

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

Лабораторна робота №6

Тема: Патерни проектування.

Виконала
студентка групи IA-32:
Слюсарєва А.А.

Київ 2025

ЗМІСТ

[6.1 Завдання](#)

[6.2 Теоретичні відомості](#)

[6.3 Хід роботи](#)

[6.4 Висновок](#)

[6.5 Контрольні запитання](#)

ЛАБОРАТОРНА РОБОТА № 6

Тема: Патерни проектування.

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

6.1. Завдання

- Ознайомитись з короткими теоретичними відомостями.
- Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
- Реалізувати один з розглянутих шаблонів за обраною темою.
- Реалізувати не менше 3-х класів відповідно до обраної теми.
- 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.

6.2. Теоретичні відомості

Шаблон «Observer»

Призначення: Шаблон визначає залежність «один-до-багатьох» таким чином, що коли один об'єкт змінює власний стан, усі інші об'єкти отримують про це сповіщення і мають можливість змінити власний стан також.

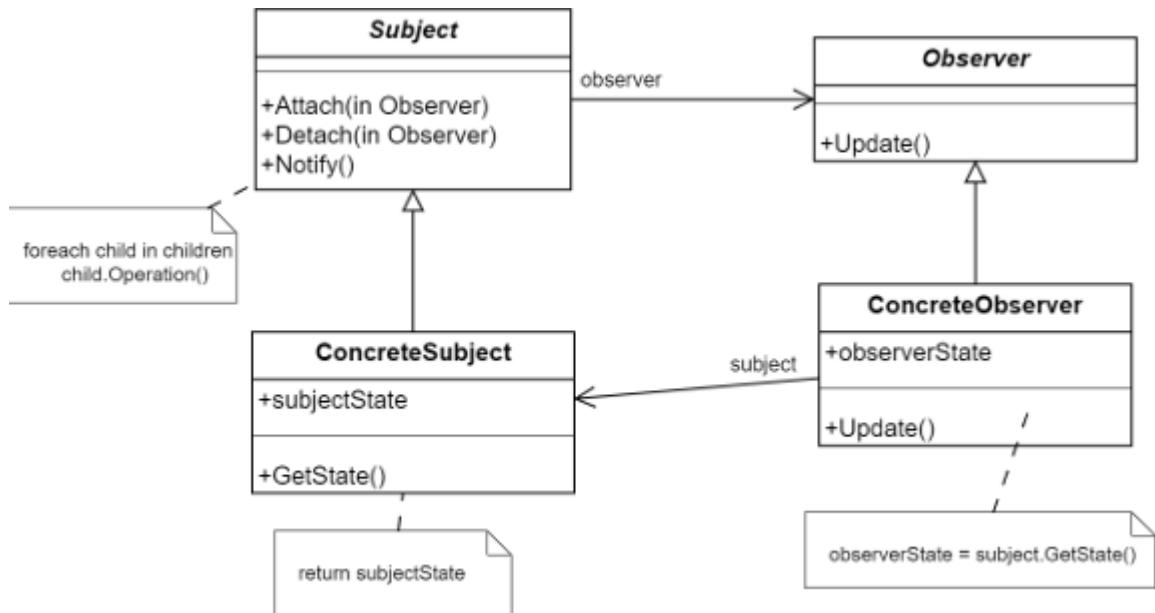


Рисунок 6.4. Структура патерна «Спостерігач»

Розглянемо цей шаблон на прикладі. Припустимо, є деяка банківська система і декілька користувачів переглядають баланс на рахунку пана І. У цей момент пан І. кладе на свій рахунок деяку суму, яка міняє загальний баланс. Кожен з користувачів, що переглядали баланс, отримує про це звістку (для користувачів ця звістка може бути прозорою – просто зміна цифр, або попередження про те, що баланс змінився). Раніше неможливі дії для користувачів (переведення до іншої категорії клієнтів і тому подібне) стають доступними. Цей шаблон дуже широко поширений в шаблоні MVVM і механізмі «прив'язок» (bindings) в WPF і частково в WinForms. Інша назва шаблону – підписка/розсилка. Кожен з оглядачів власноручно підписується на зміни конкретного об'єкту, а об'єкти зобов'язані сповіщати своїх передплатників про усі свої зміни (на даний момент конкретних механізмів автоматичного сповіщення про зміну стану в .NET мовах не існує).

Приклад з життя: Коли ви підписуєтесь на канал на YouTube і натискаєте на «дзвіночок» ви фактично оформлюєте підписку на отримання

повідомлень про вихід нових відео. Вам не потрібно заходити на канал і перевіряти чи не вийшло нове відео. Ви будете отримувати повідомлення про вихід нових відео на каналах Фактично на сервісі YouTube є список підписників на канал, хто має отримувати повідомлення про вихід нового відео. В якості прикладу також можна згадати підписку «Повідомити про появу товару» на сторінці товару в магазині «Розетка».

Переваги та недоліки:

- + Можливість паралельної та асинхронної обробки повідомлень про оновлення.
- + Спостерігачів можна добавляти та видаляти в будь-який момент часу.
- + Спостерігач і суб'єкт можуть працювати в різних потоках.
- + Реалізує принцип слабкого зв'язку між об'єктами.
- Послідовність розсылки повідомлень підписникам не підтримується.

6.3 Хід роботи

Діаграма класів:

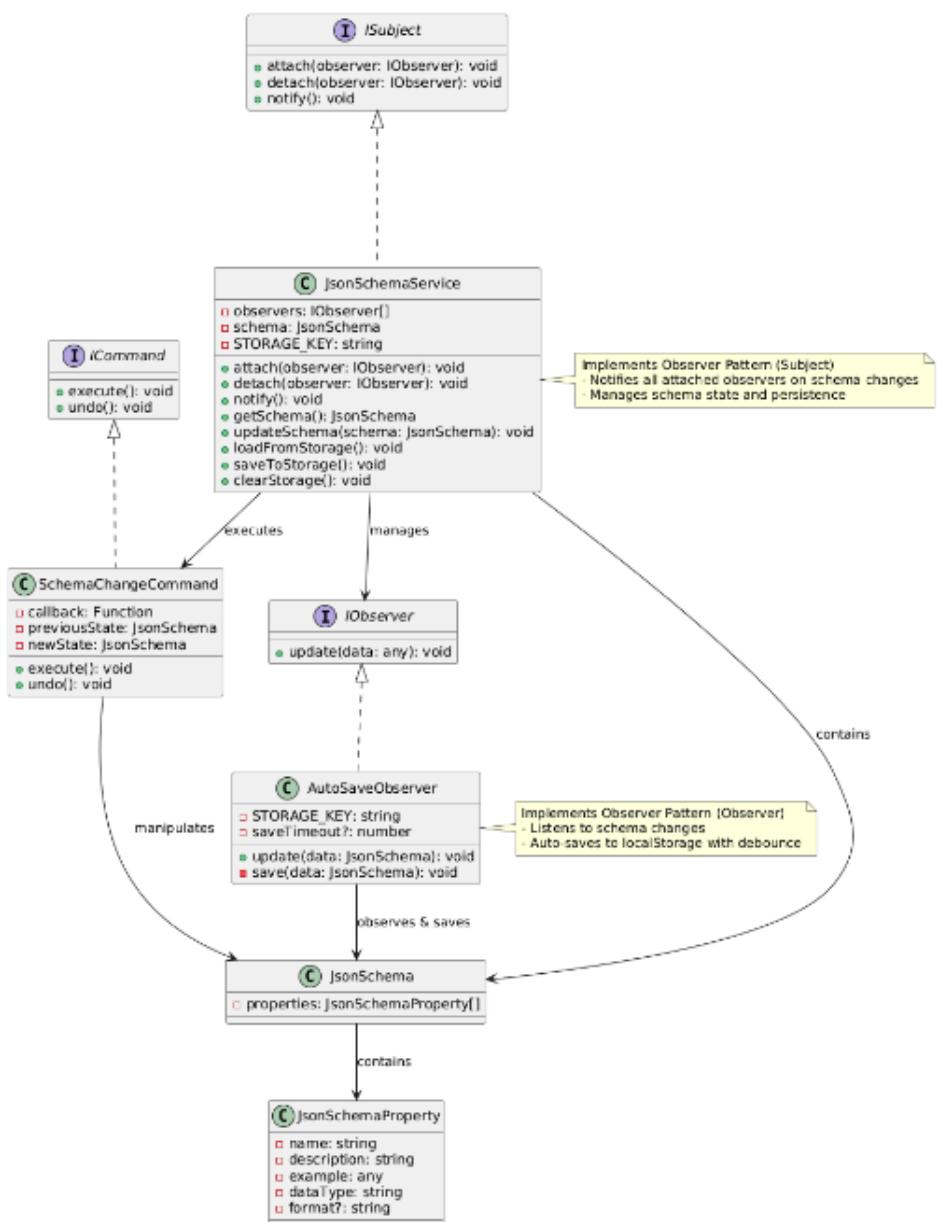


Рис 1. Діаграма класів системи

Фрагменти коду:

```

import { Injectable } from '@angular/core';
import { IOwner } from
'../../models/observer.model';
  
```

```
import { JsonSchema } from
'./models/json-schema.model';

@Injectable({
  providedIn: 'root'
})

export class AutoSaveObserver implements IObserver {
  private readonly STORAGE_KEY =
  'json_schema_autosave';
  private saveTimeout?: number;

  update(data: JsonSchema): void {
    if (this.saveTimeout) {
      clearTimeout(this.saveTimeout);
    }

    this.saveTimeout = window.setTimeout(() => {
      this.save(data);
    }, 1000);
  }

  private save(data: JsonSchema): void {
    localStorage.setItem(this.STORAGE_KEY,
    JSON.stringify(data));
  }
}

export interface IObserver {
  update(data: any): void;
}
```

```
}

export interface ISubject {
    attach(observer: IObserver): void;
    detach(observer: IObserver): void;
    notify(): void;
}
```

У реалізації використано патерн Observer (Спостерігач), який забезпечує автоматичне реагування на зміну стану JSON-схеми без жорсткого зв'язування компонентів між собою. Об'єкт повідомляє всіх підписаних спостерігачів через метод `notify()`, не знаючи, що саме вони роблять із даними. Клас `AutoSaveObserver` реалізує інтерфейс `IObserver` і реагує на зміни схеми, запускаючи автозбереження з затримкою. Завдяки цьому логіка збереження повністю відокремлена від логіки редагування, а підписники можуть додаватися або видалятися динамічно, не змінюючи код об'єкта.

6.4 Висновок

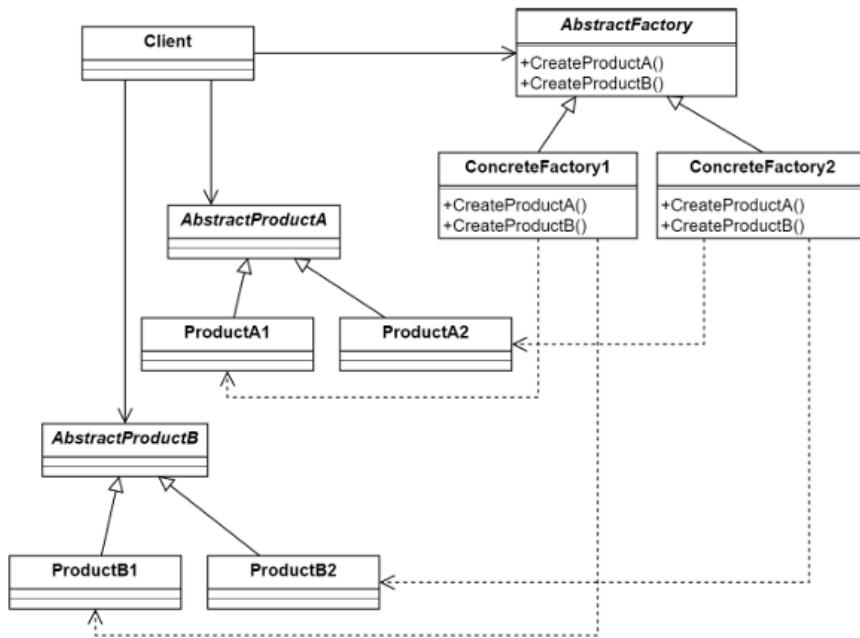
Під час даної роботи було реалізовано патерн Спостерігач у додатку JSON Tool.

6.5 Контрольні питання

1. Яке призначення шаблону «Абстрактна фабрика»?

«Абстрактна фабрика» забезпечує створення сімейств пов'язаних об'єктів без необхідності вказувати їх конкретні класи.

2. Нарисуйте структуру шаблону «Абстрактна фабрика».



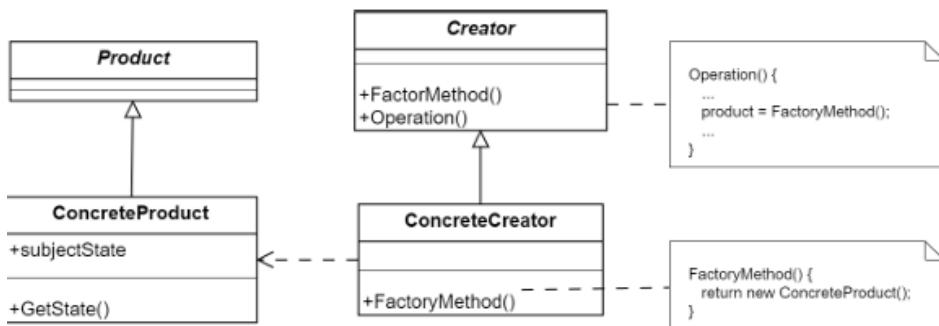
3. Які класи входять в шаблон «Абстрактна фабрика», та яка між ними взаємодія?

Абстрактна фабрика, Конкретні фабрики, Абстрактні продукти, Конкретні продукти, Клієнт.

4. Яке призначення шаблону «Фабричний метод»?

«Фабричний метод» передає створення об'єктів підкласам, дозволяючи змінювати тип створюваних продуктів через успадкування.

5. Нарисуйте структуру шаблону «Фабричний метод».



6. Які класи входять в шаблон «Фабричний метод», та яка між ними взаємодія?

Абстрактний творець, конкретні творці, абстрактний продукт та конкретні продукти. Творець визначає фабричний метод, а конкретні творці реалізують його для створення потрібних продуктів.

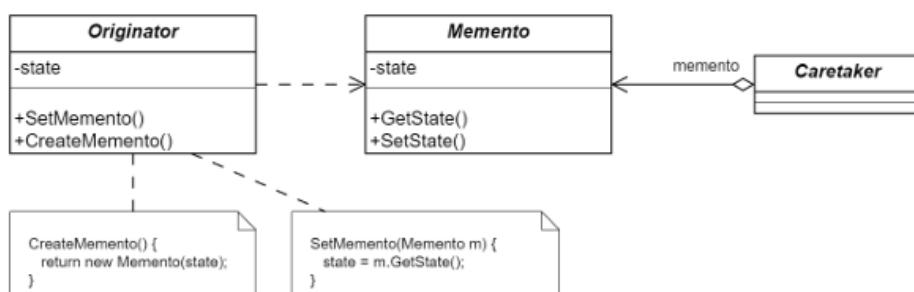
7. Чим відрізняється «Абстрактна фабрика» від «Фабричний метод»?

Абстрактна фабрика створює *сімейства* продуктів через об'єкти-фабрики, тоді як фабричний метод створює *один продукт*, делегуючи це підкласам.

8. Яке призначення шаблону «Знімок» (Memento)?

«Знімок» зберігає внутрішній стан об'єкта без розкриття його деталей, щоб пізніше можна було відновити цей стан.

9. Нарисуйте структуру шаблону «Знімок».



10. Які класи входять в шаблон «Знімок», та яка між ними взаємодія?

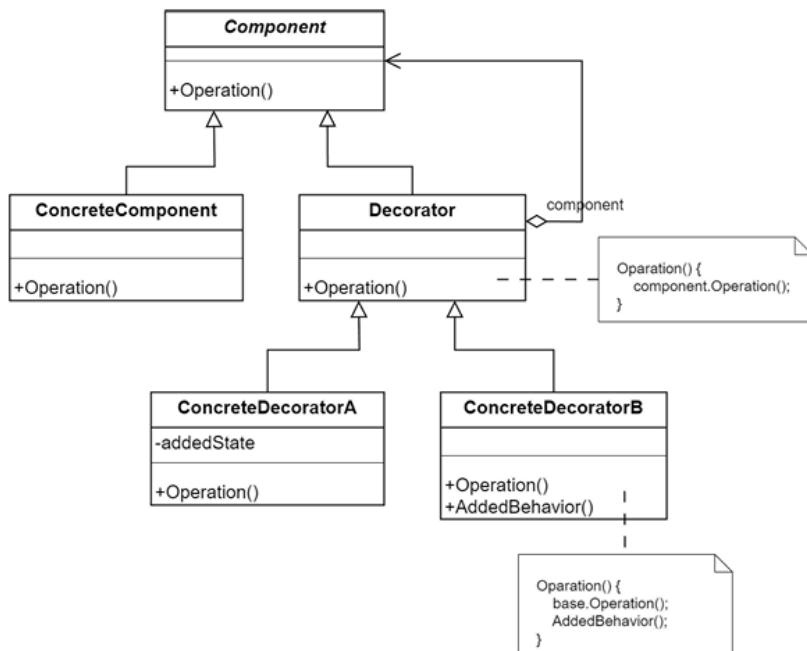
Originator (створює та відновлює стан), Memento (зберігає стан), Caretaker (зберігає знімки, але не має доступу до їх внутрішнього змісту).

11. Яке призначення шаблону «Декоратор»?

«Декоратор» динамічно додає нові обов'язки об'єктам, обгортуючи їх у

додаткові класи без зміни існуючого коду.

12. Нарисуйте структуру шаблону «Декоратор».



13. Які класи входять в шаблон «Декоратор», та яка між ними взаємодія?

Компонент (інтерфейс), конкретний компонент, базовий декоратор, конкретні декоратори. Декоратор містить компонент і делегує йому роботу, додаючи свій функціонал.

14. Які є обмеження використання шаблону «Декоратор»?

Декоратор ускладнює структуру класів, може створити надмірну кількість дрібних об'єктів, важче налагоджувати ланцюжки обгортань та контролювати порядок застосування декораторів.