Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота № 6

із дисципліни «Технології розроблення програмного забезпечення»

Тема: «Патерни проектування»

Виконав
Студент групи ІА-31:
Підковка Д. О.

Перевірив:
Мягкий М. Ю.

Київ 2025

Зміст

1. **Мета:**

Вивчити структуру шаблонів «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator» та навчитися застосовувати їх в реалізації програмної системи.

2. **Теоретичні відомості**

Abstract Factory: Шаблон для створення сімейств пов’язаних об’єктів без вказівки їх конкретних класів. Використовується, коли потрібно забезпечити узгодженість об’єктів одного стилю або типу. Приклад: створення об’єктів різних стилів у грі (стіни, двері, меблі).

Factory Method: Визначає інтерфейс для створення об’єктів, дозволяючи підкласам вирішувати, який саме об’єкт створювати. Підходить для розширення системи новими типами без зміни існуючого коду.

Memento: (Знімок) Дозволяє зберігати і відновлювати стан об’єкта без порушення інкапсуляції. Стан зберігається в об’єкті-знімку, доступному лише вихідному об’єкту.

Observer: (Спостерігач) Визначає залежність «один-до-багатьох»: зміна стану одного об’єкта сповіщає всіх підписаних. Приклад: підписка на канал або повідомлення про зміну даних.

Decorator: (Декоратор) Дозволяє динамічно додавати об’єктам нову функціональність без зміни їхнього коду. Декоратор «обгортає» базовий об’єкт і розширює його поведінку.

3. **Хід роботи**

Тема :

6. **Web-browser** (proxy, chain of responsibility, factory method, template method, visitor, p2p)

Веб-браузер повинен мати можливість зробити наступне: мати адресний рядок для введення адреси сайту, переміщатися і відображати структуру html документа, переглядати підключений javascript та css файли, перегляд всіх підключених ресурсів (зображень), коректна обробка відповідей з сервера (коди відповідей HTTP) – переходи при перенаправленнях, відображення сторінок 404 і 502/503.

Патерн: Factory Method для створення контенту вкладок різного типу (Web-сторінка, Службова сторінка).
Чому Factory Method: Різні типи адрес (http://, browser://, view-source:) вимагають різних візуальних компонентів для відображення (WebView2 для інтернету, TextBox або StackPanel для внутрішніх сторінок). Фабрика інкапсулює логіку аналізу URL та створює правильний об'єкт, позбавляючи головне вікно від складних умовних конструкцій.

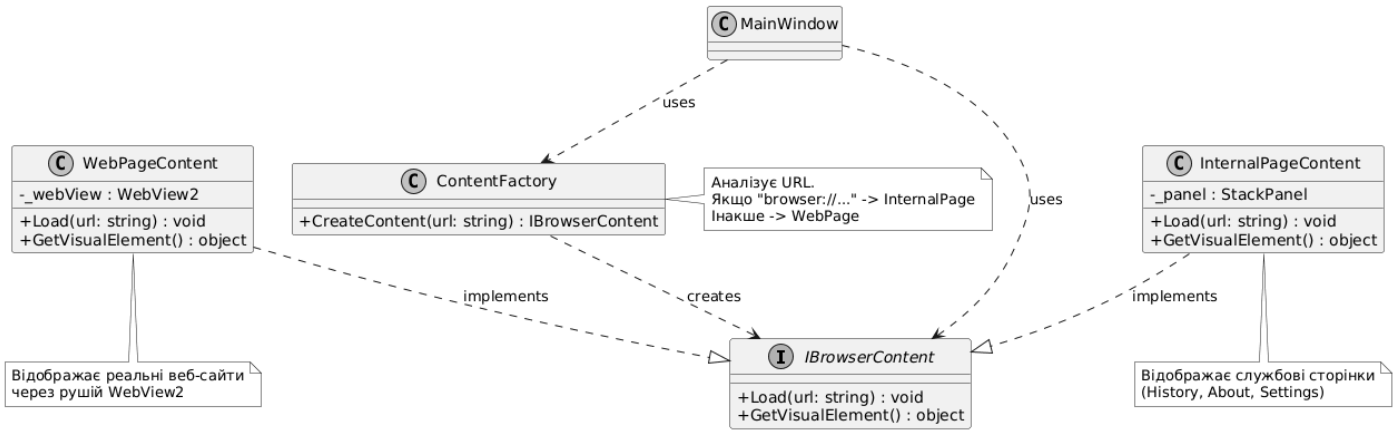


Рисунок 1 - Структура патерну Factory Method

Класи: IBrowserContent (інтерфейс), WebPageContent, InternalPageContent (конкретні продукти), ContentFactory (фабрика), MainWindow (клієнт).

IBrowserContent – це інтерфейс, який описує методи Load(url) та GetVisualElement(). Він задає контракт для будь-якого типу контенту, який може бути завантажений у вкладку браузера.

ContentFactory – це клас фабрики, який визначає метод CreateContent. Він містить логіку розпізнавання протоколу (http vs browser) та приймає рішення, який саме об'єкт створити.

WebPageContent – це конкретна реалізація контенту для звичайних сайтів. Вона ініціалізує та повертає важкий компонент WebView2 для рендерингу HTML.

InternalPageContent – це конкретна реалізація для службових цілей (довідка, історія, налаштування). Вона формує легковажний інтерфейс на базі стандартних контролів WPF (TextBlock, StackPanel).

MainWindow – це клієнт, який працює з фабрикою. Він не знає наперед, чи буде відкрито сайт, чи налаштування, а просто викликає метод CreateContent у фабрики та вбудовує отриманий візуальний елемент у вікно.

```
namespace WebBrowser.Coursework.Factories
{
    2 references
    public class ContentFactory
    {
        1 reference
        public IBrowserContent CreateContent(string url)
        {
            if (url.StartsWith("browser://"))
            {
                return new InternalPageContent();
            }
            else
            {
                return new WebPageContent();
            }
        }
    }
}
```

Рисунок 2 - ContentFactory.cs

```
using System.Windows;

namespace WebBrowser.Coursework.Factories
{
    4 references
    public interface IBrowserContent
    {
        3 references
        void Load(string url);
        4 references
        UIElement GetVisualElement();
    }
}
```

Рисунок 3 - IBrowserContent.cs

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;

namespace WebBrowser.Coursework.Factories
{
    2 references
    public class InternalPageContent : IBrowserContent
    {
        private readonly StackPanel _panel;
        private readonly TextBlock _textBlock;

        1 reference
        public InternalPageContent()
        {
            _panel = new StackPanel { Background = Brushes.WhiteSmoke, Margin = new Thickness(10) };
            _textBlock = new TextBlock { FontSize = 24, HorizontalAlignment = HorizontalAlignment.Center, Margin = new Thickness(0, 50, 0, 0) };
            _panel.Children.Add(_textBlock);
        }

        2 references
        public void Load(string url)
        {
            string command = url.Replace("browser://", "").ToLower();

            switch (command)
            {
                case "about":
                    _textBlock.Text = "Студентський Веб-Браузер v1.0\nКурсова робота";
                    _textBlock.Foreground = Brushes.DarkBlue;
                    break;
                case "help":
                    _textBlock.Text = "Довідка користувача:\n1. Введіть URL для переходу.\n2. Використовуйте browser://about для інформації.";
                    _textBlock.Foreground = Brushes.Green;
                    break;
                default:
                    _textBlock.Text = $"Невідома службова сторінка: {command}";
                    _textBlock.Foreground = Brushes.Red;
                    break;
            }
        }

        2 references
        public UIElement GetVisualElement()
        {
            return _panel;
        }
    }
}
```

Рисунок 4 - InternalPageContent.cs

```
using System;
using System.Windows;
using Microsoft.Web.WebView2.Wpf;

namespace WebBrowser.Coursework.Factories
{
    3 references
    public class WebPageContent : IBrowserContent
    {
        private readonly WebView2 _webView;

        1 reference
        public WebPageContent()
        {
            _webView = new WebView2();
        }

        2 references
        public async void Load(string url)
        {
            await _webView.EnsureCoreWebView2Async();

            if (Uri.TryCreate(url, UriKind.Absolute, out var uri))
            {
                _webView.Source = uri;
            }
        }

        3 references
        public UIElement GetVisualElement()
        {
            return _webView;
        }
    }
}
```

Рисунок 5 - WebPageContent.cs

```
private void NavigateToUrl()
{
    var url = AddressBar.Text;
    if (string.IsNullOrWhiteSpace(url)) return;

    if (!url.StartsWith("browser://") && !url.StartsWith("http"))
    {
        url = "https://" + url;
    }

    try
    {
        _currentContent = _contentFactory.CreateContent(url);

        _currentContent.Load(url);

        MainContentContainer.Content = _currentContent.GetVisualElement();

        AddressBar.Text = url;

        _historyManager.AddEntry("Visited Page", url);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Помилка фабрики: {ex.Message}");
    }
}
```

Рисунок 6 – Оновлений метод NavigateToUrl

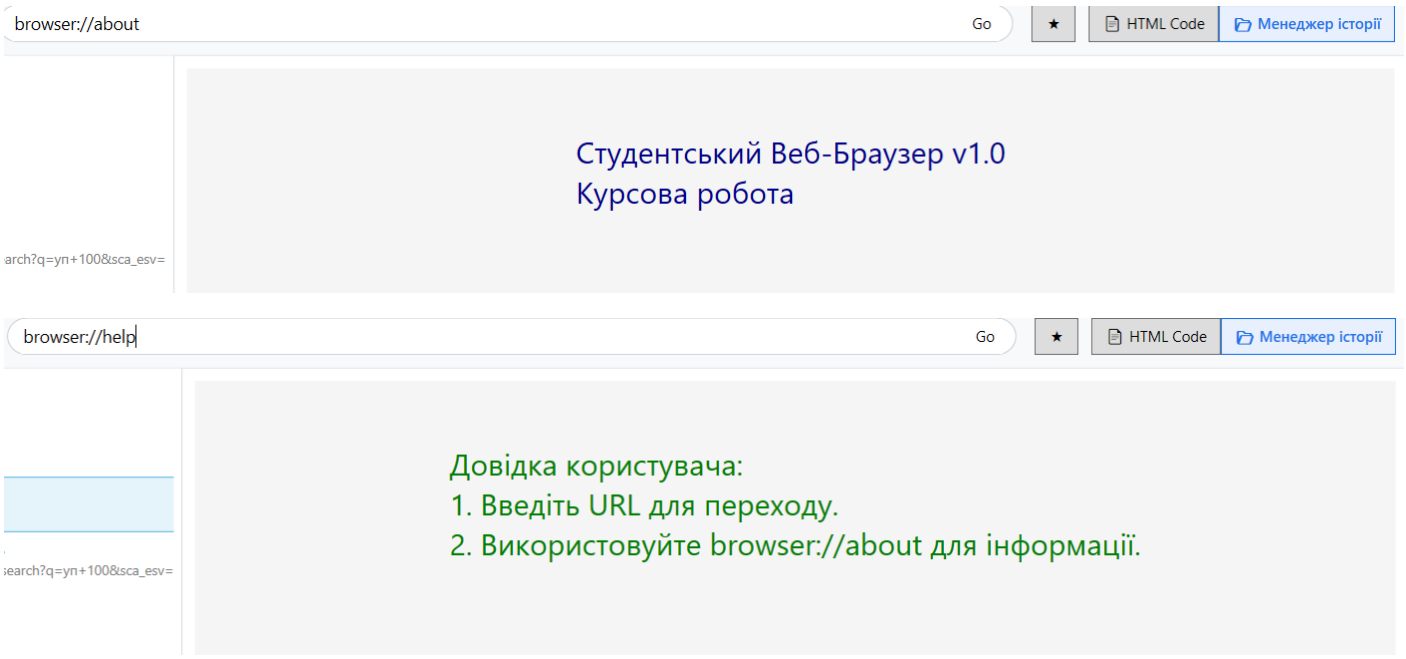


Рисунок 7 та 8 – Результат

4. Висновок

У ході виконання лабораторної роботи було розглянуто породжуючі та поведінкові шаблони проєктування, зокрема Abstract Factory, Factory Method, Memento, Observer та Decorator. Отримані знання допомогли зрозуміти їхню роль у створенні гнучкої, масштабованої та підтримувальної архітектури програмних систем.

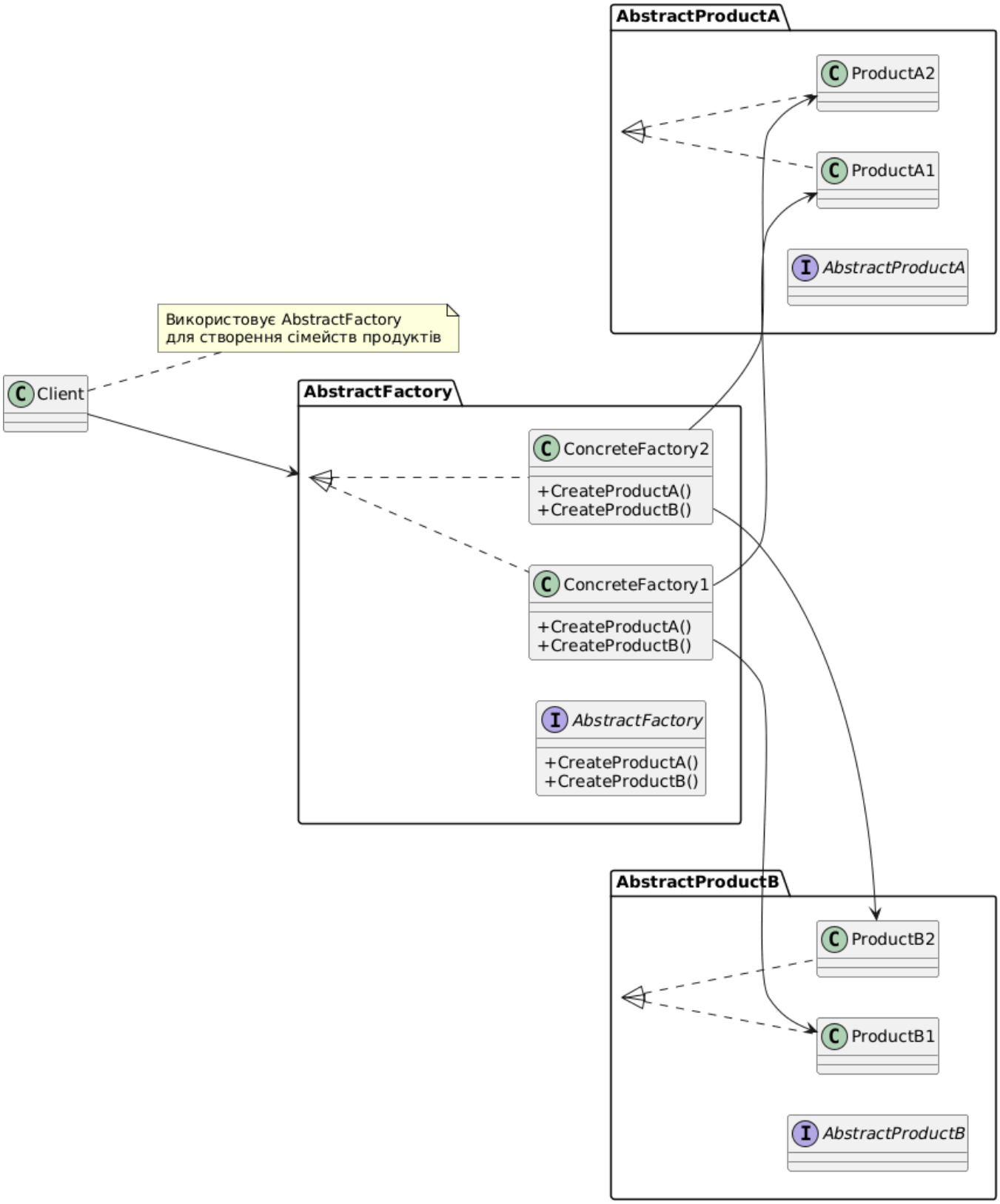
На прикладі настільного застосунку «Web-browser» було реалізовано шаблон Factory Method для організації створення контенту вкладок різного типу (звичайні веб-сторінки та службові сторінки). Логіка створення винесена в окремий клас-фабрику (ContentFactory), що дозволило відокремити процес інстанціювання компонентів відображення від клієнтського коду (MainWindow).

Такий підхід спрощує керування типами контенту, робить код головного вікна чистішим і дає змогу легко розширювати функціонал — наприклад, додати підтримку нових протоколів (ftp://, file://) або режимів перегляду (view-source) — без зміни основної логіки навігації.

5. Контрольні питання

1. Яке призначення шаблону «Абстрактна фабрика» Шаблон «Абстрактна фабрика» використовується для створення сімейств об'єктів без вказівки їх конкретних класів.

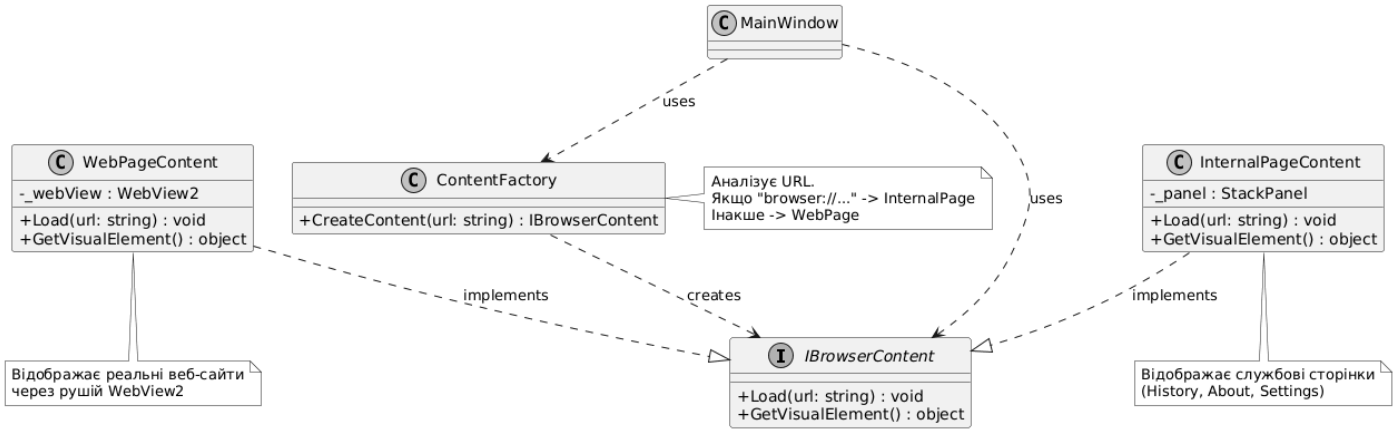
2. Нарисуйте структуру шаблону «Абстрактна фабрика»



3. Які класи входять в шаблон «Абстрактна фабрика», та яка між ними взаємодія? Класи: AbstractFactory, ConcreteFactory, AbstractProductA/B, ConcreteProductA/B, Client. Взаємодія: Client використовує AbstractFactory для створення об'єктів через інтерфейси продуктів; ConcreteFactory реалізує методи створення конкретних продуктів одного сімейства.

4. Яке призначення шаблону «Фабричний метод»? Шаблон «Фабричний метод» визначає інтерфейс для створення об'єктів певного базового типу, залишаючи реалізацію підтипам.

5. Нарисуйте структуру шаблону «Фабричний метод»

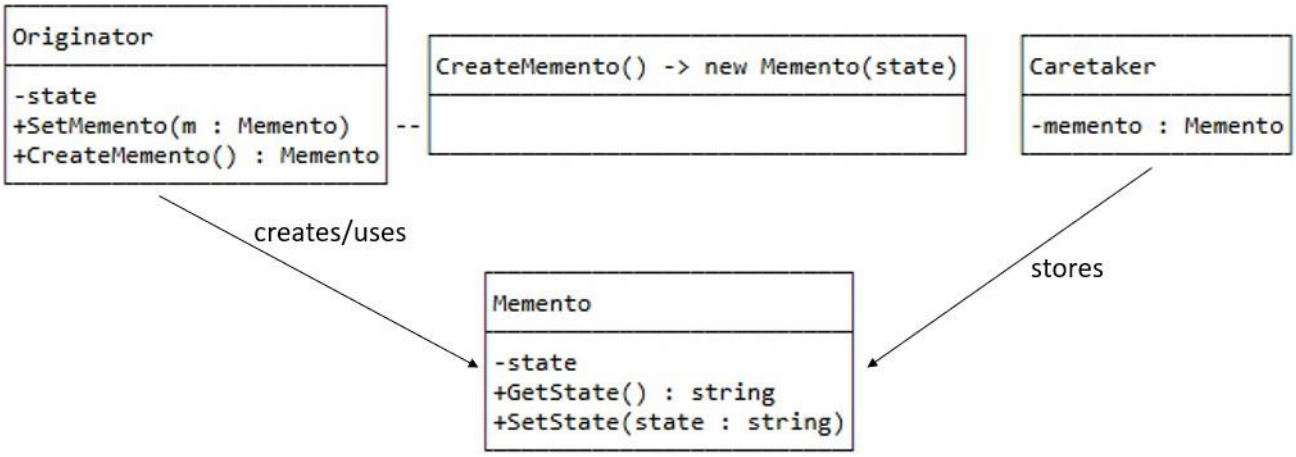


6. Які класи входять в шаблон «Фабричний метод», та яка між ними взаємодія? Класи: Creator, ConcreteCreator, Product, ConcreteProduct. Взаємодія: Creator викликає FactoryMethod() для створення об'єкта; ConcreteCreator повертає конкретний ConcreteProduct.

7. Чим відрізняється шаблон «Абстрактна фабрика» від «Фабричний метод»? «Абстрактна фабрика» створює сімейства пов’язаних об'єктів через набір методів; «Фабричний метод» створює один об'єкт через один віртуальний метод, реалізований у підкласах.

8. Яке призначення шаблону «Знімок»? Шаблон «Знімок» використовується для збереження і відновлення стану об'єктів без порушення інкапсуляції.

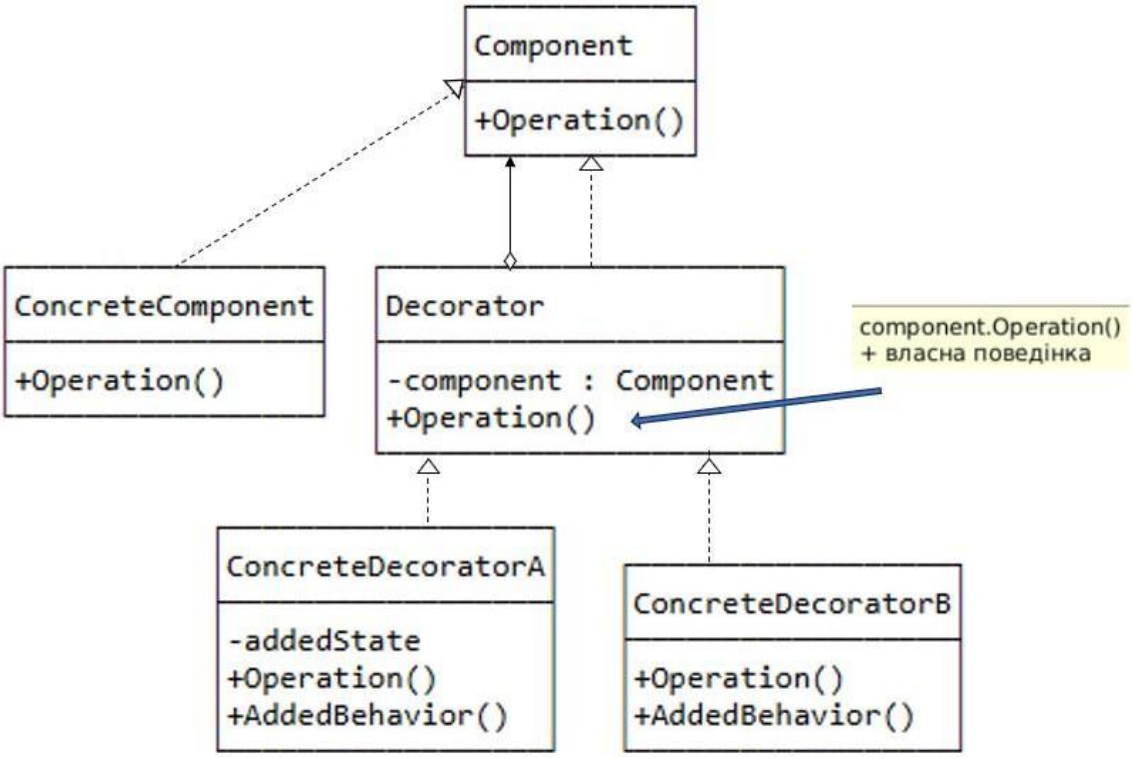
9. Нарисуйте структуру шаблону «Знімок»



10. Які класи входять в шаблон «Знімок», та яка між ними взаємодія? Класи: Originator, Memento, Caretaker. Взаємодія: Originator створює/відновлює Memento; Caretaker зберігає Memento, не маючи доступу до його вмісту.

11. Яке призначення шаблону «Декоратор»? Шаблон «Декоратор» призначений для динамічного додавання функціональних можливостей об'єкту під час роботи програми.

12. Нарисуйте структуру шаблону «Декоратор»



13. Які класи входять в шаблон «Декоратор», та яка між ними взаємодія? Класи: Component, ConcreteComponent, Decorator, ConcreteDecorator. Взаємодія: Decorator обгортає Component, викликає його Operation() і додає власну поведінку.

14. Які є обмеження використання шаблону «Декоратор»? Велика кількість крихітних класів; важко конфігурувати об’єкти, загорнуті в декілька обгорток одночасно.