Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут» імені Ігоря Сікорського
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота № 4

із дисципліни: «Технології розроблення програмного забезпечення»

на тему: «Вступ до паттернів проектування.**»**

Виконав:
студент групи ІА-34
Вінницький Г.Р.
Перевірил:
Мягкий Михайло Юрійович

Київ-2025

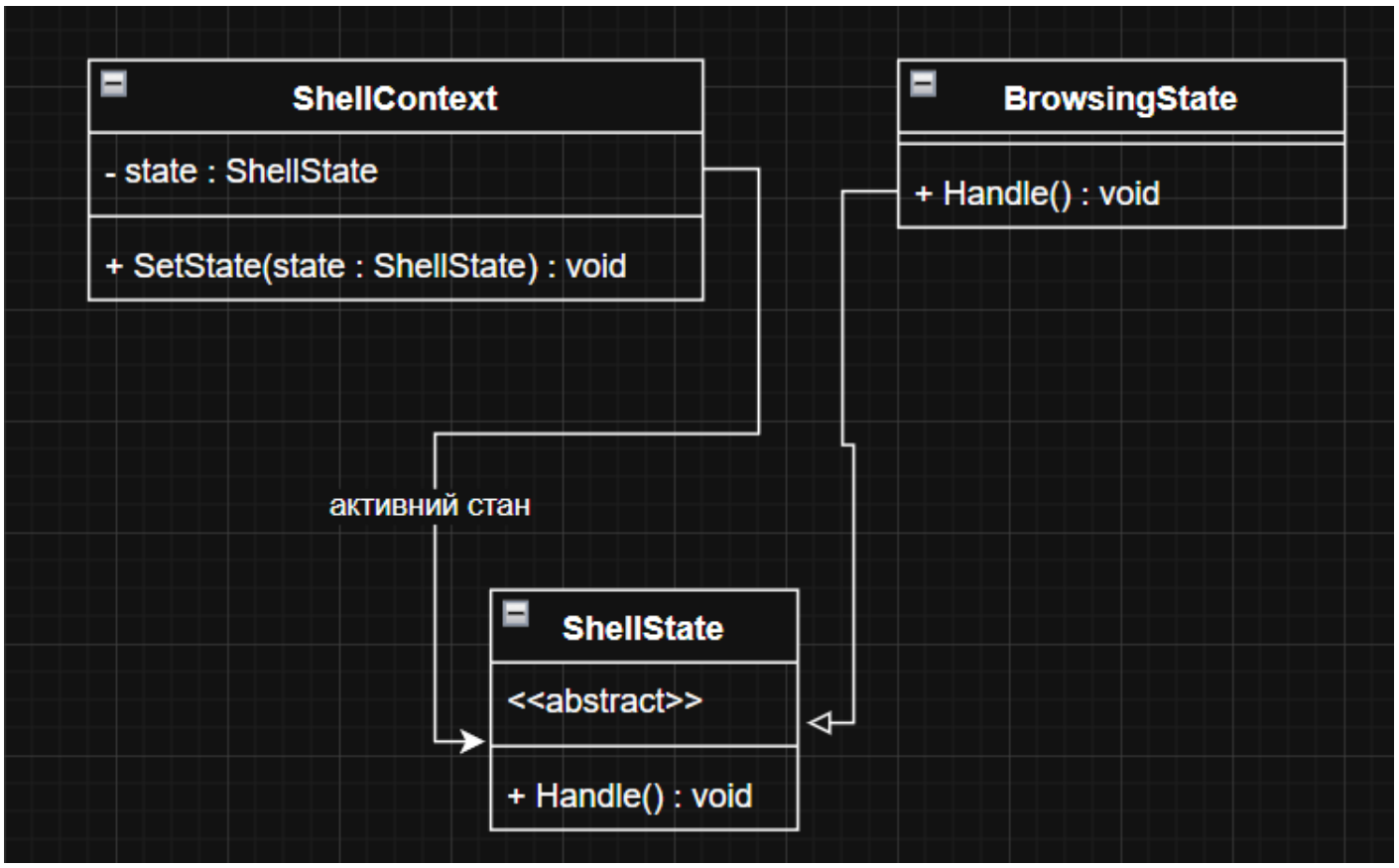
Мета: Вивчити структуру шаблонів «Singleton», «Iterator», «Proxy», «State», «Strategy» та навчитися застосовувати їх в реалізації програмної системи.

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.

18. **Shell (total commander)** (state, prototype, factory method, template method, interpreter, client-server)

Оболонка повинна вміти виконувати основні дії в системі – перегляд файлів папок в файлової системі, перемикання між дисками, копіювання, видалення, переміщення об'єктів, пошук.



На діаграмі зображено фактичну реалізацію патерну **State**, використаного у курсовій роботі Shell Total Commander. Патерн застосовується для розділення поведінки системи залежно від активного режиму роботи.

ShellState — абстрактний клас, який визначає метод Handle().

Він представляє інтерфейс для всіх можливих станів оболонки.

BrowsingState — конкретний стан, що використовується у проєкті.

У цьому стані система обробляє стандартні команди: перегляд каталогів, відкриття, копіювання, видалення файлів тощо.

ShellContext — клас контексту, який містить активний стан.

Усі команди користувача передаються поточному стану через метод Execute().

Завдяки цьому контекст не містить умовних конструкцій (if-else), а делегує поведінку класу стану, що відповідає структурі патерну **State**.

Патерн **State** реально інтегрований у проєкт **Shell Total Commander**

Реалізація шаблону State

У проєкті Shell Total Commander шаблон проектування State (Стан) використано для керування режимом роботи файлової оболонки. Цей патерн дозволяє змінювати поведінку системи залежно від її внутрішнього стану без використання розгалужень if-else або switch. У застосунку активний стан контролює спосіб обробки введених користувачем команд та визначає логіку їх виконання.

У системі визначено три ключові елементи патерну State:

1. Абстрактний клас ShellState

Він виступає базовим типом для всіх можливих станів оболонки та визначає спільний метод

2. Конкретний стан BrowsingState

Це єдиний стан, реалізований у курсовому проєкті. Він відповідає режиму стандартної роботи файлової оболонки — перегляду каталогів, відкриття папок та виконання файлових операцій.

ls, cd, copy, move, delete, search, open.

3. Клас контексту ShellContext

Контекст зберігає активний стан та делегує йому обробку введених команд

На початку роботи застосунку активним станом є BrowsingState.
Якщо в майбутньому будуть додані інші режими (наприклад, пошук, перегляд логів, режим адміністратора), їх достатньо буде оформити як нові класи-нащадки ShellState та перемикати через метод SetState().

Переваги такої реалізації
Застосування патерну State у проєкті дозволило:

1. відокремити логіку станів від логіки самого контексту;
2. спростити підтримку та розширення системи;
3. уникнути великих конструкцій розгалуження;
4. забезпечити модульну й чисту архітектуру;
5. легко додавати нові режими роботи без зміни існуючих класів.

Висновок: у ході виконання роботи було досліджено та реалізовано патерн State на основі реального проєкту Shell Total Commander. Патерн дозволив відокремити поведінку файлової системи від її контексту та структурувати обробку команд через активний стан. Реалізація включає абстракцію ShellState, конкретний стан BrowsingState та контекст ShellContext, який делегує обробку команд активному стану. Застосування патерну State зробило систему більш гнучкою, розширюваною та простою для підтримки.

Питання до лабораторної роботи:

1. Що таке шаблон проєктування?

Шаблон проєктування (design pattern) — це типове, перевірене рішення поширеної задачі в об’єктно-орієнтованому проєктуванні.

2. Навіщо використовувати шаблони проєктування?

Шаблони використовують для:

1) покращення структури коду;
2) повторного використання рішень;
3) зменшення зв’язності;
4) спрощення розширення та підтримки системи;
5) зменшення кількості помилок при проєктуванні.

3. Яке призначення шаблону «Стратегія»?

Шаблон Стратегія визначає сімейство алгоритмів, інкапсулює їх і дозволяє замінювати один алгоритм іншим під час роботи програми.

4. Нарисуйте структуру шаблону «Стратегія».

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

1) Context — використовує стратегію.
2) Strategy — інтерфейс алгоритму.
3) ConcreteStrategyA/B — конкретні реалізації алгоритмів.

Взаємодія:

Context має посилання на Strategy і викликає її метод без знання деталей реалізації.

6. Яке призначення шаблону «Стан»?

Шаблон Стан (State) дозволяє об’єкту змінювати свою поведінку залежно від внутрішнього стану.

Зовні здається, що змінюється клас об’єкта.

7. Нарисуйте структуру шаблону «Стан».

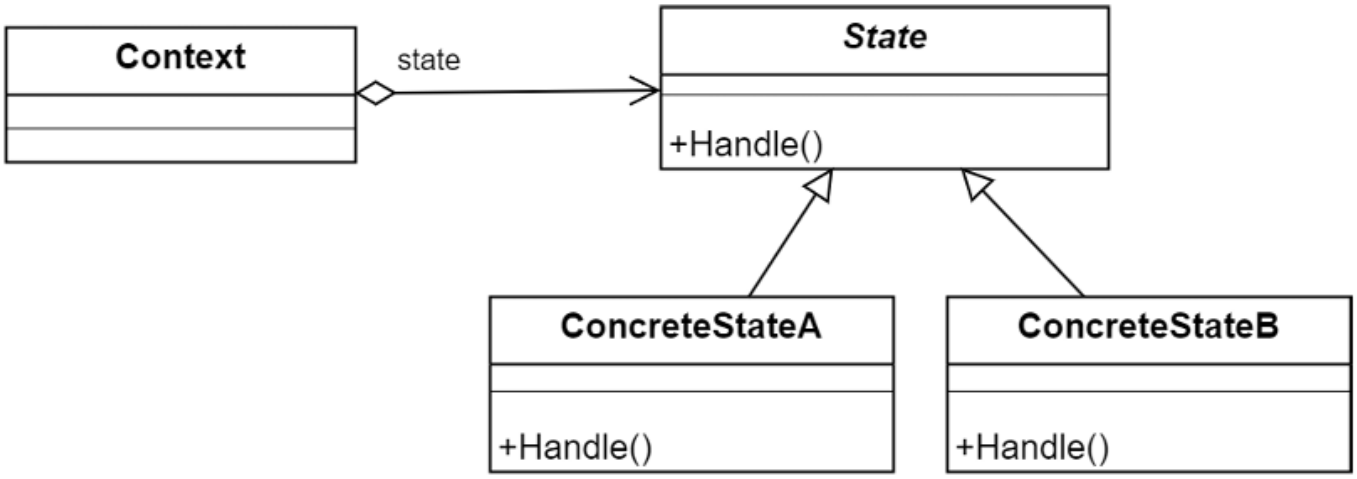


Рисунок 4.4. Структура патерну Стан

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

1) Context — містить поточний стан і делегує йому роботу.
2) State — інтерфейс для станів.
3) ConcreteStateA/B — різні стани.

Взаємодія:

Context викликає метод State; ConcreteState може змінити стан Context.

9. Яке призначення шаблону «Ітератор»?

Шаблон Ітератор (Iterator) надає стандартний спосіб послідовного доступу до елементів колекції без розкриття її внутрішньої структури.

10. Нарисуйте структуру шаблону «Ітератор».

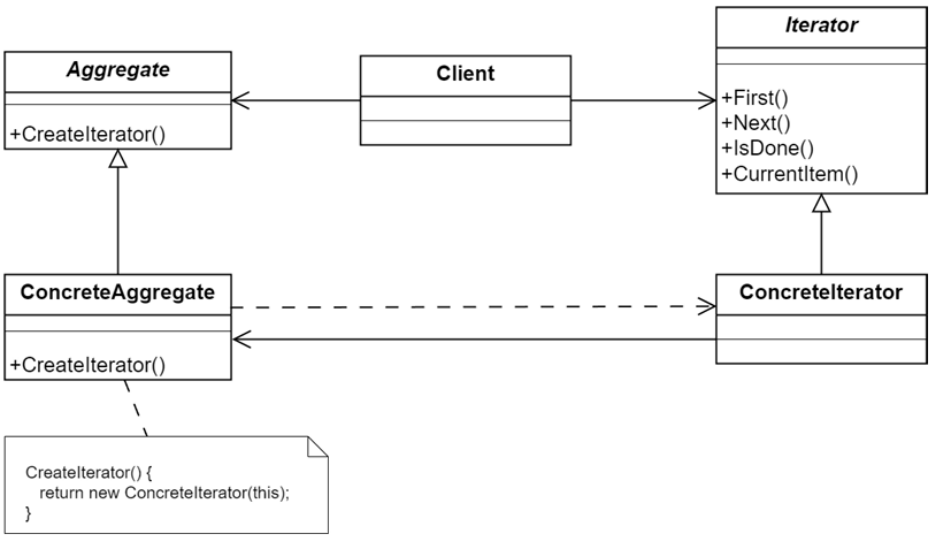


Рисунок 4.2. Структура патерну Ітератор

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

1) Aggregate — інтерфейс колекції.
2) ConcreteAggregate — конкретна колекція.
3) Iterator — інтерфейс ітератора.
4) ConcreteIterator — конкретний ітератор.

Взаємодія:

ConcreteIterator отримує доступ до ConcreteAggregate і забезпечує послідовний обхід елементів.

12. В чому полягає ідея шаблона «Одинак»?

Шаблон Одинак (Singleton) гарантує, що клас має лише один екземпляр і надає глобальну точку доступу до нього.

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

Тому що він:

1) порушує принципи SOLID (зокрема Single Responsibility);
2) створює глобальний стан;
3) ускладнює тестування;
4) призводить до прихованої залежності між класами;
5) часто використовується надмірно.

14. Яке призначення шаблону «Проксі»?

Шаблон Проксі (Proxy) створює сурогатний об’єкт, який контролює доступ до реального об’єкта (наприклад, додає кешування, логування, доступ за правами, відкладене створення).

15. Нарисуйте структуру шаблону «Проксі».

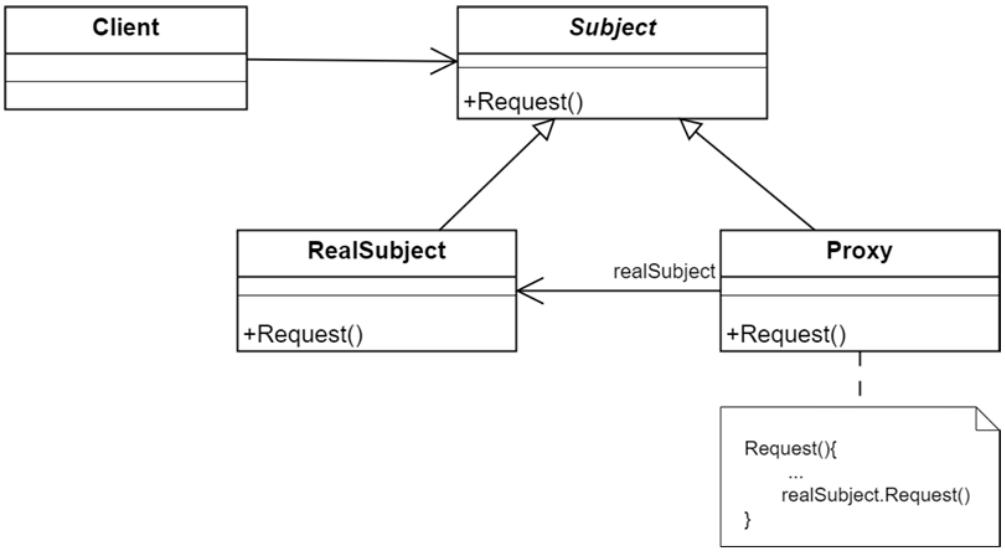


Рисунок 4.3. Структура патерну Proxy

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

1) Subject — інтерфейс реального об’єкта.
2) RealSubject — об’єкт, до якого потрібно надати доступ.
3) Proxy — містить посилання на RealSubject і контролює доступ до нього.

Взаємодія:

Клієнт працює з Proxy так само, як із RealSubject, але Proxy може перехоплювати виклики й додавати додаткову логіку.