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: «Вступ до паттернів проектування»

Виконав
Студент групи ІА-31:
Олея М. С.

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. Завдання:

1) Ознайомитись з короткими теоретичними відомостями.
2) Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
3) Реалізувати один з розглянутих шаблонів за обраною темою.
4) Реалізувати не менше 3-х класів відповідно до обраної теми.
5) Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму класів, яка представляє використання шаблону в реалізації системи, навести фрагменти коду по реалізації цього шаблону.

4. Хід роботи:

Варіант - 8

Powershell terminal (strategy, command, abstract factory, bridge, interpreter, client-server)

Термінал для powershell повинен нагадувати типовий термінал з можливістю налаштування кольорів синтаксичних конструкцій, розміру вікна, фону вікна, а також виконання команд powershell і виконуваних файлів, а також працювати в декількох вікнах терміналу (у вкладках або одночасно шляхом розділення вікна)

Призначення та доцільність:

У розробленій системі "PowerShell Terminal" існує необхідність створення інтерфейсу користувача, який чітко відповідає його ролі (Адміністратор або Користувач). Кожна роль визначає набір візуальних та текстових елементів, які мають бути узгодженими між собою:

- Адміністратор: Має бачити спеціальний заголовок вікна ("ADMIN TERMINAL") та специфічний промпт (PS ADMIN #), що підкреслює підвищені права.
- Користувач: Має бачити стандартний заголовок та звичайний промпт (PS User $).

Використання простих умов (if-else) для налаштування кожного елемента окремо може призвести до помилок (наприклад, заголовок адміна, але промпт користувача). Паттерн Abstract Factory (Абстрактна фабрика) є доцільним, оскільки він:

- Гарантує цілісність сімейства об'єктів: Фабрика адміністратора завжди створює комплект "Заголовок Адміна" + "Промпт Адміна". Змішування елементів різних ролей технічно неможливе.
- Ізолює код створення: Головна форма (TerminalForm) не знає, які саме класи вона використовує. Вона працює лише з абстрактними інтерфейсами IPrompt та IHeader.
- Спрощує розширення: Додавання нової ролі (наприклад, "Guest") потребуватиме лише створення нової фабрики, без зміни коду самої форми.

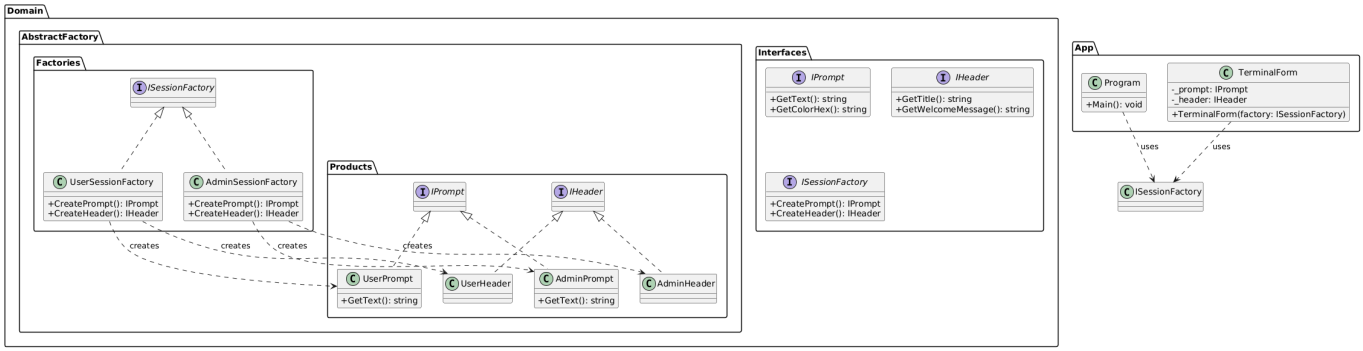


Рисунок 1 - Структура патерну Abstract Factory

Діаграма (зображена на рисунку 1) відображає структуру патерну, де виділяються чотири групи учасників:

1. Abstract Factory (ISessionFactory): Оголошує інтерфейс для створення абстрактних продуктів (CreatePrompt, CreateHeader). Це єдиний "контракт", який бачить клієнт (термінал).
2. Concrete Factories (AdminSessionFactory, UserSessionFactory): Реалізують методи фабрики.
    - AdminSessionFactory створює об'єкти AdminPrompt (з символом #) та AdminHeader (з написом "ADMIN TERMINAL").
    - UserSessionFactory створює об'єкти UserPrompt (з символом $) та UserHeader.
3. Abstract & Concrete Products (IPrompt, IHeader): Інтерфейси та їх реалізації для різних ролей. Вони містять конкретну логіку (текст заголовка, колір промпта).
4. Client (TerminalForm): Отримує посилання на ISessionFactory через конструктор і використовує його для побудови інтерфейсу. Він не перевіряє роль користувача, а просто довіряє фабриці.

Програмна реалізація:

а) Інтерфейс Абстрактної Фабрики ISessionFactory

```
public interface ISessionFactory
{
    IPrompt CreatePrompt();
    IHeader CreateHeader();
}
```

б) Реалізація продуктів Адміністратора (AdminProducts.cs)

```
public class AdminPrompt : IPrompt
    {
        public string GetText() => "PS ADMINISTRATOR # > ";
        public string GetColorHex() => "#FF0000";
    }

    public class AdminHeader : IHeader
```

```
    {
        public string GetTitle() => "ADMINISTRATOR: PowerShell Terminal";
        public string GetWelcomeMessage() => "*** WARNING: YOU HAVE ELEVATED PRIVILEGES
***";
    }
```

в) Фабрика Адміністратора (AdminSessionFactory.cs)

```
public class AdminSessionFactory : ISessionFactory
    {
        public IPrompt CreatePrompt()
        {
            return new AdminPrompt();
        }

        public IHeader CreateHeader()
        {
            return new AdminHeader();
        }
    }
```

г) Логіка вибору фабрики (Program.cs): Тут ми використовуємо нове поле Role з бази даних.

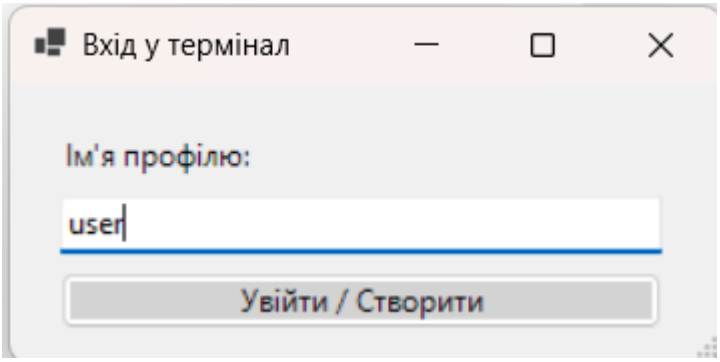
```
if (loginForm.ShowDialog() == DialogResult.OK && loginForm.LoggedInUser != null)
            {
                var user = loginForm.LoggedInUser;
                ISessionFactory factory;

                if (user.Role == "Admin")
                {
                    factory = new AdminSessionFactory();
                }
                else
                {
                    factory = new UserSessionFactory();
                }

                TerminalForm terminal = new TerminalForm(factory, user);
                terminal.ShowDialog();
            }
```

	ProfileId	ProfileName	Role	ThemeId	CreatedAt
	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	admin	Admin	1	0001-01-01 00:00:00
2	2	user	User	2	0001-01-01 00:00:00

Рисунок 2 – Users в бд SQLite



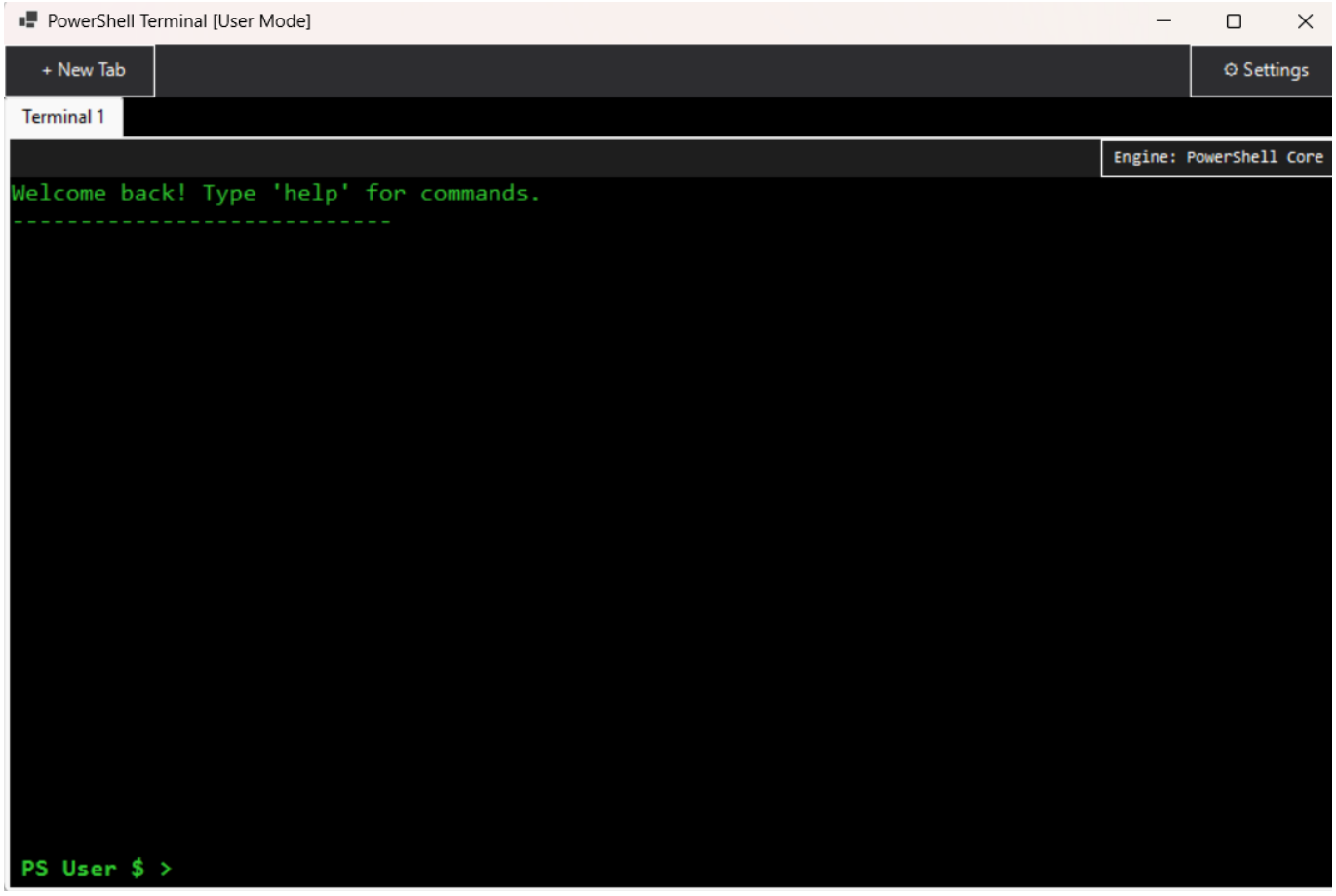


Рисунок 3 – авторизація від імені юзера

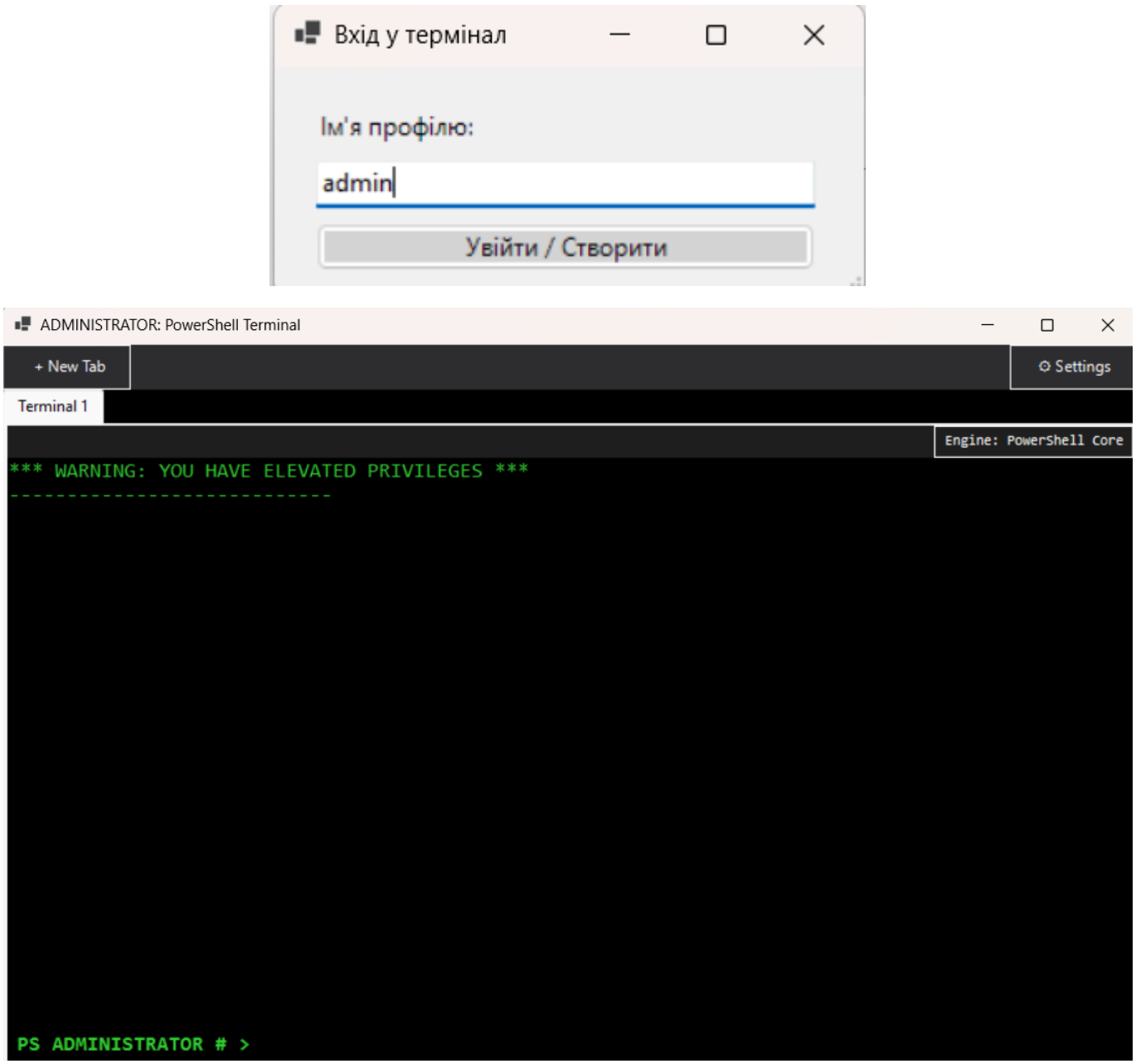


Рисунок 4 – авторизація від імені адміністратора

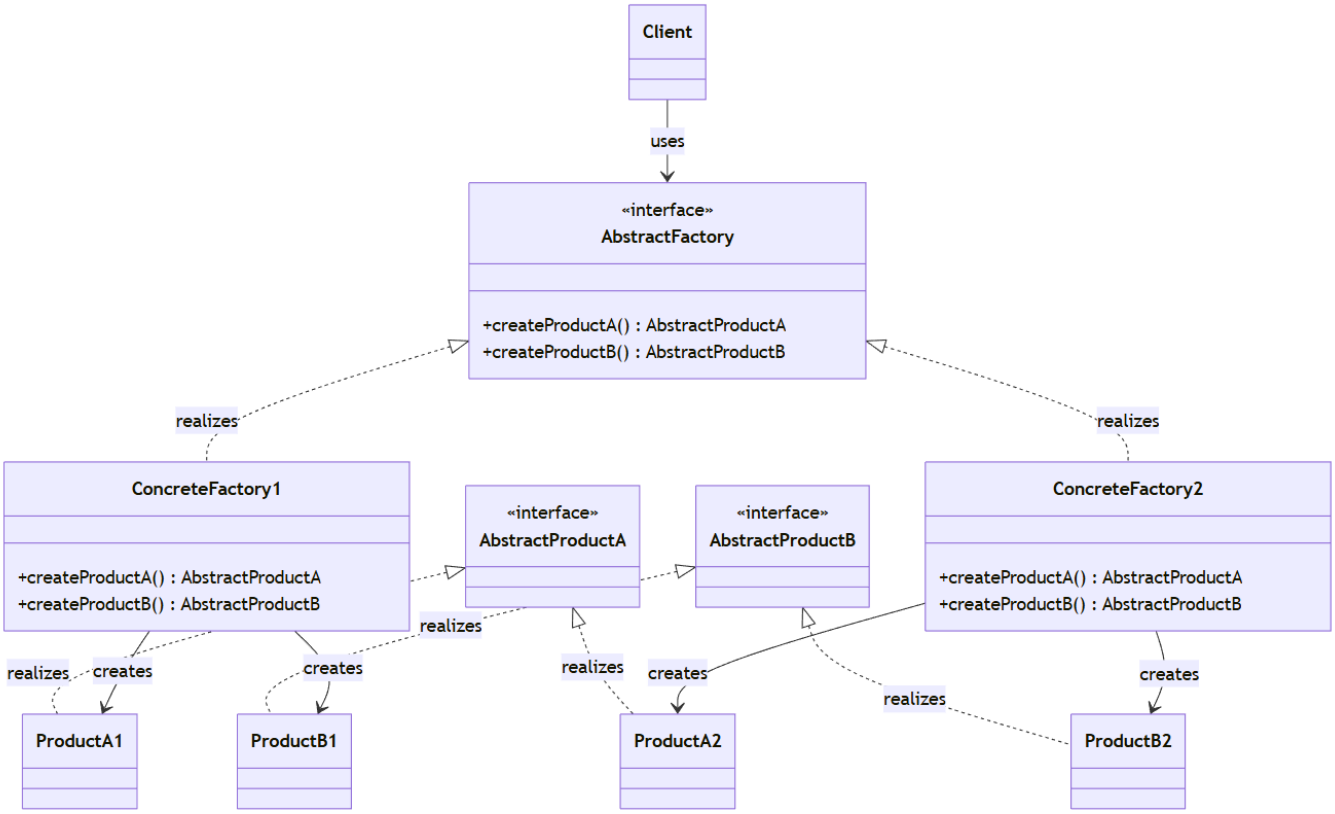
5. Висновок

У ході виконання лабораторної роботи було досліджено та реалізовано породжувальний патерн проєктування «Абстрактна фабрика» (Abstract Factory). Цей патерн дозволив створити гнучку архітектуру для ініціалізації інтерфейсу користувача залежно від його ролі (Admin або User), визначеної у базі даних. Завдяки введенню інтерфейсу ISessionFactory та конкретних фабрик AdminSessionFactory і UserSessionFactory, було забезпечено гарантоване створення узгоджених сімейств об'єктів (заголовків вікон та запрошень командного рядка). Це рішення ізолювало логіку створення компонентів від бізнес-логіки терміналу, що спрощує підтримку коду та дозволяє легко додавати нові ролі користувачів без модифікації існуючого коду клієнта.

6. Контрольні питання:

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

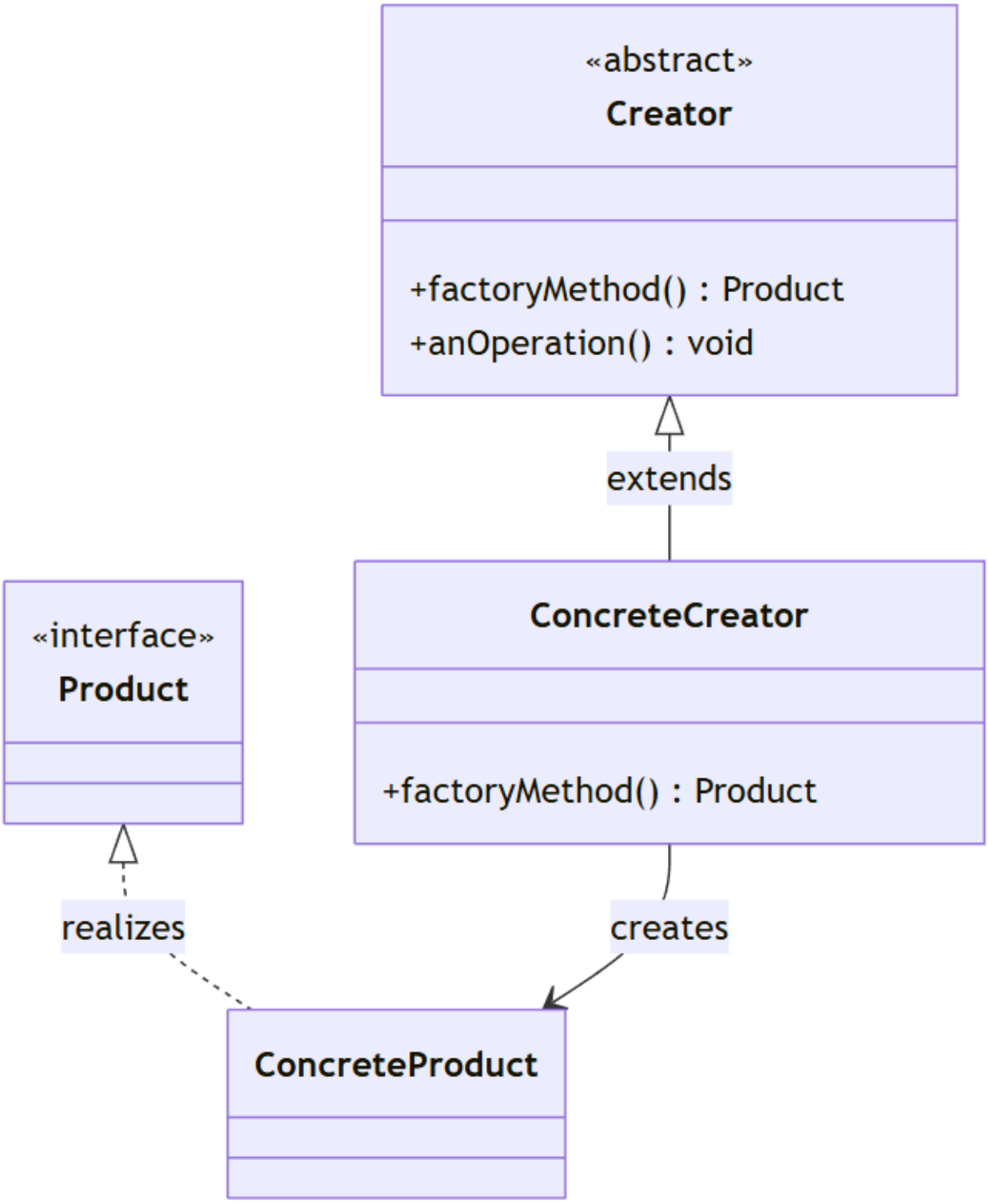
2. Нарисуйте структуру шаблону «Абстрактна фабрика»



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»



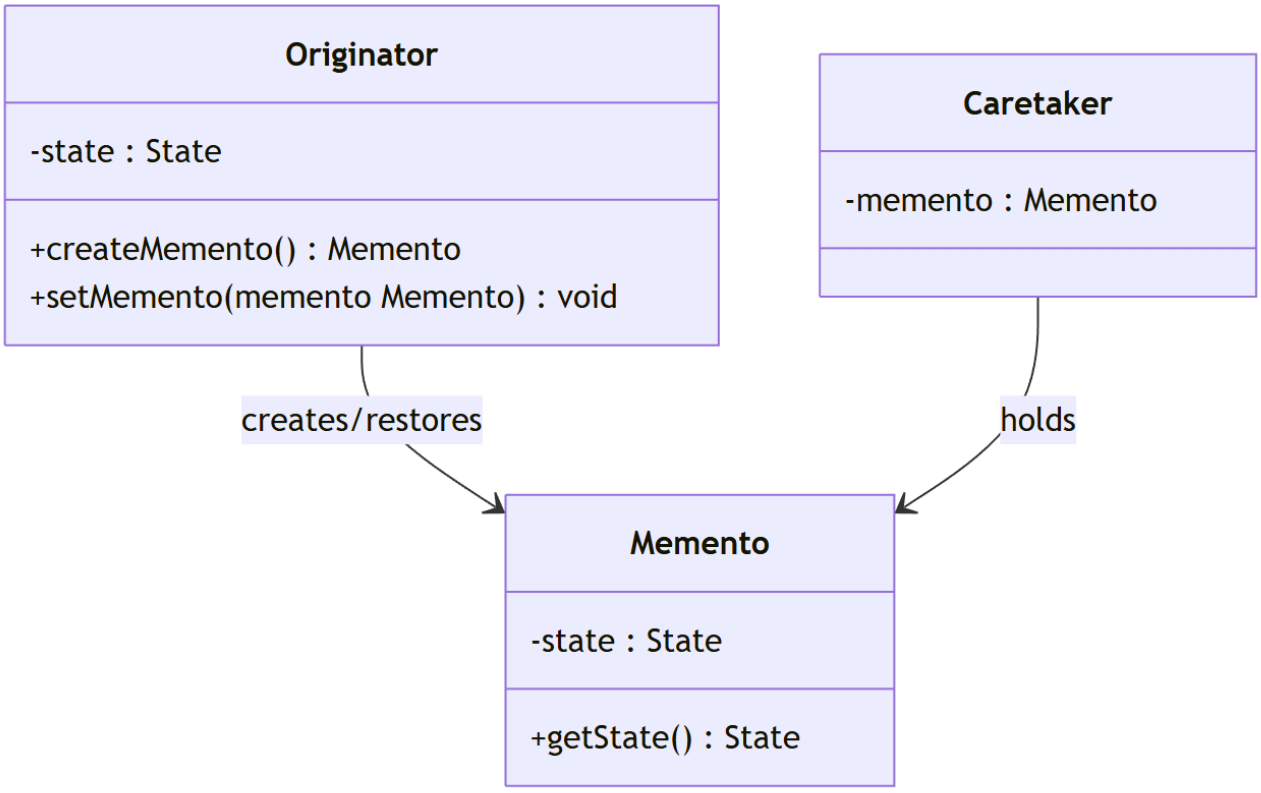
6. Які класи входять в шаблон «Фабричний метод», та яка між ними взаємодія? Класи: Creator, ConcreteCreator, Product, ConcreteProduct. Взаємодія: Creator викликає FactoryMethod() для створення об'єкта; ConcreteCreator повертає конкретний ConcreteProduct.

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.

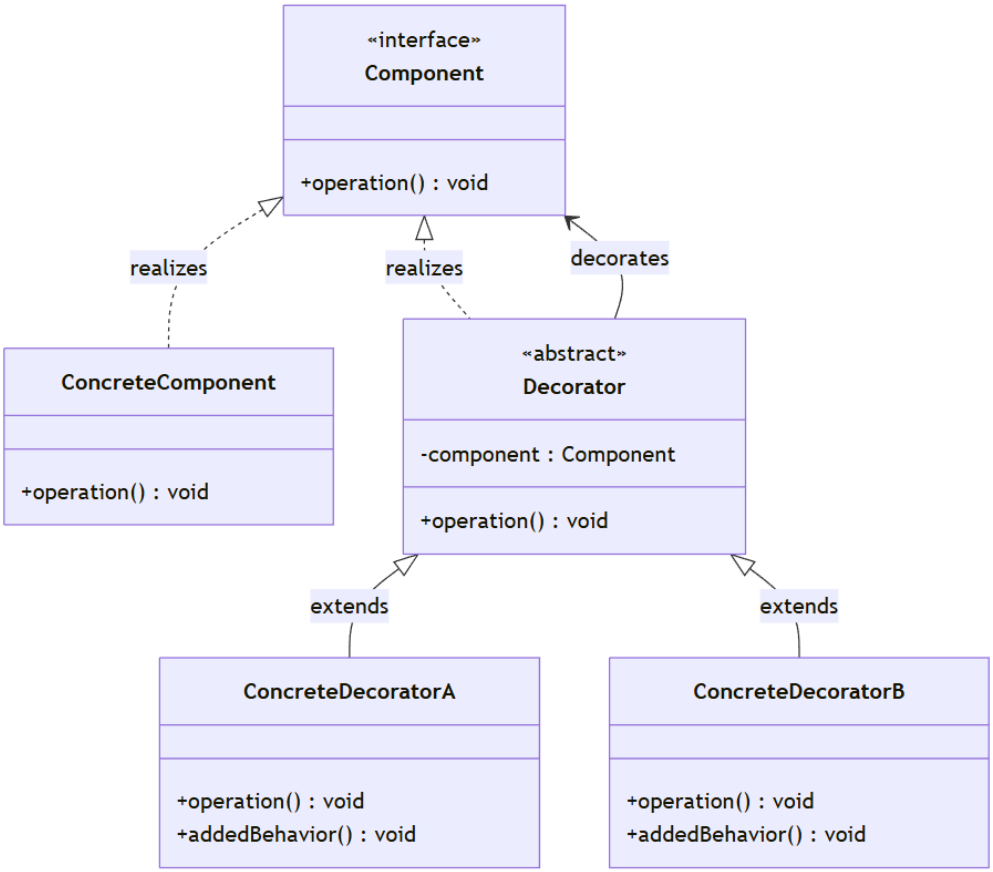
9. Нарисуйте структуру шаблону «Знімок»



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.