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 № 5

із дисципліни: «Технології розроблення програмного забезпечення»

на тему: «Патерни проектування.**»**

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

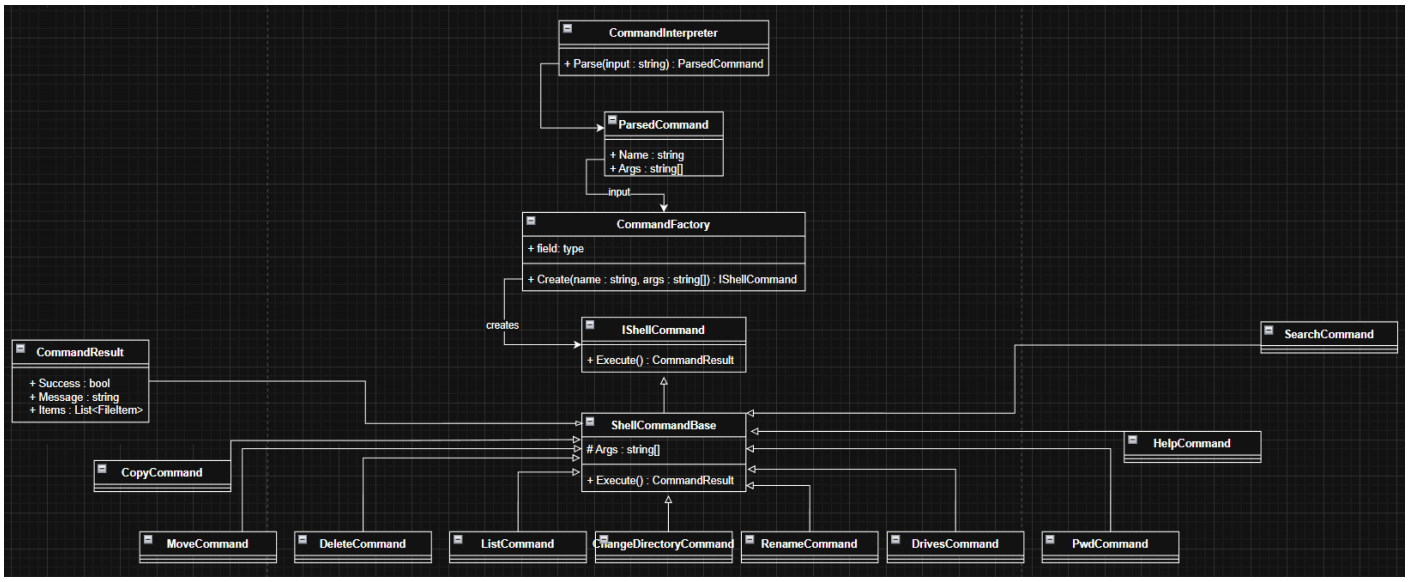
Київ-2025

Мета: Вивчити структуру шаблонів «Adapter», «Builder», «Command», «Chain of responsibility», «Prototype» та навчитися застосовувати їх в реалізації програмної системи.

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.



Опис діаграми класів патерну Command

1. Призначення патерну

Патерн **Command** використовується для інкапсуляції операцій у вигляді окремих об’єктів.

Кожна команда відповідає за одну дію файлової системи: копіювання, переміщення, видалення, перегляд каталогу, пошук тощо.

Це дозволяє:

1) розширювати систему новими командами без зміни існуючого коду;
2) відокремити логіку обробки команд від логіки введення;
3) легко інтегрувати серверну та локальну обробку команд;
4) уніфіковано логувати результати команд.

2. Опис елементів діаграми

IShellCommand

Інтерфейс, який визначає метод:

Execute() : CommandResult — виконує команду й повертає результат.

Це контракт для всіх команд системи.

ShellCommandBase (abstract)

Абстрактний клас, який:

реалізує інтерфейс IShellCommand;

містить спільні поля (Args) та базову логіку;

є основою для всіх конкретних команд.

CommandResult

Об’єкт-результат виконання команди:

Success — успіх/помилка,

Message — текстовий опис,

Items — список знайдених файлів/каталогів.

Команди повертають тільки цей тип.

CommandInterpreter

Відповідає за:

розбір текстової команди користувача (Parse),

створення структури ParsedCommand.

Використовується для інтеграції з UI.

ParsedCommand

Містить:

Name — назву команди,

Args — масив аргументів.

Передається у фабрику команд.

CommandFactory

Фабрика створює об’єкти-команди на основі імені:

Create(name, args) — повертає об’єкт IShellCommand.

Це реалізація патерну **Factory Method** у поєднанні з **Prototype**.

Конкретні команди

Кожна конкретна команда успадковує ShellCommandBase:

CopyCommand

MoveCommand

DeleteCommand

ListCommand

ChangeDirectoryCommand

SearchCommand

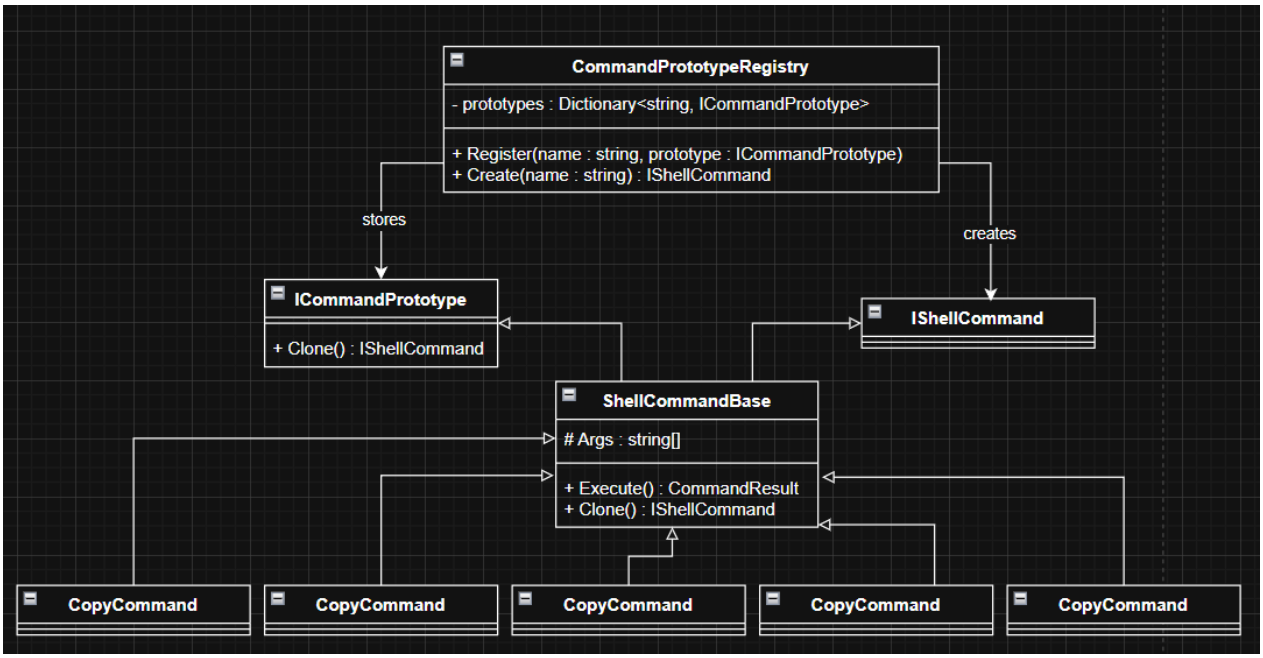
HelpCommand

RenameCommand

PwdCommand

DrivesCommand

Кожна з них реалізує власну поведінку у методі Execute.



Опис діаграми класів патерну Prototype

1. Призначення патерну

Патерн **Prototype** використовується у системі для створення об’єктів команд через їх клонування.

Замість створення команд через new, фабрика бере *вже зареєстровані прототипи команд* та створює їх копії.

Це дозволяє:

уникнути великої кількості if / switch-case,

централізовано керувати доступними командами,

швидко додавати нові команди,

підвищити гнучкість фабрики команд.

2. Опис елементів діаграми

ICommandPrototype

Інтерфейс, що визначає метод:

Clone() : IShellCommand

Будь-яка команда може бути прототипом і вміє створювати свій клон.

CommandPrototypeRegistry

Реєстр прототипів, який:

зберігає словник "ім'я команди" → прототип,

має метод Register(name, prototype) для реєстрації,

має метод Create(name) для створення нового екземпляру через клонування.

Це ядро всього патерну Prototype.

ShellCommandBase

Абстрактна команда реалізує два інтерфейси:

IShellCommand — виконання команди,

ICommandPrototype — клонування прототипу.

Це означає, що **будь-яка команда може бути і прототипом, і виконавцем**.

Конкретні команди

Кожна з команд (CopyCommand, MoveCommand, DeleteCommand, ListCommand, SearchCommand тощо):

успадковує ShellCommandBase,

автоматично стає прототипом,

може бути зареєстрована в CommandPrototypeRegistry.

Посилання: https://github.com/gervinn/Shell-total-commander-/tree/main/ShellTotalCommander1/Shell

Висновок: У ході виконання лабораторної роботи було опрацьовано та практично реалізовано ключові шаблони проектування, зокрема Command та Prototype, на прикладі програмної системи Shell Total Commander. Реалізація цих патернів дозволила підвищити структурованість, гнучкість і розширюваність програмного забезпечення.

Шаблон Command дозволив інкапсулювати всі файлові операції (копіювання, переміщення, видалення, ренейм, пошук, перегляд каталогів) у вигляді незалежних об’єктів. Це забезпечило чітке розділення відповідальностей, спростило реалізацію локального та серверного виконання команд, а також надало можливість легко додавати нові дії без зміни існуючої архітектури.

Патерн Prototype, реалізований через реєстр прототипів команд, уможливив створення нових екземплярів команд шляхом клонування. Це дозволило відмовитись від жорстко закодованих конструкцій типу switch-case, спростило модифікацію функціоналу та надало системі додаткову гнучкість.

Мета лабораторної роботи була повністю досягнута. Отримані знання та практичні навички засвідчують доцільність використання шаблонів проектування для побудови масштабованих, зручних у підтримці та якісно спроєктованих програмних систем.

Питання до лабораторної роботи:

1. Яке призначення шаблону «Адаптер»?

Шаблон «Адаптер» призначений для узгодження інтерфейсів двох класів, які зазвичай несумісні. Він дозволяє одному класу використовувати інший без зміни їхнього коду.

2. Нарисуйте структуру шаблону «Адаптер».

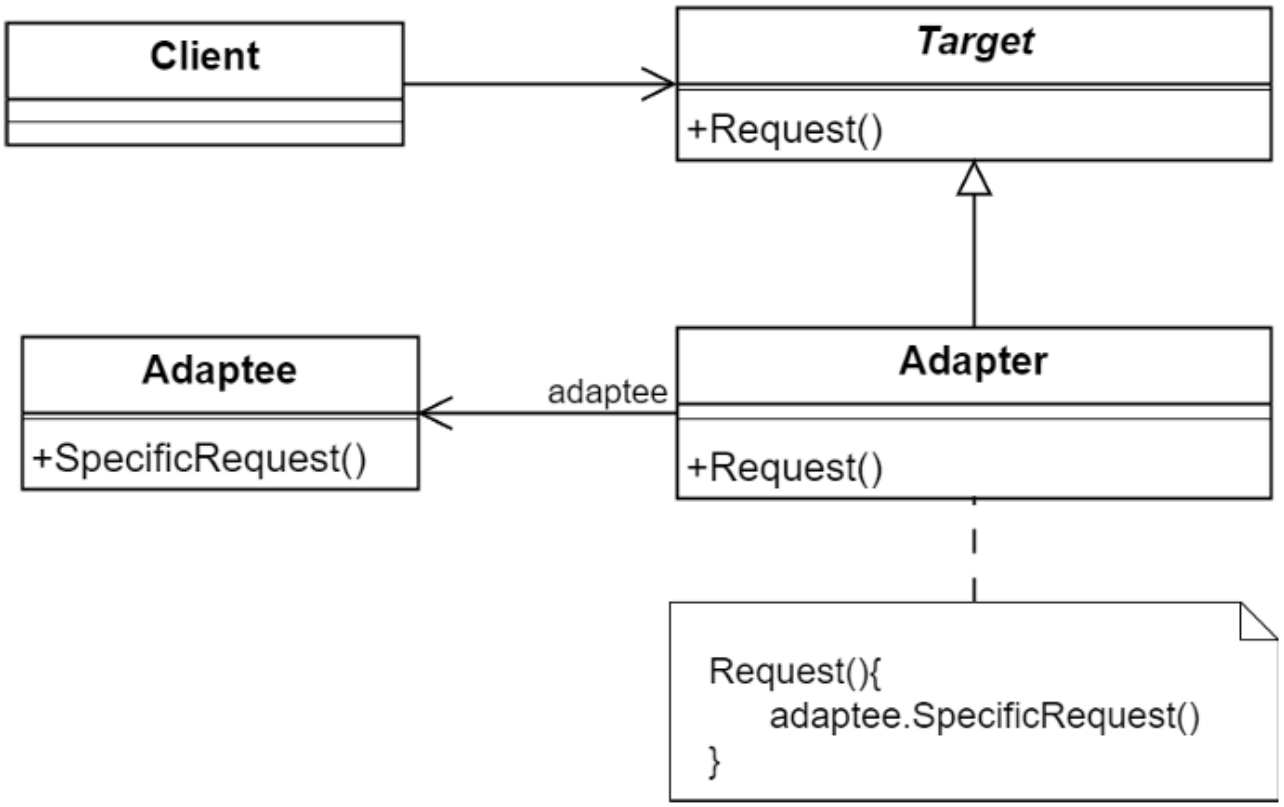


Рисунок 5.1. Структура патерну Адаптер на рівні об’єктів

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

1) Client — працює через інтерфейс Target.
2) Target — інтерфейс, який очікує Client.
3) Adaptee — клас з іншим інтерфейсом.
4) Adapter — клас, який адаптує Adaptee до Target.

Взаємодія: Client звертається до Adapter, а Adapter викликає методи Adaptee.

4. Яка різниця між реалізацією «Адаптера» на рівні об’єктів та на рівні класів?

1) На рівні об’єктів адаптер містить екземпляр Adaptee (композиція). Гнучкіший варіант.

2) На рівні класів адаптер успадковується одночасно від Target та Adaptee (потрібне множинне успадкування). Рідкісний варіант.

5. Яке призначення шаблону «Будівельник»?

Шаблон «Будівельник» дає змогу поетапно створювати складні об’єкти, розділяючи процес побудови та кінцевий вигляд об’єкта.

6. Нарисуйте структуру шаблону «Будівельник».

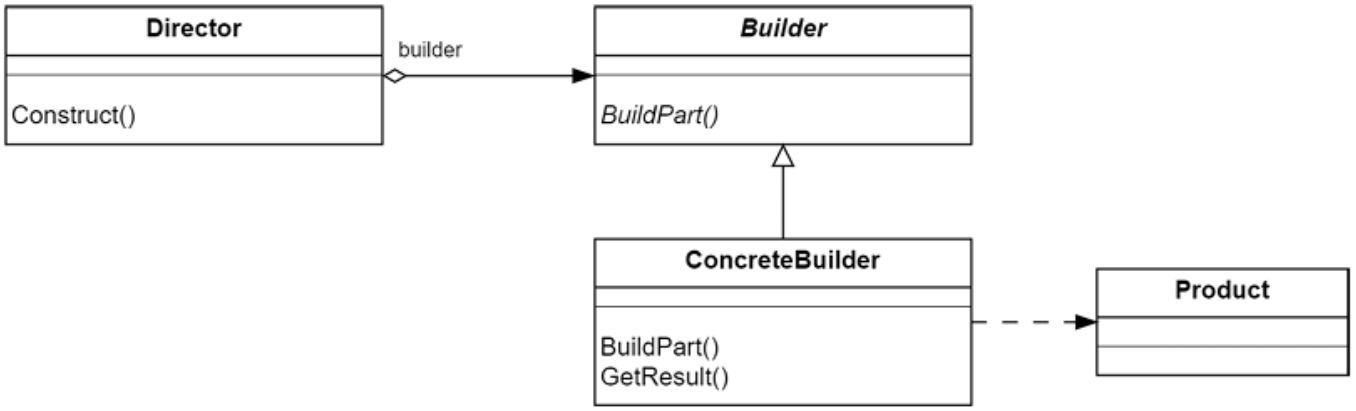


Рисунок 5.2. Структура патерну Builder

7. Які класи входять в шаблон «Будівельник», та яка між ними взаємодія?

1) Director — визначає порядок побудови.
2) Builder — описує методи створення частин продукту.
3) ConcreteBuilder — реалізує ці кроки та формує продукт.
4) Product — кінцевий об’єкт.

Взаємодія: Director викликає методи Builder, який будує об’єкт Product.

8. У яких випадках варто застосовувати шаблон «Будівельник»?

1) Коли об’єкт складається з багатьох частин.
2) Коли потрібні різні варіанти одного і того ж об’єкта.
3) Коли потрібно контролювати процес створення об’єкта.
4) Коли конструктор мав би надто багато параметрів.

9. Яке призначення шаблону «Команда»?

Шаблон «Команда» перетворює дію у об’єкт. Це дозволяє зберігати дії в історії, скасувати, повторювати, ставити в чергу та логувати.

10. Нарисуйте структуру шаблону «Команда».

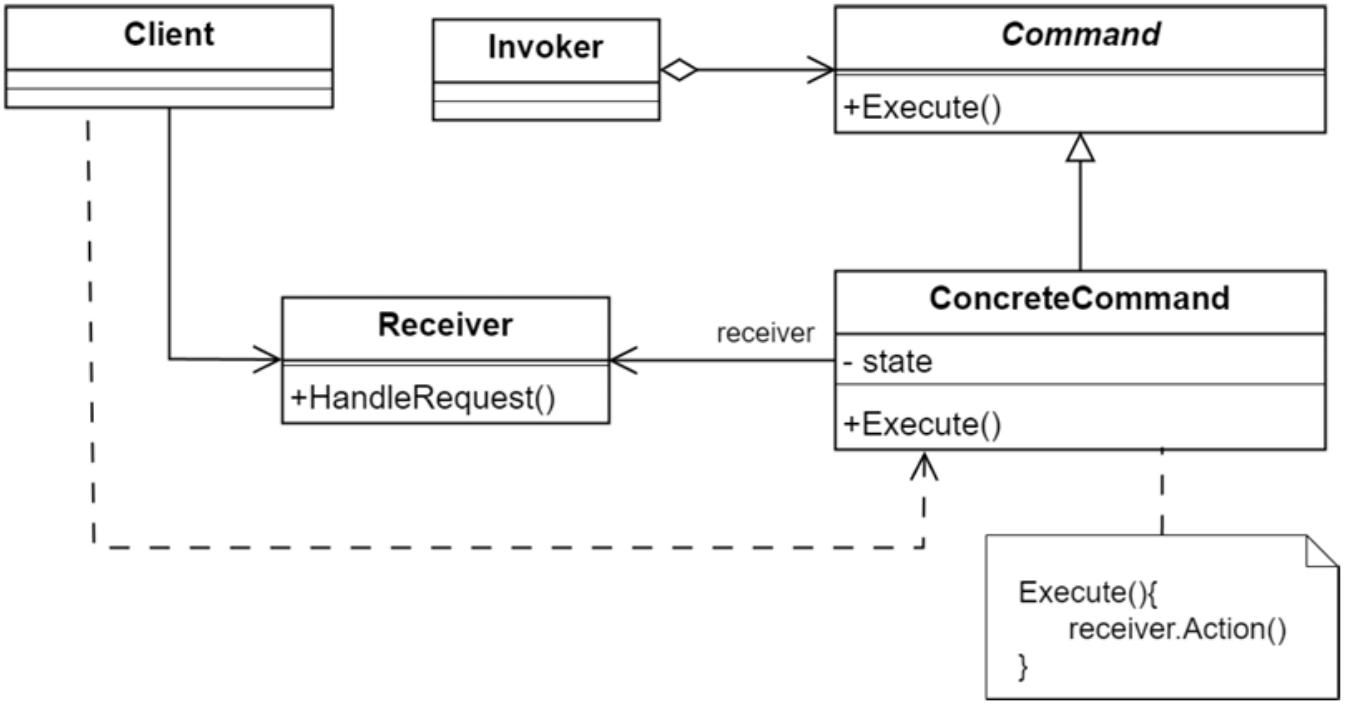


Рисунок 5.3. Структура патерну Команда

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

1) Command — інтерфейс з методом execute().
2) ConcreteCommand — реалізує execute(), викликаючи певну дію у Receiver.
3) Receiver — виконує реальну операцію.
4) Invoker — зберігає команду та викликає її.
5) Client — створює команду і передає Invoker.

Взаємодія: Invoker викликає execute(), а команда звертається до Receiver.

12. Розкажіть як працює шаблон «Команда».

Кожна дія оформляється як об’єкт-команда. Клієнт створює команду, передає її ініціатору. Коли треба виконати дію — ініціатор не знає деталей, він просто викликає execute(). Це дозволяє зберігати історію дій, робити скасування, логування тощо.

13. Яке призначення шаблону «Прототип»?

Шаблон «Прототип» дозволяє створювати нові об’єкти шляхом копіювання існуючих, без використання конструктора.

14. Нарисуйте структуру шаблону «Прототип».

15. Які класи входять в шаблон «Прототип», та яка між ними взаємодія?

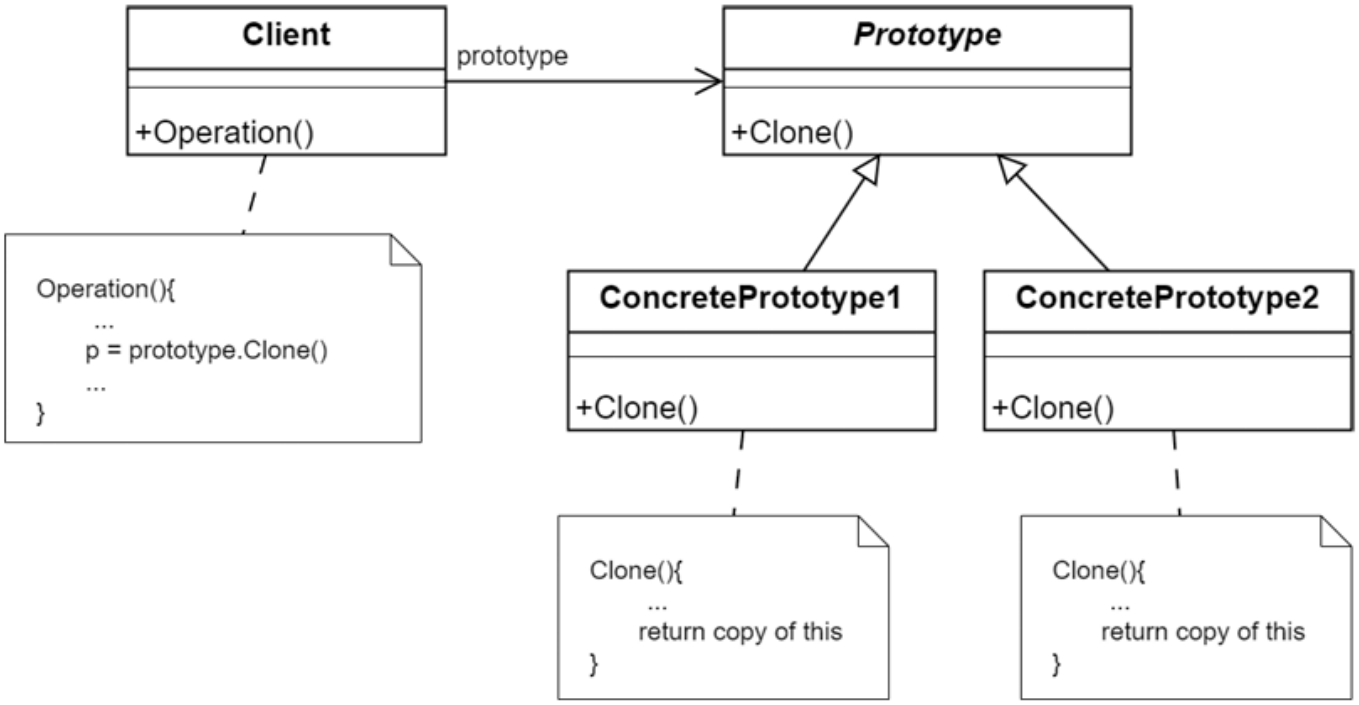


Рисунок 5.5. Структура патерну «Прототип»

16. Які можна привести приклади використання шаблону «Ланцюжок відповідальності»?

1) Обробка HTTP-запитів у веб-фреймворках (middleware, фільтри).
2) Логування на різних рівнях (debug → info → warning → error).
3) Передача подій у графічних інтерфейсах.
4) Валідація даних, де кожен валідатор перевіряє своє правило.
5) Системи техпідтримки з рівнями ескалації.
6) Обробка команд у терміналі, де кожен handler перевіряє, чи може виконати команду.