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 № 7

із дисципліни: «Технології розроблення програмного забезпечення»

на тему: «Патерни проектування.**»**

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

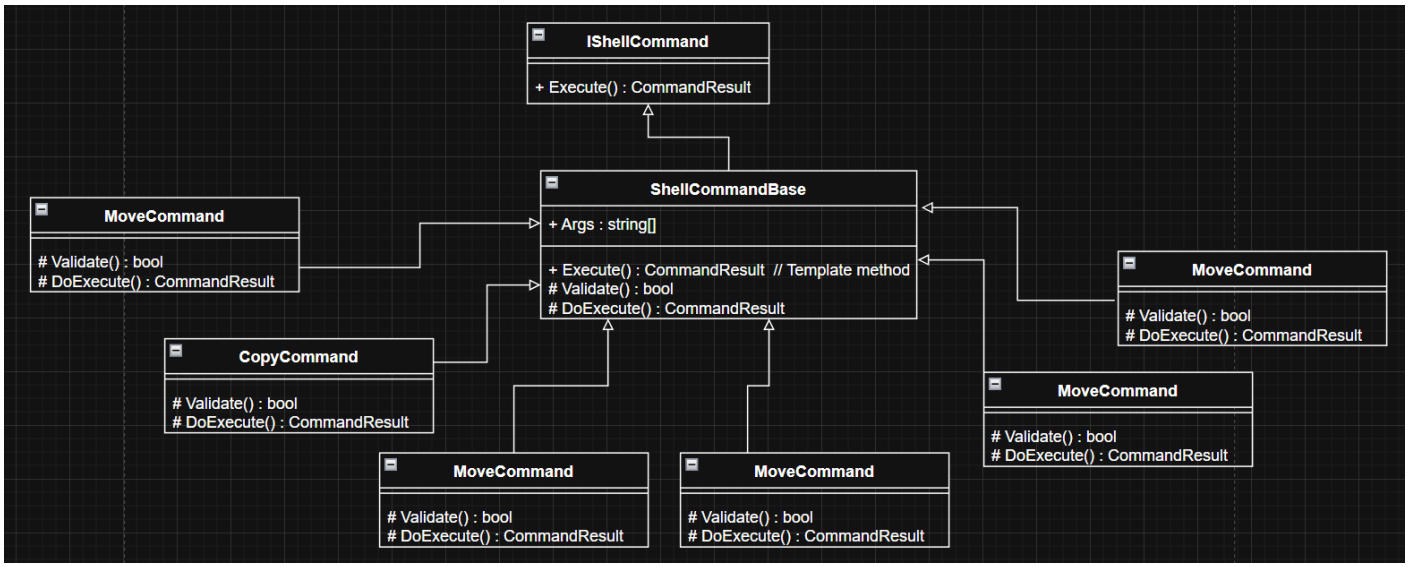
Київ-2025

Мета: Вивчити структуру шаблонів «Mediator», «Facade», «Bridge», «Template method» 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

Діаграма демонструє використання патерну Template Method у системі команд файлового менеджера.

1. ShellCommandBase (abstract)

Це ключовий клас, який містить шаблон виконання команди:

```
Execute()
{
   if (!Validate()) return error;
   return DoExecute();
}
```

Це і є Template Method:

1) він визначає структуру алгоритму,
2) не дозволяє змінювати кроки,
3) делегує змінні частини підкласам.

Також клас містить захищені методи:

1) Validate() — перевірка аргументів
2) DoExecute() — конкретне виконання

2. Конкретні команди

Кожна команда успадковує ShellCommandBase і реалізує тільки специфічну частину:

1) CopyCommand — копіювання файлів
2) MoveCommand — переміщення
3) DeleteCommand — видалення
4) ListCommand — показ каталогу
5) ChangeDirectoryCommand — перехід між папками
6) SearchCommand — пошук

Таким чином усі команди мають однаковий шаблон виконання але різну реалізацію поведінки.

Чому це Template Method?

Тому що:

1) шаблон алгоритму визначений один раз у базовому класі;
2) підкласи не можуть змінити структуру алгоритму;
3) змінюються лише його внутрішні кроки;
4) логіка повторних кроків не дублюється в десятках команд;
5) розширюваність системи реалізовано елегантно.

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

Висновок: У ході виконання лабораторної роботи було досліджено та реалізовано шаблон проектування Template Method. На прикладі системи Shell Total Commander продемонстровано, як абстрактний базовий клас ShellCommandBase визначає фіксовану структуру алгоритму виконання команди, тоді як конкретні команди (CopyCommand, MoveCommand, DeleteCommand тощо) реалізують лише окремі кроки цього алгоритму.

Такий підхід забезпечує:

1) уникнення дублювання коду,
2) просту модифікацію поведінки команд,
3) можливість легко додавати нові команди без зміни існуючих,
4) високу модульність та чисту архітектуру.

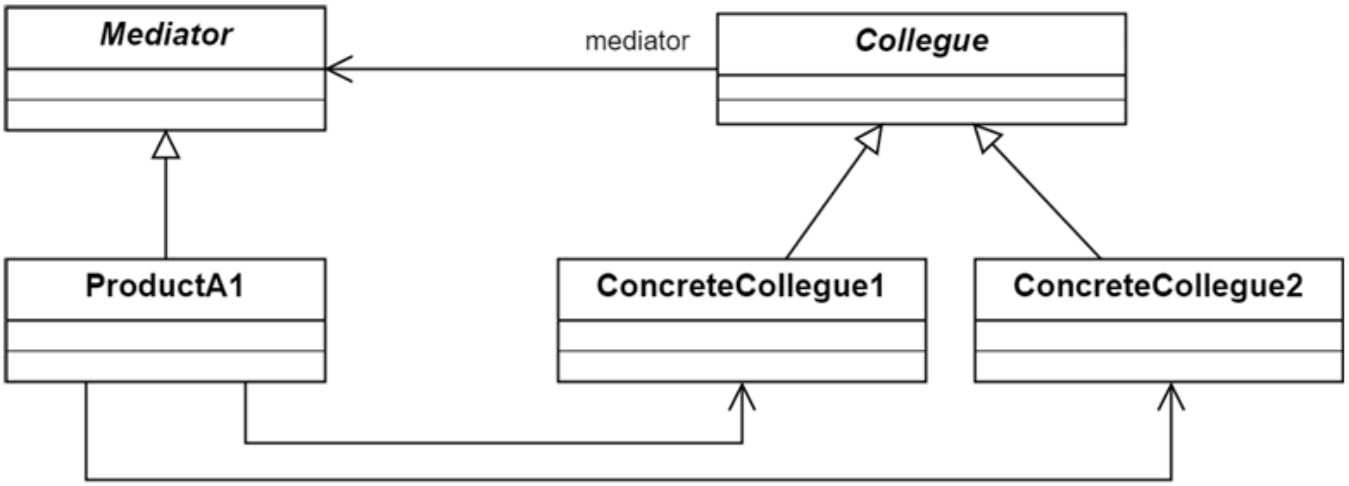
Патерн Template Method у твоєму проєкті реалізовано повністю та коректно, а його використання значно покращило структуру системи.

Питання до лабораторної роботи:

1. Яке призначення шаблону «Посередник»?

Шаблон «Посередник» призначений для організації взаємодії між об’єктами таким чином, щоб вони не залежали один від одного напряму. У складних системах часто виникає ситуація, коли різні компоненти повинні взаємодіяти між собою, але прямі посилання між ними створюють тісну зв’язність, ускладнюють модифікацію та розширення коду. Посередник бере на себе роль центрального координатора. Усі об’єкти (так звані «колеги») відправляють повідомлення не один одному, а через посередника, який вирішує, кому і як передати інформацію. Це дозволяє зменшити залежності між колегами, спростити структуру взаємодій та підвищити гнучкість системи.

2. Нарисуйте структуру шаблону «Посередник»



3. Які класи входять у шаблон «Посередник» та яка між ними взаємодія?

1) Mediator — визначає загальний інтерфейс взаємодії.
2) ConcreteMediator — реалізує механізм комунікації між учасниками, зберігає посилання на колег.
3) Colleague — абстрактний учасник, який знає про існування посередника.
4) ConcreteColleague — конкретні об’єкти, що відправляють повідомлення через посередника.

Взаємодія:

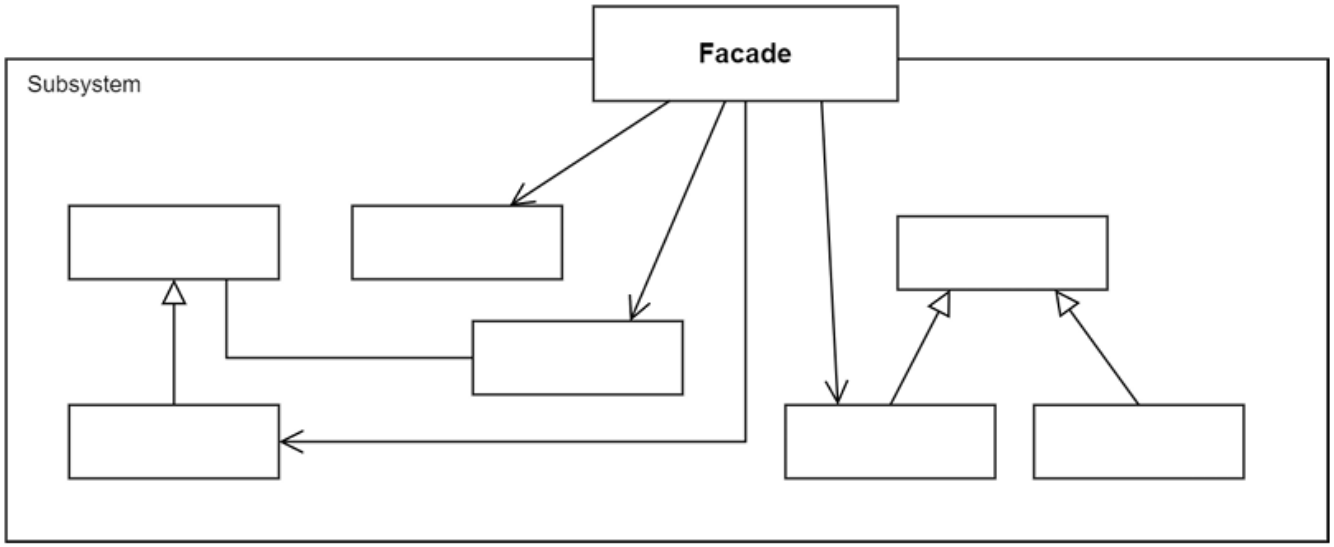
Колегія не звертаються одна до одної напряму. Кожен ConcreteColleague викликає метод Mediator, передаючи інформацію, а Mediator вирішує, як саме інші колеги повинні реагувати. Таким чином централізується логіка взаємодії.

4. Яке призначення шаблону «Фасад»?

Шаблон «Фасад» спрощує роботу з великою або складною підсистемою, надаючи до неї єдиний, високорівневий інтерфейс. Це дозволяє приховати складність внутрішньої логіки за простою обгорткою. Клієнт взаємодіє з фасадом, не заглиблюючись у деталі внутрішніх класів та методів.

Шаблон зменшує залежність між клієнтом і компонентами підсистеми, робить код чистішим і легшим в обслуговуванні.

5. Нарисуйте структуру шаблону «Фасад»



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

1) Facade — надає спрощений інтерфейс;
2) Subsystem — внутрішні класи, що виконують реальні операції;
3) Client — використовує фасад для доступу до складної логіки.

Взаємодія:

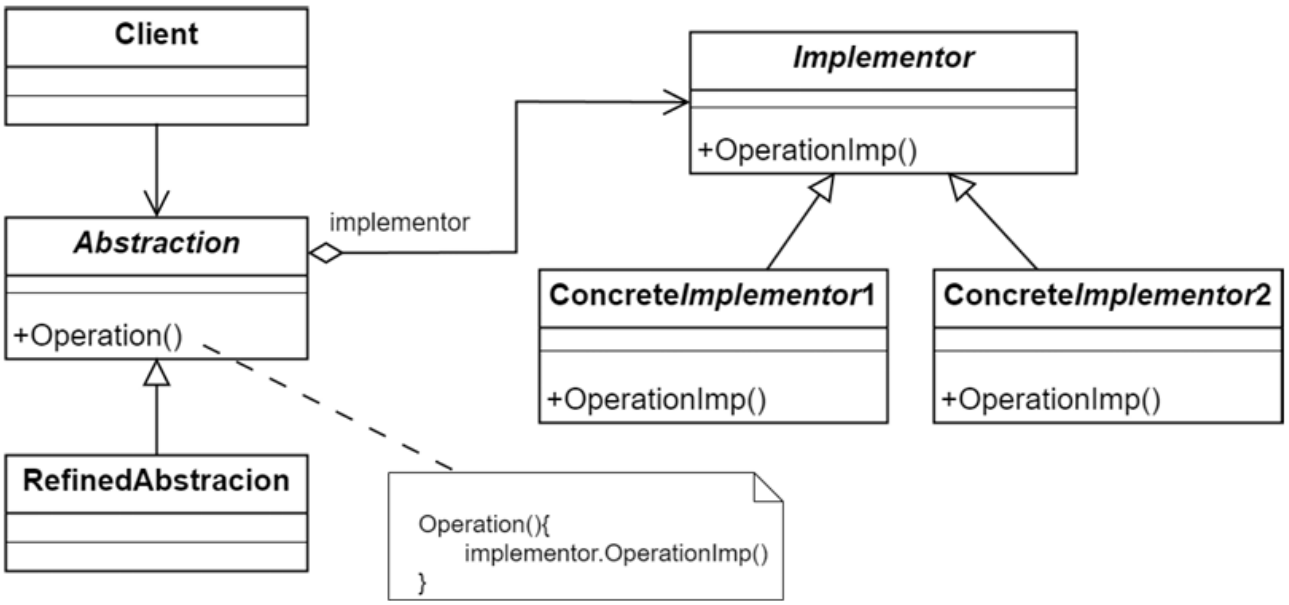
Клієнт викликає методи фасаду, а фасад у свою чергу викликає відповідні методи підсистем. Клієнт ніколи не звертається до підсистем напряму.

7. Яке призначення шаблону «Міст»?

Шаблон «Міст» використовується для розділення абстракції та реалізації так, щоб вони могли змінюватися незалежно одна від одної. У традиційному спадкуванні зміна реалізації часто вимагає створення великої кількості підкласів. «Міст» вирішує цю проблему через композицію: абстракція містить посилання на об’єкт реалізації, що

дозволяє незалежно змінювати обидві частини. Це підвищує гнучкість системи, дозволяє легко додавати нові абстракції або реалізації без переписування коду.

8. Нарисуйте структуру шаблону «Міст»



9. Які класи входять у шаблон «Міст», та яка між ними взаємодія?

1) Abstraction — визначає логіку високого рівня й містить посилання на реалізатор;
2) RefinedAbstraction — розширює абстракцію;
3) Implementor — інтерфейс методів реалізації;
4) ConcreteImplementor — конкретне втілення реалізації.

Взаємодія:

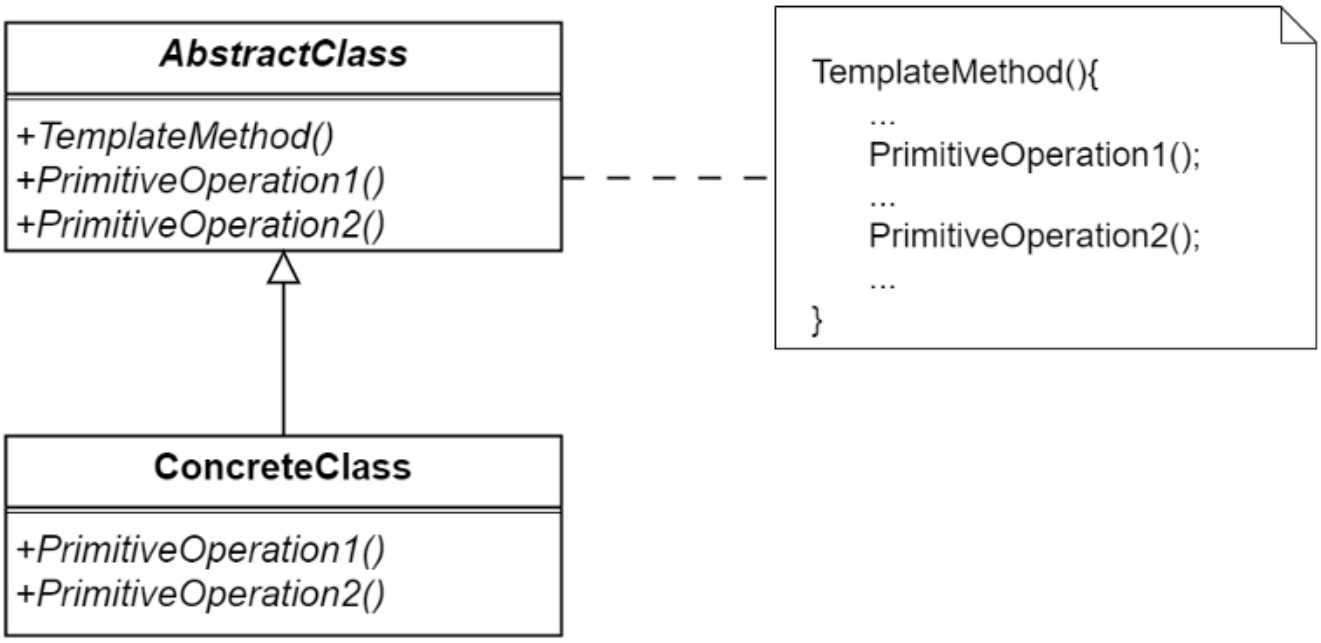
Абстракція делегує частину роботи реалізатору. Це дозволяє підключати різні реалізації без зміни коду абстракції.

10. Яке призначення шаблону «Шаблонний метод»?

Шаблон «Шаблонний метод» дозволяє визначити загальний алгоритм у базовому класі, при цьому дозволяючи підкласам змінювати окремі кроки цього алгоритму. Базовий клас встановлює структуру, тобто порядок виконання дій, але не нав’язує реалізацію кожного етапу. Це допомагає уникнути дублювання коду та забезпечує гнучкість при розширенні.

11. Нарисуйте структуру шаблону «Шаблонний метод»

Структура шаблону:



12. Які класи входять у шаблон «Шаблонний метод», та яка між ними взаємодія?

1) AbstractClass — містить основний алгоритм та абстрактні кроки;
2) ConcreteClass — реалізує конкретні варіанти кроків.

Взаємодія:

Метод-шаблон викликає змінювані кроки, які реалізують підкласи. Завдяки цьому структура алгоритму залишається незмінною, а конкретні дії можуть різнитися.

13. Чим відрізняється шаблон «Шаблонний метод» від «Фабричного методу»?

Шаблон «Фабричний метод» спрямований на створення об’єктів — він визначає, який саме тип продукту буде створено. Підклас вирішує, який конкретний об’єкт повертати. Шаблон «Шаблонний метод» визначає структуру алгоритму та дозволяє підкласам перевизначати окремі його етапи, не змінюючи загальний порядок виконання.

Отже:

1) «Фабричний метод» — про створення об’єктів.
2) «Шаблонний метод» — про поведінку та зміну кроків алгоритму.

14. Яку функціональність додає шаблон «Міст»?

Шаблон «Міст» додає можливість незалежного варіювання як абстракції, так і реалізації. Це означає, що можна додавати нові типи абстракцій, не змінюючи реалізацію, і додавати реалізації без змін у коді абстракцій. Також він:

1) запобігає вибуху класів при комбінуванні різних функцій;

2) дозволяє замінювати реалізації «на льоту»;
3) зменшує залежність між рівнями системи;
4) полегшує тестування й розширення.