Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 4

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Вступ до паттернів проектування»

«2. HTTP-сервер»

Виконав:
студент групи - ІА-32
Непотачев Іван Дмитрович

Перевірив:
Мягкий Михайло Юрійович

Київ 2025

Тема: Вступ до паттернів проектування.

Мета: Вивчити структуру шаблонів «Singleton», «Iterator», «Proxy», «State», «Strategy» та навчитися застосовувати їх в реалізації програмної системи.

Тема проєкту: HTTP-сервер (state, builder, factory method, mediator, composite, p2p) Сервер повинен мати можливість розпізнавати вхідні запити і формувати коректні відповіді (згідно протоколу HTTP), надавати сторінки chtml (html сторінки з додаванням найпростіших C# конструкцій на розсуд студента), вести статистику вхідних запитів, обробку запитів у багатопотоковому/подієвому режимах.

Зміст

Теоретичні відомості

Поняття шаблону проєктування

Будь-який патерн проєктування, використовуваний при розробці інформаційних систем, являє собою формалізований опис, який часто зустрічається в завданнях проєктування, вдале рішення даної задачі, а також рекомендації по застосуванню цього рішення в різних ситуаціях [5]. Крім того, патерн проєктування обов’язково має загальновживане найменування. Правильно сформульований патерн проєктування дозволяє, відшукавши одного разу вдале рішення, користуватися ним знову і знову. Варто підкреслити, що важливим початковим етапом при роботі з патернами є адекватне моделювання розглянутої предметної області. Це є необхідним як для отримання належним чином формалізованої постановки задачі, так і для вибору відповідних патернів проєктування.

Відповідне використання патернів проєктування дає розробнику ряд незаперечних переваг. Наведемо деякі з них. Модель системи, побудована в межах патернів проєктування, фактично є структурованим виокремленням тих елементів і зв’язків, які значимі при вирішенні поставленого завдання. Крім цього, модель, побудована з використанням патернів проєктування, більш проста і наочна у вивченні, ніж стандартна модель. Проте, не дивлячись на простоту і наочність, вона дозволяє

глибоко і всебічно опрацювати архітектуру розроблюваної системи з використанням спеціальної мови.

Застосування патернів проєктування підвищує стійкість системи до зміни вимог та спрощує неминуче подальше доопрацювання системи. Крім того, важко переоцінити роль використання патернів при інтеграції інформаційних систем організації. Також слід зазначити, що сукупність патернів проєктування, по суті, являє собою єдиний словник проєктування, який, будучи уніфікованим засобом, незамінний для спілкування розробників один одним.

Таким чином шаблони представляють собою, підтверджені роками розробок в різних компаніях і на різних проєктах, «ескізи» архітектурних рішень, які зручно застосовувати у відповідних обставинах.

Шаблон «State»

Призначення: Шаблон «State» (Стан) дозволяє змінювати логіку роботи об'єктів у випадку зміни їх внутрішнього стану [6]. Наприклад, відсоток нарахованих на картковий рахунок грошей залежить від стану картки: Visa Electron, Classic, Platinum і т.д. Або обсяг послуг, які надані хостинг компанією, змінюється в залежності від обраного тарифного плану (стану членства – бронзовий, срібний або золотий клієнт). Реалізація даного шаблону полягає в наступному: пов'язані зі станом поля, властивості, методи і дії виносяться в окремий загальний інтерфейс (State); кожен стан являє собою окремий клас (ConcreteStateA, ConcreteStateB), які реалізують загальний інтерфейс [6, 8].

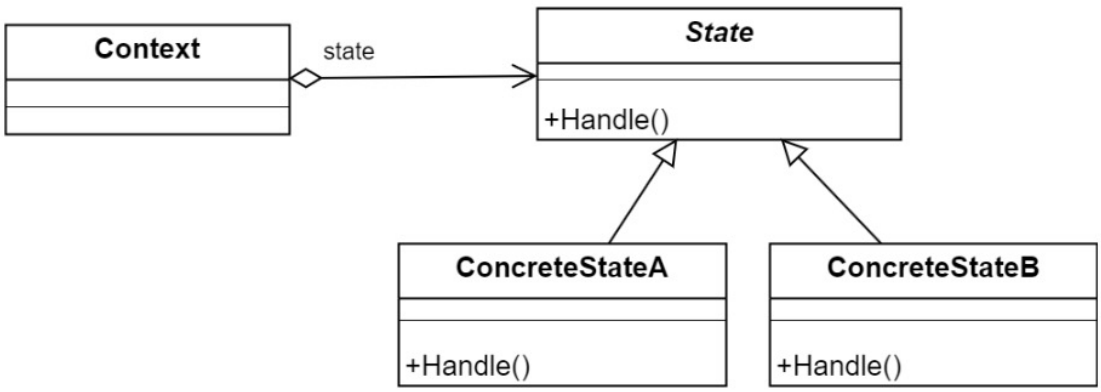


Рисунок 4.4. Структура патерну Стан

Об’єкти, що мають стан (Context), при зміні стану просто записують новий об'єкт в поле state, що призводить до повної зміни поведінки об'єкта.

Це дозволяє легко додавати в майбутньому і обробляти нові стани,

відокремлювати залежні від стану елементи об'єкта в інших об'єктах, і відкрито проводити заміну стану (що має сенс у багатьох випадках).

Проблема: Ви розробляєте систему, яка складається з мобільних клієнтів та центрального сервера.

Для отримання запитів від клієнтів ви створюєте модуль Listener. Але вам потрібно щоб під час старту, поки сервер не запустився Listener не приймав запити від клієнтів, а після команди shutdown, поки сервер зупиняється, Listener вже не приймав запити від клієнтів, але відповіді, якщо вони будуть готові, відправив.

Рішення: Тут явно видно три стани для Listener з різною поведінкою: initializing, open, closing. Тому для вирішення краще за все підходить патерн State.

Визначаємо загальний інтерфейс IListenerState з методами, які по різному працюють в різних станах. Під кожний стан створюємо клас з реалізацією цього інтерфейсе. Контекст Listener при цьому всі виклики буде перенаправляти на об'єкт стану простим делегуванням. При старті він (Listener) буде посилатися на об'єкт стану InitializingState, знаходячись в якому Listener, фактично, буде ігнорувати всі вхідні запити. Після того як система запуститься і завантажить всі базові дані для роботи, Listener буде переключено в робочий стан, наприклад, викликом методу Open(). Після цього Listener буде посилатися на об'єкт OpenState і буде відпрацьовувати всі вхідні повідомлення в звичайному режимі. При запуску виключення системи, Listener буде переключено в стан ClosingState, викликом методу Close().

Переваги та недоліки:

+ Код специфічний для окремого стану реалізується в класі стану.
+ Класи та об’єкти станів можна використовувати з різними контекстами, за рахунок чого збільшується гнучкість системи.
+ Код контексту простіше читати, тому що вся залежна від станів логіка винесена в інші класи.
+ Відносно легко додавати нові стани, головне правильно змінити переходи між станами.
- Клас контекст стає складніше через ускладнений механізм переключення станів.

Хід роботи

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.

Реалізація шаблону проєктування для майбутньої системи

Для реалізації HTTP-сервера оберемо шаблон проєктування State, оскільки він дозволяє ефективно керувати різними станами сервера та відповідною поведінкою для кожного стану. У випадку з HTTP-сервером важливо мати чіткий контроль над його робочими станами для забезпечення коректної обробки запитів на різних етапах життєвого циклу.

```
package server;

import model.HttpRequest;
import model.HttpResponse;
import state.IServerState;
import state.InitializingState;

public class HttpServer {  22 usages  IvanGodPro24
    private IServerState state;  5 usages
    private int port;  2 usages
    public Statistics Statistics = new Statistics();  1 usage

    public HttpServer(int port) {  1 usage  IvanGodPro24
        this.port = port;
        this.state = new InitializingState();
    }

    public void SetState(IServerState newState) {  2 usages  IvanGodPro24
        this.state = newState;
    }

    public void Start() {  1 usage  IvanGodPro24
        state.Start( server: this);
    }

    public void Stop() {  1 usage  IvanGodPro24
        state.Stop( server: this);
    }

    public HttpResponse HandleRequest(HttpRequest req) {  3 usages  IvanGodPro24
        return state.HandleRequest( server: this, req);
    }

    public int getPort() {  1 usage  IvanGodPro24
        return port;
    }
}
```

Рис. 1 – Код класу HTTP-сервер

Клас HTTPServer реалізований за патерном State, що є важливим для HTTP-сервера, оскільки він забезпечує:

1. Чітке розділення поведінки за станами: Завдяки використанню окремих класів для кожного стану (InitializingState, RunningState, ClosingState) сервер може мати різну логіку обробки запитів залежно від поточного стану. Це запобігає обробці запитів під час ініціалізації або завершення роботи сервера.
2. Гнучкість управління станами: Методи переходу між станами (start(), stop(), shutdown()) забезпечують контрольовану зміну поведінки сервера. Наприклад, у стані InitializingState сервер ігнорує вхідні запити, тоді як у стані RunningState - повноцінно їх обробляє.
3. Легкість розширення: Додавання нових станів сервера не вимагає змін в основному класі HTTPServer. Достатньо реалізувати новий клас стану, що відповідає інтерфейсу ServerState, і оновити логіку переходів між станами.

У нашому випадку патерн State забезпечує надійне управління життєвим циклом HTTP-сервера та запобігає потенційним помилкам, пов'язаним з обробкою запитів у неправильний момент. Це значно підвищує стабільність та безпеку роботи сервера, особливо в критичних ситуаціях, таких як запуск або зупинка системи.

Зображення структури шаблону

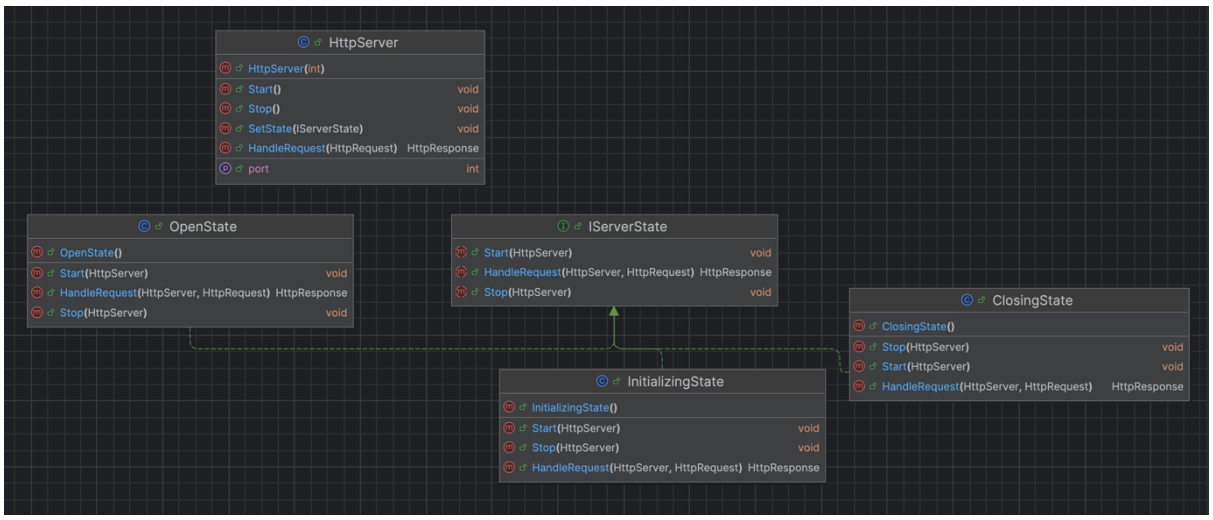


Рис. 2 – Структура шаблону State

Структура реалізації:

- IServerState - інтерфейс, що визначає методи для різних станів
- InitializingState - стан ініціалізації (запити не обробляються)
- OpenState - робочий стан (повноцінна обробка запитів)
- ClosingState - стан завершення (обробляються лише завершуючі операції)
- HTTPServer - контекст, що делегує обробку запитів поточному стану

Такий підхід дозволяє легко керувати складними переходами між станами сервера та забезпечує чітке розділення відповідальності між різними фазами його роботи.

Посилання на репозиторій: https://github.com/IvanGodPro24/trpz

Висновки

Висновки: під час виконання лабораторної роботи, ми вивчили структуру шаблонів «Singleton», «Iterator», «Proxy», «State», «Strategy» та навчилися застосовувати шаблон «State» в реалізації програмної системи.

Питання до лабораторної роботи

1. Що таке шаблон проєктування?

Шаблон проєктування (Design Pattern) – це перевірене рішення типової задачі проєктування програмного забезпечення, що виникає у певному контексті.

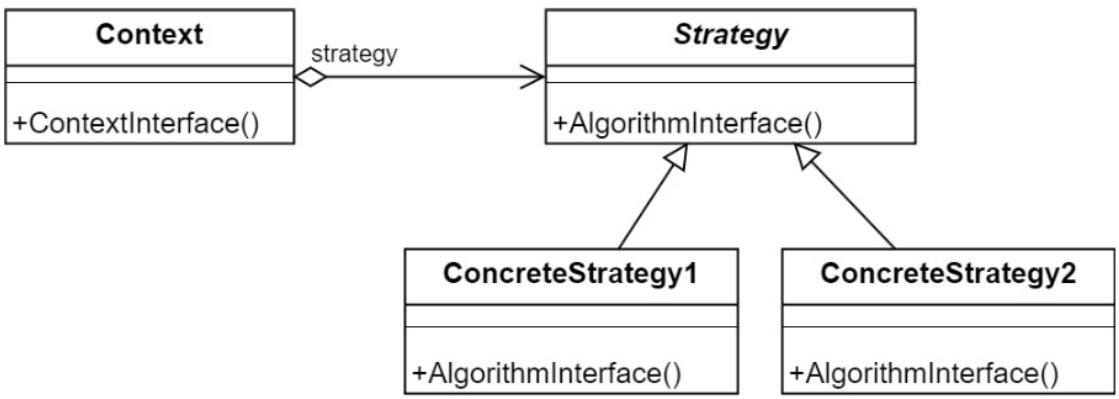
2. Навіщо використовувати шаблони проєктування?

1. Забезпечують повторне використання коду.
2. Полегшують підтримку та розширення системи.
3. Покращують зрозумілість і структуру коду.
4. Допомагають уникнути типових помилок.

3. Яке призначення шаблону «Стратегія»?

Шаблон «Стратегія» дозволяє змінювати алгоритм роботи об’єкта під час виконання програми, інкапсулюючи різні алгоритми в окремі класи.

4. Нарисуйте структуру шаблону «Стратегія».



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

Context – використовує об’єкт Strategy для виконання алгоритму.

Strategy – абстрактний клас або інтерфейс для алгоритмів.

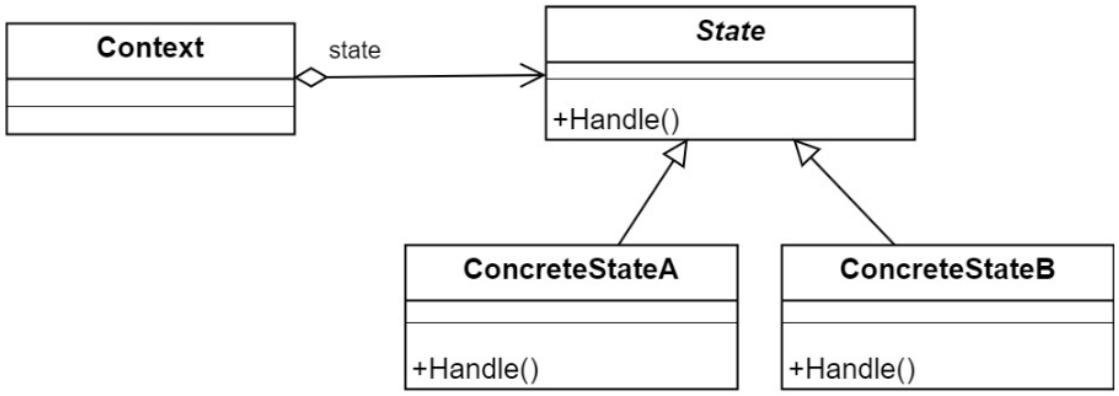
ConcreteStrategy – конкретна реалізація алгоритму.

Context делегує виконання алгоритму об’єкту Strategy, який можна змінювати динамічно.

6. Яке призначення шаблону «Стан»?

Шаблон «Стан» дозволяє змінювати поведінку об’єкта залежно від його внутрішнього стану, не змінюючи сам клас об’єкта.

7. Нарисуйте структуру шаблону «Стан».



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

Context – містить посилання на поточний стан та делегує йому поведінку.

State – абстрактний клас або інтерфейс стану.

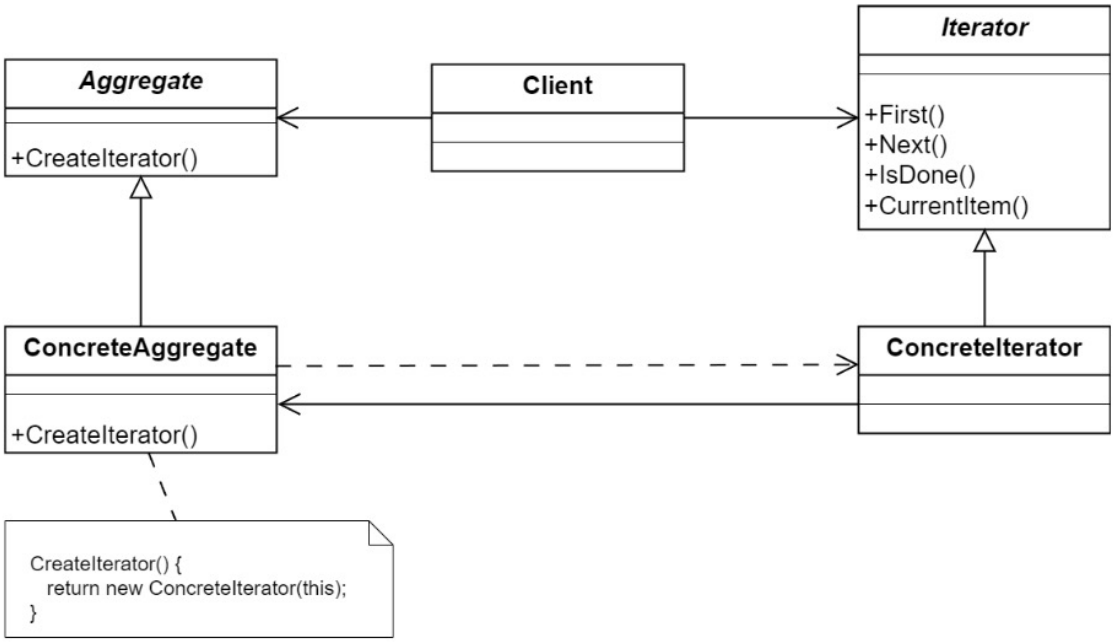
ConcreteState – реалізує конкретну поведінку для певного стану.

Context змінює стан об’єкта, а ConcreteState визначає поведінку в цьому стані.

9. Яке призначення шаблону «Ітератор»?

Шаблон «Ітератор» дозволяє послідовно отримувати доступ до елементів колекції без розкриття її внутрішньої структури.

10. Нарисуйте структуру шаблону «Ітератор».



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

ConcreteAggregate – реалізація колекції.

Iterator – інтерфейс ітератора.

ConcreteIterator – реалізує послідовний доступ до елементів ConcreteAggregate.

Клас ConcreteIterator використовує ConcreteAggregate для обходу елементів.

12. В чому полягає ідея шаблона «Одинак»?

Шаблон «Одинак» (Singleton) забезпечує, що клас матиме лише один екземпляр, і надає глобальну точку доступу до нього.

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

Створює глобальний стан, що ускладнює тестування.

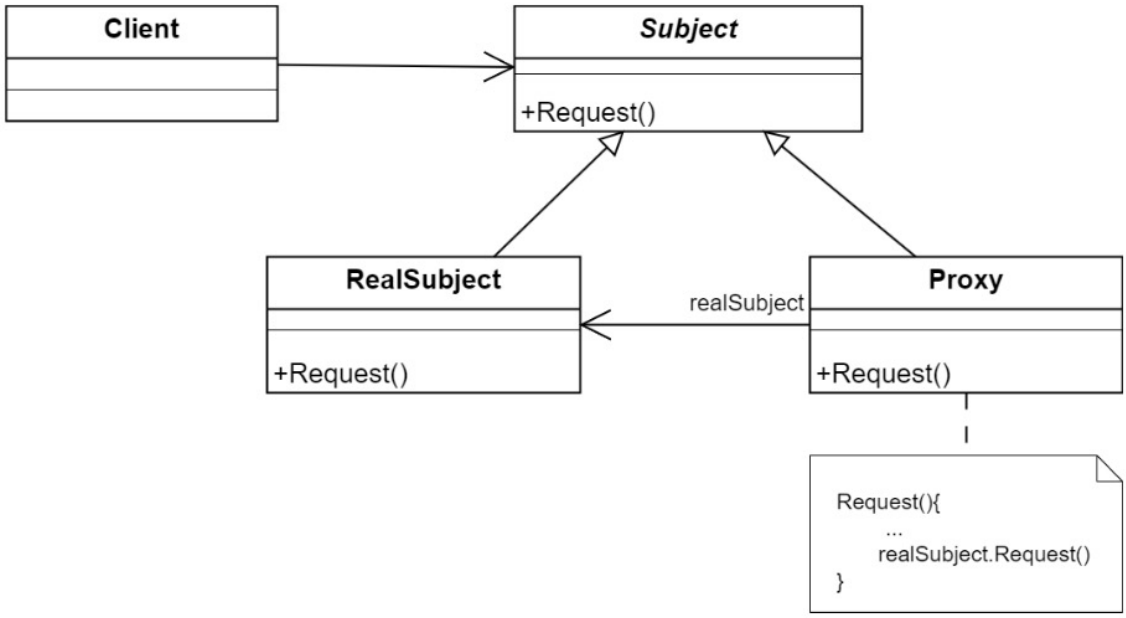
Порушує принцип інверсії залежностей.

Може призводити до жорстких залежностей між класами.

14. Яке призначення шаблону «Проксі»?

Шаблон «Проксі» забезпечує заміну об’єкта замінником, який контролює доступ до справжнього об’єкта.

15. Нарисуйте структуру шаблону «Проксі».



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

Subject – інтерфейс для RealSubject і Proxy.

RealSubject – реальний об’єкт, що виконує роботу.

Proxy – контролює доступ до RealSubject.
Proxy реалізує Subject і делегує виклики методів RealSubject, при необхідності додаючи додаткову поведінку.