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 № 6

із дисципліни: «Технології розроблення програмного забезпечення»

на тему: «Патерни проектування.**»**

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

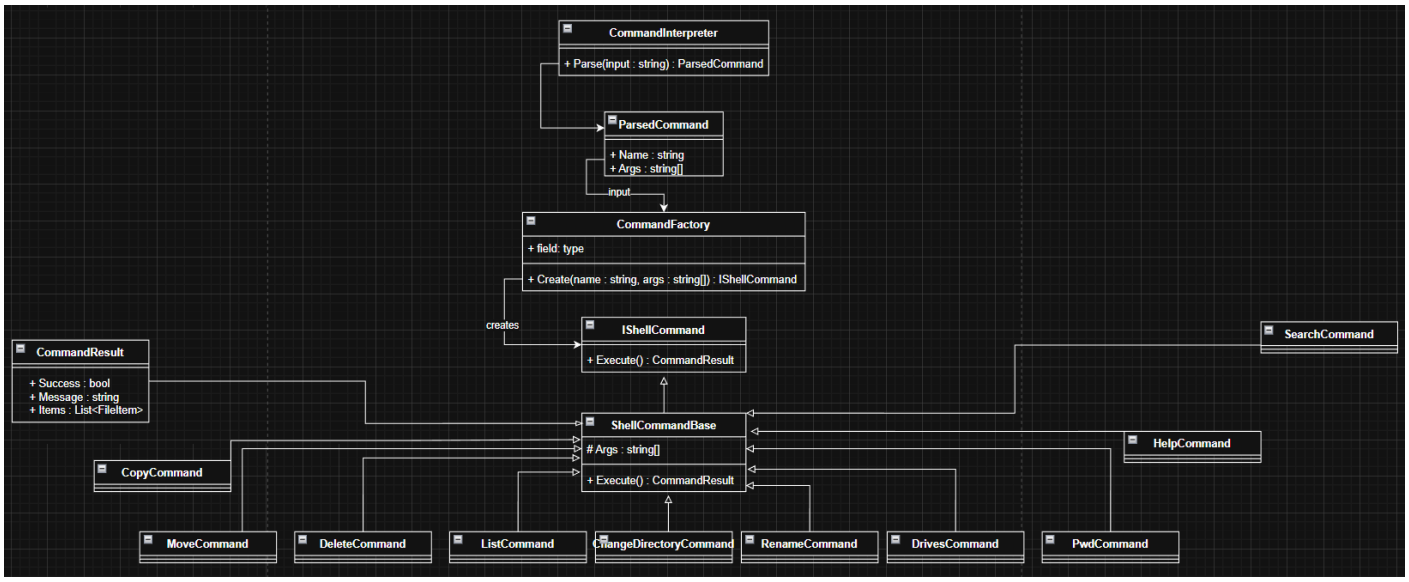
Київ-2025

Мета: Вивчити структуру шаблонів «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator» 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.



2. Опис діаграми

Factory Method — призначення

Патерн Factory Method дозволяє інкапсулювати створення об’єктів у спеціальний клас-фабрику.

У системі це дає можливість:

створювати команди за ім’ям (copy, move, delete, …);

не використовувати switch-case або великі if-else;

розширювати систему новими командами без зміни фабрики.

Учасники діаграми

1. IShellCommand

Інтерфейс усіх команд системи.

Містить метод:

Execute() : CommandResult

2. ShellCommandBase (abstract)

Базовий клас, який реалізує:

загальну логіку команд,

спільні поля, такі як Args.

Всі команди успадковують саме його.

3. Конкретні команди

Кожна команда — окремий клас:

1) CopyCommand
2) MoveCommand
3) DeleteCommand

4) ListCommand
5) SearchCommand
6) RenameCommand
7) ChangeDirectoryCommand
8) DrivesCommand
9) PwdCommand
10) HelpCommand

Кожна команда реалізує власну поведінку через Execute().

4. CommandFactory — творець (Creator)

Має метод:

Create(name, args) : IShellCommand

Фабрика:

отримує ім’я команди,

створює відповідний об’єкт,

повертає команду у вигляді інтерфейсу.

5. CommandInterpreter

Не створює команд сам, лише розбирає текст:

"copy C:\a.txt D:\b.txt" → ParsedCommand → Factory

6. ParsedCommand

Містить:

1) ім’я команди,
2) аргументи.

Саме його дані передаються фабриці.

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

Висновок: У ході виконання лабораторної роботи було розглянуто та реалізовано шаблон проектування Factory Method на прикладі файлового менеджера Shell Total Commander. У системі фабрика команд (CommandFactory) відповідає за створення об’єктів-команд відповідно до введеної користувачем текстової інструкції.

Завдяки застосуванню патерну:

1) система отримала гнучку та розширювану архітектуру;
2) логіка створення об’єктів була відокремлена від логіки їх використання;

3) спростилося додавання нових команд;
4) було досягнуто високого рівня модульності та слабкого зв’язку між компонентами.

Реалізація Factory Method у проекті повністю відповідає класичній структурі патерну та демонструє його доцільність у розробці масштабованих програмних систем.

Питання до лабораторної роботи:

1. Яке призначення шаблону «Абстрактна фабрика»?

Шаблон «Абстрактна фабрика» використовується для створення груп взаємопов’язаних об’єктів, не вказуючи їх конкретних класів. Дозволяє легко змінювати сімейства продуктів.

2. Нарисуйте структуру шаблону «Абстрактна фабрика».

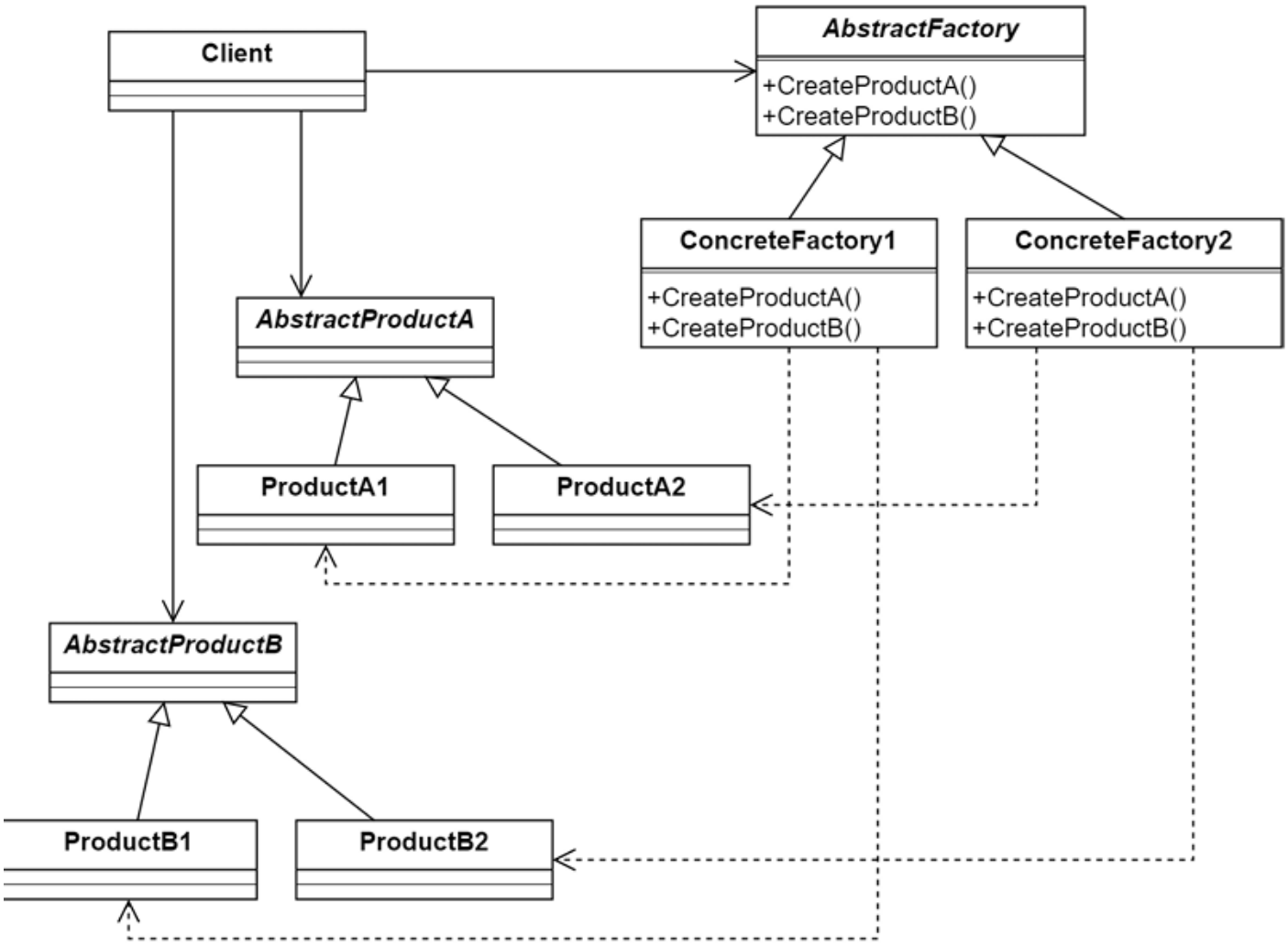


Рисунок 6.1. Структура патерну «Абстрактна фабрика»

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

1) AbstractFactory — визначає інтерфейс для створення родини продуктів.
2) ConcreteFactory — створює конкретні продукти однієї сім'ї.
3) AbstractProduct — інтерфейс продукту.
4) ConcreteProduct — конкретний продукт.
5) Client — працює з фабрикою через AbstractFactory.

Взаємодія: Client використовує завод для створення об'єктів, не знаючи їх конкретних класів.

4. Яке призначення шаблону «Фабричний метод»?

Шаблон «Фабричний метод» дозволяє створювати об'єкти через метод, який повертає інтерфейс або абстрактний тип. Дозволяє підкласам визначити, який саме об'єкт створювати.

5. Нарисуйте структуру шаблону «Фабричний метод».

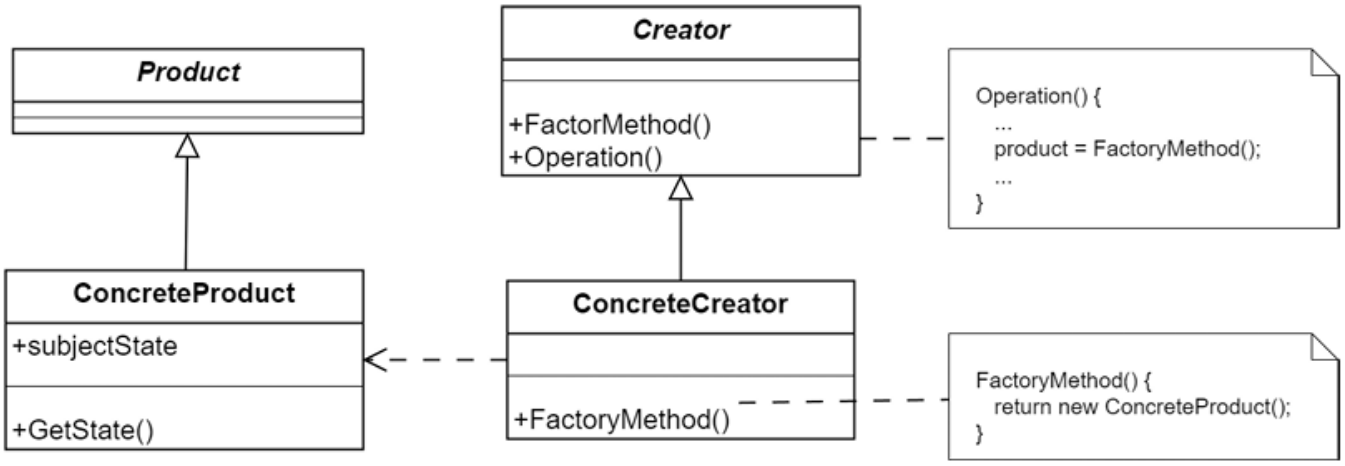


Рисунок 6.2. Структура патерну «Фабричний Метод»

6. Які класи входять в шаблон «Фабричний метод», та яка між ними взаємодія?

1) Creator — містить фабричний метод.
2) ConcreteCreator — перевизначає фабричний метод для створення конкретного продукту.
3) Product — абстрактний продукт.
4) ConcreteProduct — конкретний продукт.

Взаємодія: Creator викликає фабричний метод, ConcreteCreator повертає потрібний об'єкт Product.

7. Чим відрізняється шаблон «Абстрактна фабрика» від «Фабричний метод»?

Шаблон «Абстрактна фабрика» використовується тоді, коли потрібно створювати цілу групу взаємопов'язаних об'єктів. Він визначає набір фабричних методів, кожен з яких створює свій тип продукту, і таким чином дозволяє перемикати цілі сімейства

продуктів, не змінюючи код клієнта. Головна ідея — забезпечити узгодженість об’єктів, що створюються разом.

Шаблон «Фабричний метод» використовується тоді, коли потрібно делегувати створення одного конкретного типу об’єкта підкласам. Він визначає один метод, який підкласи перевизначають, вирішуючи, який саме клас продукту створити.

Таким чином:

1) Абстрактна фабрика створює кілька пов’язаних об’єктів одночасно.
2) Фабричний метод створює лише один об’єкт, і його вибір залежить від підкласу.
3) Абстрактна фабрика об’єднує багато фабричних методів.
4) Фабричний метод — це один метод для створення одного продукту.

8. Яке призначення шаблону «Знімок» (Memento)?

Шаблон «Знімок» дозволяє зберігати стан об’єкта та відновлювати його в майбутньому без порушення інкапсуляції.

9. Нарисуйте структуру шаблону «Знімок».

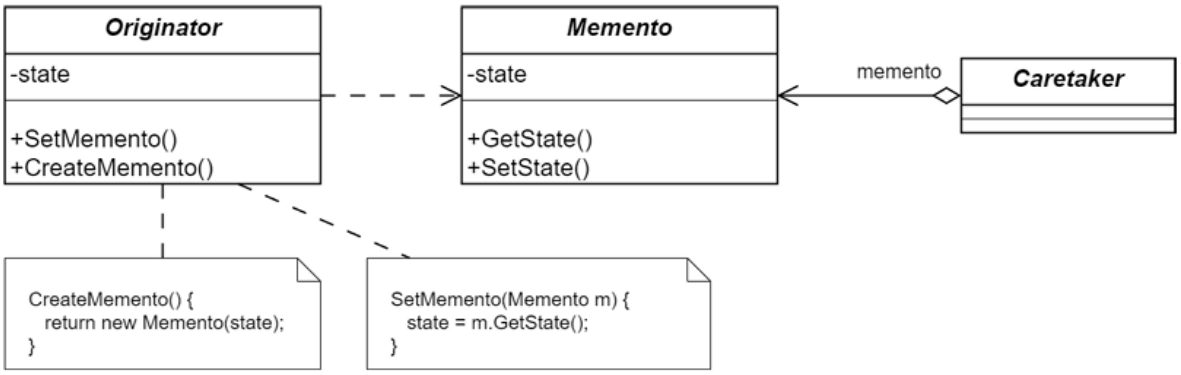


Рисунок 6.3. Структура шаблону «Знімок»

10. Які класи входять в шаблон «Знімок», та яка між ними взаємодія?

1) Originator — створює знімок та відновлює свій стан.
2) Memento — зберігає внутрішній стан Originator.
3) Caretaker — зберігає знімки, але не може змінювати їх.

Взаємодія: Originator → створює Memento; Caretaker зберігає; при потребі Originator відновлює стан із Memento.

11. Яке призначення шаблону «Декоратор»?

Шаблон «Декоратор» дозволяє динамічно додавати об’єкту нову поведінку або функціональність, не змінюючи його клас.

12. Нарисуйте структуру шаблону «Декоратор».

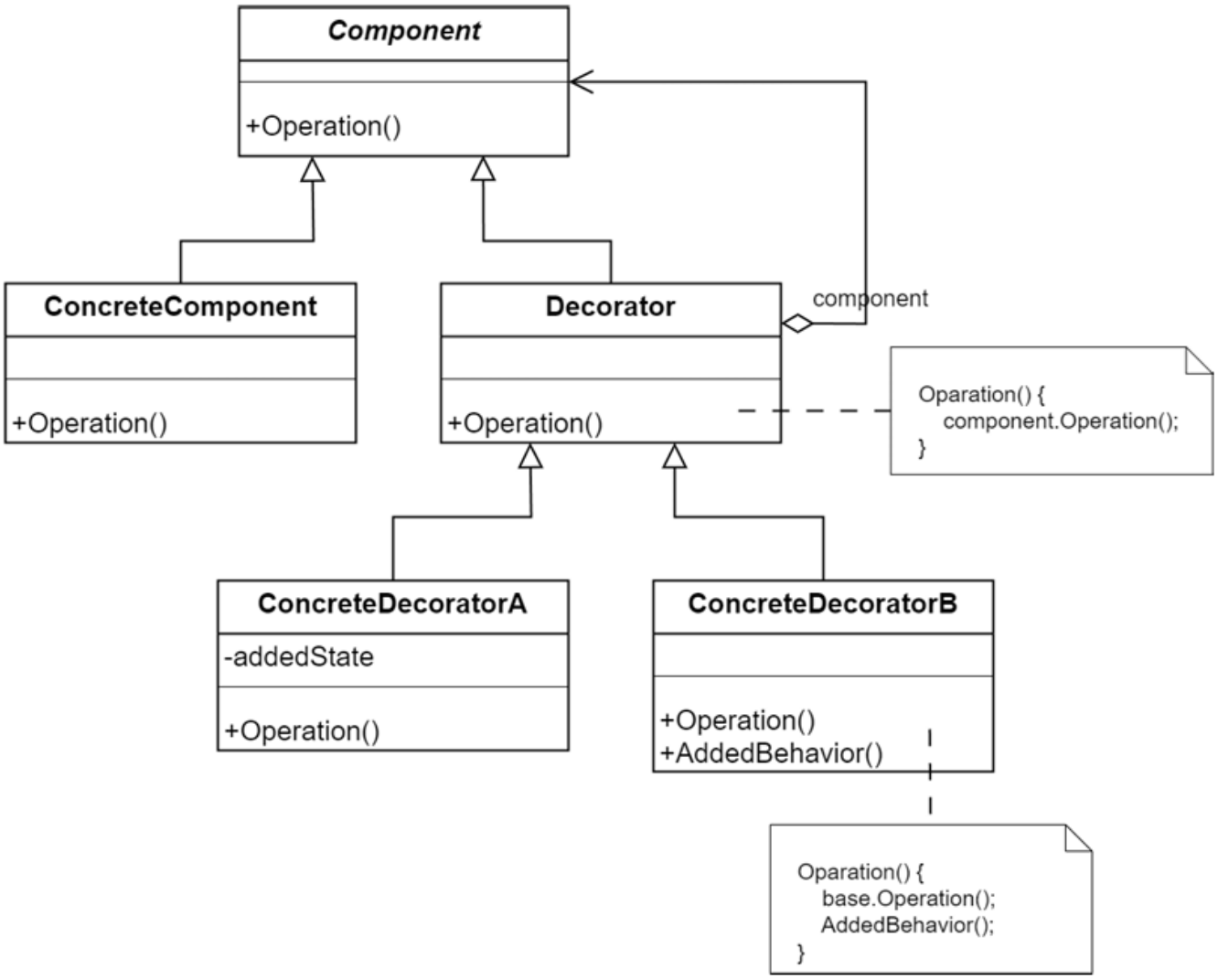


Рисунок 6.5. Структура патерну «Декоратор»

13. Які класи входять в шаблон «Декоратор», та яка між ними взаємодія?

1) Component — базовий інтерфейс.
2) ConcreteComponent — основна реалізація об’єкта.
3) Decorator — містить компонент і реалізує той самий інтерфейс.
4) ConcreteDecorator — додає нову функціональність.

Взаємодія: декоратор «обгортає» компонент і делегує йому виклик, додаючи власну поведінку.

14. Які є обмеження використання шаблону «Декоратор»?

1) Може ускладнити структуру програми через велику кількість дрібних класів.
2) Ланцюг декораторів може стати важким для розуміння й налагодження.
3) Не підходить, якщо необхідно змінювати структуру об’єкта, а не лише додавати поведінку.
4) При неправильному використанні може бути важко контролювати порядок застосування декораторів.