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:
Підковка Д. О.

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. **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.

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.

Для системи веб-браузера реалізовано патерн Proxy (Замісник) для оптимізації завантаження веб-сторінок. Користувач вводить URL => Browser звертається до інтерфейсу завантажувача => SmartProxy перевіряє наявність коду в кеші => RealLoader завантажує з мережі тільки за необхідності.

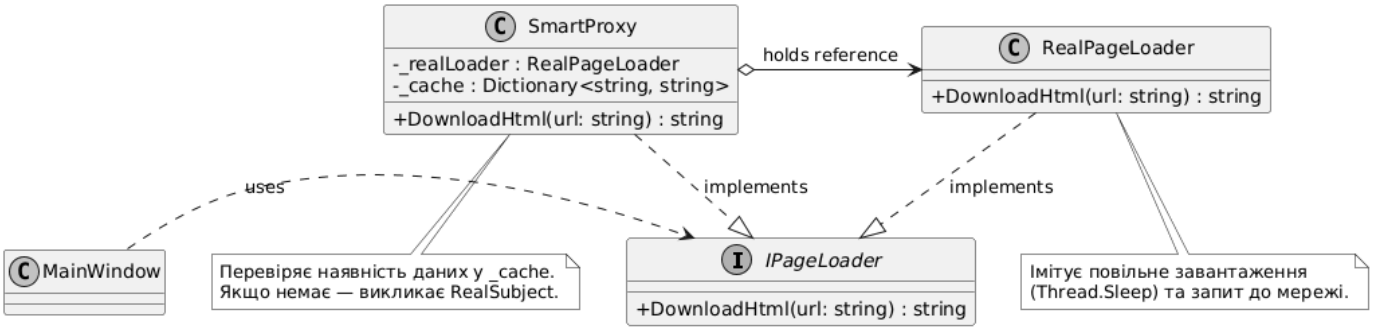


Рисунок 1 - Структура патерну Iterator

IPageLoader – це спільний інтерфейс, який описує метод DownloadHtml(url). Він задає контракт для всіх об'єктів, що вміють отримувати контент сторінки, дозволяючи клієнтському коду працювати з ними однаково, незалежно від реалізації.

RealPageLoader – це конкретна реалізація сервісу ("Реальний суб'єкт"). Цей клас виконує фактичну, ресурсомістку роботу: встановлює HTTP-з'єднання з сервером, очікує відповідь та завантажує "сирий" HTML-код. У нашій системі це імітується затримкою (Thread.Sleep) для демонстрації навантаження на мережу.

SmartProxy – це клас-замісник, який реалізує той самий інтерфейс IPageLoader. Він зберігає посилання на екземпляр реального сервісу (RealPageLoader) та додає додаткову логіку: перевірку внутрішнього словника (кешу). Якщо сторінка вже була завантажена, проксі повертає дані миттєво з пам'яті, не турбуючи реальний завантажувач.

MainWindow (Client) – це клієнтський код, який використовує завантажувач. Він працює виключно через інтерфейс IPageLoader і не знає, чи отримав він дані з мережі, чи з кешу. Це дозволяє прозоро впроваджувати оптимізацію без зміни коду відображення.

В результаті досягається суттєва економія часу та трафіку: повторні запити до однієї й тієї ж сторінки обробляються миттєво. Це і є суть Замісника – контроль доступу до реального об'єкта для додавання нової поведінки (у нашому випадку – кешування).

```
namespace WebBrowser.Coursework.Services
{

    3 references
    public interface IPageLoader
    {
        4 references
        string DownloadHtml(string url);
    }
}
```

Рисунок 2 - IPageLoader.cs

```
using System.Net.Http;
using System.Threading;

namespace WebBrowser.Coursework.Services
{
    2 references
    public class RealPageLoader : IPageLoader
    {
        3 references
        public string DownloadHtml(string url)
        {

            Thread.Sleep(3000);


            using (var client = new HttpClient())
            {

                try
                {
                    var response = client.GetAsync(url).Result;
                    return response.Content.ReadAsStringAsync().Result;
                }
                catch
                {
                    return $"<html><body><h1>Error loading {url}</h1></body></html>";
                }
            }
        }
    }
}
```

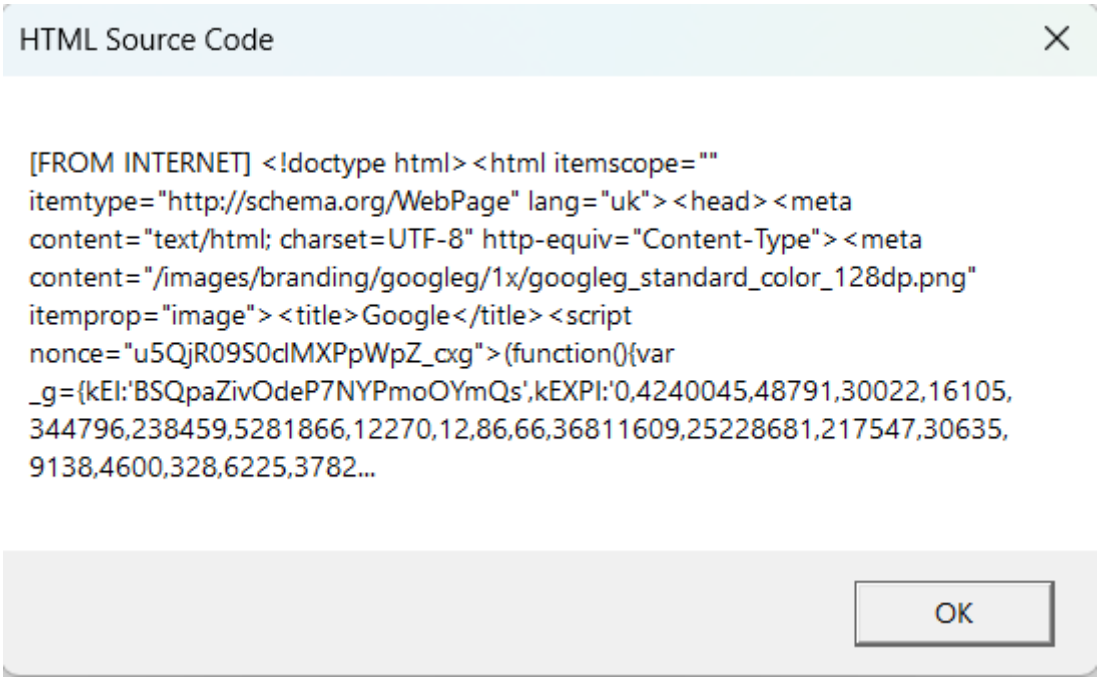
Рисунок 3 - RealPageLoader.cs

```
using System.Collections.Generic;

namespace WebBrowser.Coursework.Services
{
    1 reference
    public class SmartProxy : IPageLoader
    {

        private RealPageLoader _realLoader;


        private Dictionary<string, string> _cache = new Dictionary<string, string>();

        2 references
        public string DownloadHtml(string url)
        {

            if (_cache.ContainsKey(url))
            {

                return $"[FROM CACHE] {_cache[url]}";
            }


            if (_realLoader == null)
            {
                _realLoader = new RealPageLoader();
            }


            string html = _realLoader.DownloadHtml(url);


            _cache[url] = html;

            return $"[FROM INTERNET] {html}";
        }
    }
}
```

Рисунок 4 - SmartProxy.cs

```
private async void BtnViewSource_Click(object sender, RoutedEventArgs e)
{
    string url = AddressBar.Text;
    if (string.IsNullOrEmpty(url)) return;


    BtnViewSource.Content = "Завантаження...";
    BtnViewSource.IsEnabled = false;


    string htmlContent = await Task.Run(() =>
    {
        return _pageLoader.DownloadHtml(url);
    });

    //перші 500 символів, щоб не спамити
    string preview = htmlContent.Length > 500 ? htmlContent.Substring(0, 500) + "..." : htmlContent;

    MessageBox.Show(preview, "HTML Source Code");


    BtnViewSource.Content = "  HTML Code";
    BtnViewSource.IsEnabled = true;
}
```

Рисунок 5 – Обробник події BtnViewSource_Click



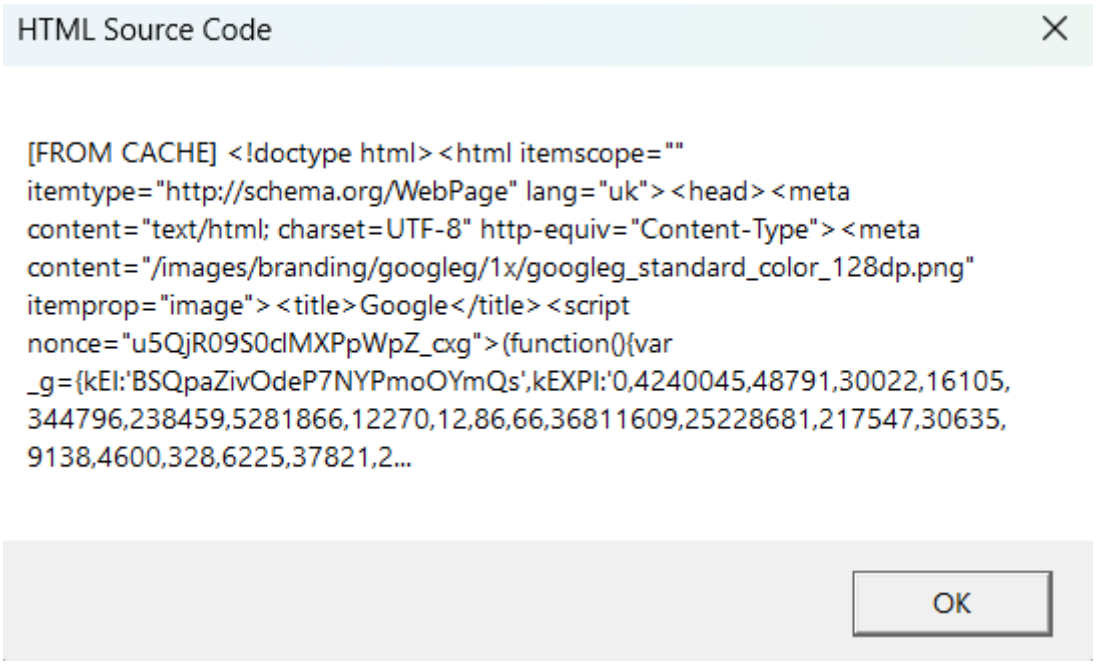


Рисунок 7 та 8 – Результат виконання

4. Висновок

У ході виконання лабораторної роботи було розглянуто базові шаблони проєктування, зокрема Singleton, Iterator, Proxy, State та Strategy. Отримані знання допомогли зрозуміти їхню роль у створенні гнучкої та масштабованої архітектури.

На прикладі настільного застосунку «Web-browser» було реалізовано шаблон **Proxy** (Замісник) для оптимізації процесу завантаження веб-сторінок. Логіка кешування винесена в окремий клас-замісник (**SmartProxy**), що дозволило відокремити алгоритм перевірки кешу від реальної логіки завантаження даних з мережі (**RealPageLoader**).

Такий підхід дозволив клієнтському коду працювати через загальний інтерфейс, не знаючи про внутрішні механізми оптимізації, що підвищило швидкодію системи та дозволило уникнути дублювання важких мережевих запитів без зміни основної логіки відображення. Це підтверджує ефективність патернів проєктування як інструменту для побудови зрозумілого, підтримуваного та готового до розвитку коду.

5. **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.

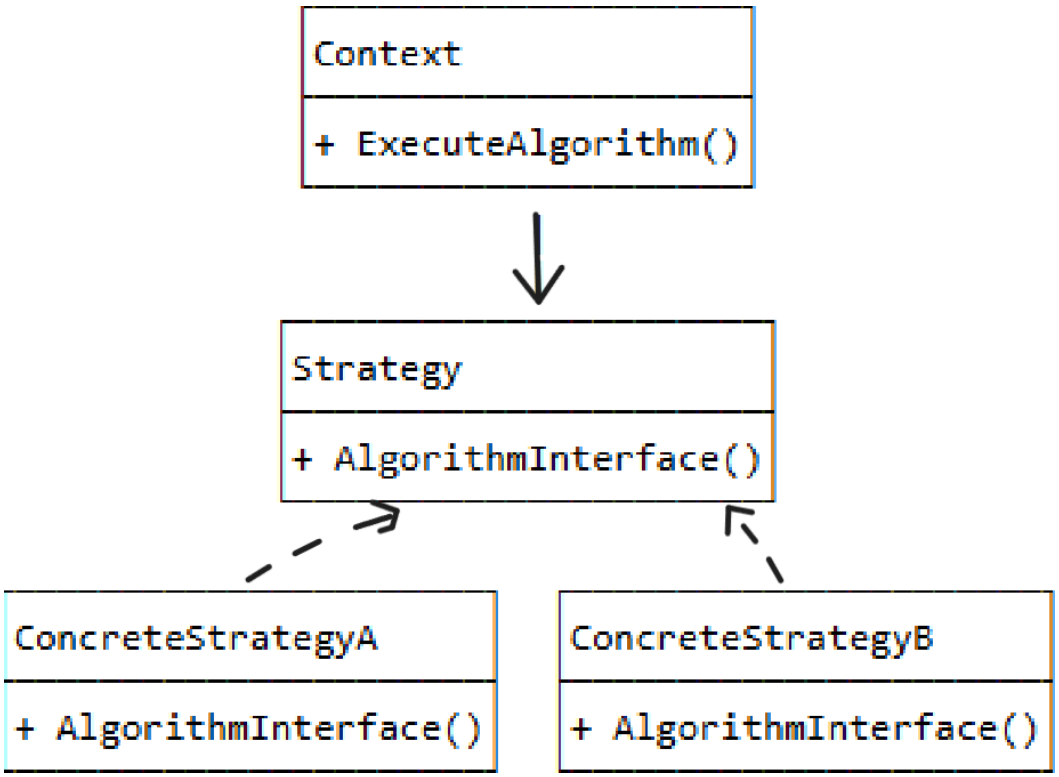
2. Навіщо використовувати шаблони проєктування?

- 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.

6. Нарисуйте структуру шаблону «Стратегія».



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

Класи:

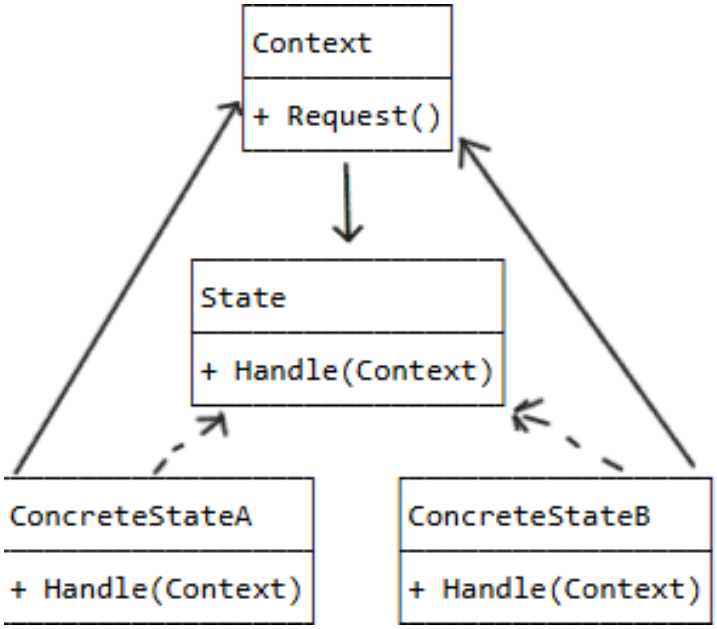
- Strategy: Інтерфейс із методом algorithm().
- ConcreteStrategy: Реалізації алгоритмів.
- Context: Використовує Strategy через setStrategy() і викликає algorithm().

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:

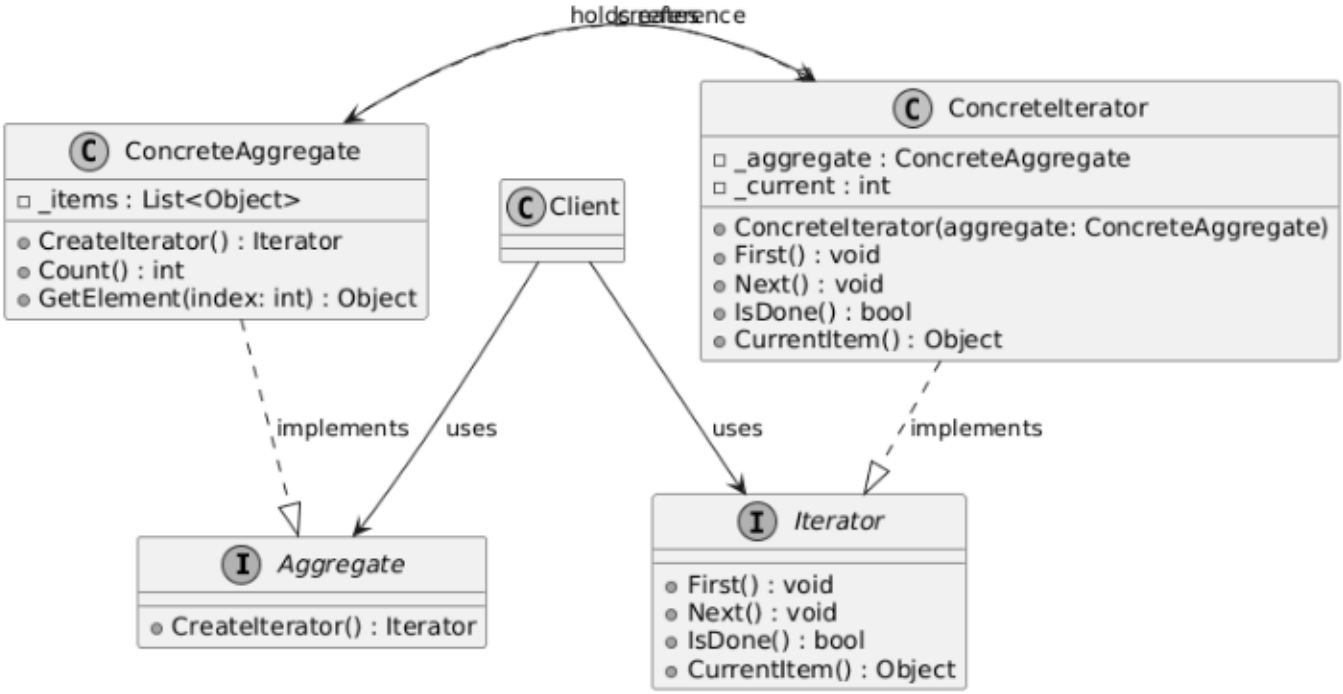
- State: Інтерфейс із методом handle().
- ConcreteState: Реалізації поведінки для стану.
- Context: Зберігає стан, делегує виконання handle().

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.

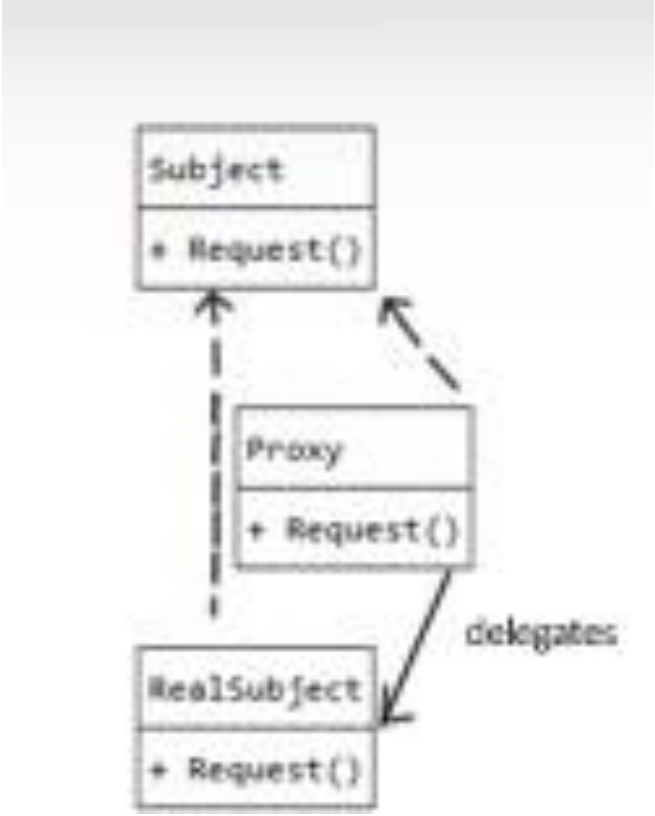
13. Чому шаблон «Одинак» вважають «анти-шаблоном»?

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.