
        {
            case 1:
                return new DomesticSubscription();
            case 2:
                return new EducationalSubscription();
            case 3:
                return new PremiumSubscription();
            default:
                throw new ArgumentException("Invalid choice");
        }
    }
}
public class MobileApp : SubscriptionCreator
{
    public override Subscription CreateSubscription()
    {
        Console.WriteLine("Creating subscription through mobile app...");
        Console.WriteLine("Please choose the type of subscription:");
        Console.WriteLine("1. Domestic");
        Console.WriteLine("2. Educational");
        Console.WriteLine("3. Premium");
        int choice = int.Parse(Console.ReadLine());
        switch (choice)
        {
            case 1:
                return new DomesticSubscription();
            case 2:
                return new EducationalSubscription();
            case 3:
                return new PremiumSubscription();
            default:
                throw new ArgumentException("Invalid choice");
        }
    }
}
public class ManagerCall : SubscriptionCreator
{
    public override Subscription CreateSubscription()
    {
        Console.WriteLine("Creating subscription through manager call...");
        Console.WriteLine("Please choose the type of subscription:");
        Console.WriteLine("1. Domestic");
```

```
        Console.WriteLine("2. Educational");
        Console.WriteLine("3. Premium");
        int choice = int.Parse(Console.ReadLine());

        switch (choice)
        {
            case 1:
                return new DomesticSubscription();
            case 2:
                return new EducationalSubscription();
            case 3:
                return new PremiumSubscription();
            default:
                throw new ArgumentException("Invalid choice");
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        SubscriptionCreator website = new WebSite();
        Subscription subscription1 = website.CreateSubscription();
        Console.WriteLine("Місячна плата: " + subscription1.MonthlyFee);
        Console.WriteLine("Мінімальний період: " + subscription1.MinPeriod );
        Console.WriteLine(string.Join(", ", subscription1.Channels ) + '\n');
        SubscriptionCreator mobileApp = new MobileApp();
        Subscription subscription2 = mobileApp.CreateSubscription();
        Console.WriteLine("Місячна плата: " + subscription2.MonthlyFee);
        Console.WriteLine("Мінімальний період: " + subscription2.MinPeriod);
        Console.WriteLine(string.Join(", ", subscription2.Channels) + '\n');
        SubscriptionCreator managerCall = new ManagerCall();
        Subscription subscription3 = managerCall.CreateSubscription();
        Console.WriteLine("Місячна плата: " + subscription3.MonthlyFee);
        Console.WriteLine("Мінімальний період: " + subscription3.MinPeriod);
        Console.WriteLine(string.Join(", ", subscription3.Channels));
    }
}
```

Результат Виконання:

```
Creating subscription through website...
Please choose the type of subscription:
1. Domestic
2. Educational
3. Premium
1
Місячна плата: 10
Мінімальний період: 1
Domestic Channels

Creating subscription through mobile app...
Please choose the type of subscription:
1. Domestic
2. Educational
3. Premium
2
Місячна плата: 20
Мінімальний період: 3
Educational Channels

Creating subscription through manager call...
Please choose the type of subscription:
1. Domestic
2. Educational
3. Premium
3
Місячна плата: 30
Мінімальний період: 6
Premium Channels
```

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.

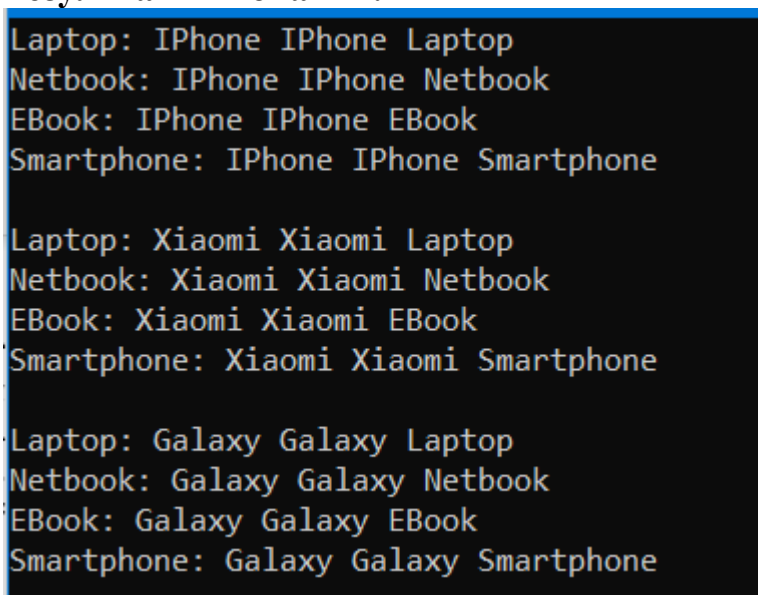
Код:

```
using System;

public abstract class Device
{
    public string Model { get; set; }
    public string Brand { get; set; }

    public abstract void DisplayInfo();
}

public class Laptop : Device
{
    public override void DisplayInfo()
    {
        Console.WriteLine($"Laptop: {Brand} {Model}");
    }
}

public class Netbook : Device
{
    public override void DisplayInfo()
    {
        Console.WriteLine($"Netbook: {Brand} {Model}");
    }
}

public class EBook : Device
{
    public override void DisplayInfo()
    {
        Console.WriteLine($"EBook: {Brand} {Model}");
    }
}
```

```
public class Smartphone : Device
{
    public override void DisplayInfo()
    {
        Console.WriteLine($"Smartphone: {Brand} {Model}\n");
    }
}

public abstract class DeviceFactory
{
    public abstract Laptop CreateLaptop();
    public abstract Netbook CreateNetbook();
    public abstract EBook CreateEBook();
    public abstract Smartphone CreateSmartphone();
}

public class IPhoneFactory : DeviceFactory
{
    public override Laptop CreateLaptop()
    {
        return new Laptop { Brand = "IPhone", Model = "IPhone Laptop" };
    }

    public override Netbook CreateNetbook()
    {
        return new Netbook { Brand = "IPhone", Model = "IPhone Netbook" };
    }

    public override EBook CreateEBook()
    {
        return new EBook { Brand = "IPhone", Model = "IPhone EBook" };
    }

    public override Smartphone CreateSmartphone()
    {
        return new Smartphone { Brand = "IPhone", Model = "IPhone Smartphone" };
    }
}

public class XiaomiFactory : DeviceFactory
{
    public override Laptop CreateLaptop()
    {
        return new Laptop { Brand = "Xiaomi", Model = "Xiaomi Laptop" };
    }

    public override Netbook CreateNetbook()
    {
        return new Netbook { Brand = "Xiaomi", Model = "Xiaomi Netbook" };
    }

    public override EBook CreateEBook()
    {
        return new EBook { Brand = "Xiaomi", Model = "Xiaomi EBook" };
    }

    public override Smartphone CreateSmartphone()
    {
        return new Smartphone { Brand = "Xiaomi", Model = "Xiaomi Smartphone" };
    }
}

public class GalaxyFactory : DeviceFactory
{
```

```
    public override Laptop CreateLaptop()
    {
        return new Laptop { Brand = "Galaxy", Model = "Galaxy Laptop" };
    }

    public override Netbook CreateNetbook()
    {
        return new Netbook { Brand = "Galaxy", Model = "Galaxy Netbook" };
    }

    public override EBook CreateEBook()
    {
        return new EBook { Brand = "Galaxy", Model = "Galaxy EBook" };
    }

    public override Smartphone CreateSmartphone()
    {
        return new Smartphone { Brand = "Galaxy", Model = "Galaxy Smartphone" };
    }
}

class Program
{
    static void Main(string[] args)
    {
        DeviceFactory factory1 = new IPhoneFactory();
        Device laptop1 = factory1.CreateLaptop();
        Device netbook1 = factory1.CreateNetbook();
        Device ebook1 = factory1.CreateEBook();
        Device smartphone1 = factory1.CreateSmartphone();

        laptop1.DisplayInfo();
        netbook1.DisplayInfo();
        ebook1.DisplayInfo();
        smartphone1.DisplayInfo();

        DeviceFactory factory2 = new XiaomiFactory();
        Device laptop2 = factory2.CreateLaptop();
        Device netbook2 = factory2.CreateNetbook();
        Device ebook2 = factory2.CreateEBook();
        Device smartphone2 = factory2.CreateSmartphone();

        laptop2.DisplayInfo();
        netbook2.DisplayInfo();
        ebook2.DisplayInfo();
        smartphone2.DisplayInfo();

        DeviceFactory factory3 = new GalaxyFactory();
        Device laptop3 = factory3.CreateLaptop();
        Device netbook3 = factory3.CreateNetbook();
        Device ebook3 = factory3.CreateEBook();
        Device smartphone3 = factory3.CreateSmartphone();

        laptop3.DisplayInfo();
        netbook3.DisplayInfo();
        ebook3.DisplayInfo();
        smartphone3.DisplayInfo();
    }
}
```

Результат Виконання:



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.

Код:

```
public class Authenticator
{
    private static Authenticator instance;

    private Authenticator() { }

    public static Authenticator GetInstance()
    {
        if (instance == null)
        {
            instance = new Authenticator();
        }
        return instance;
    }
}

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

        Console.WriteLine(authenticator1 == authenticator2); // Виведе "True", якщо
обидва об'єкти є посиланням на один і той же екземпляр
    }
}
```

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.

Код:

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

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

    public object Clone()
    {
        Virus clone = new Virus(this.Weight, this.Age, this.Name, this.Type);
        foreach (var child in Children)
        {
            clone.Children.Add((Virus)child.Clone());
        }
        return clone;
    }

    public void DisplayInfo()
    {
        Console.WriteLine($"Name: {Name}, Type: {Type}, Weight: {Weight}, Age: {Age}");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Virus grandparentVirus = new Virus(2.5, 1, "Grandparent Virus", "A");
        Virus parentVirus1 = new Virus(1.8, 2, "Parent Virus 1", "B");
        Virus parentVirus2 = new Virus(2.0, 2, "Parent Virus 2", "C");
```

```
        grandparentVirus.Children.Add(parentVirus1);
        grandparentVirus.Children.Add(parentVirus2);

        Virus childVirus1 = new Virus(1.2, 1, "Child Virus 1", "D");
        Virus childVirus2 = new Virus(1.0, 1, "Child Virus 2", "E");
        parentVirus1.Children.Add(childVirus1);
        parentVirus1.Children.Add(childVirus2);

        Virus grandChildVirus1 = new Virus(0.8, 1, "Grandchild Virus 1", "F");
        Virus grandChildVirus2 = new Virus(0.7, 1, "Grandchild Virus 2", "G");
        childVirus1.Children.Add(grandChildVirus1);
        childVirus2.Children.Add(grandChildVirus2);

        Virus clonedGrandparent = (Virus)grandparentVirus.Clone();
        Virus clonedParent1 = (Virus)parentVirus1.Clone();

        Console.WriteLine("Cloned Grandparent Virus:");
        clonedGrandparent.DisplayInfo();
        Console.WriteLine("\nCloned Parent Virus 1:");
        clonedParent1.DisplayInfo();
    }
}
```

Результат Виконання:

```
Cloned Grandparent Virus:
Name: Grandparent Virus, Type: A, Weight: 2,5, Age: 1

Cloned Parent Virus 1:
Name: Parent Virus 1, Type: B, Weight: 1,8, Age: 2
```

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.

Код:

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

// Клас для створення героїв
public class HeroBuilder
{
    protected string Name;
    protected string Gender;
    protected int Height;
    protected string HairColor;
    protected string EyeColor;
    protected string Clothing;
    protected List<string> Inventory;

    public HeroBuilder SetName(string name)
    {
        Name = name;
        return this;
    }

    public HeroBuilder SetGender(string gender)
    {
        Gender = gender;
        return this;
```

```
    }

    public HeroBuilder SetHeight(int height)
    {
        Height = height;
        return this;
    }

    public HeroBuilder SetHairColor(string hairColor)
    {
        HairColor = hairColor;
        return this;
    }

    public HeroBuilder SetEyeColor(string eyeColor)
    {
        EyeColor = eyeColor;
        return this;
    }

    public HeroBuilder SetClothing(string clothing)
    {
        Clothing = clothing;
        return this;
    }

    public HeroBuilder AddToInventory(string item)
    {
        if (Inventory == null)
        {
            Inventory = new List<string>();
        }
        Inventory.Add(item);
        return this;
    }

    public Hero Build()
    {
        return new Hero(Name, Gender, Height, HairColor, EyeColor, Clothing, Inventory);
    }

    public Enemy BuildEnemy(List<string> evilDeeds)
    {
        return new Enemy(Name, Gender, Height, HairColor, EyeColor, Clothing, Inventory,
evilDeeds);
    }
}

// Клас героя
public class Hero
{
    public string Name { get; }
    public string Gender { get; }
    public int Height { get; }
    public string HairColor { get; }
    public string EyeColor { get; }
    public string Clothing { get; }
    public List<string> Inventory { get; }

    public Hero(string name, string gender, int height, string hairColor, string
eyeColor, string clothing, List<string> inventory)
    {
        Name = name;
        Gender = gender;
```

```
        Height = height;
        HairColor = hairColor;
        EyeColor = eyeColor;
        Clothing = clothing;
        Inventory = inventory;
    }

    public void ShowInfo()
    {
        Console.WriteLine($"Name: {Name}, Gender: {Gender}, Height: {Height}, Hair Color:
{HairColor}, Eye Color: {EyeColor}, Clothing: {Clothing}");
        Console.WriteLine("Inventory:");
        foreach (var item in Inventory)
        {
            Console.WriteLine("- " + item);
        }
    }
}

// Клас ворога
public class Enemy : Hero
{
    public List<string> EvilDeeds { get; }

    public Enemy(string name, string gender, int height, string hairColor, string
eyeColor, string clothing, List<string> inventory, List<string> evilDeeds)
        : base(name, gender, height, hairColor, eyeColor, clothing, inventory)
    {
        EvilDeeds = evilDeeds;
    }

    public void ShowEvilDeeds()
    {
        Console.WriteLine("Evil Deeds:");
        foreach (var deed in EvilDeeds)
        {
            Console.WriteLine("- " + deed);
        }
    }
}

class Program
{
    static void Main(string[] args)
    {
        HeroBuilder heroBuilder = new HeroBuilder();
        Hero hero = heroBuilder
            .SetName("Hero")
            .SetGender("Male")
            .SetHeight(180)
            .SetHairColor("Brown")
            .SetEyeColor("Blue")
            .SetClothing("Armor")
            .AddToInventory("Sword")
            .AddToInventory("Shield")
            .Build();

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

        List<string> evilDeeds = new List<string> { "Destroyed the village", "Stole the
king's treasure" };
        Enemy enemy = heroBuilder
```

```
            .SetName("Enemy")
            .SetGender("Female")
            .SetHeight(160)
            .SetHairColor("Black")
            .SetEyeColor("Red")
            .SetClothing("Dark Robe")
            .AddToInventory("Magic Staff")
            .BuildEnemy(evilDeeds);

        Console.WriteLine("\nEnemy:");
        enemy.ShowInfo();
        enemy.ShowEvilDeeds();
    }
}
```

Результат Виконання:

```
Hero:
Name: Hero, Gender: Male, Height: 180, Hair Color: Brown, Eye Color: Blue, Clothing: Armor
Inventory:
- Sword
- Shield


Enemy:
Name: Enemy, Gender: Female, Height: 160, Hair Color: Black, Eye Color: Red, Clothing: Dark Robe
Inventory:
- Sword
- Shield
- Magic Staff
Evil Deeds:
- Destroyed the village
- Stole the king's treasure
```

Висновок: У ході виконання лабораторної роботи я навчився реалізовувати породжувальні шаблони проєктування.