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 № 7

Технології розробки програмного забезпечення

“ Патерни проектування.”

“Download manager”

Виконав студент групи ІА–33:
Ященко К.А.

Київ 2025

Тема: Патерни проєктування
Мета: Вивчити структуру шаблонів “Mediator”, “Facade”, “Bridge”, “Template method” та навчитися застосовувати їх в реалізації програмної частини.

Зміст

Завдання

- Ознайомитись з короткими теоретичними відомостями.
- Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
- Реалізувати один з розглянутих шаблонів за обраною темою.
- Реалізувати не менше 3-х класів відповідно до обраної теми.
- 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.

Теоретичні відомості

Mediator (Посередник)
Цей патерн поведінки призначений для зменшення хаотичних зв'язків між об'єктами. Замість того, щоб об'єкти (колеги) спілкувалися один з одним напряму, створюючи складну мережу залежностей, вони взаємодіють лише через спеціальний об'єкт-посередник. Посередник інкапсулює логіку взаємодії та

вирішує, якому компоненту передати запит або як на нього відреагувати. Це дозволяє послабити зв'язність коду, оскільки компоненти нічого не знають про реалізацію інших компонентів, а зміна логіки спілкування відбувається лише в одному класі посередника, не зачіпаючи решту системи.

Facade (Фасад)
Фасад — це структурний патерн, який надає простий інтерфейс до складної системи класів, бібліотеки або фреймворку. Він приховує всю складність ініціалізації, налаштування та взаємодії внутрішніх об'єктів підсистеми за одним класом-обгорткою. Клієнтський код взаємодіє тільки з фасадом, викликаючи прості методи, а фасад вже перенаправляє ці виклики потрібним об'єктам у правильному порядку. Це дозволяє відокремити бізнес-логіку програми від деталей реалізації сторонніх бібліотек та спростити використання системи для клієнта, хоча іноді це може обмежувати гнучкість, якщо потрібен доступ до низькорівневих функцій.

Bridge (Міст)
Цей структурний патерн використовується для розділення абстракції та її реалізації так, щоб вони могли змінюватися незалежно одна від одної. Це особливо корисно, коли клас має кілька вимірів розширення (наприклад, "Форма" і "Колір"), що при звичайному успадкуванні призвело б до створення величезної кількості підкласів для кожної комбінації. Міст пропонує замінити успадкування композицією: виділити одну з ієрархій в окремий набір класів (реалізацію) і посилатися на об'єкт цього набору з основної ієрархії (абстракції). Таким чином, ви можете змінювати або додавати нові класи в обидві ієрархії, не ламаючи існуючий код.

Template Method (Шаблонний метод)
Це патерн поведінки, який визначає скелет алгоритму в суперкласі, але дозволяє підкласам перевизначати певні кроки цього алгоритму без зміни його

структури. У базовому класі створюється метод, який викликає ряд інших методів у чітко визначеній послідовності. Частина цих методів може мати реалізацію за замовчуванням, а частина може бути абстрактною. Підкласи реалізують або перевизначають ці конкретні кроки, наповнюючи шаблон своєю логікою, але загальний порядок виконання операцій залишається незмінним, що сприяє повторному використанню коду.

Тема проєкту
26. Download manager (iterator, command, observer, template method, composite, p2p)
Інструмент для скачування файлів з інтернету по протоколах http або https з можливістю продовження завантаження в зупиненому місці, розподілу швидкостей активним завантаженням, ведення статистики завантажень, інтеграції в основні браузери (firefox, opera, internet explorer, chrome).

Паттерн: Template method
Посилання на GitHub: https://github.com/fl1ckyexe/trpz

Хід роботи

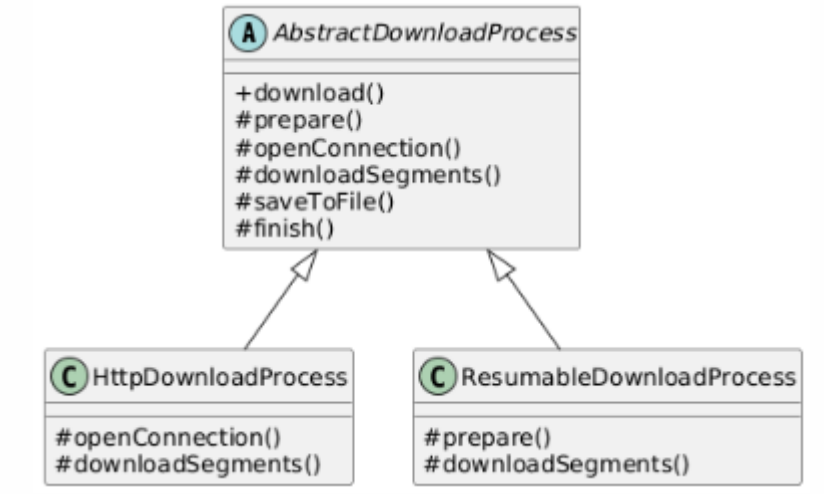


Рис. 1 – Діаграма класів для Template method

Діаграма класів відображає застосування шаблону проєктування Template Method для організації процесу завантаження файлів у менеджері завантажень. Загальний алгоритм завантаження визначається в абстрактному класі, тоді як конкретні кроки можуть бути перевизначені у підкласах.

Основні елементи діаграми

Клас AbstractDownloadProcess є абстрактним базовим класом, який визначає шаблонний метод download(). Саме цей метод фіксує послідовність кроків завантаження і не може бути змінений у підкласах.

Методи prepare, openConnection, downloadSegments, saveToFile та finish оголошені як захищені. Частина з них може мати стандартну реалізацію, а частина — перевизначатися у нащадках.

Клас HttpDownloadProcess реалізує стандартний процес завантаження файлів по HTTP/HTTPS.

Клас ResumableDownloadProcess спеціалізується на завантаженні з можливістю відновлення, зокрема перевизначає етапи підготовки та завантаження сегментів.

Зв’язки та взаємодія між елементами

Підкласи HttpDownloadProcess та ResumableDownloadProcess наслідують AbstractDownloadProcess, отримуючи загальний алгоритм завантаження.

Метод download() викликає кроки алгоритму у визначеній послідовності, незалежно від конкретної реалізації підкласу.

Кожен підклас може змінювати лише окремі кроки алгоритму, не порушуючи загальної структури процесу завантаження.

Таким чином забезпечується повторне використання коду та єдина логіка роботи для різних типів завантажень.

Переваги використання шаблону Template method у проєкті

У межах нашого проєкту шаблон Template Method було використано для моделювання процесу завантаження файлів. Загальна логіка завантаження була винесена в абстрактний клас, що дозволило уникнути дублювання коду та чітко зафіксувати послідовність дій.

Конкретні варіанти завантаження (звичайне HTTP-завантаження та завантаження з відновленням) були реалізовані у вигляді підкласів, які перевизначають лише необхідні етапи алгоритму.

Такий підхід дозволив продемонструвати розуміння шаблону Template Method, забезпечив структурованість коду та створив основу для подальшого розширення функціональності менеджера завантажень.

Частина коду програми з Template method

```
// AbstractDownloadProcess.java
public abstract class AbstractDownloadProcess {

    public final void download() {
        prepare();
        openConnection();
        downloadSegments();
        saveToFile();
        finish();
    }

    protected void prepare() {
        // Підготовка до завантаження (перевірка URL, файлу, БД)
    }

    protected abstract void openConnection();

    protected abstract void downloadSegments();

    protected void saveToFile() {
        // Запис завантажених сегментів у файл
    }

    protected void finish() {
        // Завершення процесу завантаження, оновлення статусу
    }
}

// HttpDownloadProcess.java
```

```
public class HttpDownloadProcess extends AbstractDownloadProcess {

    @Override
    protected void openConnection() {
        // Встановлення HTTP/HTTPS-з’єднання
    }

    @Override
    protected void downloadSegments() {
        // Послідовне або паралельне завантаження сегментів
    }
}

// ResumableDownloadProcess.java
public class ResumableDownloadProcess extends AbstractDownloadProcess {

    @Override
    protected void prepare() {
        // Перевірка наявних сегментів для відновлення завантаження
    }

    @Override
    protected void openConnection() {
        // Встановлення HTTP range-з’єднання
    }

    @Override
    protected void downloadSegments() {
        // Завантаження лише відсутніх сегментів
    }
}

// DownloadService.java
public class DownloadService {

    public void startDownload(AbstractDownloadProcess process) {
        process.download();
    }
}

// Main.java
public class Main {
    public static void main(String[] args) {

        DownloadService service = new DownloadService();

        AbstractDownloadProcess httpDownload =
                new HttpDownloadProcess();
```

```
        AbstractDownloadProcess resumableDownload =
                new ResumableDownloadProcess();

        service.startDownload(httpDownload);
        service.startDownload(resumableDownload);
    }
}
```

Опис програмного коду

Поданий програмний код реалізує шаблон проєктування Template Method для організації процесу завантаження файлів у десктопному менеджері завантажень. Абстрактний клас AbstractDownloadProcess визначає загальний алгоритм завантаження у вигляді методу download, який фіксує послідовність виконання основних етапів процесу.

Окремі кроки алгоритму реалізовані у вигляді захищених методів, частина з яких має стандартну реалізацію, а частина оголошена абстрактною. Це дозволяє підкласам змінювати лише окремі етапи завантаження без порушення загальної логіки роботи.

Класи HttpDownloadProcess та ResumableDownloadProcess наслідують базовий клас і реалізують специфічні варіанти завантаження. Перший відповідає за стандартне HTTP/HTTPS-завантаження, а другий — за завантаження з можливістю відновлення зупиненого процесу.

Клас DownloadService використовує абстрактний тип AbstractDownloadProcess, що дозволяє запускати будь-який варіант завантаження без знання його конкретної реалізації.

Висновок

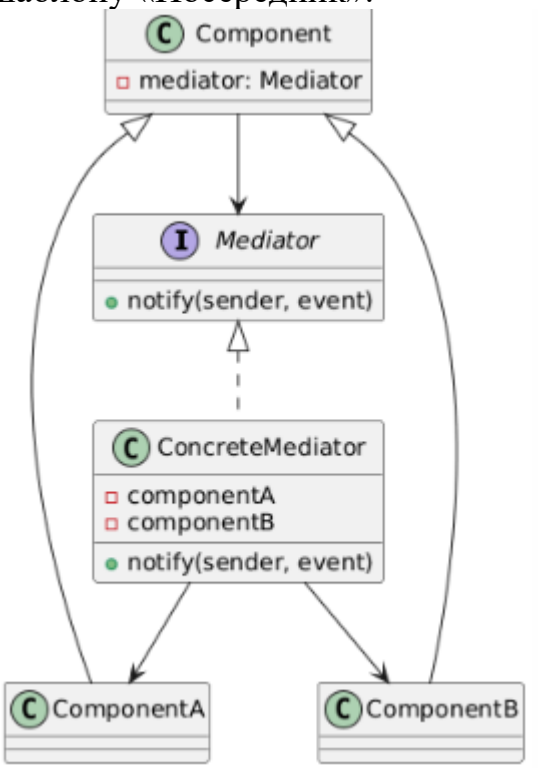
У ході виконання лабораторної роботи було розглянуто та реалізовано шаблон проєктування Template Method у межах десктопного менеджера завантажень файлів. Даний шаблон дозволив зафіксувати загальну структуру алгоритму завантаження та винести змінні етапи у підкласи.

Використання Template Method забезпечило повторне використання коду, зменшило дублювання логіки та спростило розширення системи новими варіантами завантаження. Отримана реалізація демонструє коректне застосування шаблону та створює основу для подальшого розвитку функціональності менеджера завантажень.

Відповіді на контрольні питання

1. Яке призначення шаблону «Посередник»?
Шаблон проєктування «Посередник» (Mediator) призначений для зменшення кількості прямих зв’язків між об’єктами шляхом винесення логіки їх взаємодії в окремий об’єкт-посередник. Це спрощує структуру системи та зменшує зв’язаність компонентів.

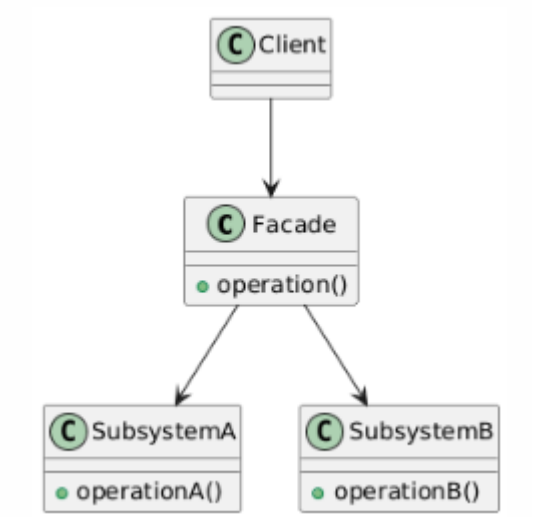
2. Нарисуйте структуру шаблону «Посередник».



3. Які класи входять в шаблон «Посередник», та яка між ними взаємодія?
У шаблоні беруть участь інтерфейс Mediator, конкретний клас ConcreteMediator та компоненти Component. Компоненти не взаємодіють напряму між собою, а передають інформацію через посередника, який координує їхню роботу.

4. Яке призначення шаблону «Фасад»?
Шаблон «Фасад» призначений для надання спрощеного інтерфейсу до складної підсистеми. Він приховує внутрішню складність системи та дозволяє клієнту працювати з нею через єдину точку доступу.

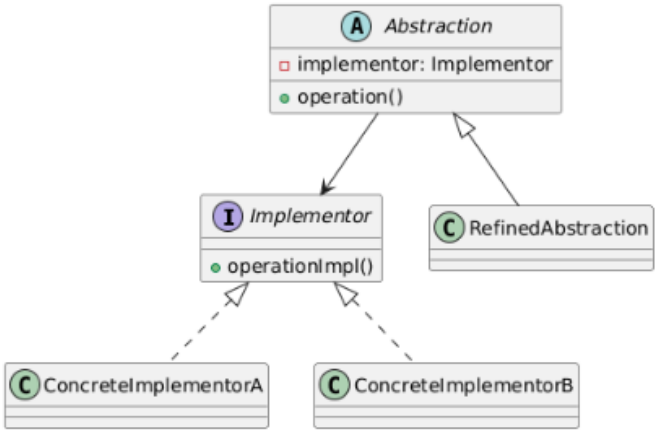
5. Нарисуйте структуру шаблону «Фасад».



6. Які класи входять в шаблон «Фасад», та яка між ними взаємодія?
У шаблоні використовуються клас Facade, декілька класів підсистеми та Client. Клієнт звертається лише до фасаду, а фасад координує виклики методів підсистеми у правильному порядку.

7. Яке призначення шаблону «Міст»?
Шаблон «Міст» (Bridge) призначений для відокремлення абстракції від її реалізації таким чином, щоб обидві могли змінюватися незалежно одна від одної.

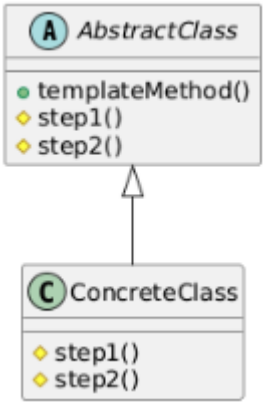
8. Нарисуйте структуру шаблону «Міст».



9. Які класи входять в шаблон «Міст», та яка між ними взаємодія?
Шаблон включає абстракцію Abstraction, конкретні абстракції, інтерфейс Implementor та його реалізації. Абстракція містить посилання на реалізацію і делегує їй виконання низькорівневих операцій.

10. Яке призначення шаблону «Шаблонний метод»?
Шаблон «Шаблонний метод» визначає скелет алгоритму в базовому класі та дозволяє підкласам перевизначати окремі кроки алгоритму без зміни його загальної структури.

11. Нарисуйте структуру шаблону «Шаблонний метод».



12. Які класи входять в шаблон «Шаблонний метод», та яка між ними взаємодія?
У шаблоні використовуються абстрактний клас AbstractClass, який визначає шаблонний метод, та конкретні підкласи, що реалізують окремі кроки алгоритму. Шаблонний метод керує порядком виклику цих кроків.

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

Шаблонний метод визначає структуру алгоритму, дозволяючи змінювати лише його окремі етапи, тоді як Фабричний метод спеціалізується на створенні об’єктів і дозволяє підкласам визначати тип створюваного продукту.

14. Яку функціональність додає шаблон «Міст»?
Шаблон «Міст» додає можливість незалежного розвитку ієрархій абстракцій та реалізацій, зменшує кількість підкласів та підвищує гнучкість і масштабованість програмної системи.