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 № 4

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 «Singleton», «Iterator», «Proxy», «State», «Strategy» та навчитися застосовувати їх в реалізації програмної системи.

2. Теоретичні відомості:

Singleton: Забезпечує єдиний екземпляр класу з глобальним доступом. Використовується для унікальних ресурсів (наприклад, конфігураційні файли), але вважається антипатерном через глобальний стан.
Iterator: Дозволяє послідовний доступ до елементів колекції без розкриття її внутрішньої структури. Підтримує різні методи обходу (наприклад, у глибину, випадково).
Proxy: Діє як замінник або заглушка для іншого об’єкта, додаючи функціонал (наприклад, ледаче завантаження, контроль доступу). Зменшує кількість запитів до зовнішніх сервісів (наприклад, оптимізація DocuSign).
State: Дозволяє змінювати поведінку об’єкта залежно від його стану (наприклад, типи карток або режими системи). Використовує окремі класи для кожного стану.
Strategy: Уможливлює заміну алгоритмів поведінки об’єкта (наприклад, методи сортування чи маршрути). Відокремлює логіку алгоритмів від контексту для гнучкості.

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" необхідно забезпечити можливість динамічної зміни візуального оформлення (теми) інтерфейсу користувача. Реалізація цієї функції за допомогою великих умовних конструкцій (if-else або switch) безпосередньо у коді форми призвела б до порушення принципу відкритості/закритості (OCP) та ускладнила б підтримку коду.

Паттерн Strategy (Стратегія) є доцільним, оскільки він дозволяє винести алгоритми налаштування кольорів (стилізації) у окремі незалежні класи. Це дозволяє:

- Змінювати алгоритм поведінки (стиль відображення) під час виконання програми.
- Уникнути дублювання коду налаштування кольорів для різних форм.
- Легко додавати нові теми, створюючи нові класи стратегій, без зміни існуючого коду контексту чи форм.

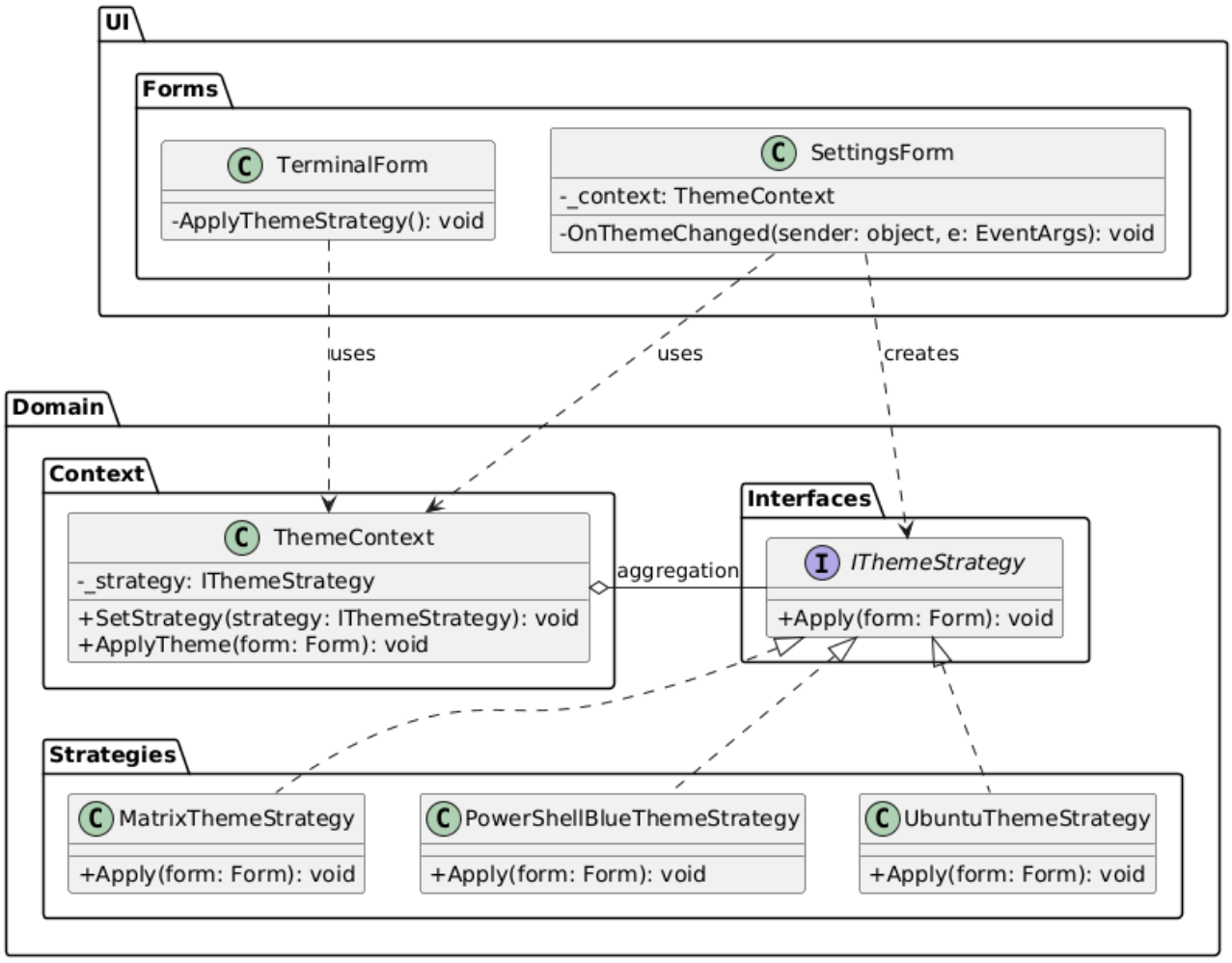


Рисунок 1 - Структура патерну Strategy

Діаграма(рис. 1) демонструє класичну структуру патерну "Стратегія", адаптовану під задачу стилізації:

- IThemeStrategy (Strategy Interface): Оголошує загальний інтерфейс для всіх варіантів стилізації. Він містить метод Apply(Form form), який приймає форму для налаштування.
- MatrixThemeStrategy, PowerShellBlueThemeStrategy, UbuntuThemeStrategy (Concrete Strategies): Реалізують конкретні алгоритми розфарбовування елементів керування у відповідні кольори (чорно-зелений, синьо-білий, фіолетовий).
- ThemeContext (Context): Клас, який зберігає посилання на поточну стратегію (_strategy). Він не реалізує алгоритм самостійно, а делегує виконання методу Apply об'єкту конкретної стратегії. Клієнт може змінювати стратегію через метод SetStrategy.
- SettingsForm / TerminalForm (Client): Клієнтський код, який створює об'єкт контексту та обирає конкретну стратегію залежно від вибору користувача (у випадаючому списку) або налаштувань профілю.

Програмна реалізація:

а) Інтерфейс Стратегії (IThemeStrategy.cs)

```
using System.Windows.Forms;

namespace PowerShellTerminal.App.Domain.Interfaces
{
    public interface IThemeStrategy
    {
        void Apply(Form form);
    }
}
```

б) Приклад Конкретної Стратегії (UbuntuThemeStrategy.cs):

```
public class UbuntuThemeStrategy : IThemeStrategy
{
    public void Apply(Form form)
    {
        var ubuntuPurple = Color.FromArgb(48, 10, 36);
        var ubuntuOrange = Color.FromArgb(221, 72, 20);

        form.BackColor = ubuntuPurple;
        foreach (Control c in form.Controls)
        {
            c.BackColor = ubuntuPurple;
            c.ForeColor = Color.White;
            if (c is Button) c.BackColor = ubuntuOrange;
        }
    }
```

```
}
```

в) Контекст

```
public class ThemeContext
{
    private IThemeStrategy _strategy;

    public void SetStrategy(IThemeStrategy strategy)
    {
        _strategy = strategy;
    }

    public void ApplyTheme(Form form)
    {
        _strategy?.Apply(form);
    }
}
```

г) Використання у Клієнті (SettingsForm.cs або TerminalForm.cs):

```
private void ApplyThemeStrategy()
{
    IThemeStrategy strategy = new MatrixThemeStrategy(); // Default

    switch (_currentUser.ThemeId)
    {
        case 2: strategy = new PowerShellBlueThemeStrategy(); break;
        case 3: strategy = new UbuntuThemeStrategy(); break;
    }

    ThemeContext context = new ThemeContext();
    context.SetStrategy(strategy);
    context.ApplyTheme(this);
}
```

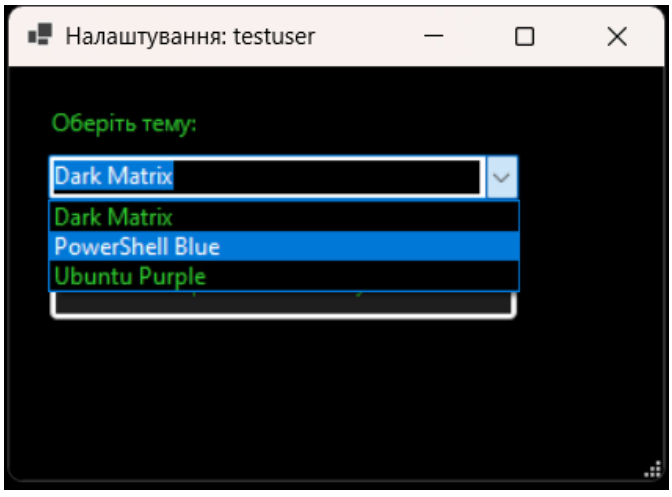


Рисунок 2 – Вибір теми

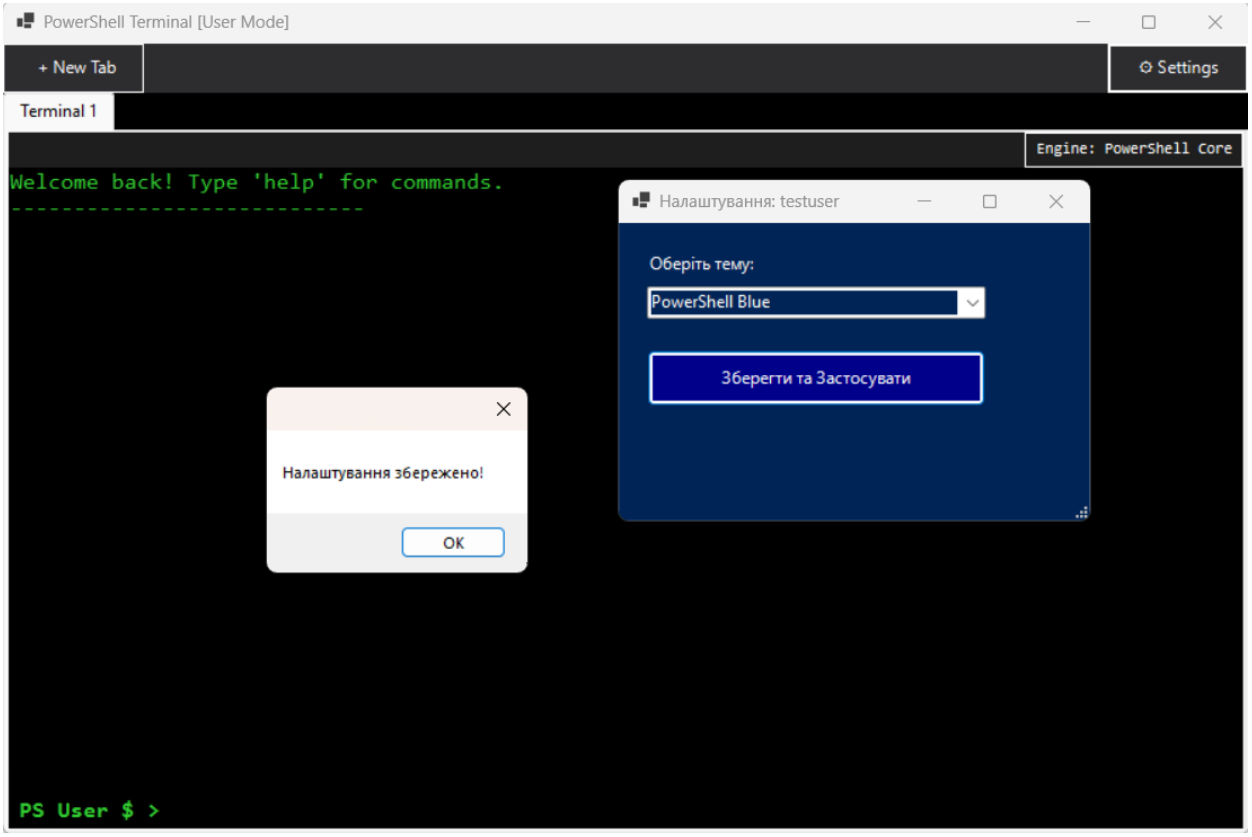


Рисунок 2 – Налаштовуємо інший колір

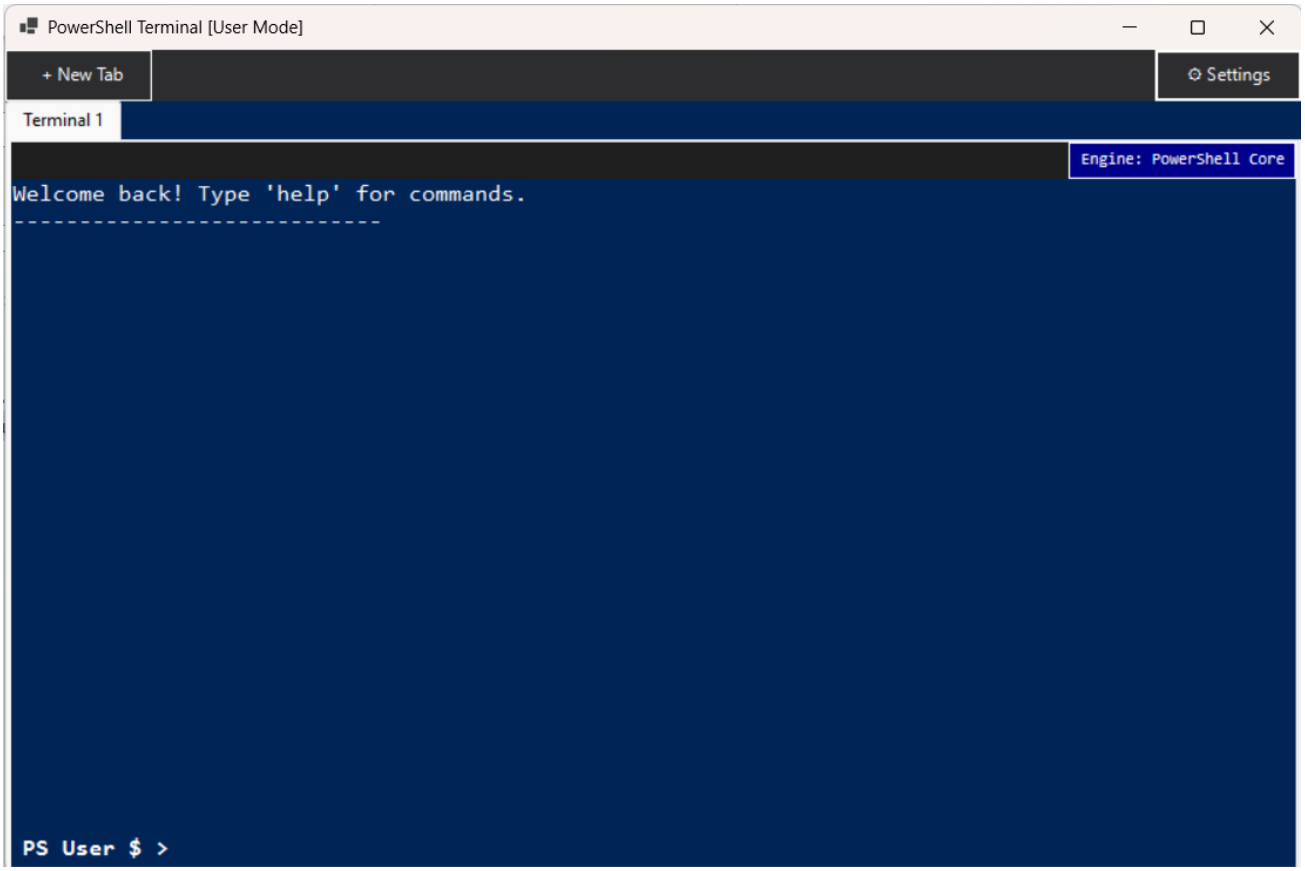


Рисунок 2 – Застосовано синій колір

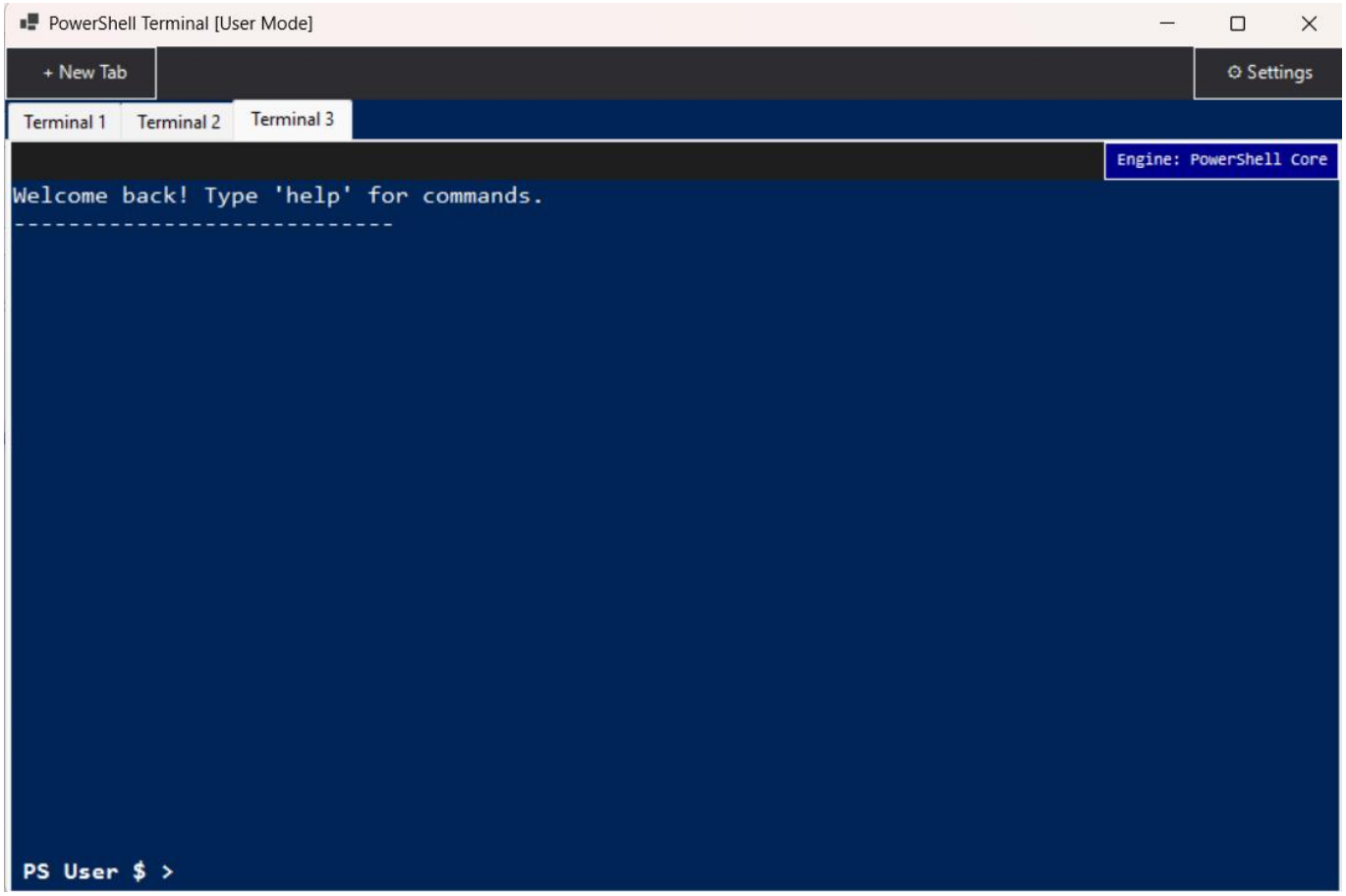


Рисунок 2 – Дизайн зберігається на інших вкладках

5. Висновок

У ході виконання лабораторної роботи було вивчено та реалізовано структурний патерн проєктування «Стратегія» (Strategy). Даний патерн було застосовано для реалізації механізму динамічної зміни тем оформлення в додатку «PowerShell Terminal». Використання цього патерну дозволило інкапсулювати алгоритми стилізації вікон у незалежні класи (MatrixThemeStrategy, UbuntuThemeStrategy тощо), що відокремило логіку візуалізації від бізнес-логіки форм. Це забезпечило дотримання принципу відкритості/закритості (OCP), оскільки додавання нових тем не потребує зміни існуючого коду контексту, а також дозволило змінювати поведінку об'єктів під час виконання програми, підвищивши гнучкість та розширюваність системи.

6. Контрольні питання:

1. Що таке шаблон проєктування?

Шаблон проєктування — це універсальне рішення для типових проблем у розробці ПЗ, яке описує структуру та взаємодію об’єктів.

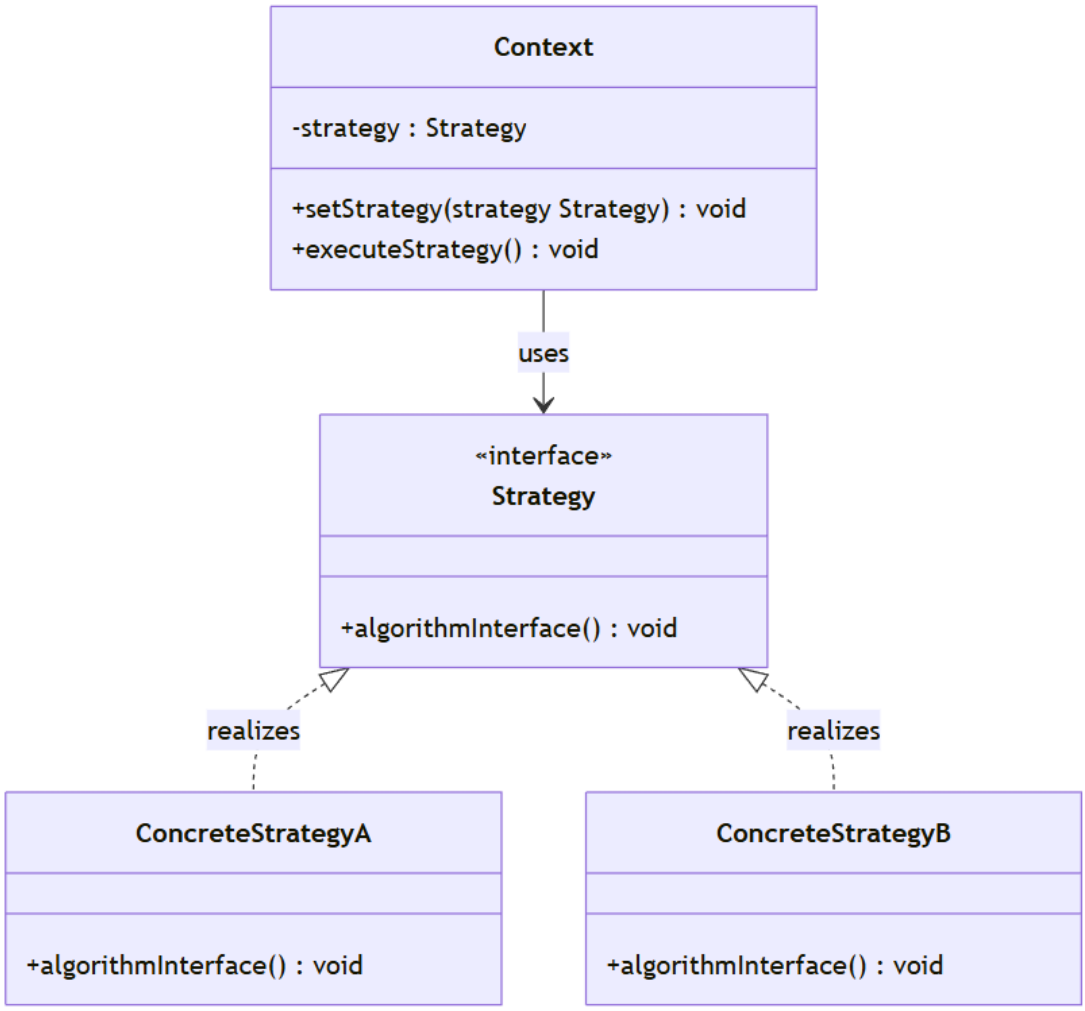
2. Навіщо використовувати шаблони проєктування?

- Спрощують розробку.
- Покращують читабельність і масштабування коду.
- Допомагають уникнути помилок.
- Забезпечують гнучкість і повторне використання.

3. Яке призначення шаблону «Стратегія»?

Шаблон Стратегія дозволяє динамічно вибирати алгоритми, інкапсулюючи їх у взаємозамінні класи, щоб уникнути умовних конструкцій.

7. Нарисуйте структуру шаблону «Стратегія».



5. Які класи входять в шаблон «Стратегія», та яка між ними взаємодія?

Класи:

- Strategy: Інтерфейс із методом algorithm().
- ConcreteStrategy: Реалізації алгоритмів.

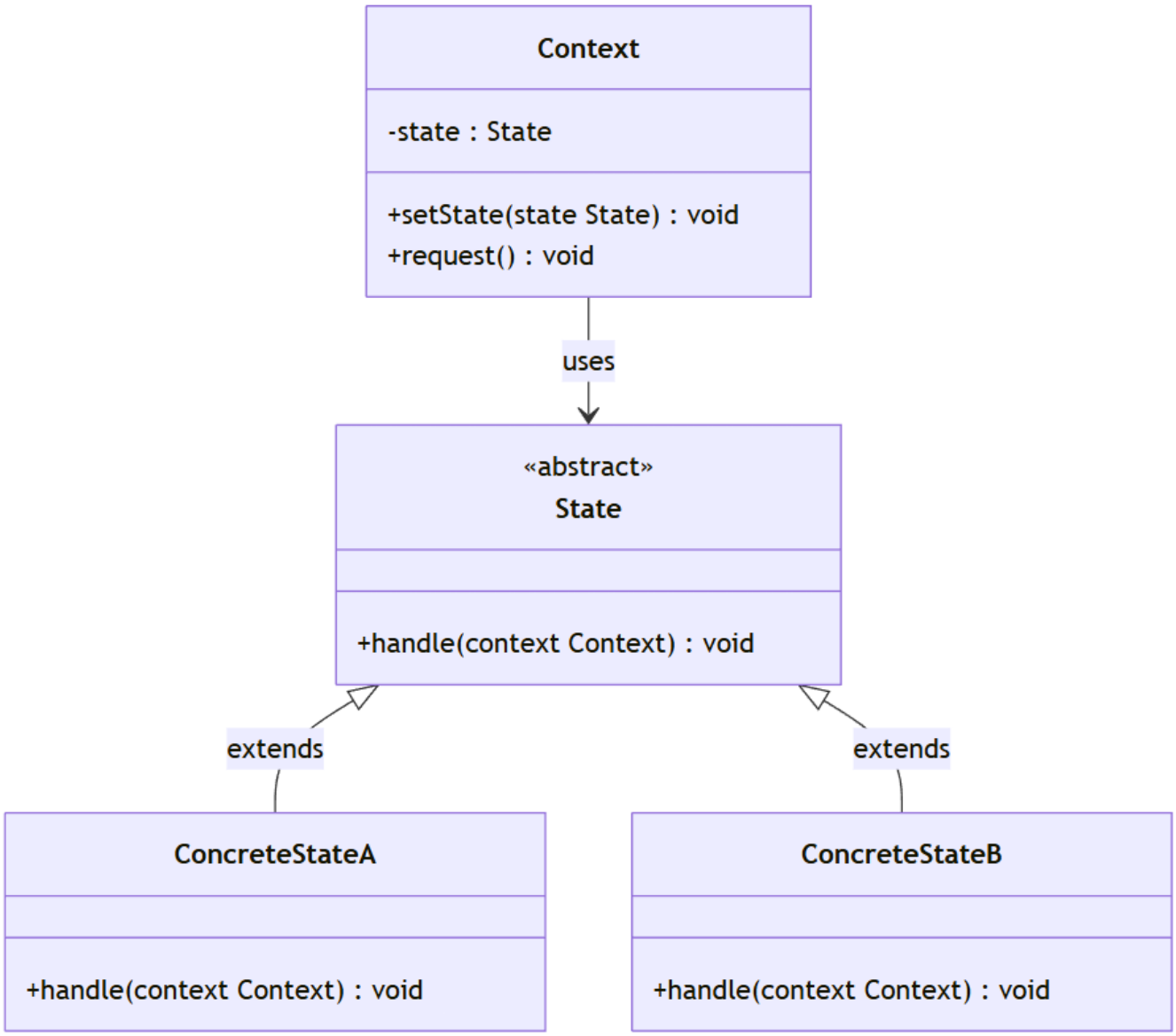
- Context: Використовує Strategy через setStrategy() і викликає algorithm().

Взаємодія: Контекст делегує виконання алгоритму об’єкту Strategy, який клієнт встановлює динамічно.

6. Яке призначення шаблону «Стан»?

Шаблон «Стан» дозволяє об’єкту змінювати поведінку залежно від стану, інкапсулюючи стани в окремі класи.

7. Нарисуйте структуру шаблону «Стан».



8. Які класи входять в шаблон «Стан», та яка між ними взаємодія?

Класи:

- State: Інтерфейс із методом handle().
- ConcreteState: Реалізації поведінки для стану.

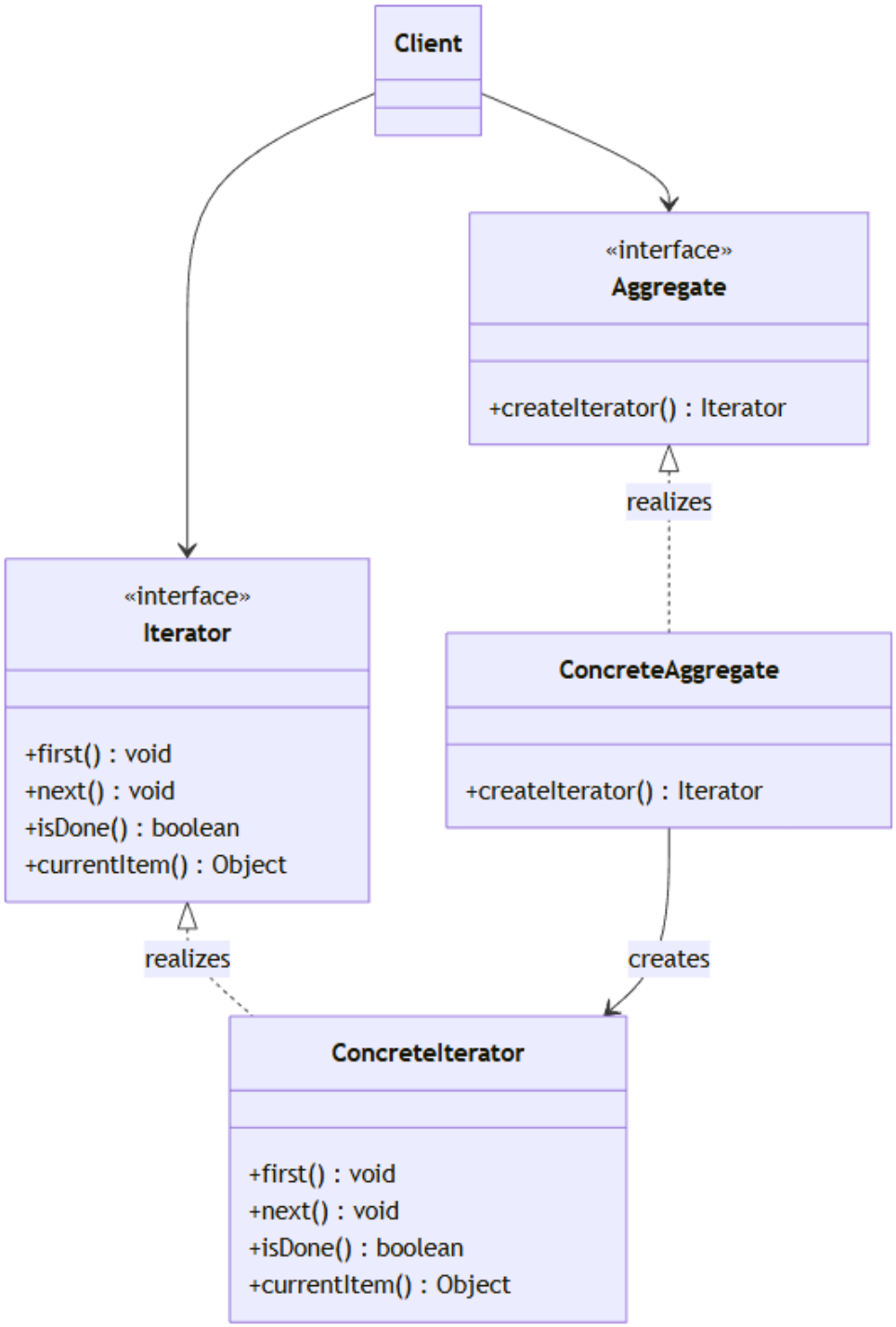
- Context: Зберігає стан, делегує виконання handle().

Взаємодія: Контекст викликає handle() поточного стану, який може змінити стан через setState().

9. Яке призначення шаблону «Ітератор»?

Шаблон «Ітератор» забезпечує послідовний доступ до елементів колекції, приховуючи її внутрішню структуру.

10. Нарисуйте структуру шаблону «Ітератор».



11. Які класи входять в шаблон «Ітератор», та яка між ними взаємодія?

Класи:

- Iterator: Інтерфейс із методами next(), hasNext().
- ConcreteIterator: Реалізація обходу.
- Aggregate: Інтерфейс із createIterator().
- ConcreteAggregate: Створює ConcreteIterator.

Взаємодія: Клієнт отримує ітератор через createIterator() і використовує його для обходу колекції.

12. В чому полягає ідея шаблона «Одинак»?

Шаблон «Одинак» забезпечує єдиний екземпляр класу з глобальним доступом до нього.

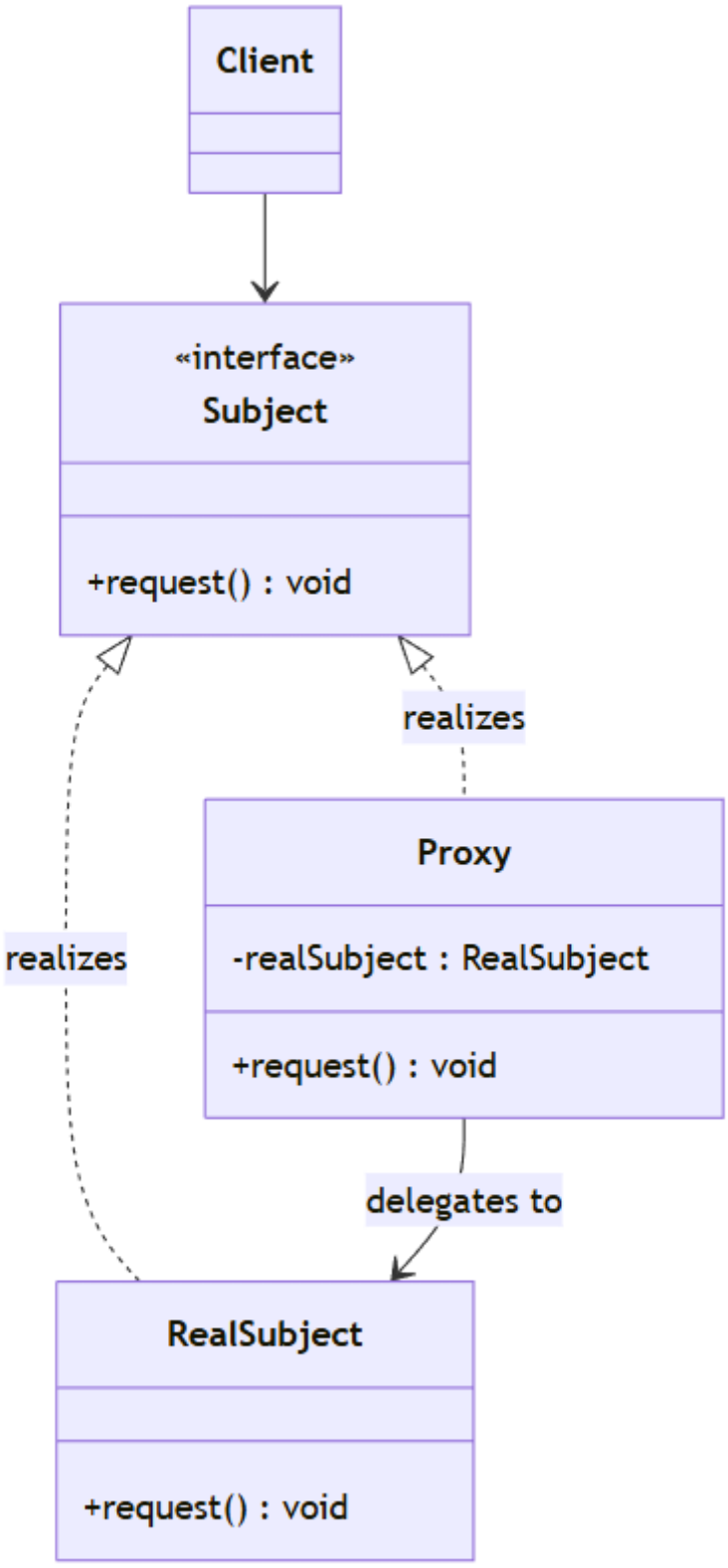
13. Чому шаблон «Одинак» вважають «анти-шаблоном»?

Бо він порушує принципи ООП: ускладнює тестування, створює приховані залежності та глобальний стан

14. Яке призначення шаблону «Проксі»?

Шаблон «Проксі» контролює доступ до об’єкта, додаючи функціонал, як-от ледарська ініціалізація чи перевірка прав.

15. Нарисуйте структуру шаблону «Проксі».



16. Які класи входять в шаблон «Проксі», та яка між ними взаємодія?

Класи:

- Subject: Інтерфейс із методом request().

- RealSubject: Виконує основну роботу.
- Proxy: Контролює доступ до RealSubject.

Взаємодія: Клієнт викликає request() через Proxy, який делегує виклик до RealSubject або додає логіку.