Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №6

Технології розроблення програмного забезпечення

Тема проєкту ‘Shell (total commander)’

Виконала: студентка групи ІА-33

Маляревич Анжела Валентинівна

Перевірив:

Мягкий Михайло Юрійович

Київ - 2025

Зміст

Тема: Патерни проектування.

Мета: Вивчити структуру шаблонів «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator» та навчитися застосовувати їх в реалізації програмної системи.

Тема роботи:

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.

Теоретичні відомості

Abstract Factory — патерн для створення цілих сімейств пов’язаних об’єктів без прив’язки до їх конкретних класів. Дозволяє легко міняти «стиль» або групу продуктів, забезпечуючи їхню узгодженість.

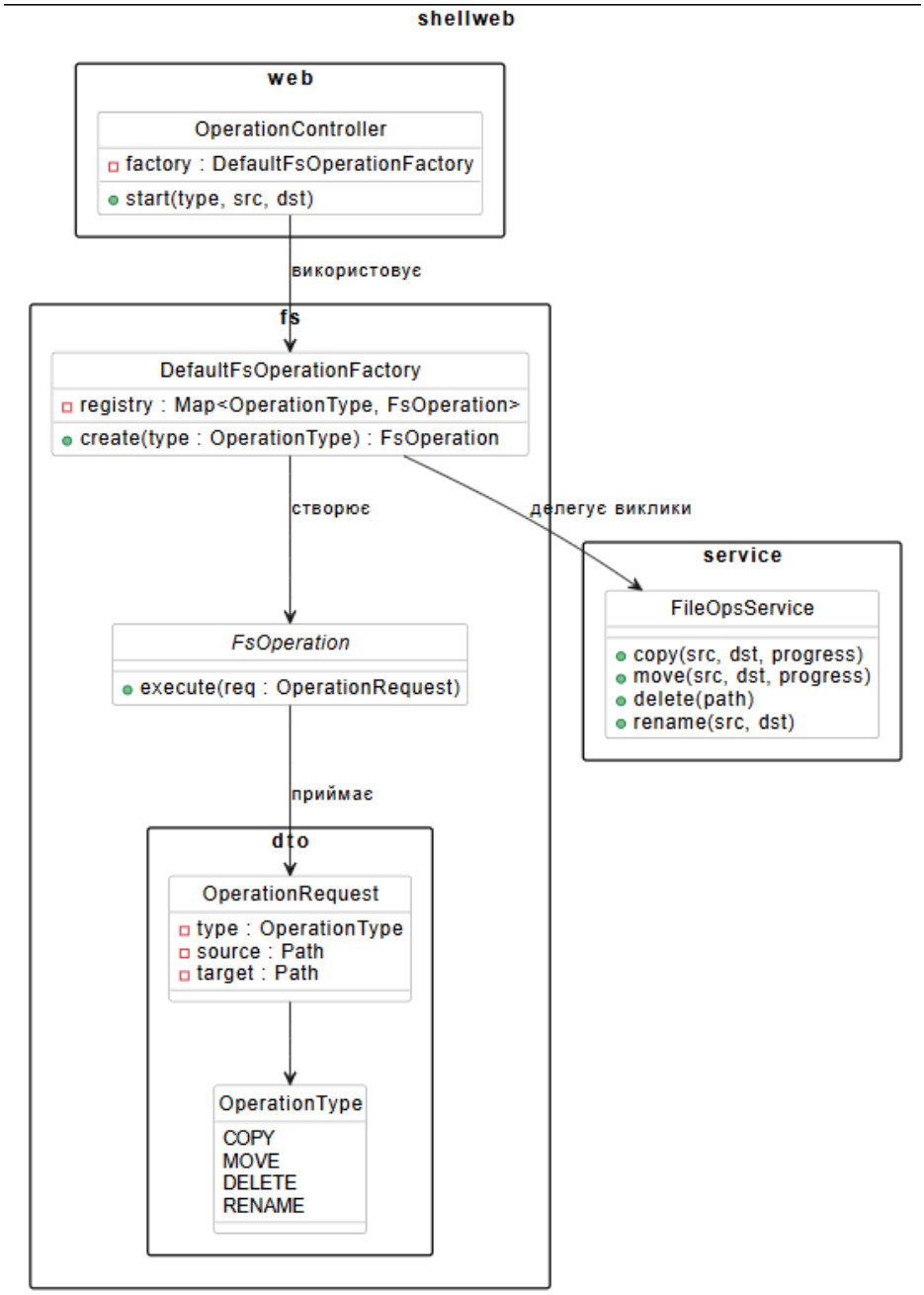
Factory Method — визначає спільний спосіб створення об’єктів базового типу, дозволяючи підкласам вирішувати, який конкретний об’єкт повертати. Зменшує залежність від конкретних класів і спрощує розширення системи.

Memento — дає можливість зберігати та відновлювати стан об’єктів без порушення інкапсуляції. Використовується для реалізації “undo”.

Observer — встановлює зв’язок «один-до-багатьох»: коли один об’єкт змінюється, усі підписники автоматично отримують сповіщення. Використовується в подієвих та реактивних системах.

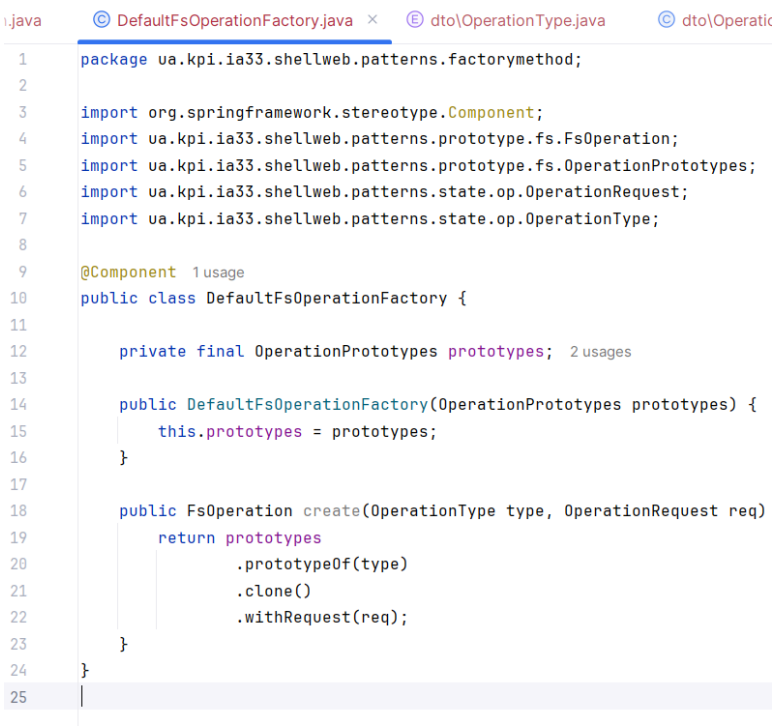
Decorator — дозволяє динамічно додавати нову поведінку об’єктам шляхом «обгортання», не змінюючи їх внутрішню реалізацію. Гнучкіший за спадкування, але може збільшувати кількість дрібних класів.

Хід роботи



Показано взаємодію між контролером (OperationController), фабрикою (DefaultFsOperationFactory), продуктом (FsOperation), DTO-класами та сервісом файлових операцій. Діаграма демонструє, як створення конкретної операції інкапсульовано у фабриці, а контролер працює лише через абстракцію.

DefaultFsOperationFactory



Фабрика отримує тип операції та об’єкт OperationRequest, обирає відповідний прототип файлової операції, клонує його та повертає готовий до виконання об’єкт. Таким чином, фабрика виступає “творцем” у патерні Factory Method.

OperationRequest

```java
package ua.kpi.ia33.shellweb.patterns.factorymethod.dto;

import java.nio.file.Path;

public class OperationRequest {

    private OperationType type;  2 usages
    private Path source;  2 usages
    private Path target;  2 usages

    public OperationType getType() { return type; }
    public void setType(OperationType type) { this.type = type; }

    public Path getSource() { return source; }
    public void setSource(Path source) { this.source = source; }

    public Path getTarget() { return target; }
    public void setTarget(Path target) { this.target = target; }
}

```

Цей DTO передає у фабрику й далі в операцію всі необхідні параметри: тип, шлях джерела та шлях призначення. Factory Method працює з ним як із універсальним контейнером параметрів

FsOperation

```java
package ua.kpi.ia33.shellweb.patterns.factorymethod;

import ua.kpi.ia33.shellweb.patterns.factorymethod.dto.OperationRequest;

@FunctionalInterface  no usages
public interface FsOperation {
    void execute(OperationRequest req) throws Exception;
}
```

Усі файлові операції успадковують цей інтерфейс і реалізують метод execut(). Це дозволяє фабриці повертати будь-яку конкретну операцію в єдиному узагальненому вигляді

OperationType

```java
package ua.kpi.ia33.shellweb.patterns.factorymethod.dto;

public enum OperationType {  5 usages
    COPY, MOVE, DELETE, RENAME  1 usage
}
```

Enum визначає можливі дії файлового менеджера. Саме цей тип використовується фабрикою для вибору правильного прототипу й створення конкретної операції.

Висновок

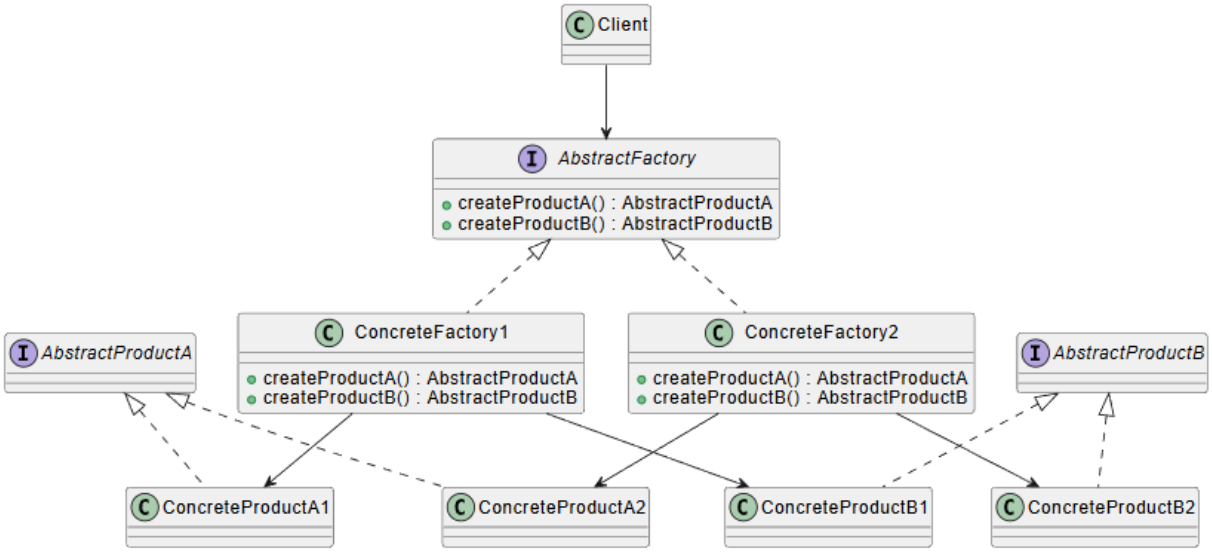
У цій роботі я реалізувала патерн Factory Method для створення файлових операцій у проєкті “Shell”. Завдяки фабриці процес створення операцій було відокремлено від контролера, що зробило код простішим, гнучкішим і легшим у підтримці. Фабрика використовує прототипи операцій та повертає готові об’єкти, тому контролер не залежить від їх конкретних реалізацій. У результаті система стала більш модульною та розширюваною

Питання до лабораторної роботи

1. Яке призначення шаблону «Абстрактна фабрика»?

Створювати сімейства взаємопов’язаних об’єктів (продуктів) одного стилю без прив’язки до їх конкретних класів, забезпечуючи узгодженість цих об’єктів між собою.

2. Нарисуйте структуру шаблону «Абстрактна фабрика».



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

AbstractFactory — інтерфейс/абстрактний клас фабрики, описує методи створення всіх типів продуктів.

ConcreteFactoryX — конкретні фабрики, реалізують AbstractFactory і створюють конкретні продукти одного сімейства (стилю).

AbstractProductA / AbstractProductB — інтерфейси продуктів різних типів.

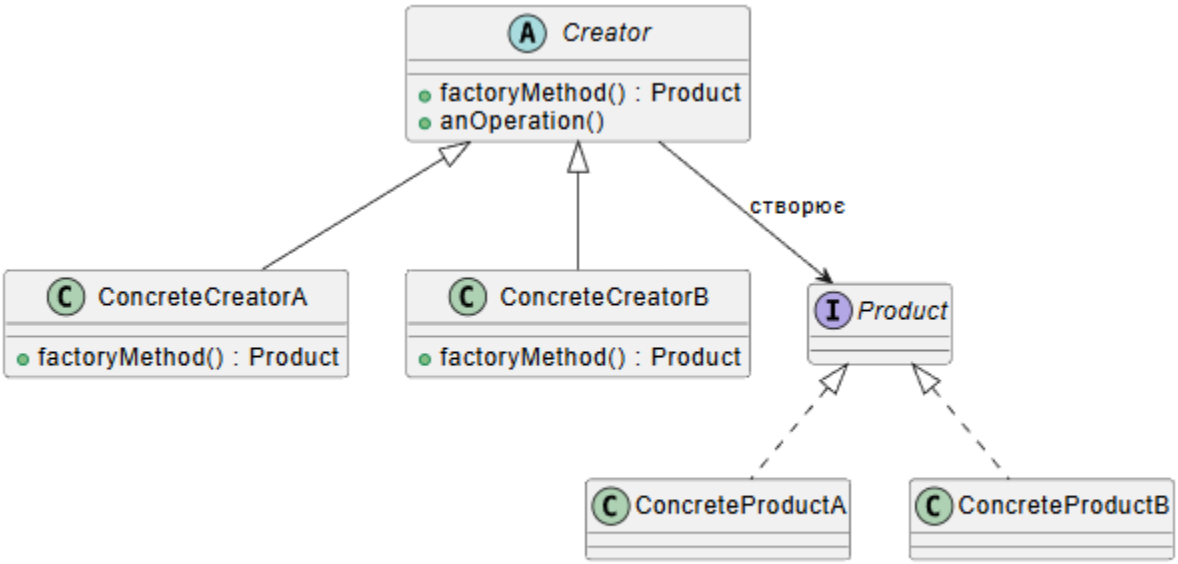
ConcreteProductAX / ConcreteProductBX — конкретні реалізації продуктів певного стилю.

Client — працює з фабрикою та продуктами тільки через абстрактні інтерфейси;

4. Яке призначення шаблону «Фабричний метод»?

Надати єдиний інтерфейс для створення об’єктів базового типу, дозволяючи підкласам вирішувати, який конкретний підтип буде створено. Це «віртуальний конструктор», що ізолює клієнта від конкретних класів продуктів

5. Нарисуйте структуру шаблону «Фабричний метод».



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

Creator — базовий клас/інтерфейс з фабричним методом factoryMethod() і, зазвичай, з операцією anOperation(), що використовує створений продукт.

ConcreteCreatorX — підкласи, які перевизначають factoryMethod() і створюють конкретні продукти.

Product — інтерфейс або базовий клас продукту.

ConcreteProductX — конкретні реалізації продукту.

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

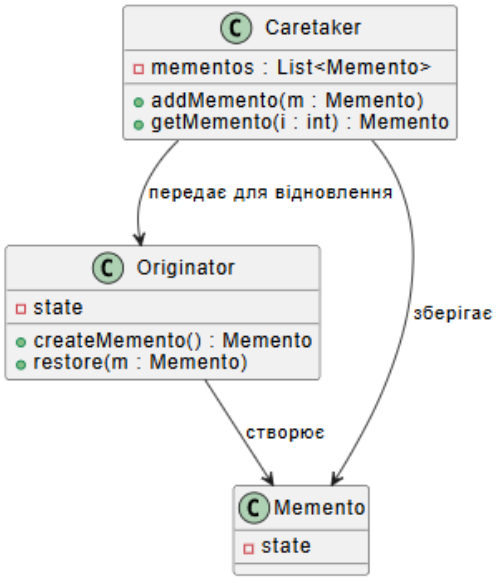
Абстрактна фабрика створює цілі сімейства різних продуктів, які узгоджені між собою (кілька фабричних методів в одному інтерфейсі).

Фабричний метод зосереджений на створенні одного типу продукту і зазвичай описує один віртуальний метод створення.

8. Яке призначення шаблону «Знімок»?

Зберігати і відновлювати стан об’єкта без порушення його інкапсуляції: стан виноситься в окремий Memento-об’єкт, який може бути збережений і пізніше використаний для відкату змін

9. Нарисуйте структуру шаблону «Знімок».



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

Originator — об’єкт, стан якого треба зберігати/відновлювати; вміє створювати Memento і відновлюватися з нього

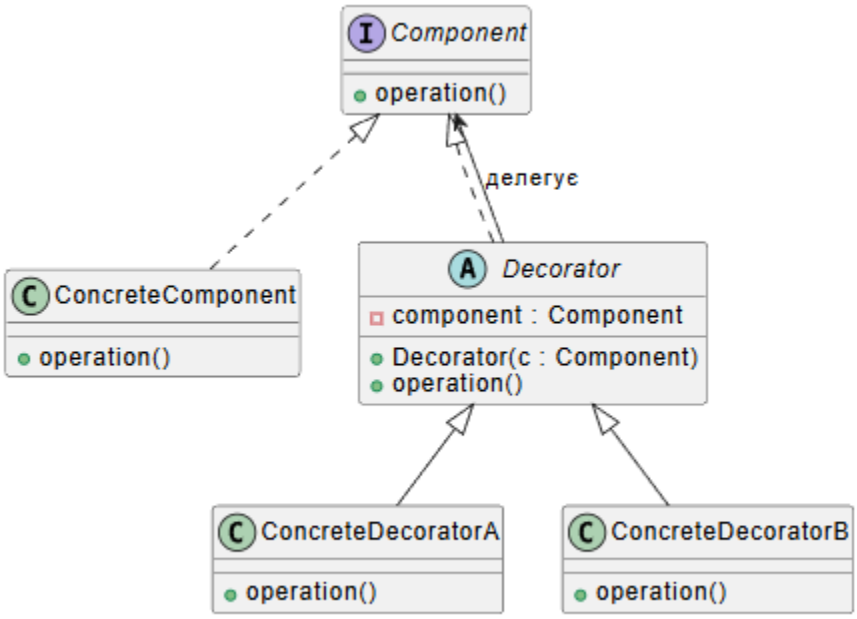
Memento — об’єкт-знімок, що зберігає внутрішній стан Originator; його вміст прихований від інших

Caretaker — зберігає Memento, керує історією, але не читає його вміст

11. Яке призначення шаблону «Декоратор»?

Динамічно додавати нову поведінку об’єктам під час виконання, обгортаючи їх у спеціальні об’єкти-декоратори, не змінюючи початковий клас і не торкаючись інших екземплярів цього класу

12. Нарисуйте структуру шаблону «Декоратор».



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

Component — базовий інтерфейс/клас з операцією operation().

ConcreteComponent — початкова реалізація компонента з базовою поведінкою.

Decorator — абстрактний клас, що реалізує Component і містить посилання на інший Component; у operation() делегує виклик внутрішньому компоненту.

ConcreteDecoratorX — конкретні декоратори, які перевизначають operation(), додаючи свою поведінку до виклику базового компонента.

14. Які є обмеження використання шаблону «декоратор»?

Може призвести до великої кількості дрібних класів, якщо є багато комбінацій поведінки. Об’єкти, обгорнуті декількома декораторами, важко конфігурувати та відлагоджувати, бо складно розібратися, в якому порядку застосовуються всі обгортки