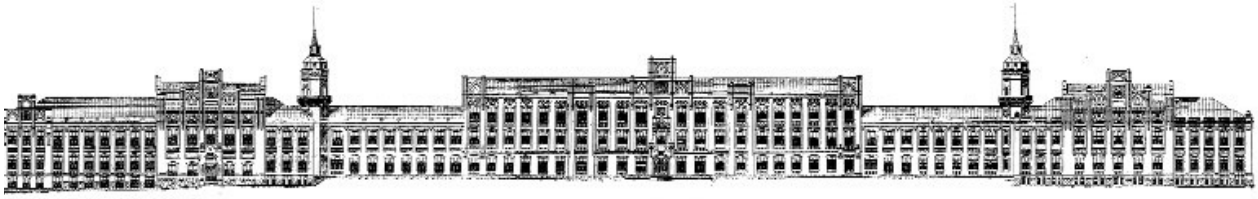


Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №7
Тема: «Патерни проектування»

Виконала
студентка групи ІА-32:
Іванова Анастасія
Юріївна

Перевірив:
Мягкий Михайло
Юрійович

Зміст

Тема проекту

Варіант: 26
Опис теми: 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).

Теоретичні відомості

Патерни проєктування (Design Patterns) — це типові, перевірені на практиці способи розв'язання поширених проблем, які виникають під час проєктування програмного забезпечення.
Вони описують загальні принципи побудови структури класів і взаємодії об'єктів, не прив'язуючись до конкретної мови програмування. Патерни не є готовим кодом — це шаблони, за якими можна побудувати гнучке, масштабоване й зрозуміле рішення.
Використання патернів дозволяє:

- підвищити гнучкість і розширюваність системи;
- зменшити дублювання коду й спростити супровід;
- покращити зрозумілість архітектури для інших розробників;
- забезпечити повторне використання перевірених рішень у різних проєктах.

Template Method (Шаблонний метод) — це поведінковий шаблон проєктування, який дозволяє визначити скелет алгоритму в базовому класі та дозволити підкласам перевизначати окремі кроки цього алгоритму без зміни його загальної структури.
У цьому патерні «каркас» алгоритму задається один раз у абстрактному класі, а специфічні дії реалізуються в похідних класах.
Таким чином, загальна логіка зберігається в одному місці, а змінюються лише ті частини, які відрізняються для різних варіантів використання.

Призначення шаблону

Template Method використовується для випадків, коли алгоритм складається з кількох кроків, а для різних задач або форматів даних реалізація деяких кроків повинна бути різною, але сам порядок виконання залишається незмінним .

Основна ідея

- Виносити спільну частину алгоритму у базовий клас.
- Дозволяти підкласам реалізовувати лише ту логіку, що відрізняється.
- Забезпечувати повторне використання коду та уникати дублювання.

Типові елементи шаблону

- **AbstractClass** — містить шаблонний метод (алгоритм) і визначає абстрактні методи для змінних кроків.
- **ConcreteClass** — перевизначає окремі кроки алгоритму.

Шаблонний метод завжди викликає ті самі підметоди в однаковому порядку, але реалізація цих підметодів змінюється в підкласах.

Переваги

- Значно підвищує повторне використання коду.
- Зменшує дублювання алгоритмів.
- Робить додавання нових реалізацій простим — достатньо створити новий підклас.

Недоліки

– Жорстко задає структуру алгоритму — важко змінювати порядок кроків.
– Може порушити принцип підстановки Лісков, якщо підклас надто змінює поведінку.
– Велика кількість абстрактних кроків у складних алгоритмах ускладнює підтримку.

Хід роботи

Створення діаграми класів патерну

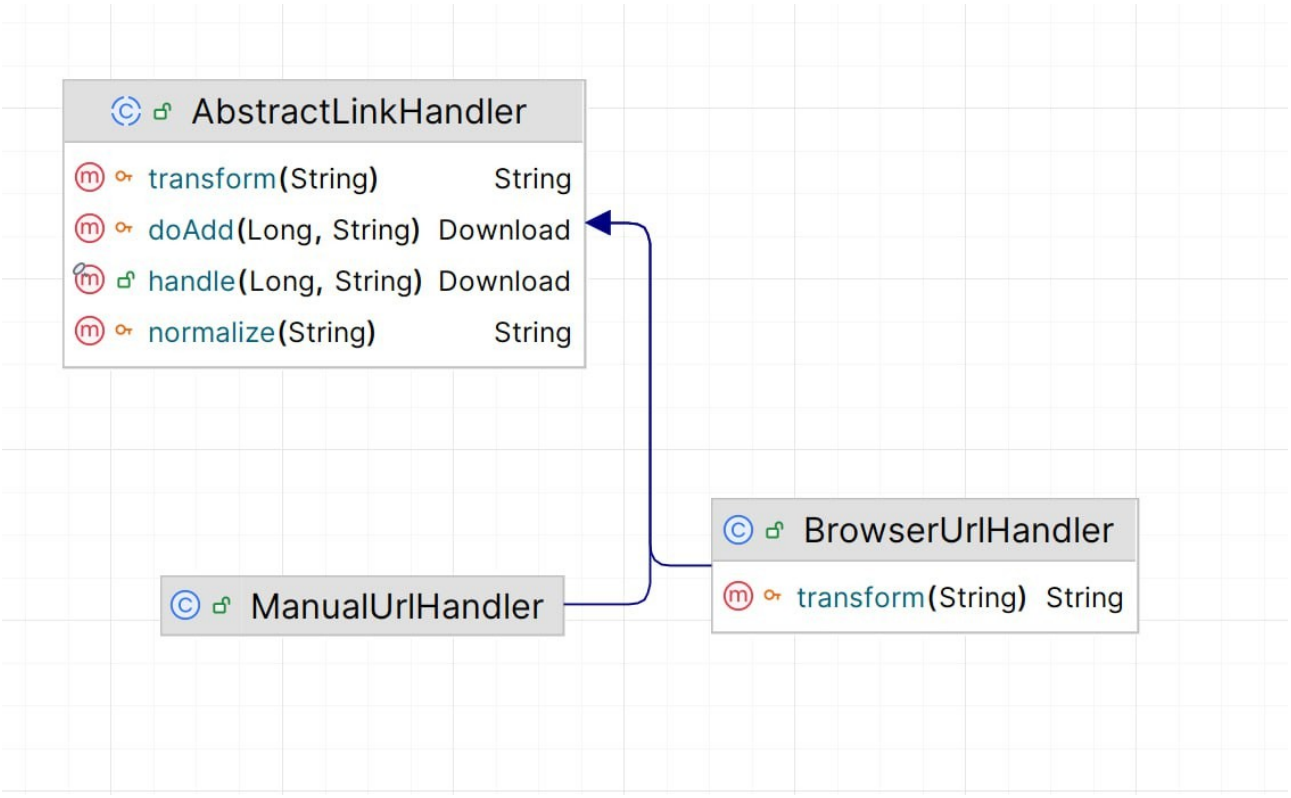


Рис. 1. Діаграма класів патерну Template Method

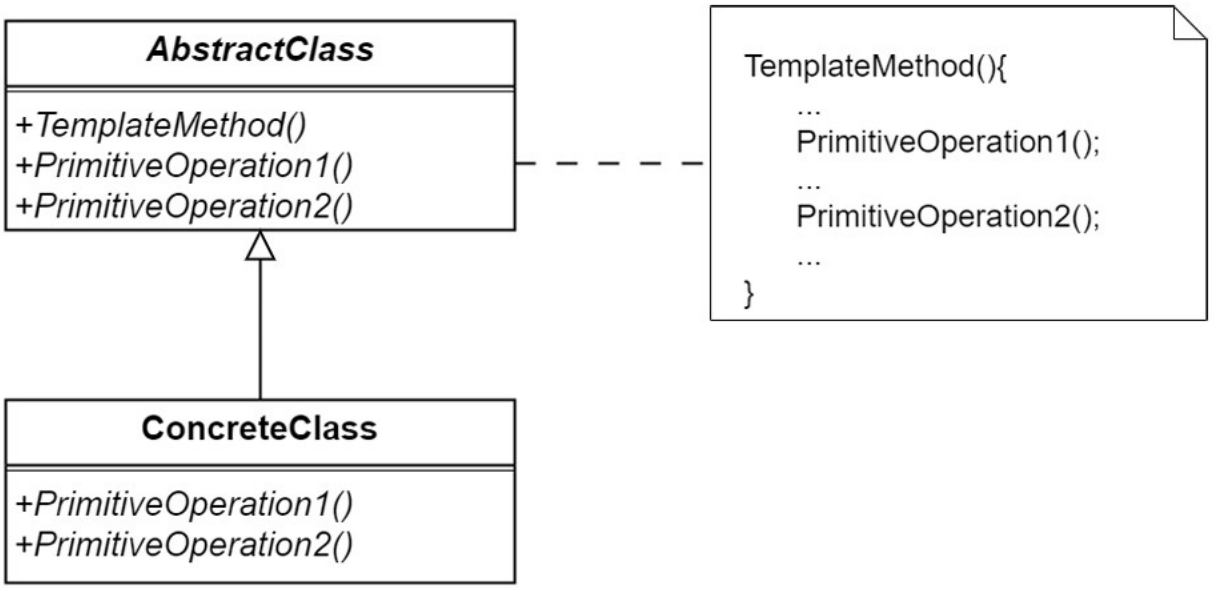


Рис. 2. Шаблон Template Method

1. AbstractClass → AbstractLinkHandler
Призначення:
Задає єдиний алгоритм обробки будь-якого посилання, яке потім перетворюється в завантаження в системі.

Фрагмент коду:

```
public abstract class AbstractLinkHandler {

    protected final DownloadService downloadService;

    public final Download handle(Long userId, String rawUrl) {
        String normalized = normalize(rawUrl);
        String transformed = transform(normalized);
        return doAdd(userId, transformed);
    }

    protected String normalize(String rawUrl) {
        return rawUrl != null ? rawUrl.trim() : "";
    }

    protected String transform(String url) {
        return url;
    }

    protected Download doAdd(Long userId, String url) {
        return downloadService.addUrl(userId, url);
    }
}
```

Роль:
Абстрактний клас.

Метод handle(...) — це TemplateMethod(): він задає фіксовану послідовність кроків:

1. normalize(rawUrl) – привести рядок до нормального вигляду;
2. transform(url) – виконати специфічні перетворення для конкретного джерела посилання;
3. doAdd(userId, url) – створити завантаження через DownloadService.

Методи normalize, transform, doAdd — це primitive operations (примітивні операції), які можуть бути перевизначені в підкласах.
Завдяки ключовому слову final у handle(...) підкласи не можуть змінити сам алгоритм, а лише деталі його кроків — саме так і працює Template Method.
У моєму застосунку будь-яка логіка “додати завантаження з URL” проходить через цей шаблонний метод, що гарантує однакову послідовність дій для всіх джерел посилань.

2. ConcreteClass → ManualUrlHandler
Призначення:
Конкретна реалізація Template Method для ручного введення URL на сторінці /downloads.
Фрагмент коду:

```
@Service
public class ManualUrlHandler extends AbstractLinkHandler {

   public ManualUrlHandler(DownloadService downloadService) {
      super(downloadService);
   }
}
```

Роль:
ManualUrlHandler це конкретний клас.

- успадковує handle(...) з AbstractLinkHandler.
  Тобто для ручних URL використовується той самий шаблон алгоритму.
- Викликається в DownloadController при додаванні посилання з форми:

```
manualUrlHandler.handle(user.getId(), form.getUrl());
```

3. ConcreteClass → BrowserUrlHandler
Призначення:
Конкретна реалізація Template Method для посилань, що приходять з браузерних розширень (Chrome / Edge / Firefox).
Фрагмент коду:

```
@Service
public class BrowserUrlHandler extends AbstractLinkHandler {

   public BrowserUrlHandler(DownloadService downloadService) {
      super(downloadService);
   }

   @Override
   protected String transform(String url) {
```

```
        if (url != null && url.startsWith("view-source:")) {
            return url.substring("view-source:".length()).trim();
        }
        return url;
    }
}
```

Роль:

BrowserUrlHandler — ще один ConcreteClass у структурі Template Method.

- Так само успадковує шаблонний метод handle(...), але перевизначає примітивну операцію transform():
  може обрізати службові префікси (наприклад, view-source:);

Висновки:

Патерн Template Method у проєкті використано для уніфікації процесу обробки посилань перед створенням завантаження.
Єдина послідовність кроків визначена в абстрактному класі AbstractLinkHandler, а конкретні реалізації можуть змінювати лише окремі етапи алгоритму, не порушуючи його структури. Це дозволяє централізувати загальну логіку, уникнути дублювання коду та легко розширювати систему новими способами обробки URL.
У моєму застосунку:

- AbstractLinkHandler є шаблонним "каркасом" алгоритму, який задає фіксовані кроки: нормалізація → трансформація → створення завантаження.
- ManualUrlHandler та BrowserUrlHandler — конкретні реалізації, що використовують спільний алгоритм, але можуть адаптувати окремі операції під свій сценарій (наприклад, трансформацію посилання в браузерній інтеграції).

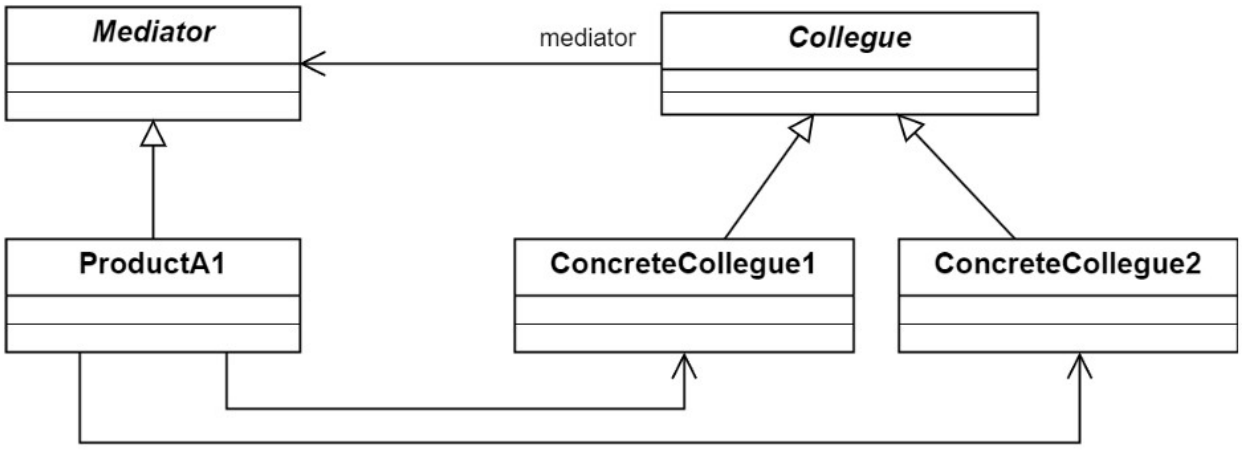
Завдяки Template Method процес додавання завантажень став розширюваним, контрольованим і структуровано єдиним, а зміна логіки обробки URL не потребує переписування контролерів чи дублювання коду — достатньо створити новий Handler.

Контрольні питання

1. Яке призначення шаблону «Посередник»?

Шаблон Посередник (Mediator) використовується для організації взаємодії між багатьма об’єктами через один центральний об’єкт — *медіатора*. Це дозволяє зменшити кількість прямих залежностей між класами та спростити підтримку системи.

2. Нарисуйте структуру шаблону «Посередник».



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

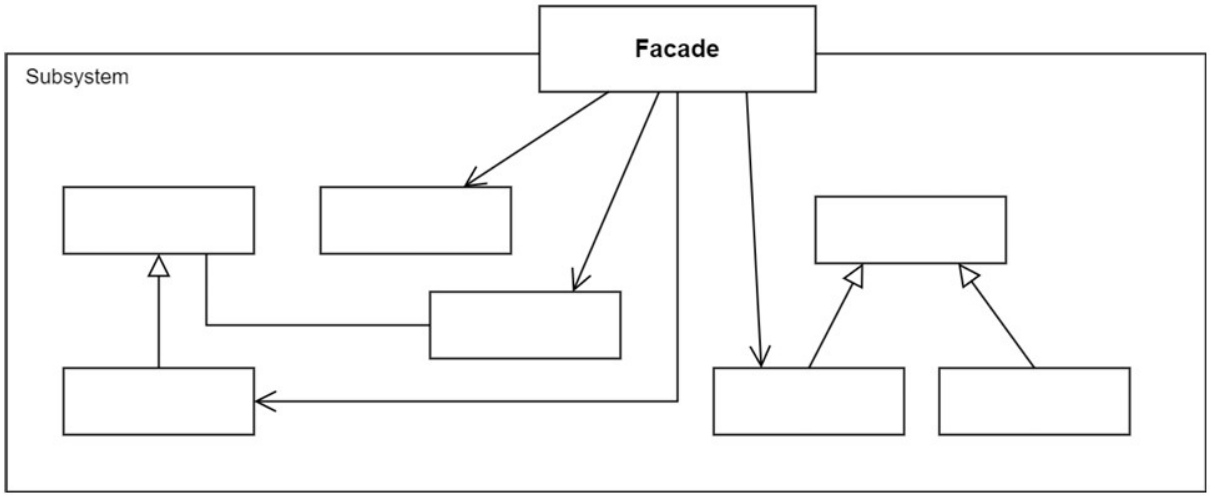
- **Mediator** — інтерфейс або абстрактний клас посередника.
- **ConcreteMediator** — реалізація, що координує взаємодію між об’єктами.
- **Colleague** — клас-учасник, який взаємодіє через медіатора.
- **ConcreteColleague** — конкретні об’єкти, що передають повідомлення медіатору.

Усі об’єкти-компоненти взаємодіють *лише через медіатора*, а не між собою напряму.

4. Яке призначення шаблону «Фасад»?

Шаблон Фасад (Facade) надає спрощений, уніфікований інтерфейс до складної підсистеми, приховуючи деталі реалізації. Він допомагає зменшити залежність клієнта від внутрішніх класів системи.

5. Нарисуйте структуру шаблону «Фасад».



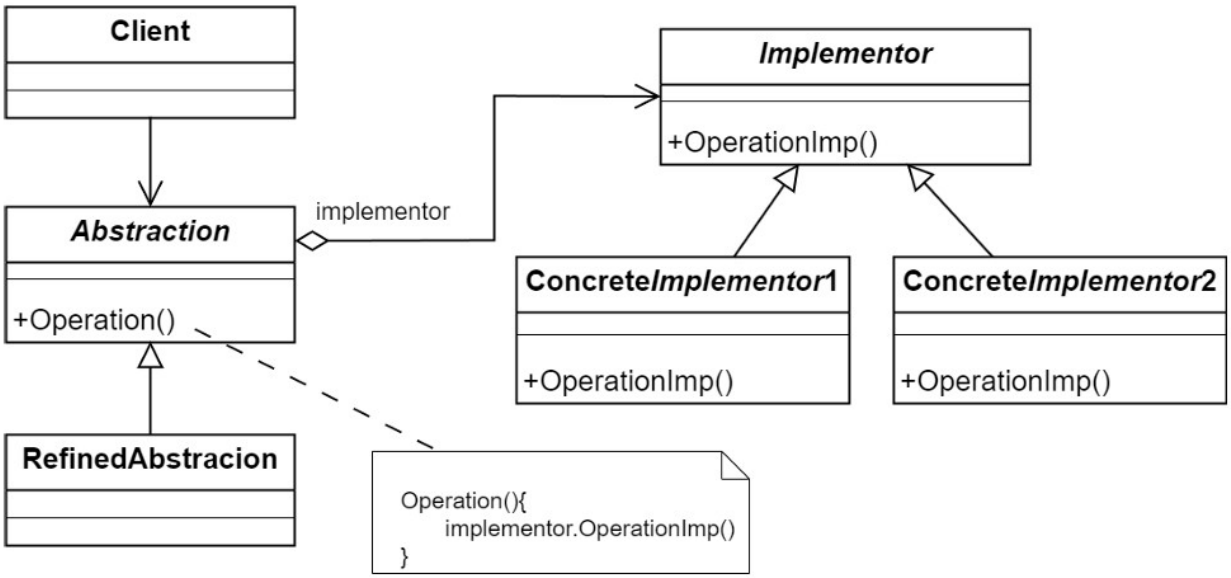
6. Які класи входять у шаблон «Фасад», та яка між ними взаємодія?

- **Facade** — єдиний доступний клієнту клас, що інкапсулює складну логіку.
- **Subsystem classes** — набір внутрішніх класів підсистеми.
  Клієнт працює тільки з Facade, який передає виклики внутрішнім класам.

7. Яке призначення шаблону «Міст»?

Шаблон Міст (Bridge) розділяє абстракцію та її реалізацію на різні ієрархії, дозволяючи розвивати їх незалежно одна від одної. Він усуває вибухове зростання кількості підкласів при поєднанні різних абстракцій із різними реалізаціями.

8. Нарисуйте структуру шаблону «Міст».



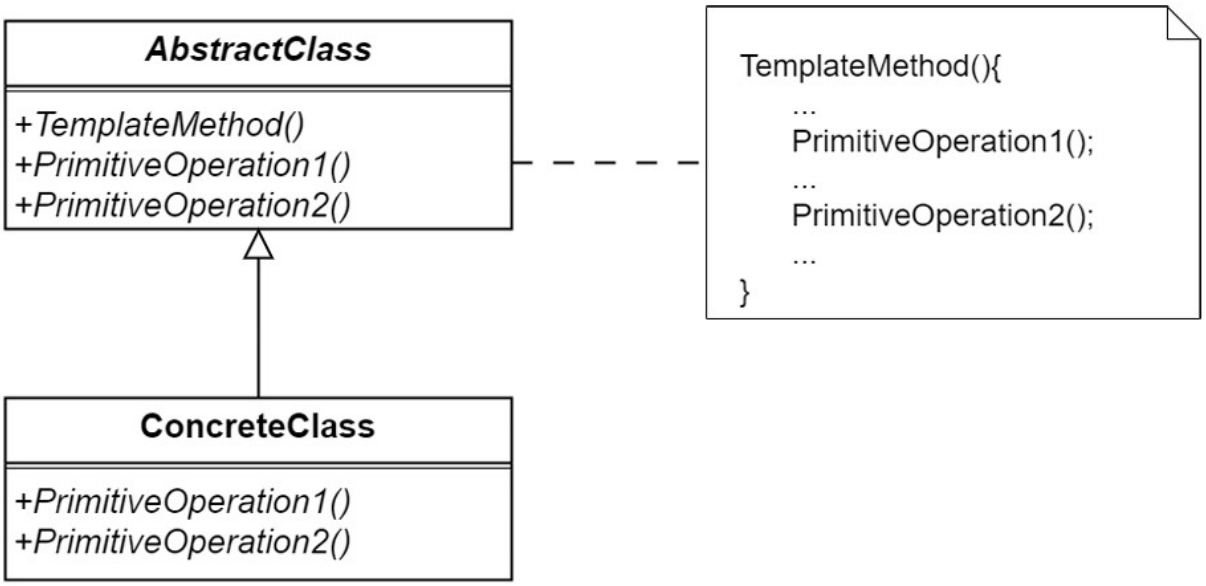
9. Які класи входять у шаблон «Міст», та яка між ними взаємодія?

- **Abstraction** — базовий інтерфейс абстракції.
- **RefinedAbstraction** — розширена абстракція.
- **Implementor** — інтерфейс реалізації.
- **ConcreteImplementor** — конкретна реалізація.

Abstraction містить посилання на Implementor і делегує йому конкретні операції.

10. Яке призначення шаблону «Шаблонний метод»?
Патерн Шаблонний метод (Template Method) визначає загальний алгоритм у базовому класі та дозволяє підкласам перевизначати окремі його кроки, не змінюючи структуру алгоритму.

11. Нарисуйте структуру шаблону «Шаблоний метод».



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

- **AbstractClass** — містить шаблонний метод (алгоритм).
- **ConcreteClass** — реалізує окремі кроки алгоритму.

Шаблонний метод викликає кроки в певному порядку; реалізація цих кроків визначається в підкласах.

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

- **Template Method** регулює послідовність виконання алгоритму, дозволяючи змінювати окремі кроки.
- **Factory Method** регулює створення об’єктів і дозволяє підкласам визначати, який саме об’єкт створити.
  Template Method описує алгоритм, а Factory Method описує створення екземплярів.

14. Яку функціональність додає шаблон «Міст»?
Патерн дає змогу незалежно розвивати абстракцію та її реалізацію, легко розширювати систему новими абстракціями або реалізаціями без зміни існуючих класів.