Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

# Лабораторна робота № 7

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Патерни проектування»

«Web crawler»

Виконала
студентка групи – ІА–31
Горлач Дар’я Дмитрівна

Перевірив:
Мягкий Михайло Юрійович

Київ 2025

Тема: Патерни проектування.

Мета: Вивчити структуру шаблонів «Mediator», «Facade», «Bridge», «Template method» та навчитися застосовувати їх в реалізації програмної системи.

Тема проєкту: Web crawler (proxy, chain of responsibility, memento, template method, composite, p2p) Веб–сканер повинен вміти розпізнавати структуру сторінок сайту, переходити за посиланнями, збирати необхідну інформацію про зазначений термін, видаляти не семантичні одиниці (рекламу, об'єкти javascript і т.д.), зберігати знайдені дані у вигляді структурованого набору html файлів вести статистику відвіданих сайтів і метадані.

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.

## Зміст



## Теоретичні відомості

### Шаблон «Template method»

Призначення патерну: Шаблон «Template Method» (шаблонний метод) дозволяє реалізувати покроково алгоритм в абстрактному класі, але залишити специфіку реалізації підкласам [6]. Можна привести в приклад формування веб-сторінки: необхідно додати заголовки, вміст сторінки, файли, що додаються, і нижню частину сторінки. Код для додавання вмісту сторінки може бути абстрактним і реалізовуватися в різних класах

– AspNetCompiler, HtmlCompiler, PhpCompiler і т.п. Додавання всіх інших елементів виконується за допомогою вихідного абстрактного класу з алгоритмом.

Даний шаблон дещо нагадує шаблон «Фабричний метод», однак область його використання абсолютно інша – для покрокового визначення конкретного алгоритму; більш того, даний шаблон не обов'язково створює нові об'єкти – лише визначає послідовність дій.

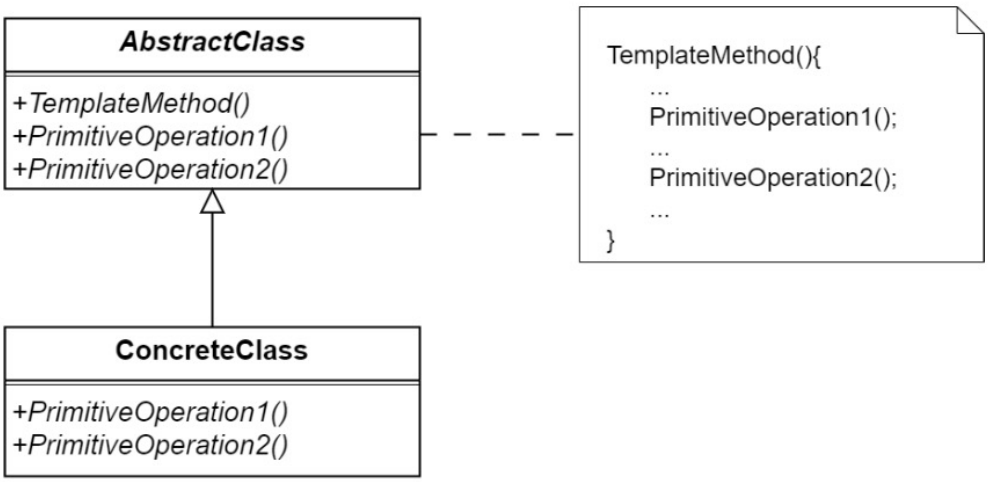



Рисунок 1. Структура патерну «Шаблонний метод»

Проблема: Ви працюєте в команді, що займається розробкою застосунку для редагування відео-файлів. Застосунок вже працює з форматом відео MPEG- 4, а саме дозволяє читати такі файли, виконувати попередню обробку даних для відображення в відео-редакторі.

Ви отримуєте нову задачу на реалізацію можливості роботи з більш старим форматом MPEG-2. Ви бачите два варіанта: зробити копію існуючого класу, що працює з MPEG-4, або вносити зміни в уже існуючий клас. Щоб прийняти рішення ви більш детально розбираєтеся з існуючим алгоритмом і бачите, що близько 70 відсотків коду має бути таким самим. Тому ви вирішуєте змінити вже існуючий клас для роботи з MPEG-4 добавляючи в місцях де це потрібно умови з перевіркою, що якщо формат MPEG-2 то відпрацьовувати новий код, який ви добавили. Через деякий час, на запити від користувачів, вам на реалізацію приходить задача добавити підтримку ще більш старого формату MPEG-1. Ви вносите зміни так само в існуючий клас, тільки умови стали більш складними, тому що розгалуження логіки йде на три гілки.

Ще через деякий час приходить аналогічна задача на додавання читання даних з файлів формату H.262. Ви починаєте працювати над задачею і бачите, що код, який до цього був ще більш-менш зрозумілим стає зовсім важким для читання та внесення змін.

Рішення: Патерн «Шаблонний метод» (Template Method) пропонує загальний алгоритм винести в базовий клас, а частини алгоритма, які для різних задач виконуються по різному, виділити в окремі методи. Ці методи будуть викликатися в алгоритмі, що реалізований в базовому класі. В дочірніх класах ці виділені методи будуть перевизначатися. Таким чином загальна логіка залишається в базовому класі, а специфічна частина реалізується в дочірніх класах.

Якщо подивитися на задачу з відео-редактором, то застосування «Шаблонного методу» наведе лад в коді і спростить його зміни.

Як це зробити: По перше, в алгоритмі всі блоки коду де є вибір гілки на основі типу формату виділяються в окремі методи. У випадку з відео-редактором, це скоріш за все будуть блоки коду пов'язані з читанням даних та розпакування їх в кадри, а також читання звукових доріжок. Далі створюється загальний базовий клас в який переноситься загальний алгоритм, а також об'являються віртуальні методи (фактично беремо сигнатуру тих методів, що виділили на попередньому кроці). Далі створюємо дочірні класи під кожен формат файлу і перевизначаємо віртуальні методи. Фактично при цьому в кожному такому методі в дочірньому класі із реалізації цих методів, що була виділена на першому кроці, залишається код гілки який відповідав вибраному формату.

Після всіх цих змін ми маємо реалізацію патерна «Шаблоний метод»: в базовому класі реалізовано базовий алгоритм (по суті більша частина алгоритму) і в дочірніх класах перевизначені методи зі специфічною логікою.

Після таких змін, добавити підтримку нового формату стає легше, тому що достатньо буде добавити лише новий дочірній клас і перевизначити в ньому необхідні методи.

Слід зауважити, що якщо у вас алгоритми співпадають більше ніж на 50 відсотків, то застосування шаблонного методу буде доцільним, але якщо у вас алгоритми співпадають лише відсотків на 10 або 20, то скоріш за все, краще буде використати патерн «Стратегія».

Переваги та недоліки:

+ Полегшує повторне використання коду.

- Ви жорстко обмежені скелетом існуючого алгоритму.

- Ви можете порушити принцип підстановки Барбари Лісков, змінюючи базову поведінку одного з кроків алгоритму через підклас.

- З ростом складності загального алгоритму шаблонний метод стає занадто складно підтримувати, особливо, коли є багато віртуальних методів для перевизначення в підкласах.

## Хід роботи

### Реалізація патерну проєктування

Для реалізації Web Crawler використано патерн проєктування Template Method, оскільки він забезпечує визначення загального алгоритму краулінгу веб-сторінок з можливістю гнучкої зміни конкретних кроків обробки. Це дозволяє створити уніфіковану структуру процесу збору даних, яка може бути легко адаптована для різних типів веб-сайтів та завдань.

Патерн Template Method реалізовано у класах AbstractCrawler та SimpleWebCrawler, де базовий клас визначає загальний алгоритм краулінгу, а конкретна реалізація надає специфічну логіку для кожного кроку. Такий підхід є особливо ефективним для веб-краулерів, де процес збору даних зазвичай включає стандартні етапи, але з різною реалізацією залежно від контексту.

```
public abstract class AbstractCrawler {
    2 usages
    public final void crawl(String url) {
        System.out.println("\nCrawl: " + url);
        String html = fetchPage(url);
        html = processHtml(html);
        saveHtml(url, html);
        collectStats(url);
        for (String link : extractLinks(html)) {
            crawl(link);
        }
    }

    1 usage  1 implementation
    protected abstract String fetchPage(String url);
    1 usage  1 implementation
    protected abstract String processHtml(String html);
    1 usage  1 implementation
    protected abstract void saveHtml(String url, String html);
    1 usage  1 implementation
    protected abstract void collectStats(String url);
    1 usage  1 implementation
    protected abstract List<String> extractLinks(String html);
}
```

Рис. 2 – Код класу AbstractCrawler

```java
public class SimpleWebCrawler extends AbstractCrawler {

    3 usages
    private final PageFetcherProxy proxy;
    2 usages
    private final PageHandler htmlProcessor;

    1 usage
    public SimpleWebCrawler(PageFetcherProxy proxy, PageHandler htmlProcessor) {
        this.proxy = proxy;
        this.htmlProcessor = htmlProcessor;
    }
    1 usage
    @Override
    protected String fetchPage(String url) {
        proxy.fetchPage(url);
        return proxy.getPageContent(url);
    }


    1 usage
    @Override
    protected String processHtml(String html) {
        PageContext context = new PageContext(html, keyword: "example");
        htmlProcessor.handle(context);
        return context.getHtmlContent();
    }

    1 usage
    @Override
    protected void saveHtml(String url, String html) { System.out.println("Зберігає

    1 usage
    @Override
    protected void collectStats(String url) {
        System.out.println("Збираємо статистику для " + url);
    }

    1 usage
    @Override
    protected List<String> extractLinks(String html) { return Collections.emptyList(); }
}
```

Рис. 3 – Код класу SimpleWebCrawler

Використання цього патерну надає:

1. Уніфікований алгоритм краулінгу: Клас AbstractCrawler визначає фінальний метод crawl(), який задає загальну послідовність дій: завантаження сторінки, обробку HTML, збереження результатів, збір статистики та витягування посилань для подальшого обходу.

2. Гнучкість у реалізації окремих кроків: Кожен крок алгоритму представлений як абстрактний метод, що дозволяє різним нащадкам реалізовувати їх по-різному.
3. Можливість розширення без зміни загальної структури: Новий тип краулера може бути створений шляхом наслідування від AbstractCrawler та реалізації абстрактних методів, без необхідності переписувати загальну логіку обходу.
4. Контрольований порядок виконання: Базовий клас гарантує, що всі етапи краулінгу будуть виконані в правильній послідовності, незалежно від конкретної реалізації.

У нашому випадку патерн Template Method забезпечує структурований підхід до веб-краулінгу, дозволяючи легко створювати спеціалізовані краулери для різних типів веб-сайтів, забезпечувати консистентність процесу збору даних та масштабувати систему шляхом додавання нових реалізацій.

## Структура патерну

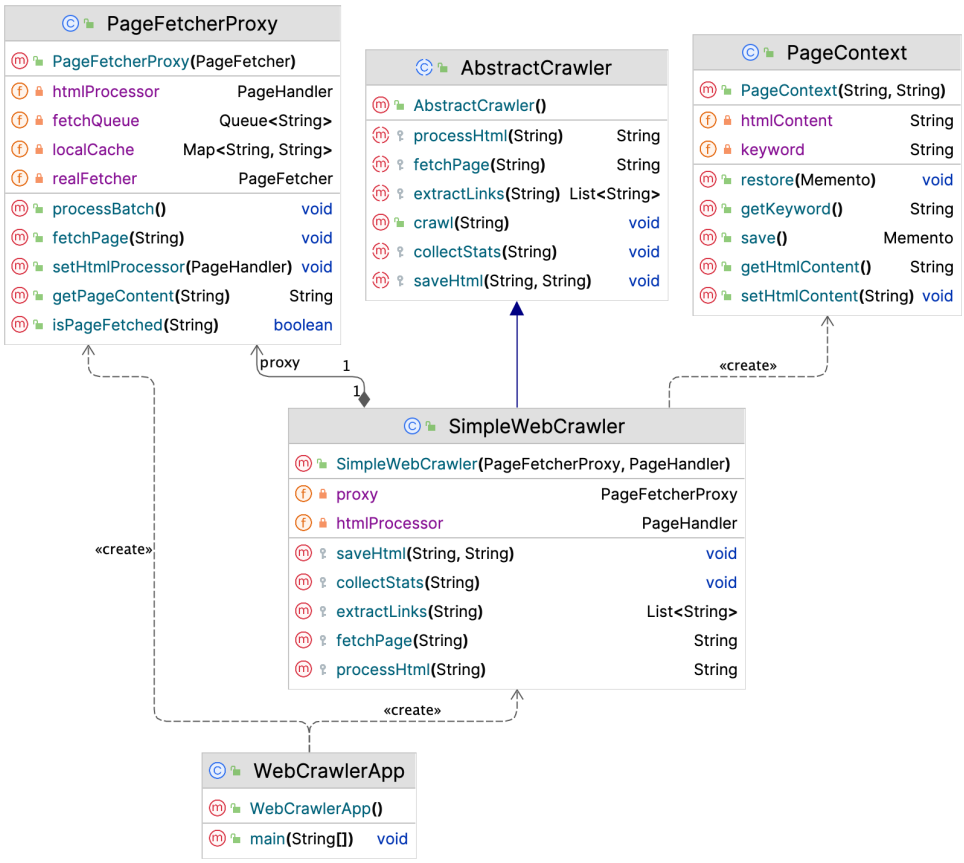



Рис. 4 – Структура патерну Template Method

Опис реалізації:

• AbstractCrawler - абстрактний клас, що визначає шаблонний метод crawl() та абстрактні кроки алгоритму

• SimpleWebCrawler - конкретна реалізація, яка інтегрує Proxy для завантаження сторінок та ланцюжок обробників для трансформації HTML

• crawl() - шаблонний метод, що координує всі етапи обробки веб-сторінки

Такий підхід особливо ефективний для веб–краулерів, де процес збору даних має конкретну послідовність дій, необхідно підтримувати різні стратегії обробки для різних типів контенту та важливо забезпечити можливість легкого додавання нових типів краулерів.

Посилання на репозиторій: https://github.com/dariahorlach/Lab–TRPZ

## Висновки

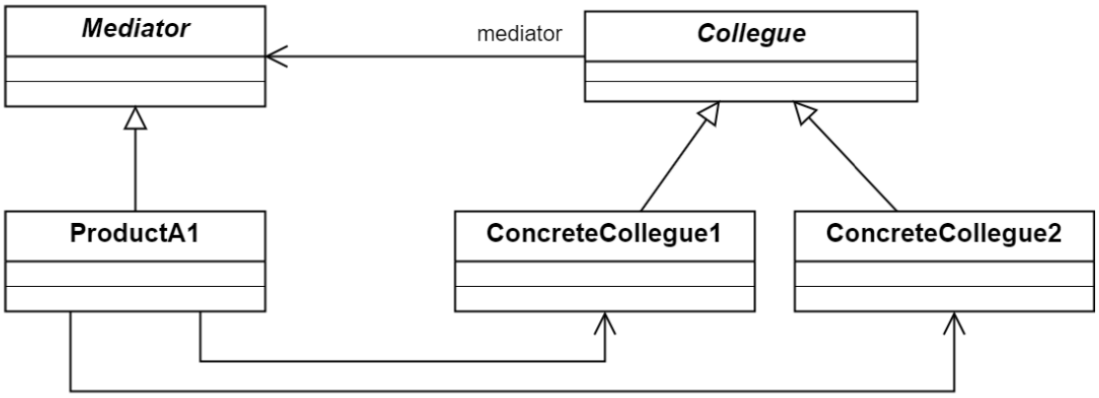
Висновки: під час виконання лабораторної роботи, було реалізовано патерн Template Method для веб–краулера, що дозволило створити структурований та гнучкий фреймворк для обходу веб-сторінок. Абстрактний клас AbstractCrawler визначив загальний алгоритм краулінгу через шаблонний метод crawl(), який інкапсулює стандартну послідовність дій: завантаження сторінки, обробку HTML, збереження результатів, збір статистики та витягування посилань. Ця реалізація заклала основу для створення цілого сімейства веб-краулерів, кожен з яких може спеціалізуватися на конкретних типах контенту або завданнях, зберігаючи при цьому загальну архітектуру та інтерфейси.

## Питання до лабораторної роботи

1. Яке призначення шаблону «Посередник»?

Шаблон «Посередник» (Mediator) забезпечує централізовану взаємодію між об’єктами, щоб вони не взаємодіяли безпосередньо один з одним, зменшуючи кількість зв’язків між класами.

2. Нарисуйте структуру шаблону «Посередник».




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

Mediator – інтерфейс, який визначає метод сповіщення.

ConcreteCollegue – реалізує координацію між компонентами.

Collegue – базовий клас об’єктів, які спілкуються через посередника.

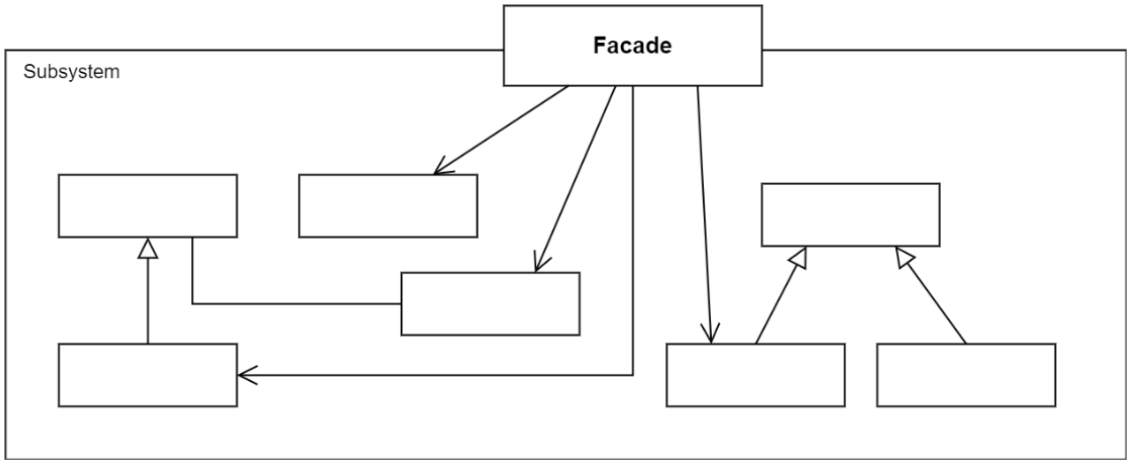
ConcreteComponent – конкретний об’єкт, що надсилає повідомлення посереднику.

Компоненти не звертаються один до одного напряму, а відправляють запити посереднику, який вирішує, хто має на них реагувати.

4. Яке призначення шаблону «Фасад»?

Шаблон «Фасад» (Facade) спрощує роботу зі складною підсистемою, надаючи уніфікований інтерфейс для взаємодії з нею.

5. Нарисуйте структуру шаблону «Фасад».




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

Facade – забезпечує спрощений інтерфейс до складної системи.

Subsystem classes – виконують конкретні функції підсистеми.

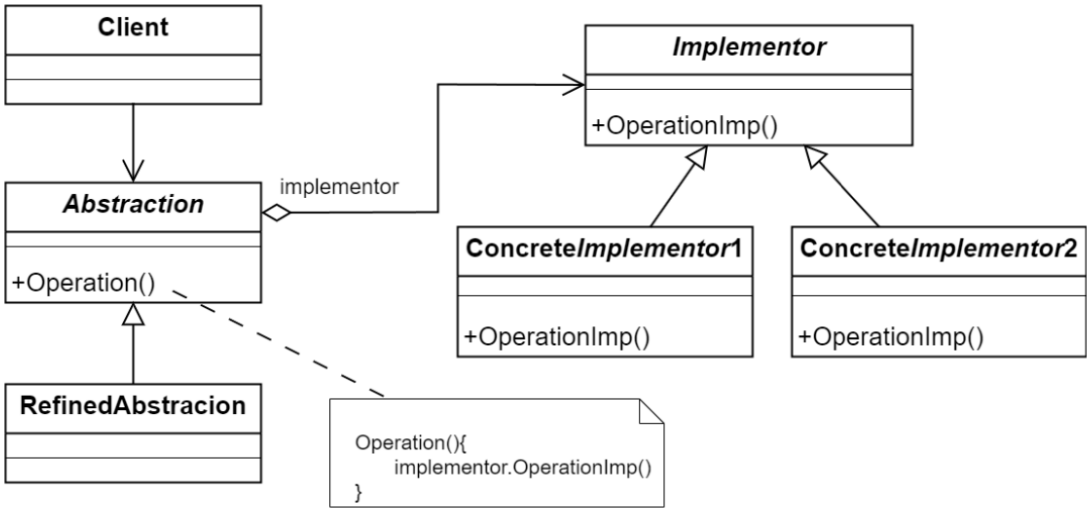
Client – звертається до Facade, а не до підсистем напряму.

Client викликає метод Facade, який усередині звертається до кількох класів підсистеми, приховуючи складність.

7. Яке призначення шаблону «Міст»?

Шаблон «Міст» (Bridge) розділяє абстракцію та реалізацію так, щоб вони могли змінюватися незалежно одна від одної.

8. Нарисуйте структуру шаблону «Міст».




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

Abstraction – визначає високорівневий інтерфейс.

RefinedAbstraction – розширює функціональність Abstraction.

Implementor – інтерфейс для реалізації.

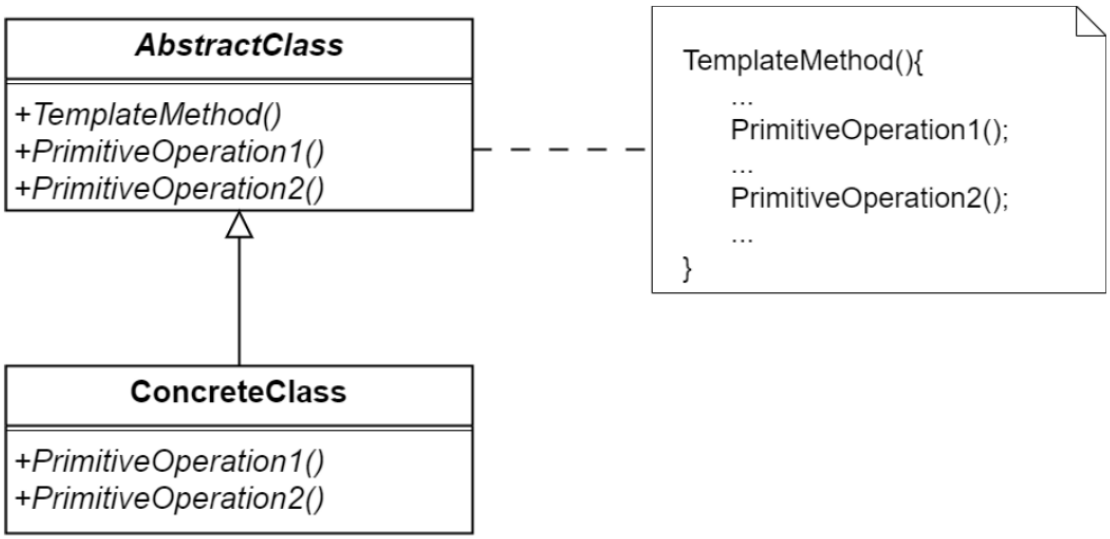
ConcreteImplementor – конкретна реалізація інтерфейсу Implementor.

Abstraction зберігає посилання на Implementor і делегує йому частину роботи. Це дозволяє змінювати реалізацію незалежно від абстракції.

10. Яке призначення шаблону «Шаблоний метод»?

Шаблон «Шаблонний метод» (Template Method) визначає кістяк алгоритму в базовому класі, дозволяючи підкласам перевизначати окремі кроки алгоритму без зміни його структури.

11. Нарисуйте структуру шаблону «Шаблоний метод».




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

AbstractClass – визначає структуру алгоритму через templateMethod() і абстрактні методи.

ConcreteClass – реалізує конкретні кроки алгоритму.

Клас AbstractClass викликає методи, реалізовані в ConcreteClass, щоб виконати певні частини алгоритму.

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

Шаблонний метод задає структуру алгоритму, а фабричний метод контролює створення об’єктів.У Шаблонному методі підкласи змінюють окремі кроки алгоритму, а в фабричному – визначають, який об’єкт створювати.

14. Яку функціональність додає шаблон «Міст»?

Шаблон «Міст» додає гнучкість у зміні або розширенні як абстракції, так і реалізації незалежно одна від одної. Це дозволяє уникнути множинного успадкуванн, розділити логіку на рівні інтерфейсу та реалізації та легко змінювати реалізації в процесі виконання програми.