

## Лабораторна робота №2 (максимально - 10 балів)

Тема: Породжувальні шаблони

Мета роботи: навчитися реалізовувати породжувальні шаблони проектування

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.

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

namespace FabrichniMethod
{
    // Абстрактний клас для представлення підписок
    public abstract class Subscription
    {
        public decimal MonthlyPrice { get; protected set; } // щомісячна оплата
        public int MinSubscriptionMonths { get; protected set; } // мінімальний термін
        підписки в місяцях
        public List<string> IncludedChannels { get; protected set; } // список включених
        каналів

        // Конструктор класу Subscription
        public Subscription(decimal monthlyPrice, int minSubscriptionMonths, List<string>
        includedChannels)
        {
            MonthlyPrice = monthlyPrice;
            MinSubscriptionMonths = minSubscriptionMonths;
            IncludedChannels = includedChannels;
        }

        // Конкретний клас для підписки на вітчизняні канали
        public class DomesticSubscription : Subscription
        {
            // Конструктор класу DomesticSubscription
            public DomesticSubscription() : base(10m, 1, new List<string> { "Вітчизняний
            канал 1", "Вітчизняний канал 2" }) { }
        }

        // Конкретний клас для освітньої підписки
        public class EducationalSubscription : Subscription
        {
            // Конструктор класу EducationalSubscription
            public EducationalSubscription() : base(15m, 3, new List<string> { "Освітній
            канал 1", "Освітній канал 2" }) { }
        }
    }
}
```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.22.121.35.000 –Лр.2				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Ширяєв М.Є			Звіт з лабораторної роботи №2	Літ.	Арк.	Акрушів	
Перевір.		М.О Фант					1		
Реценз.						ФІКТ, зр.ІПЗ-22-3[2]			
Н. Контр.									
Зав.каф.									

```

// Конкретний клас для преміальної підписки
public class PremiumSubscription : Subscription
{
    // Конструктор класу PremiumSubscription
    public PremiumSubscription() : base(25m, 6, new List<string> { "Преміум канал 1",
"Преміум канал 2", "Преміум канал 3" }) { }
}

// Інтерфейс для фабричного методу створення підписок
public interface ISubscriptionFactory
{
    Subscription CreateSubscription();
}

// Реалізація фабрики для створення підписок через веб-сайт
public class WebSite : ISubscriptionFactory
{
    public Subscription CreateSubscription()
    {
        return new DomesticSubscription(); // Тимчасово повертаємо
DomesticSubscription
    }
}

// Реалізація фабрики для створення підписок через мобільний додаток
public class MobileApp : ISubscriptionFactory
{
    public Subscription CreateSubscription()
    {
        return new EducationalSubscription(); // Тимчасово повертаємо
EducationalSubscription
    }
}

// Реалізація фабрики для створення підписок через дзвінок менеджру
public class ManagerCall : ISubscriptionFactory
{
    public Subscription CreateSubscription()
    {
        return new PremiumSubscription(); // Тимчасово повертаємо PremiumSubscription
    }
}

internal class Program
{
    public static void Main(string[] args)
    {
        Console.OutputEncoding = System.Text.Encoding.Unicode;
        // Приклад використання фабричного методу для створення підписок
        // Створення підписки через веб-сайт
        ISubscriptionFactory website = new WebSite();
        Subscription websiteSubscription = website.CreateSubscription();
        Console.WriteLine("Підписка створена через веб-сайт:");
        DisplaySubscriptionDetails(websiteSubscription);

        // Створення підписки через мобільний додаток
        ISubscriptionFactory mobileApp = new MobileApp();
        Subscription mobileAppSubscription = mobileApp.CreateSubscription();
        Console.WriteLine("\nПідписка створена через мобільний додаток:");
        DisplaySubscriptionDetails(mobileAppSubscription);

        // Створення підписки через дзвінок менеджру
        ISubscriptionFactory managerCall = new ManagerCall();
        Subscription managerCallSubscription = managerCall.CreateSubscription();
        Console.WriteLine("\nПідписка створена через дзвінок менеджру:");
        DisplaySubscriptionDetails(managerCallSubscription);
    }
}

```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.22.121.35.000 –Лр.2	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		2

```

// Метод для виведення деталей підписки на екран
public static void DisplaySubscriptionDetails(Subscription subscription)
{
    Console.WriteLine($"Щомісячна оплата: {subscription.MonthlyPrice}$");
    Console.WriteLine($"Мінімальний термін підписки:
{subscription.MinSubscriptionMonths} місяців");
    Console.WriteLine("Включені канали:");
    foreach (var channel in subscription.IncludedChannels)
    {
        Console.WriteLine($"- {channel}");
    }
}
}
}

```

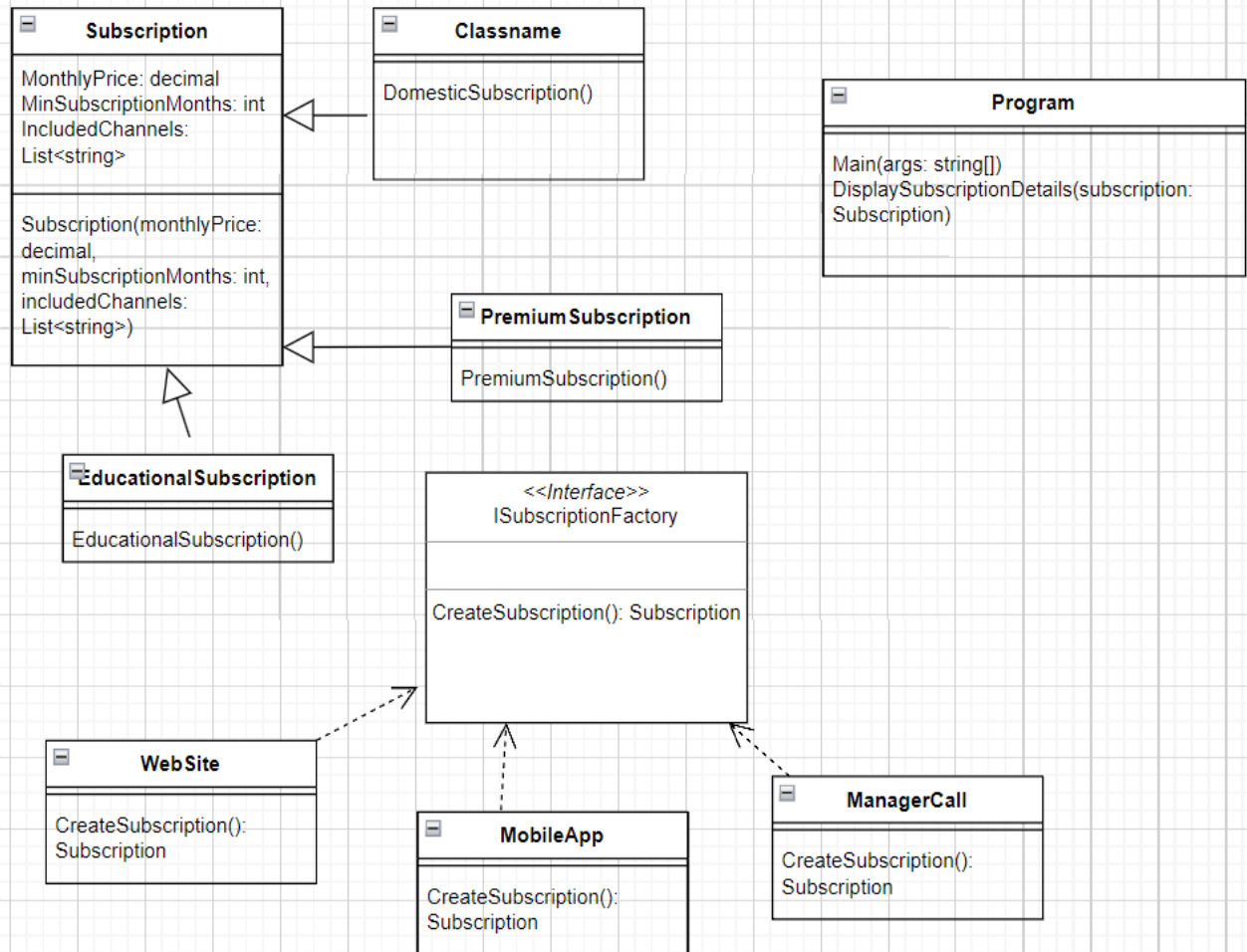
```

Microsoft Visual Studio Debug Console
Підписка створена через веб-сайт:
Щомісячна оплата: 10$
Мінімальний термін підписки: 1 місяців
Включені канали:
- Вітчизняний канал 1
- Вітчизняний канал 2

Підписка створена через мобільний додаток:
Щомісячна оплата: 15$
Мінімальний термін підписки: 3 місяців
Включені канали:
- Освітній канал 1
- Освітній канал 2

Підписка створена через дзвінок менеджеру:
Щомісячна оплата: 25$
Мінімальний термін підписки: 6 місяців
Включені канали:
- Преміум канал 1
- Преміум канал 2
- Преміум канал 3

```



Завдання 2: Абстрактна фабрика. 1. Створіть фабрику виробництва техніки. 2. На фабриці мають створюватися різні девайси (наприклад, Laptop, Netbook, EBook, Smartphone) для різних брендів (IPhone, 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.

`using System;`

```

namespace AbstractFactory
{
    // Абстрактний клас для різних типів техніки
    abstract class Device
    {
        public abstract void DisplayInfo();
    }

    // Конкретний клас для ноутбука
    class Laptop : Device
    {
        public override void DisplayInfo()
        {
            Console.WriteLine("Ноутбук");
        }
    }
}

```

```

// Конкретний клас для нетбука
class Netbook : Device
{
    public override void DisplayInfo()
    {
        Console.WriteLine("Нетбук");
    }
}

// Конкретний клас для електронної книги
class EBook : Device
{
    public override void DisplayInfo()
    {
        Console.WriteLine("Електронна книга");
    }
}

// Конкретний клас для смартфона
class Smartphone : Device
{
    public override void DisplayInfo()
    {
        Console.WriteLine("Смартфон");
    }
}

// Абстрактний клас фабрики, який визначає методи створення різних пристроїв
abstract class DeviceFactory
{
    public abstract Device CreateLaptop();
    public abstract Device CreateNetbook();
    public abstract Device CreateEBook();
    public abstract Device CreateSmartphone();
}

// Конкретна фабрика для бренду "IPhone"
class IProneFactory : DeviceFactory
{
    public override Device CreateLaptop()
    {
        return new Laptop();
    }

    public override Device CreateNetbook()
    {
        return new Netbook();
    }

    public override Device CreateEBook()
    {
        return new EBook();
    }

    public override Device CreateSmartphone()
    {
        return new Smartphone();
    }
}

```

```

// Конкретна фабрика для бренду "Kiaomi"
class XiaomiFactory : DeviceFactory
{
    public override Device CreateLaptop()
    {
        return new Laptop();
    }

    public override Device CreateNetbook()
    {
        return new Netbook();
    }

    public override Device CreateEBook()
    {
        return new EBook();
    }

    public override Device CreateSmartphone()
    {
        return new Smartphone();
    }
}

// Конкретна фабрика для бренду "Balaxy"
class BalaxyFactory : DeviceFactory
{
    public override Device CreateLaptop()
    {
        return new Laptop();
    }

    public override Device CreateNetbook()
    {
        return new Netbook();
    }

    public override Device CreateEBook()
    {
        return new EBook();
    }

    public override Device CreateSmartphone()
    {
        return new Smartphone();
    }
}

class Program
{
    static void Main(string[] args)
    {
        Console.OutputEncoding = System.Text.Encoding.Unicode;
        // Створення фабрик для різних брендів
        DeviceFactory iPhoneFactory = new iPhoneFactory();
        DeviceFactory XiaomiFactory = new XiaomiFactory();
        DeviceFactory BalaxyFactory = new BalaxyFactory();

        // Створення різних пристроїв для кожного бренду і виведення інформації про
них
        Device device1 = iPhoneFactory.CreateLaptop();
        device1.DisplayInfo();

        Device device2 = XiaomiFactory.CreateSmartphone();
        device2.DisplayInfo();
    }
}

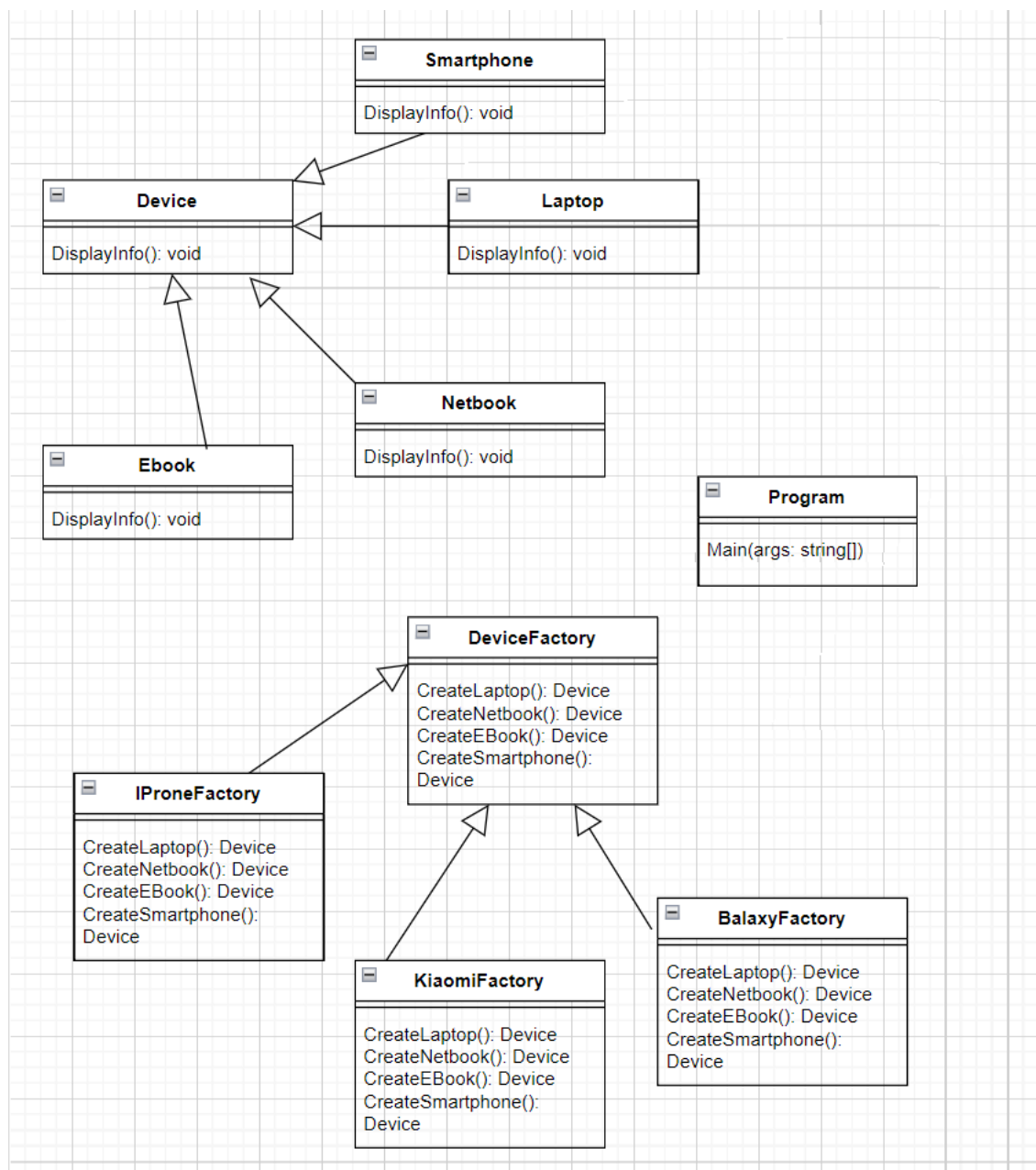
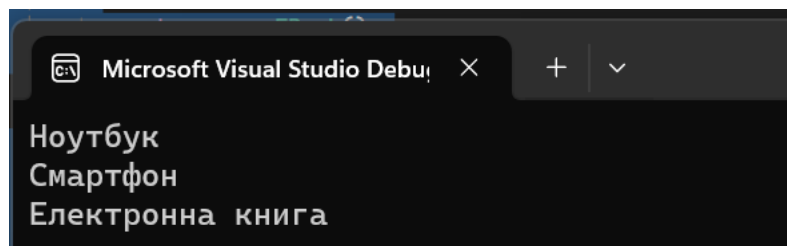
```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.22.121.35.000 –Лр.2	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

```

        Device device3 = balaxyFactory.CreateEBook();
        device3.DisplayInfo();
    }
}

```



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.

```
using System;

namespace Loner
{
    // Клас, який гарантує єдиний екземпляр
    class Authenticator
    {
        // Приватний статичний екземпляр класу
        private static Authenticator? instance;

        // Приватний конструктор, щоб заборонити створення екземплярів ззовні класу
        private Authenticator()
        {
            Console.WriteLine("Екземпляр Authenticator був створений.");
        }

        // Публічний метод для отримання єдиного екземпляру класу
        public static Authenticator GetInstance()
        {
            // Перевірка, чи екземпляр вже був створений
            if (instance == null)
            {
                // Якщо екземпляр ще не існує, створюємо новий
                instance = new Authenticator();
            }
            // Повертаємо єдиний екземпляр
            return instance;
        }

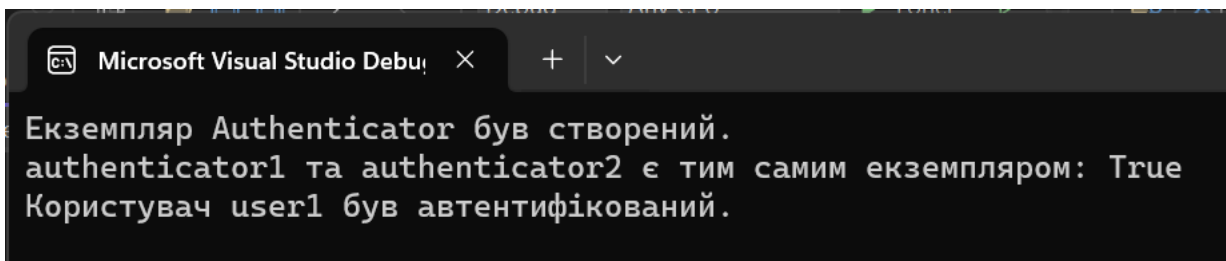
        // Метод для автентифікації користувача
        public void Authenticate(string username, string password)
        {
            Console.WriteLine("Користувач {0} був автентифікований.", username);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Console.OutputEncoding = System.Text.Encoding.Unicode;
            // Спроба отримати єдиний екземпляр Authenticator з різних місць програми
            Authenticator authenticator1 = Authenticator.GetInstance();
            Authenticator authenticator2 = Authenticator.GetInstance();

            // Перевірка на те, що отримані об'єкти є тим самим екземпляром
            Console.WriteLine("authenticator1 та authenticator2 є тим самим екземпляром: " + (authenticator1 == authenticator2));

            // Перевірка роботи методу автентифікації
            authenticator1.Authenticate("user1", "password1");
        }
    }
}
```





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.

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

namespace Prototype
{
    // Клас вірусу
    class Virus : ICloneable
    {
        public int Weight { get; set; }
        public int Age { get; set; }
        public string Name { get; set; }
        public string Species { get; set; }
        public List<Virus> Children { get; set; }

        // Конструктор
        public Virus(int weight, int age, string name, string species)
        {
            Weight = weight;
            Age = age;
            Name = name;
            Species = species;
            Children = new List<Virus>();
        }

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

        // Метод клонування
        public object Clone()
        {
            // Клонуємо основний об'єкт
            Virus clone = new Virus(Weight, Age, Name, Species);

            // Клонуємо дітей
            foreach (var child in Children)
            {
                clone.AddChild((Virus)child.Clone());
            }

            return clone;
        }
    }
}
```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.22.121.35.000 –Лр.2	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

```

// Метод для виведення інформації про вірус
public void DisplayInfo()
{
    Console.WriteLine($"Ім'я: {Name}, Вік: {Age}, Вага: {Weight}, Вид: {Species}");
    foreach (var child in Children)
    {
        child.DisplayInfo();
    }
}

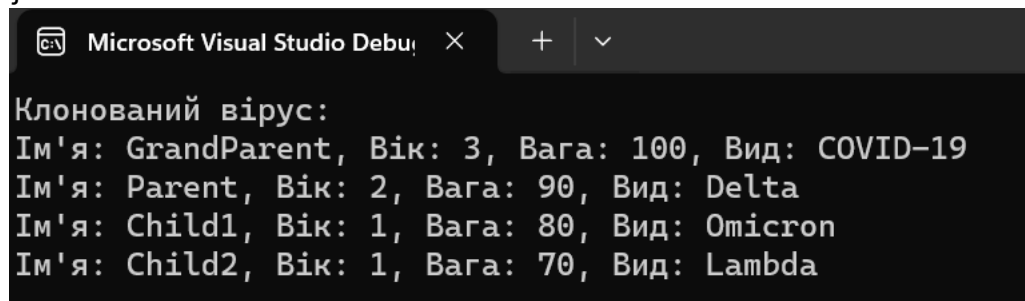
class Program
{
    static void Main(string[] args)
    {
        Console.OutputEncoding = System.Text.Encoding.Unicode;
        // Створюємо віруси
        Virus grandParentVirus = new Virus(100, 3, "GrandParent", "COVID-19");
        Virus parentVirus = new Virus(90, 2, "Parent", "Delta");
        Virus childVirus1 = new Virus(80, 1, "Child1", "Omicron");
        Virus childVirus2 = new Virus(70, 1, "Child2", "Lambda");

        // Додаємо дітей до вірусів
        grandParentVirus.AddChild(parentVirus);
        parentVirus.AddChild(childVirus1);
        parentVirus.AddChild(childVirus2);

        // Клонуємо віруси
        Virus clonedGrandParentVirus = (Virus)grandParentVirus.Clone();

        // Виводимо інформацію про клонований вірус
        Console.WriteLine("Клонований вірус:");
        clonedGrandParentVirus.DisplayInfo();
    }
}

```



```

Клонований вірус:
Ім'я: GrandParent, Вік: 3, Вага: 100, Вид: COVID-19
Ім'я: Parent, Вік: 2, Вага: 90, Вид: Delta
Ім'я: Child1, Вік: 1, Вага: 80, Вид: Omicron
Ім'я: Child2, Вік: 1, Вага: 70, Вид: Lambda

```

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 інтерфейс (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.

```

using System;
using System.Collections.Generic;

namespace Builder
{
    // Інтерфейс для будівництва персонажів
    interface ICharacterBuilder
    {
        ICharacterBuilder SetHeight(int height);
        ICharacterBuilder SetBuild(string build);
        ICharacterBuilder SetHairColor(string hairColor);
        ICharacterBuilder SetEyeColor(string eyeColor);
        ICharacterBuilder SetClothing(string clothing);
        ICharacterBuilder SetInventory(List<string> inventory);
        Character Build();
    }

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

        public void DisplayInfo()
        {
            Console.WriteLine($"Зріст: {Height}, Статура: {Build}, Волосся: {HairColor},
Очі: {EyeColor}, Одяг: {Clothing}");
            Console.WriteLine("Інвентар:");
            foreach (var item in Inventory)
            {
                Console.WriteLine($"- {item}");
            }
        }
    }

    // Клас-директор для будівництва персонажів
    class CharacterDirector
    {
        private ICharacterBuilder _builder;

        public CharacterDirector(ICharacterBuilder builder)
        {
            _builder = builder;
        }

        public void ConstructCharacter()
        {
            _builder.SetHeight(180)
                .SetBuild("Athletic")
                .SetHairColor("Brown")
                .SetEyeColor("Blue")
                .SetClothing("Armor")
                .SetInventory(new List<string> { "Sword", "Shield", "Potion" });
        }
    }
}

```

```

// Клас-бїлдер для створення героїв
class HeroBuilder : ICharacterBuilder
{
    private Character _character = new Character();

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

    public ICharacterBuilder SetBuild(string build)
    {
        _character.Build = build;
        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 Character Build()
    {
        return _character;
    }
}

// Клас-бїлдер для створення ворогїв
class EnemyBuilder : ICharacterBuilder
{
    private Character _character = new Character();

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

    public ICharacterBuilder SetBuild(string build)
    {
        _character.Build = build;
        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 Character Build()
{
    return _character;
}
}

class Program
{
    static void Main(string[] args)
    {
        Console.OutputEncoding = System.Text.Encoding.Unicode;
        // Створюємо білдера для героя
        var heroBuilder = new HeroBuilder();

        // Створюємо директора і викликаємо метод будівництва героя
        var heroDirector = new CharacterDirector(heroBuilder);
        heroDirector.ConstructCharacter();

        // Отримуємо героя
        var hero = heroBuilder.Build();

        // Виводимо інформацію про героя
        Console.WriteLine("Інформація про героя:");
        hero.DisplayInfo();

        // Тепер створимо ворога за допомогою того ж білдера
        var enemyBuilder = new EnemyBuilder();

        // Створюємо директора для ворога і викликаємо метод будівництва
        var enemyDirector = new CharacterDirector(enemyBuilder);
        enemyDirector.ConstructCharacter();

        // Отримуємо ворога
        var enemy = enemyBuilder.Build();

        // Виводимо інформацію про ворога
        Console.WriteLine("\nІнформація про ворога:");
        enemy.DisplayInfo();
    }
}

```



Інформація про героя:

Зріст: 180, Статура: Athletic, Волосся: Brown, Очі: Blue, Одяг: Armor

Інвентар:

- Sword
- Shield
- Potion

Інформація про ворога:

Зріст: 180, Статура: Athletic, Волосся: Brown, Очі: Blue, Одяг: Armor

Інвентар:

- Sword
- Shield
- Potion

