ЛАБОРАТОРНА РОБОТА № 2
Породжувальні шаблони

Мета роботи: навчитися реалізовувати породжувальні шаблони проєктування

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;

namespace SubscriptionSystem
{

    class Program
    {
        static void Main(string[] args)
        {
            Console.OutputEncoding = System.Text.Encoding.UTF8;
            Console.WriteLine("Система створення підписок");

            List<SubscriptionCreator> creators = new List<SubscriptionCreator>
            {
                new WebSite(),
                new MobileApp(),
                new ManagerCall()
            };

            foreach (var creator in creators)
            {
                ISubscription subscription = creator.CreateSubscription();
                Console.Write($"Створено через {creator.GetType().Name}: ");
                subscription.PrintDetails();
            }
        }
    }
```

					ДУ «Житомирська політехніка».24.121.21.000 – Лр2		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Полякова А.С.			Звіт з лабораторної роботи	Літ.	Арк. 1 / Аркушів 11
Перевір.		Левківський В.Л.					
Керівник						ФІКТ Гр. ІПЗ-24-2	
Н. контр.							
Зав. каф.							

```
    public interface ISubscription
    {
        decimal MonthlyPrice { get; }
        int MinSubscriptionPeriodMonths { get; }
        List<string> Channels { get; }
        void PrintDetails();
    }

    public class DomesticSubscription : ISubscription
    {
        public decimal MonthlyPrice => 10.0m;
        public int MinSubscriptionPeriodMonths => 1;
        public List<string> Channels => new List<string> { "News", "Local", "Weather" };

        public void PrintDetails() =>
            Console.WriteLine($"[Domestic] Ціна: {MonthlyPrice}$, Мін. період: {MinSub-
scriptionPeriodMonths} міс., Канали: {string.Join(", ", Channels)}");
    }

    public class EducationalSubscription : ISubscription
    {
        public decimal MonthlyPrice => 15.0m;
        public int MinSubscriptionPeriodMonths => 3;
        public List<string> Channels => new List<string> { "Science", "History", "Docu-
mentary" };

        public void PrintDetails() =>
            Console.WriteLine($"[Educational] Ціна: {MonthlyPrice}$, Мін. період: {Min-
SubscriptionPeriodMonths} міс., Канали: {string.Join(", ", Channels)}");
    }

    public class PremiumSubscription : ISubscription
    {
        public decimal MonthlyPrice => 30.0m;
        public int MinSubscriptionPeriodMonths => 1;
        public List<string> Channels => new List<string> { "All Channels", "4K Movies",
"Sports", "Music" };

        public void PrintDetails() =>
            Console.WriteLine($"[Premium] Ціна: {MonthlyPrice}$, Мін. період: {MinSub-
scriptionPeriodMonths} міс., Канали: {string.Join(", ", Channels)}");
    }

    public abstract class SubscriptionCreator
    {
        public abstract ISubscription CreateSubscription();
    }

    public class WebSite : SubscriptionCreator
    {
        public override ISubscription CreateSubscription() => new DomesticSubscription();
    }

    public class MobileApp : SubscriptionCreator
    {
        public override ISubscription CreateSubscription() => new PremiumSubscription();
    }

    public class ManagerCall : SubscriptionCreator
    {
        public override ISubscription CreateSubscription() => new EducationalSubscrip-
tion();
    }
      }
```

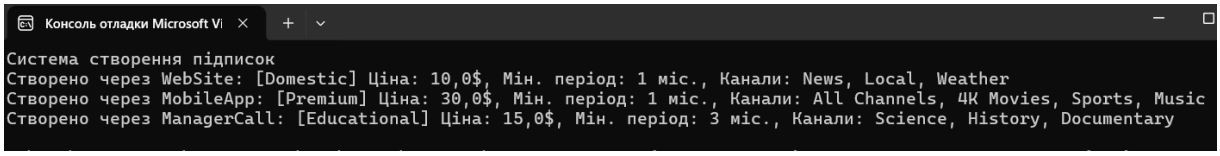

Рис.1.Результат виконання програми:

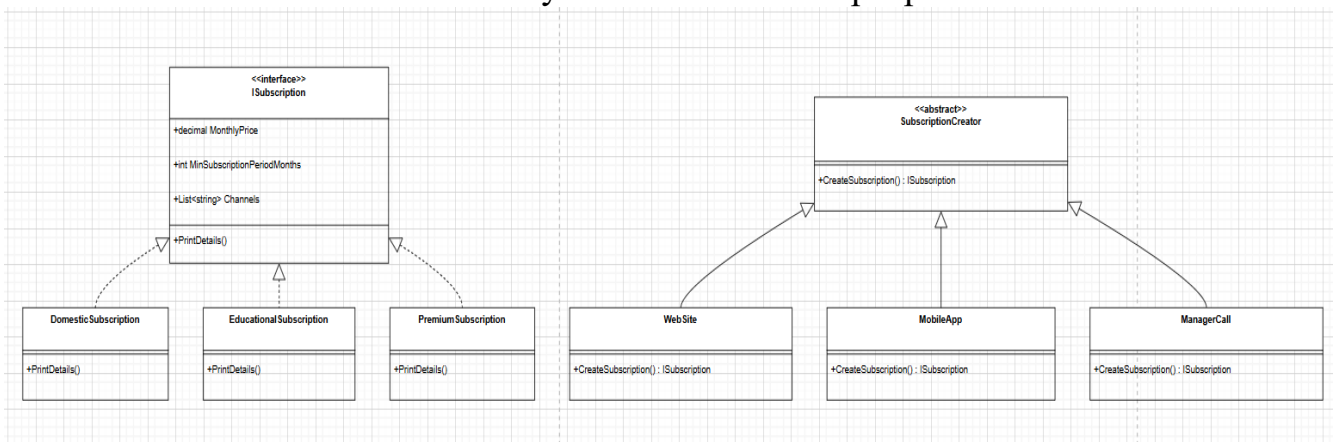

Рис.2.Даграма

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 прорами:

```
using System;

namespace AbstractFactoryTask
{

    class Program
    {
        static void Main(string[] args)
        {

            IDeviceFactory factory = new IProneFactory();
            Console.WriteLine("Фабрика IProne:");
            Console.WriteLine(factory.CreateLaptop().GetModel());
            Console.WriteLine(factory.CreateSmartphone().GetModel());


            factory = new KiaomiFactory();
            Console.WriteLine("\nФабрика Kiaomi:");
            Console.WriteLine(factory.CreateNetbook().GetModel());
            Console.WriteLine(factory.CreateEBook().GetModel());


            factory = new BalaxyFactory();
            Console.WriteLine("\nФабрика Balaxy:");
            Console.WriteLine(factory.CreateSmartphone().GetModel());
        }
    }
```

```
    public interface ILaptop { string GetModel(); }
    public interface INetbook { string GetModel(); }
    public interface IEBook { string GetModel(); }
    public interface ISmartphone { string GetModel(); }


    public interface IDeviceFactory
    {
        ILaptop CreateLaptop();
        INetbook CreateNetbook();
        IEBook CreateEBook();
        ISmartphone CreateSmartphone();
    }

    public class IProneLaptop : ILaptop { public string GetModel() => "IProne Laptop"; }
    public class IProneNetbook : INetbook { public string GetModel() => "IProne Netbook";
}
    public class IProneEBook : IEBook { public string GetModel() => "IProne EBook"; }
    public class IProneSmartphone : ISmartphone { public string GetModel() => "IProne
Smartphone"; }

    public class IProneFactory : IDeviceFactory
    {
        public ILaptop CreateLaptop() => new IProneLaptop();
        public INetbook CreateNetbook() => new IProneNetbook();
        public IEBook CreateEBook() => new IProneEBook();
        public ISmartphone CreateSmartphone() => new IProneSmartphone();
    }


    public class KiaomiLaptop : ILaptop { public string GetModel() => "Kiaomi Laptop"; }
    public class KiaomiNetbook : INetbook { public string GetModel() => "Kiaomi Netbook";
}
    public class KiaomiEBook : IEBook { public string GetModel() => "Kiaomi EBook"; }
    public class KiaomiSmartphone : ISmartphone { public string GetModel() => "Kiaomi
Smartphone"; }

    public class KiaomiFactory : IDeviceFactory
    {
        public ILaptop CreateLaptop() => new KiaomiLaptop();
        public INetbook CreateNetbook() => new KiaomiNetbook();
        public IEBook CreateEBook() => new KiaomiEBook();
        public ISmartphone CreateSmartphone() => new KiaomiSmartphone();
    }


    public class BalaxyLaptop : ILaptop { public string GetModel() => "Balaxy Laptop"; }
    public class BalaxyNetbook : INetbook { public string GetModel() => "Balaxy Netbook";
}
    public class BalaxyEBook : IEBook { public string GetModel() => "Balaxy EBook"; }
    public class BalaxySmartphone : ISmartphone { public string GetModel() => "Balaxy
Smartphone"; }

    public class BalaxyFactory : IDeviceFactory
    {
        public ILaptop CreateLaptop() => new BalaxyLaptop();
        public INetbook CreateNetbook() => new BalaxyNetbook();
        public IEBook CreateEBook() => new BalaxyEBook();
        public ISmartphone CreateSmartphone() => new BalaxySmartphone();
    }
      }
```

```
Фабрика IProne:
IProne Laptop
IProne Smartphone

Фабрика Kiaomi:
Kiaomi Netbook
Kiaomi EBook

Фабрика Balaxy:
Balaxy Smartphone
```

Рис.3Результат виконання програми:

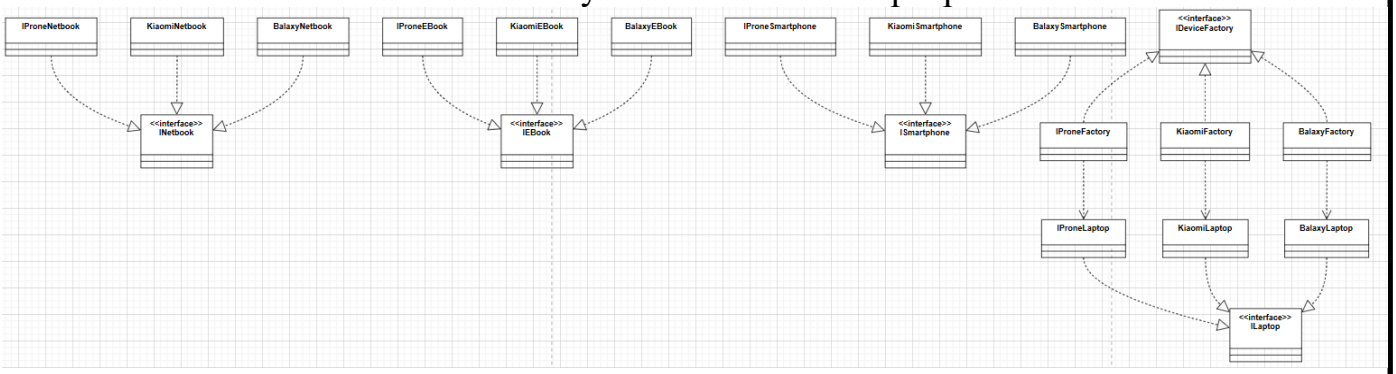

Рис.4.Діаграма

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;
using System.Text;

namespace SingletonTask
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.OutputEncoding = Encoding.UTF8;

            Console.WriteLine("Намагаємося отримати екземпляри:");

            Authenticator auth1 = Authenticator.Instance;
            Authenticator auth2 = Authenticator.Instance;


            if (Object.ReferenceEquals(auth1, auth2))
            {
                Console.WriteLine("Успіх! auth1 та auth2 — це один і той самий об'єкт.");
            }
```

```
            else
            {
                Console.WriteLine("Помилка! Це різні об'єкти.");
            }

            auth1.Login("Admin");
            auth2.Login("User");
        }
    }


    public sealed class Authenticator
    {

        private static readonly Lazy<Authenticator> _instance =
            new Lazy<Authenticator>(() => new Authenticator());


        private Authenticator()
        {
            Console.WriteLine("Створюється екземпляр Authenticator...");
        }


        public static Authenticator Instance => _instance.Value;

        public void Login(string user)
        {

            Console.WriteLine($"Користувач {user} увійшов в систему через екземпляр
{GetHashCode()}");
        }
    }
}
```

```
Намагаємося отримати екземпляри:
Створюється екземпляр Authenticator...
Успіх! auth1 та auth2 — це один і той самий об'єкт.
Користувач Admin увійшов в систему через екземпляр 35342034
Користувач User увійшов в систему через екземпляр 35342034
```

Рис.5.Результат виконання програми:

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;
using System.Text;

namespace PrototypeTask
```

```
{
    class Program
    {
        static void Main(string[] args)
        {

            Console.OutputEncoding = Encoding.UTF8;


            Virus grandparent = new Virus(10.5, 50, "Вірус-Дід", "Штам А");

            Virus parent = new Virus(5.2, 20, "Вірус-Батько", "Штам А");
            grandparent.Children.Add(parent);


            Virus child = new Virus(2.1, 5, "Вірус-Онук", "Штам А");
            parent.Children.Add(child);

            Console.WriteLine("Оригінальне сімейство");
            grandparent.PrintInfo(0);


            Virus clonedGrandparent = grandparent.Clone();

            Console.WriteLine("\nКлоноване сімейство");
            clonedGrandparent.PrintInfo(0);


            Console.WriteLine("\nПеревірка на незалежність");
            clonedGrandparent.Name = "КЛОН-Дід";
            clonedGrandparent.Children[0].Name = "КЛОН-Батько";

            Console.WriteLine($"Оригінал (дід): {grandparent.Name}, Син: {grandpar-
ent.Children[0].Name}");
            Console.WriteLine($"Клон (дід): {clonedGrandparent.Name}, Син: {clonedGrand-
parent.Children[0].Name}");
        }
    }

    public class Virus
    {
        public double Weight { get; set; }
        public int Age { get; set; }
        public string Name { get; set; }
        public string Species { get; set; }
        public List<Virus> Children { get; set; }

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


        public Virus Clone()
        {

            Virus clone = new Virus(this.Weight, this.Age, this.Name, this.Species);


            foreach (var child in this.Children)
            {
```

```
            clone.Children.Add(child.Clone());
        }

        return clone;
    }


    public void PrintInfo(int indent)
    {
        string spaces = new string(' ', indent * 4);
        Console.WriteLine($"{spaces}Вірус: {Name} (Вік: {Age}, Вага: {Weight})");
        foreach (var child in Children)
        {
            child.PrintInfo(indent + 1);
        }
    }
}
}
```

```
Консоль отладки Microsoft Vi
Оригінальне сімейство
Вірус: Вірус-Дід (Вік: 50, Вага: 10,5)
    Вірус: Вірус-Батько (Вік: 20, Вага: 5,2)
        Вірус: Вірус-Онук (Вік: 5, Вага: 2,1)

Клоноване сімейство
Вірус: Вірус-Дід (Вік: 50, Вага: 10,5)
    Вірус: Вірус-Батько (Вік: 20, Вага: 5,2)
        Вірус: Вірус-Онук (Вік: 5, Вага: 2,1)

Перевірка на незалежність
Оригінал (дід): Вірус-Дід, Син: Вірус-Батько
Клон (дід): КЛОН-Дід, Син: КЛОН-Батько
```

Рис.6.Результат виконання завдання:

Лістинг програми:

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

namespace BuilderTask
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.OutputEncoding = Encoding.UTF8;

            Director director = new Director();


            HeroBuilder heroBuilder = new HeroBuilder("Геральт");
            director.ConstructHero(heroBuilder);


            heroBuilder.AddInventory("Медальйон Вовка");

            Character hero = heroBuilder.Build();
            Console.WriteLine(" ГЕРОЙ");
            hero.ShowDetails();
```

```
            EnemyBuilder enemyBuilder = new EnemyBuilder("Ередін");
            director.ConstructEnemy(enemyBuilder);

            Character enemy = enemyBuilder.Build();
            Console.WriteLine("\n ВОРОГ");
            enemy.ShowDetails();
        }
    }


    public class Character
    {
        public string Name { get; set; }
        public string Height { get; set; }
        public string Stature { get; set; }
        public string HairColor { get; set; }
        public string EyeColor { get; set; }
        public string Outfit { get; set; }
        public List<string> Inventory { get; set; } = new List<string>();
        public List<string> Deeds { get; set; } = new List<string>();

        public void ShowDetails()
        {
            Console.WriteLine($"Персонаж: {Name}");
            Console.WriteLine($"Зріст: {Height}, Статура: {Stature}");
            Console.WriteLine($"Зовнішність: {HairColor} волосся, {EyeColor} очі, Одяг:
{Outfit}");
            Console.WriteLine($"Інвентар: {string.Join(", ", Inventory)}");
            Console.WriteLine($"Дії: {string.Join(", ", Deeds)}");
        }
    }


    public interface ICharacterBuilder
    {
        ICharacterBuilder SetHeight(string h);
        ICharacterBuilder SetStature(string s);
        ICharacterBuilder SetHairColor(string h);
        ICharacterBuilder SetEyeColor(string e);
        ICharacterBuilder SetOutfit(string outfit);
        ICharacterBuilder AddInventory(string item);
        Character Build();
    }
    public class HeroBuilder : ICharacterBuilder
    {
        private Character _character;
        public HeroBuilder(string name) { _character = new Character { Name = name }; }

        public HeroBuilder SetHeight(string h) { _character.Height = h; return this; }
        public HeroBuilder SetStature(string s) { _character.Stature = s; return this; }
        public HeroBuilder SetHairColor(string h) { _character.HairColor = h; return
this; }
        public HeroBuilder SetEyeColor(string e) { _character.EyeColor = e; return this;
}
        public HeroBuilder SetOutfit(string o) { _character.Outfit = o; return this; }
        public HeroBuilder AddInventory(string i) { _character.Inventory.Add(i); return
this; }
        public HeroBuilder AddGoodDeed(string deed) { _character.Deeds.Add("Добро: " +
deed); return this; }

        public Character Build() => _character;

        ICharacterBuilder ICharacterBuilder.SetHeight(string h) => SetHeight(h);
        ICharacterBuilder ICharacterBuilder.SetStature(string s) => SetStature(s);
```

```
        ICharacterBuilder ICharacterBuilder.SetHairColor(string h) => SetHairColor(h);
        ICharacterBuilder ICharacterBuilder.SetEyeColor(string e) => SetEyeColor(e);
        ICharacterBuilder ICharacterBuilder.SetOutfit(string o) => SetOutfit(o);
        ICharacterBuilder ICharacterBuilder.AddInventory(string i) => AddInventory(i);
    }

    public class EnemyBuilder : ICharacterBuilder
    {
        private Character _character;
        public EnemyBuilder(string name) { _character = new Character { Name = name }; }

        public EnemyBuilder SetHeight(string h) { _character.Height = h; return this; }
        public EnemyBuilder SetStature(string s) { _character.Stature = s; return this; }
        public EnemyBuilder SetHairColor(string h) { _character.HairColor = h; return
this; }
        public EnemyBuilder SetEyeColor(string e) { _character.EyeColor = e; return this;
}
        public EnemyBuilder SetOutfit(string o) { _character.Outfit = o; return this; }
        public EnemyBuilder AddInventory(string i) { _character.Inventory.Add(i); return
this; }
        public EnemyBuilder AddEvilDeed(string deed) { _character.Deeds.Add("Зло: " +
deed); return this; }

        public Character Build() => _character;

        ICharacterBuilder ICharacterBuilder.SetHeight(string h) => SetHeight(h);
        ICharacterBuilder ICharacterBuilder.SetStature(string s) => SetStature(s);
        ICharacterBuilder ICharacterBuilder.SetHairColor(string h) => SetHairColor(h);
        ICharacterBuilder ICharacterBuilder.SetEyeColor(string e) => SetEyeColor(e);
        ICharacterBuilder ICharacterBuilder.SetOutfit(string o) => SetOutfit(o);
        ICharacterBuilder ICharacterBuilder.AddInventory(string i) => AddInventory(i);
    }

    public class Director
    {
        public void ConstructHero(HeroBuilder builder)
        {
            builder.SetHeight("185см")
                   .SetStature("Атлетична")
                   .SetHairColor("Біле")
                   .SetEyeColor("Жовті")
                   .SetOutfit("Шкіряна броня")
                   .AddInventory("Срібний меч")
                   .AddGoodDeed("Врятував світ");
        }

        public void ConstructEnemy(EnemyBuilder builder)
        {
            builder.SetHeight("210см")
                   .SetStature("Могутня")
                   .SetHairColor("Відсутнє")
                   .SetEyeColor("Червоні")
                   .SetOutfit("Темні лати")
                   .AddInventory("Морозний меч")
                   .AddEvilDeed("Викрав Цірі");
        }
    }
}
```

```
 ГЕРОЙ
Персонаж: Геральт
Зріст: 185см, Статура: Атлетична
Зовнішність: Біле волосся, Жовті очі, Одяг: Шкіряна броня
Інвентар: Срібний меч, Медальйон Вовка
Дії: Добро: Врятував світ

 ВОРОГ
Персонаж: Ередін
Зріст: 210см, Статура: Могутня
Зовнішність: Відсутнє волосся, Червоні очі, Одяг: Темні лати
Інвентар: Морозний меч
Дії: Зло: Викрав Цірі
```

Рис.7.Результат виконання програми:

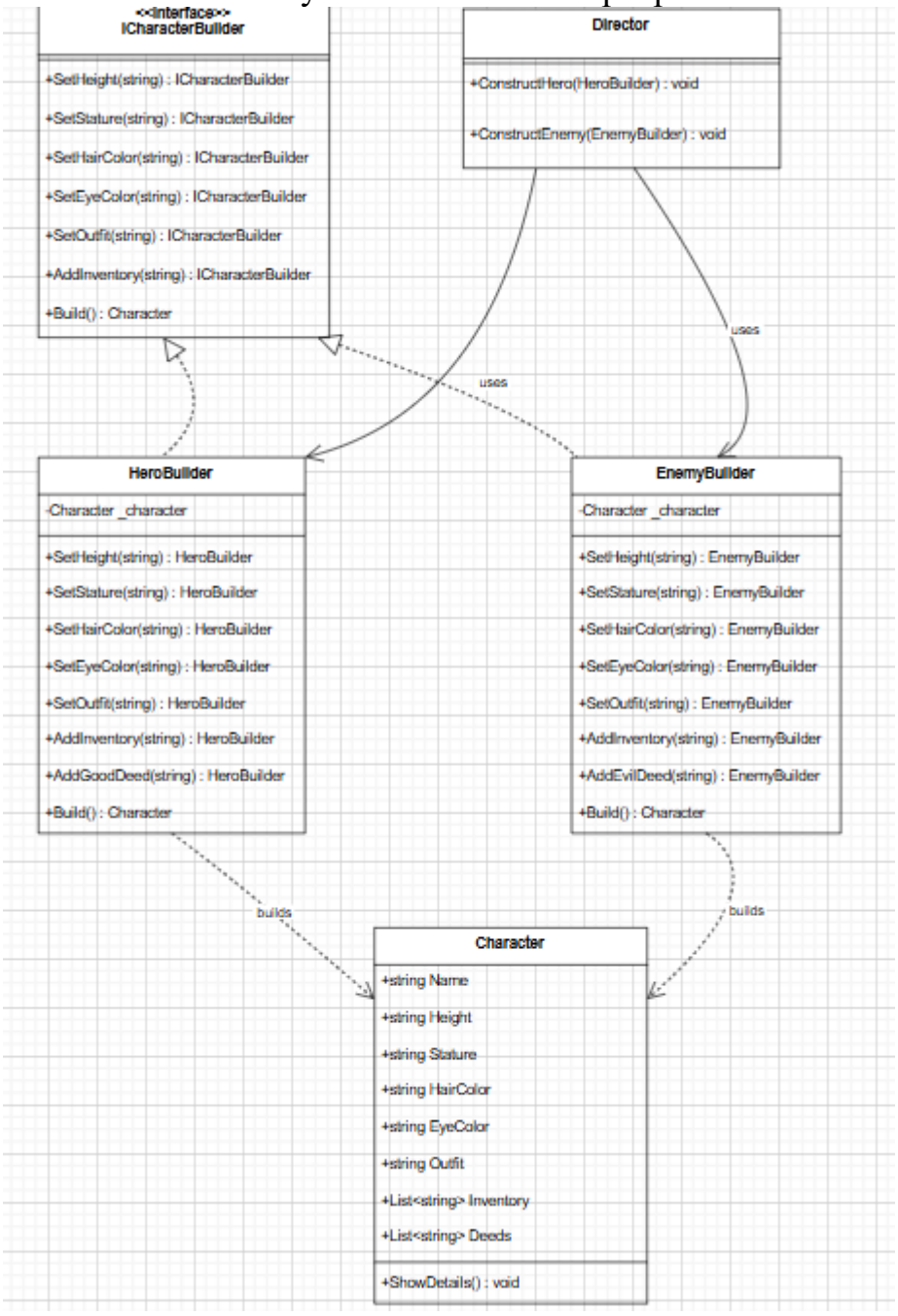

Рис.8.Діаграма