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

Лабораторна робота № 7

із дисципліни «Технології розроблення програмного забезпечення»

Тема: «Патерни проектування»

Виконав
Студент групи ІА-31: Перевірив:
Губар Б. О. Мягкий М. Ю.

Київ 2025

Зміст

1. Мета:

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

2. Теоретичні відомості

Шаблон	Опис	Переваги	Недоліки
«Mediator» (Посередник)	Призначення: Централізує взаємодію між об'єктами через посередника, замість прямих зв'язків. Кожен об'єкт зберігає лише посилання на медіатор. Застосування: Складні форми з великою кількістю взаємодіючих компонентів (наприклад, чекбокси, блоки, що ховаються/показуються).	Зменшення зв'язаності Простота розширення Легке тестування	Медіатор може стати «God Object».
«Facade» (Фасад)	Призначення: Надає спрощений уніфікований інтерфейс до складної підсистеми, приховуючи її внутрішню структуру. Застосування: Робота з різними протоколами (HTTP, TCP), складні бібліотеки.	Інкапсуляція складності Простіший API Легке оновлення внутрішньої реалізації	Зменшення гнучості
«Bridge» (Міст)	Призначення: Розділяє абстракцію та її реалізацію, дозволяючи змінювати їх незалежно. Утворює дві ієрархії. Застосування: Графічні редактори (фігури + драйвери: екран, принтер, bitmap).	Незалежний розвиток абстракції та реалізації Гнучкість	Збільшення складності
«Template Method» (Шаблонний метод)	Призначення: Визначає скелет алгоритму в базовому класі, залишаючи окремі кроки для перевизначення в похідних. Застосування: Обробка різних форматів відео (MPEG-4, MPEG-2), компіляція веб-сторінок.	Повторне використання коду Чітка структура алгоритму	Жорсткий скелет Можливе порушення принципу Лісков Складність при багатьох кроках

3. Хід роботи

Тема : CI server (state, command, decorator, mediator, visitor, soa)

Обґрунтування застосування патерну

Для системи реалізовано патерн Mediator (Посередник) для зменшення зв'язності (Coupling) між компонентами сервера. У веб-застосунках часто виникає проблема "товстих контролерів" (Fat Controllers), коли клас контролера (AgentController) не

тільки приймає HTTP-запити, а й виконує бізнес-логіку: звертається до репозиторіїв, оновлює статуси в БД, розраховує час виконання тощо. Це робить контролер складним для підтримки та тестування. Використання патерну Mediator дозволило винести логіку взаємодії між компонентами в окремий клас — CiMediator. Тепер контролер лише сповіщає посередника про подію (наприклад, "JobFinished"), а посередник сам вирішує, які сервіси чи репозиторії потрібно задіяти для обробки цієї події.

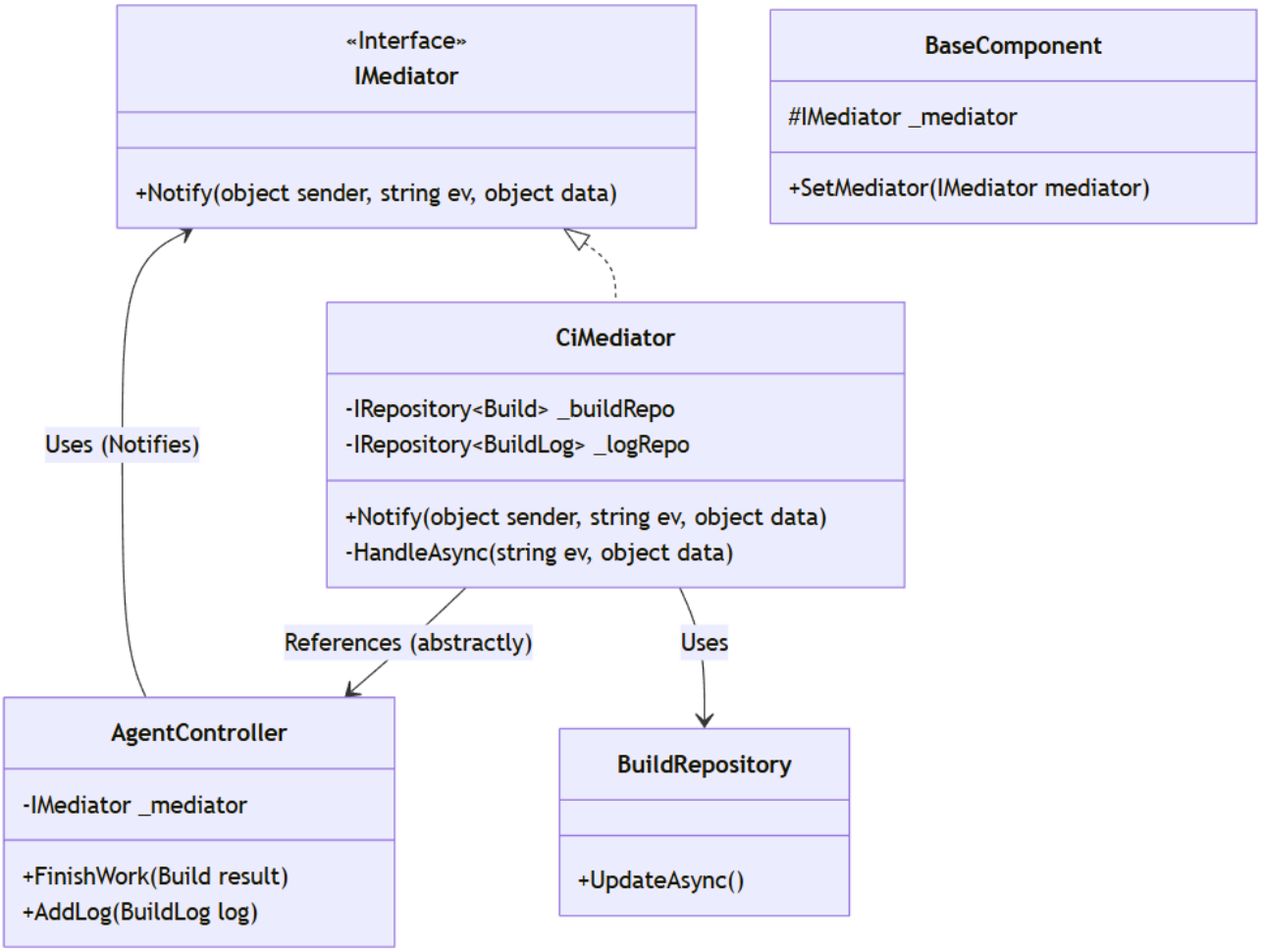


Рисунок 1 - Структура патерну Mediator

IMediator (Mediator) – інтерфейс, що визначає метод Notify. Він дозволяє компонентам спілкуватися, не знаючи конкретних класів один одного.

CiMediator (Concrete Mediator) – реалізує логіку координації. Він містить залежності від репозиторіїв (_buildRepo, _logRepo). Коли він отримує повідомлення (наприклад, "JobFinished"), він знаходить відповідний білд у базі, оновлює його статус та час завершення.

AgentController (Component/Colleague) – компонент, який ініціює події. У методах FinishWork та AddLog він не працює з базою даних напряму. Замість цього він викликає _mediator.Notify().

Перевага: Контролер нічого не знає про те, як саме зберігається результат збірки. Якщо логіка збереження зміниться (наприклад, додасться відправка email), нам треба буде змінити лише CiMediator, не чіпаючи контролер.

```
namespace CiServer.Core.Mediator;

0 references
public interface IMediator
{
    0 references
    void Notify(object sender, string ev, object? data = null);
}
```

Рисунок 2 – показуємо метод Notify

```
public class CiMediator : IMediator
{
    3 references
    private readonly IRepository<Build, Guid> _buildRepo;
    3 references
    private readonly IRepository<BuildLog, Guid> _logRepo;

    0 references
    public CiMediator(
        IRepository<Build, Guid> buildRepo,
        IRepository<BuildLog, Guid> logRepo)
    {
        _buildRepo = buildRepo;
        _logRepo = logRepo;
    }
    0 references
    public void Notify(object sender, string ev, object? data = null)
    {
        HandleAsync(ev, data).Wait();
    }
    1 reference
    private async Task HandleAsync(string ev, object? data)
    {
        if (ev == "JobFinished" && data is Build resultBuild)
        {
            var build = await _buildRepo.GetByIdAsync(resultBuild.BuildId);
            if (build != null)
            {
                build.RestoreState();
                bool isSuccess = resultBuild.Status == BuildStatus.Success;
                build.Finish(isSuccess);
                build.EndTime = DateTime.UtcNow;
                await _buildRepo.SaveChangesAsync();
                Console.WriteLine($"[MEDIATOR] Build {build.BuildId} finalized. Status: {build.Status}");
            }
        }
        else if (ev == "LogReceived" && data is BuildLog log)
        {
            await _logRepo.AddAsync(log);
            await _logRepo.SaveChangesAsync();
            Console.WriteLine($"[MEDIATOR] Log saved: {log.Content}");
        }
```

Рисунок 3 - Клас CiMediator.cs (серце патерну)

```
[HttpPost("finish")]
0 references
public IActionResult FinishWork([FromBody] Build result)
{
    _mediator.Notify(this, "JobFinished", result);
    return Ok();
}

[HttpPost("log")]
0 references
public IActionResult AddLog([FromBody] BuildLog log)
{
    _mediator.Notify(this, "LogReceived", log);
    return Ok();
}
```

Рисунок 4 - Відправник

```
[MEDIATOR] Build ab2317c5-36b3-4233-aba1-20dbdc5dca88 finalized. Status: Success
```

Рисунок 5 – Agent => Controller => Mediator => Logic

4. Висновок

У ході виконання лабораторної роботи було розглянуто поведінкові шаблони проєктування. На прикладі системи «CI Server» було реалізовано шаблон Mediator для організації взаємодії між API-контролерами та шаром даних.

Впровадження класу CiMediator дозволило ізолювати AgentController від прямої залежності з базою даних та бізнес-логікою завершення збірки. Контролер виконує лише функцію маршрутизації HTTP-запитів, делегуючи обробку подій посереднику. Це забезпечило слабку зв'язність компонентів, спростило код контролерів і дозволило централізувати логіку обробки результатів збірки в одному місці, що значно полегшує подальшу підтримку та розширення системи.

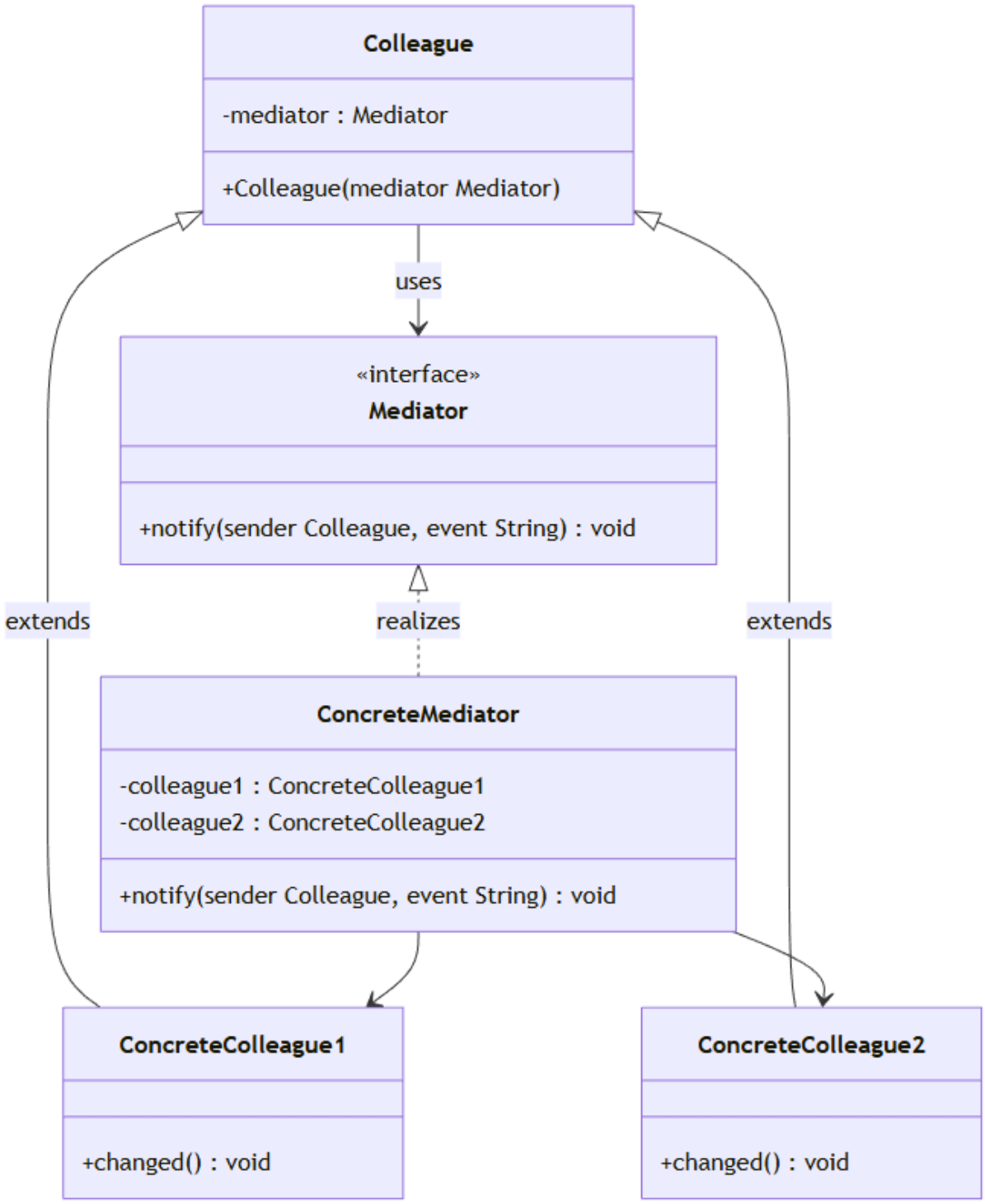
5. Контрольні питання

1. Яке призначення шаблону «Посередник»?

Централізує взаємодію між об’єктами через єдиний об’єкт-посередник, усуваючи прямі зв’язки між ними. Зменшує зв’язність, спрощує логіку взаємодії.

2. Нарисуйте структуру шаблону «Посередник».

м



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

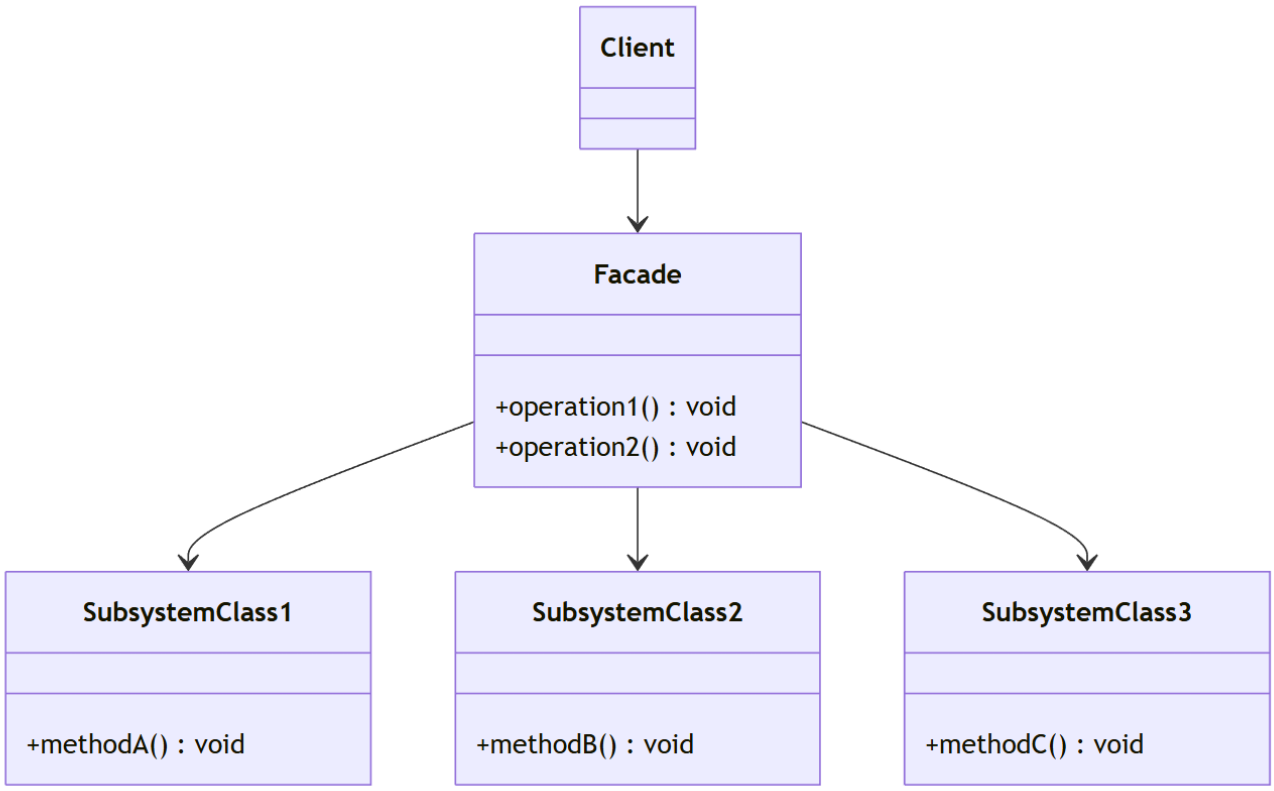
- Mediator — інтерфейс посередника
- ConcreteMediator — реалізація, координує взаємодію
- Colleague — базовий клас компонентів

- ConcreteColleague1, ConcreteColleague2 — конкретні компоненти

4. Яке призначення шаблону «Фасад»?

Надає спрощений уніфікований інтерфейс до складної підсистеми, приховуючи її внутрішню складність.

5. Нарисуйте структуру шаблону «Фасад».



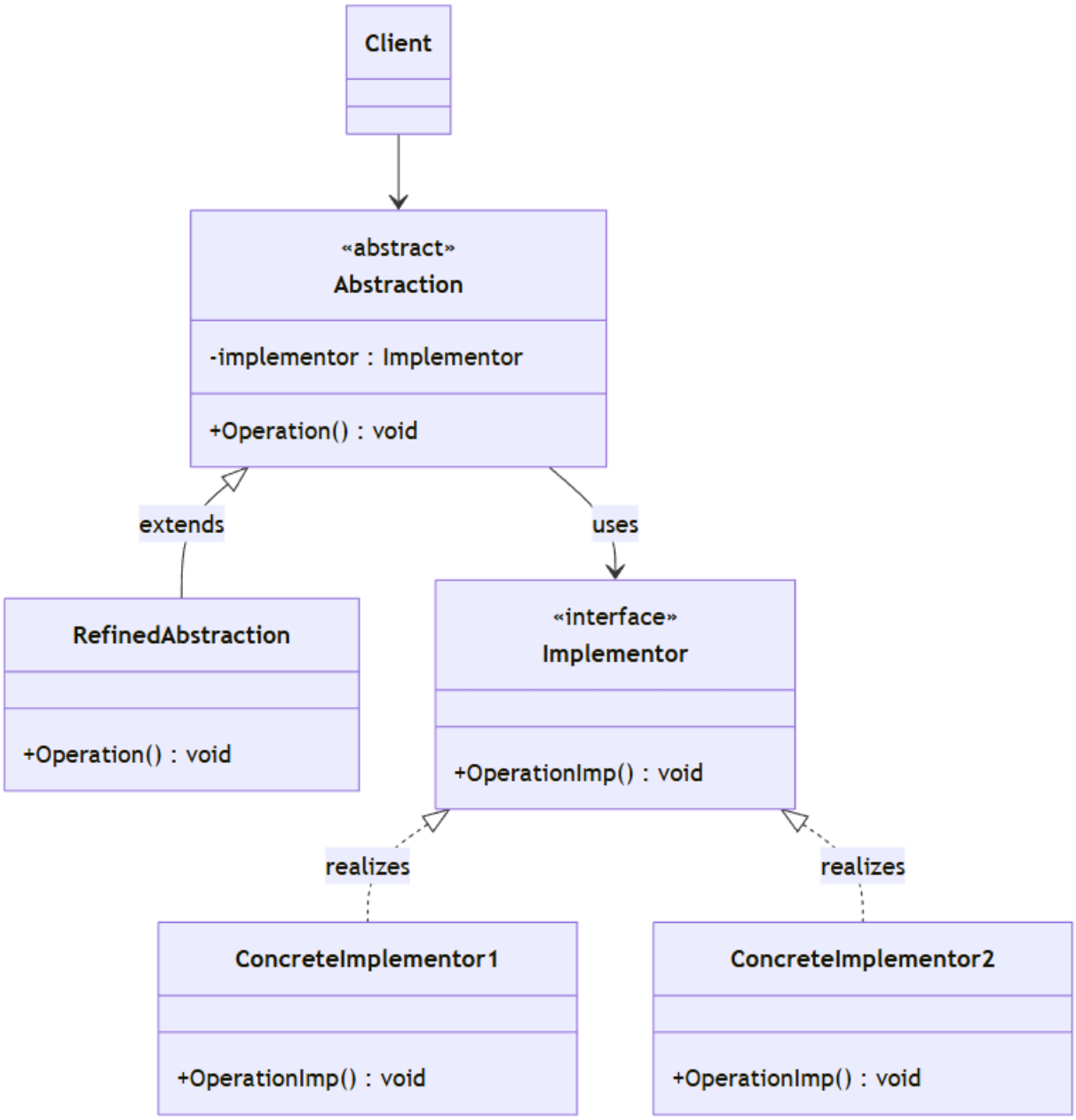
6. Які класи входять в шаблон «Фасад», та яка між ними взаємодія?

- Facade — єдиний інтерфейс
- SubsystemA, SubsystemB, SubsystemC — класи підсистеми

7. Яке призначення шаблону «Міст»?

Розділяє абстракцію та її реалізацію, дозволяючи змінювати їх незалежно (дві ієрархії).

8. Нарисуйте структуру шаблону «Міст».



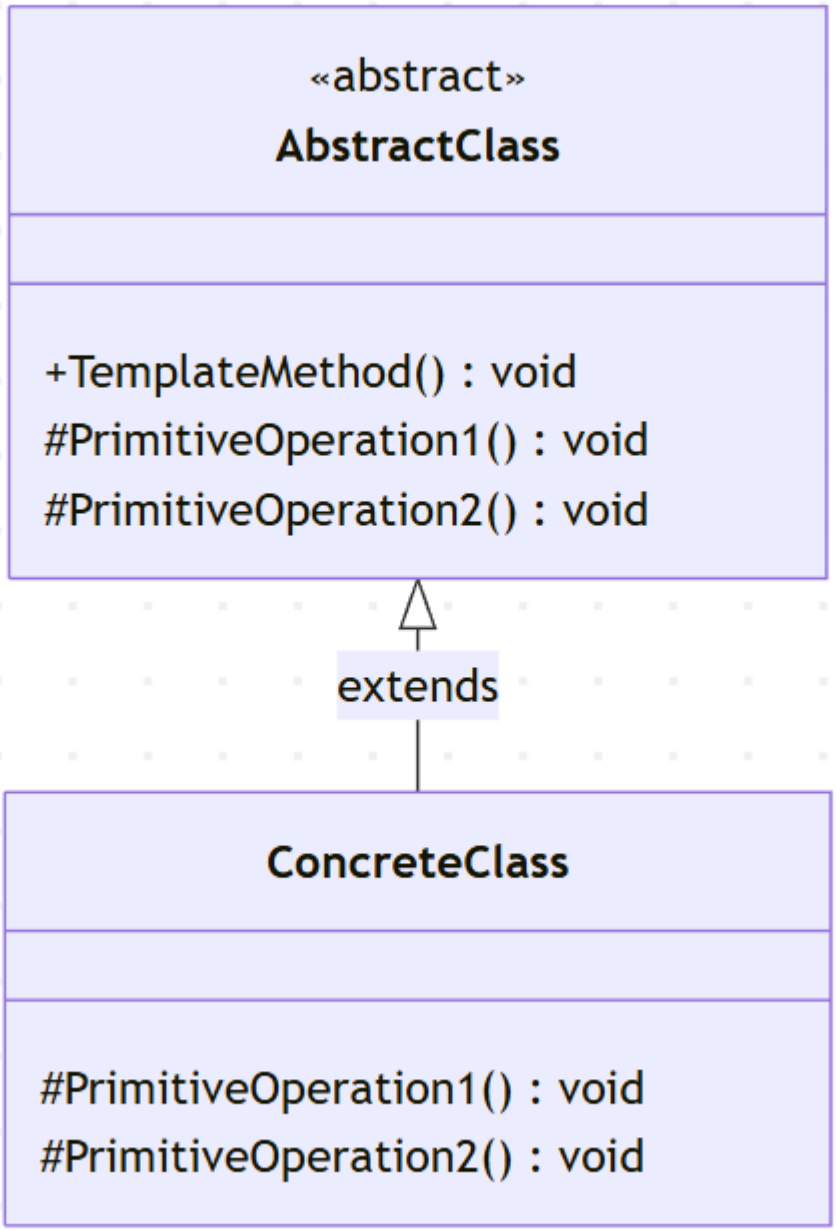
9. Які класи входять в шаблон «Міст», та яка між ними взаємодія?

- Abstraction — абстракція
- RefinedAbstraction — розширена абстракція
- Implementor — інтерфейс реалізації
- ConcreteImplementorA/B — конкретні реалізації

10. Яке призначення шаблону «Шаблонний метод»?

Визначає скелет алгоритму в базовому класі, залишаючи окремі кроки для перевизначення в похідних.

11. Нарисуйте структуру шаблону «Шаблонний метод».



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

- AbstractClass — містить TemplateMethod() і абстрактні методи
- ConcreteClass — реалізує абстрактні методи

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

Шаблонний метод» визначає скелет алгоритму, дозволяючи підкласам перевизначати окремі кроки. Фабричний метод» визначає інтерфейс створення об’єкта, залишаючи підкласам вибір конкретного класу.

14. Яку функціональність додає шаблон «Міст»?

Дозволяє незалежно розвивати абстракцію та реалізацію.

- Додає гнучкість
- Усуває експоненційне зростання підкласів
- Підтримує принцип розділення відповідальностей