**ЛАБОРАТОРНА РОБОТА № 2**

**Породжувальні шаблони**

**Мета:** навчитися реалізовувати породжувальні шаблони проєктування.

**Хід роботи:**

Завдання

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:

FactoryMethod/ISubscription.cs

namespace FactoryMethod;

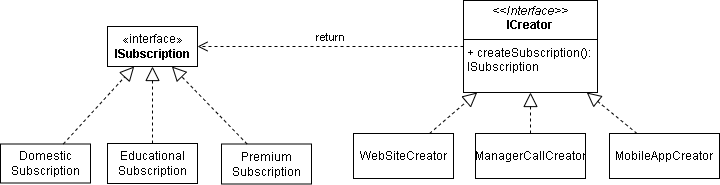
public interface ISubscription

{

List<string> Channels { get; }

TimeSpan TimeSpan { get; }

string Rent { get; }

string Name { get; }

string Description { get; }

}

FactoryMethod/DomesticSubscription.cs

namespace FactoryMethod;

public class DomesticSubscription : ISubscription

{

public List<string> Channels { get; set; } = ["ABC", "NBC", "TET"];

public TimeSpan TimeSpan { get; set; } = TimeSpan.FromDays(28);

public string Rent { get; set; } = "5 USD";

public string Name { get; set; } = "DomesticSubscription";

public string Description { get; set; } = "Basic Subscription for Family";

}

FactoryMethod/EducationalSubscription.cs

namespace FactoryMethod;

public class EducationalSubscription : ISubscription

{

public List<string> Channels { get; set; } = ["ABC", "NBC"];

public TimeSpan TimeSpan { get; set; } = TimeSpan.FromDays(28);

public string Rent { get; set; } = "2 USD";

public string Name { get; set; } = "EducationalSubscription";

public string Description { get; set; } = "Subscription for Students only";

}

FactoryMethod/PremiumSubscription.cs

namespace FactoryMethod;

public class PremiumSubscription : ISubscription

{

public List<string> Channels { get; set; } = ["ABC", "NBC", "TET", "Inter", "1+1"];

public TimeSpan TimeSpan { get; set; } = TimeSpan.FromDays(60);

public string Rent { get; set; } = "11 USD";

public string Name { get; set; } = "Premium Subscription";

public string Description { get; set; } = "Premium Subscription";

}

FactoryMethod/ICreator.cs

namespace FactoryMethod;

public interface ICreator

{

ISubscription CreateSubscription();

}

FactoryMethod/ManagerCallCreator.cs

namespace FactoryMethod;

public class ManagerCallCreator : ICreator

{

public ISubscription CreateSubscription()

{

var subscription = new DomesticSubscription();

subscription.Description += "\nCreated using Manager Call\n";

return subscription;

}

}

FactoryMethod/MobileAppCreator.cs

namespace FactoryMethod;

public class MobileAppCreator : ICreator

{

public ISubscription CreateSubscription()

{

var subscription = new EducationalSubscription();

subscription.Description += "\nCreated using Mobile App\n";

return subscription;

}

}

FactoryMethod/WebSiteCreator.cs

namespace FactoryMethod;

public class WebSiteCreator : ICreator

{

public ISubscription CreateSubscription()

{

var subscription = new PremiumSubscription();

subscription.Description += "\nCreated using Web Site\n";

return subscription;

}

}

FactoryMethod/Program.cs

namespace FactoryMethod;

internal class Program

{

private static void Main(string[] args)

{

// Some code to choose Factory

ICreator creator = new WebSiteCreator();

ISubscription subscription = creator.CreateSubscription();

Console.WriteLine($"Name: {subscription.Name}, Rent: {subscription.Rent}, Description: {subscription.Description}");

}

}

Результат виконання програми:

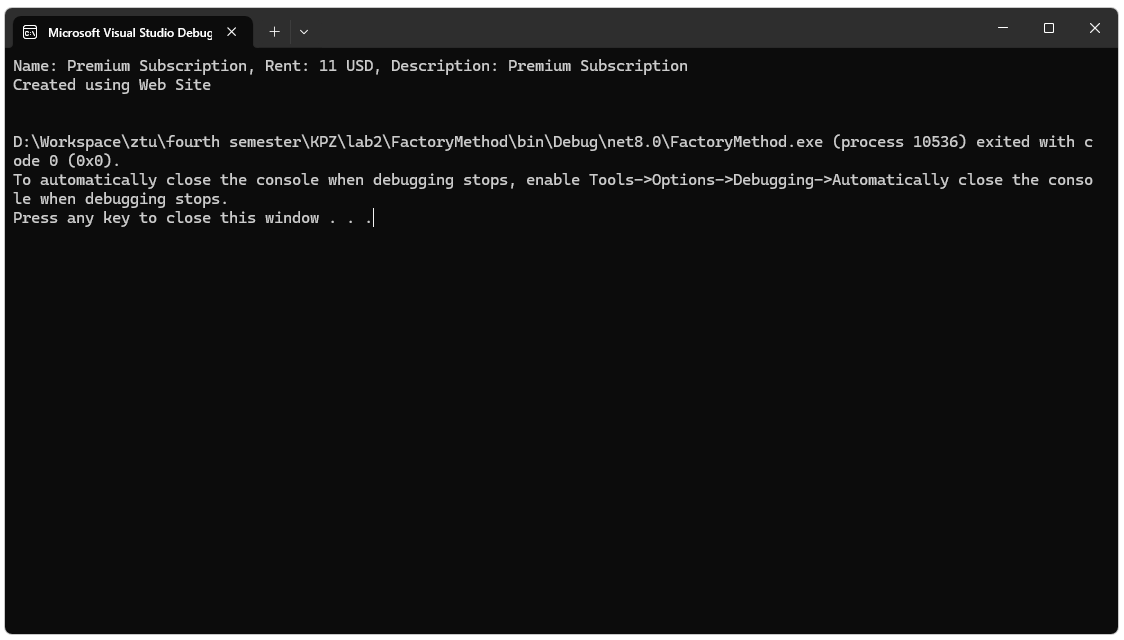


Рис 1 Результат виконання програми

Діаграма:

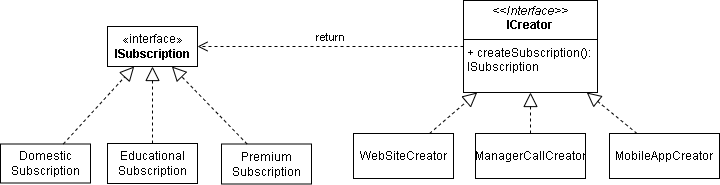


Рис 2 Діаграма класів

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:

AbstractFactory/Interfaces.cs

namespace AbstractFactory;

internal interface IGadgetCreator

{

ILaptop CreateLaptop();

IPhone CreatePhone();

}

internal interface IPhone

{

string Name { get; }

string Description { get; }

string Cost { get; }

string Battery { get; }

}

internal interface ILaptop

{

string Name { get; }

string Description { get; }

string Cost { get; }

string Screen { get; }

}

AbstractFactory/ConcreteProducts.cs

namespace AbstractFactory;

// Xiaomi

internal class XiaomiPhone : IPhone

{

public string Name { get; set; } = "Xiaomi Phone";

public string Description { get; set; } = "Xiaomi Phone Description";

public string Cost { get; set; } = "250 USD";

public string Battery { get; set; } = "4000 mah";

}

internal class XiaomiLaptop : ILaptop

{

public string Name { get; set; } = "Xiaomi Laptop";

public string Description { get; set; } = "Xiaomi Laptop Description";

public string Cost { get; set; } = "1600 USD";

public string Screen { get; set; } = "15.6";

}

// Galaxy

internal class GalaxyPhone : IPhone

{

public string Name { get; set; } = "Galaxy Phone";

public string Description { get; set; } = "Galaxy Phone Description";

public string Cost { get; set; } = "450 USD";

public string Battery { get; set; } = "5000 mah";

}

internal class GalaxyLaptop : ILaptop

{

public string Name { get; set; } = "Galaxy Laptop";

public string Description { get; set; } = "Galaxy Laptop Description";

public string Cost { get; set; } = "2100 USD";

public string Screen { get; set; } = "16.6";

}

AbstractFactory/ConcreteFactories.cs

namespace AbstractFactory;

// Here listed concreate factories, which in our example, are GadgetCreators

internal class XiaomiGadgetCreator : IGadgetCreator

{

public ILaptop CreateLaptop() => new XiaomiLaptop();

public IPhone CreatePhone() => new XiaomiPhone();

}

internal class GalaxyGadgetCreator : IGadgetCreator

{

public ILaptop CreateLaptop() => new GalaxyLaptop();

public IPhone CreatePhone() => new GalaxyPhone();

}

AbstractFactory/Program.cs

namespace AbstractFactory;

internal class Program

{

private static void Main(string[] args)

{

// Here listed basic usage of Abstract Factory Pattern

/\*

My analogy:

Interfaces: IWindow, IButton, IPanel, ILayoutCreator

ConcreteProducts: WinWindow, WinButton, WinPanel, LinuxWindow, LinuxButton, LinuxPanel

ConcreteFactories: WinLayoutCreator, LinuxLayoutCreator

In this example good solution is also use Singleton Pattern for Factory Choose and usage

// there is also Singleton class - LayoutCreator, that has Configure method that chooses which factory to use and also has GetFactory method.

LayoutCreator.Configure();

ILayoutCreator layoutCreator = LayoutCreator.GetLayoutCreator();

IPanel settingsPanel = layoutCreator.CreatePanel();

settingsPanel.Width = 800;

settingsPanel.Height = 600;

// ...

\*/

IGadgetCreator gadgetCreator = new GalaxyGadgetCreator();

ILaptop laptop = gadgetCreator.CreateLaptop();

IPhone phone = gadgetCreator.CreatePhone();

Console.WriteLine($"{phone.Name}, {phone.Cost}, {phone.Description}, Battery {phone.Battery}");

Console.WriteLine($"{laptop.Name}, {laptop.Cost}, {laptop.Description}, Battery {laptop.Screen}");

}

}

Результат виконання програми:

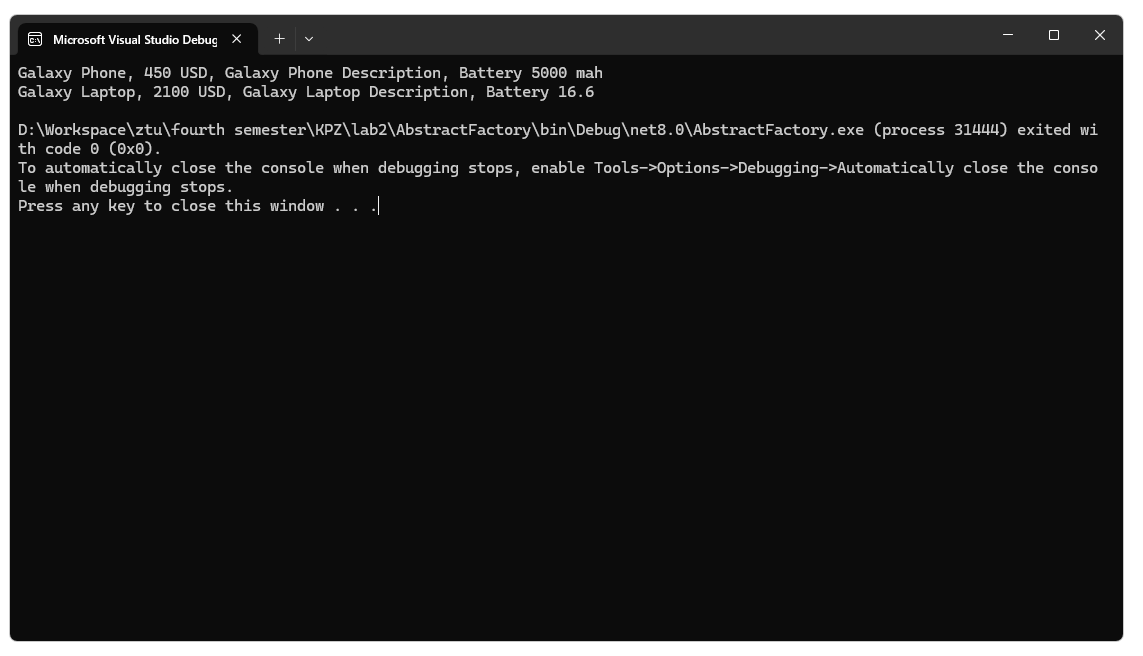


Рис 3 Результат виконання програми

Діаграма:

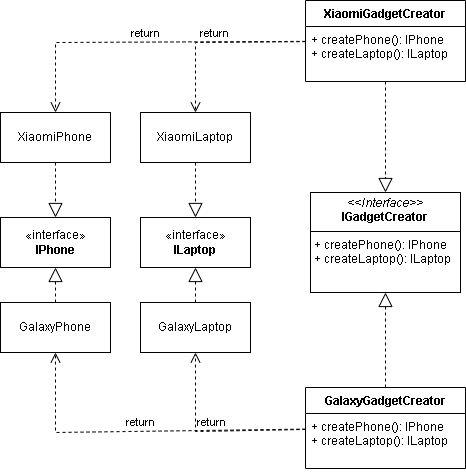


Рис 4 Діаграма класів

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:

Singleton/Authentificator.cs

namespace Singleton;

public class Authenticator

{

private Authenticator()

{

Console.WriteLine("Authenticator initialization!");

}

public void Auth(string username, string password)

{

Console.WriteLine($"Username: {username}, Password: {password}");

}

private static object \_lock = new();

private static Authenticator? instance;

public static Authenticator Instance

{

get

{

lock (\_lock)

{

if (instance == null)

instance = new Authenticator();

return instance;

}

}

}

}

Singleton/Program.cs

namespace Singleton;

internal class Program

{

private async static Task Main(string[] args)

{

List<Task> tasks = [];

for (int i = 0; i < 10; i++)

{

tasks.Add(Task.Run(() =>

{

Authenticator authenticator = Authenticator.Instance;

authenticator.Auth($"username", $"password");

}));

}

await Task.WhenAll(tasks);

}

}

Результат виконання програми:

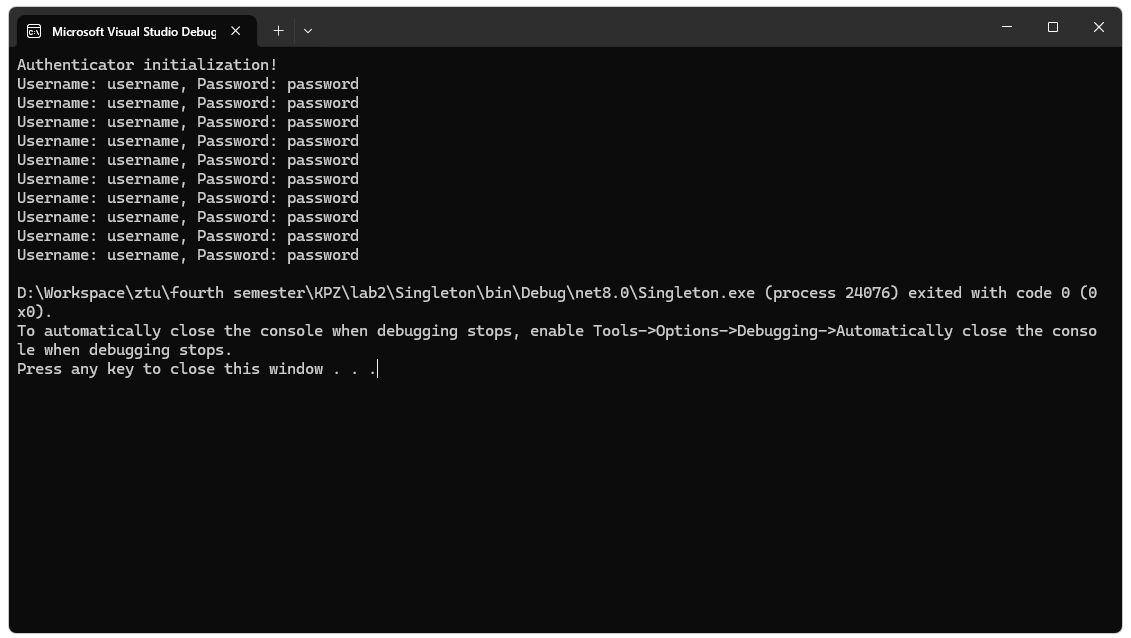


Рис 5 Результат виконання програми

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:

PrototypeTask/Virus.cs

namespace PrototypeTask;

internal class Virus : ICloneable

{

public string Name { get; set; }

public string Type { get; set; }

public decimal Weight { get; set; }

public TimeSpan Age { get; set; }

public List<Virus> Childs { get; set; }

public Virus(string name, string type, decimal weight, TimeSpan age, List<Virus> childs)

{

Name = name;

Type = type;

Weight = weight;

Age = age;

Childs = childs;

}

public void PrintInfo()

{

Console.WriteLine($"Name: {Name}, Type: {Type}, Weight: {Weight}, Age: {Age.Days} days");

foreach ( Virus virus in Childs )

virus.PrintInfo();

}

public Virus(Virus other)

{

Name = other.Name;

Type = other.Type;

Weight = other.Weight;

Age = other.Age;

Childs = [];

foreach(var child in other.Childs)

{

Childs.Add(child.Clone() as Virus);

}

}

public object Clone() => new Virus(this);

}

PrototypeTask/Program.cs

namespace PrototypeTask;

internal class Program

{

private static void Main(string[] args)

{

Virus generation1Virus = new Virus("Virus Mk1", "Mk1 type", 0.5M, TimeSpan.FromDays(30), new List<Virus>());

Virus generation2Virus = new Virus("Virus Mk2", "Mk2 type", 0.3M, TimeSpan.FromDays(20), new List<Virus>());

Virus generation3Virus = new Virus("Virus Mk3", "Mk3 type", 0.25M, TimeSpan.FromDays(10), new List<Virus>());

generation2Virus.Childs.Add(generation3Virus);

generation2Virus.Childs.Add(generation3Virus.Clone() as Virus);

generation1Virus.Childs.Add(generation2Virus);

generation1Virus.Childs.Add(generation2Virus.Clone() as Virus);

generation1Virus.PrintInfo();

Console.WriteLine();

Console.WriteLine();

Virus clone = generation1Virus.Clone() as Virus;

clone.PrintInfo();

}

}

Результат виконання програми:

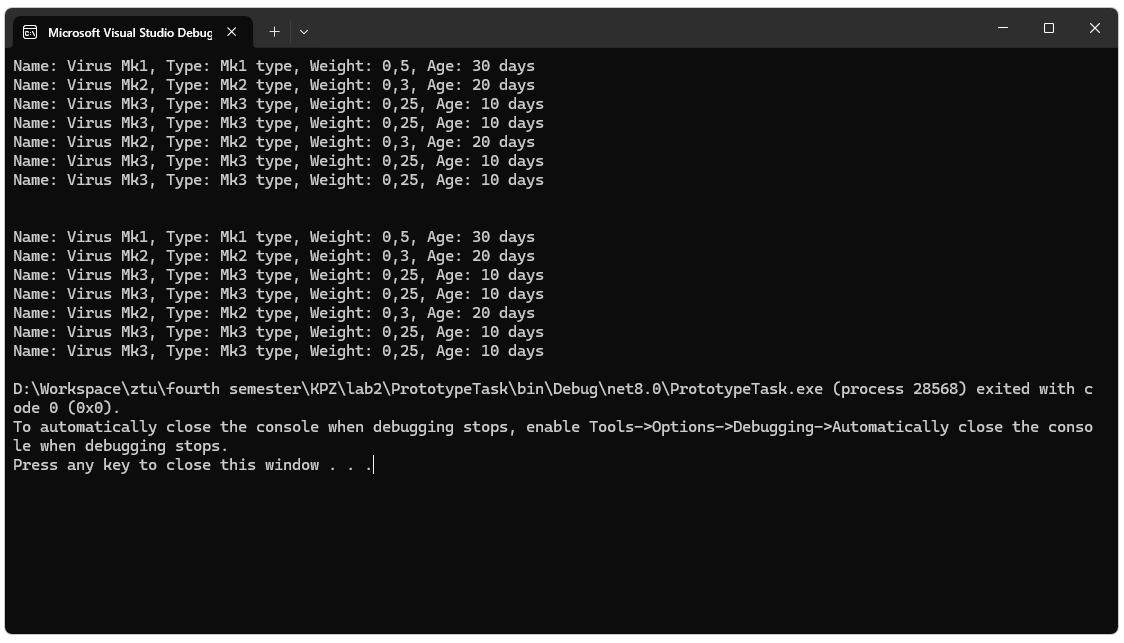


Рис 6 Результат виконання програми

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:

FactoryMethod/Program.cs

Результат виконання програми:

Рис Х Результат виконання програми

Діаграма:

Рис Х Діаграма класів

Посилання на репозиторій GitLab:

<https://github.com/ipz231-pmo/KPZ>

**Висновки:** В ході лабараторної роботи минавчилися реалізовувати породжувальні шаблони проєктування.