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 № 5

із дисципліни «Технології розроблення програмного забезпечення»

Тема: «Патерни проектування»

Виконав
Студент групи ІА-31:
Козир Я. О.

Перевірив:
Мягкий М. Ю.

Київ 2025

Зміст

1. Мета: 3
2. Теоретичні відомості: 3
3. Хід роботи: 3
4. Висновок 6
5. Контрольні питання: 7

1. **Мета:**

Вивчити структуру шаблонів «Adapter», «Builder», «Command», «Chain of responsibility», «Prototype» та навчитися застосовувати їх в реалізації програмної системи.

2. **Теоретичні відомості:**

Adapter: Адаптує інтерфейс одного об’єкта до іншого, дозволяючи несумісним компонентам працювати разом (наприклад, уніфікація інтерфейсів бібліотек принтерів). Спрощує інтеграцію без змін у коді.
Builder: Відокремлює процес створення об’єкта від його представлення, доречний для складних або багатоформних конструкцій (наприклад, відповіді веб-сервера). Забезпечує гнучкий контроль.
Command: Перетворює виклик методу в об’єкт, дозволяючи гнучкі системи команд із відміною, логуванням чи плануванням (наприклад, дії в інтерфейсі). Покращує модульність.
Chain of Responsibility: Передає запити по ланцюжку обробників, поки один не виконає (наприклад, контекстні меню). Зменшує зв’язки та спрощує зміни.
Prototype: Створює об’єкти клонуванням прототипу, корисний для складних об’єктів або динамічних змін (наприклад, редактор рівнів гри). Зменшує ієрархію спадкування.

3. **Хід роботи:**

Тема :

21. **Online radio station** (iterator, adapter, factory method, facade, visitor, client-server)

Додаток повинен служити сервером для радіостанції з можливістю мовлення на радіостанцію (64, 92, 128, 196, 224 kb/s) в потоковому режимі; вести облік підключених користувачів і статистику відвідувань і прослуховувань; налаштувати папки з піснями і можливість вести списки програвання або playlists (не відтворювати всі пісні).

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.

Патерн: **Adapter** – інтеграція зовнішньої бібліотеки FFmpeg (консольний процес) у C#-код.

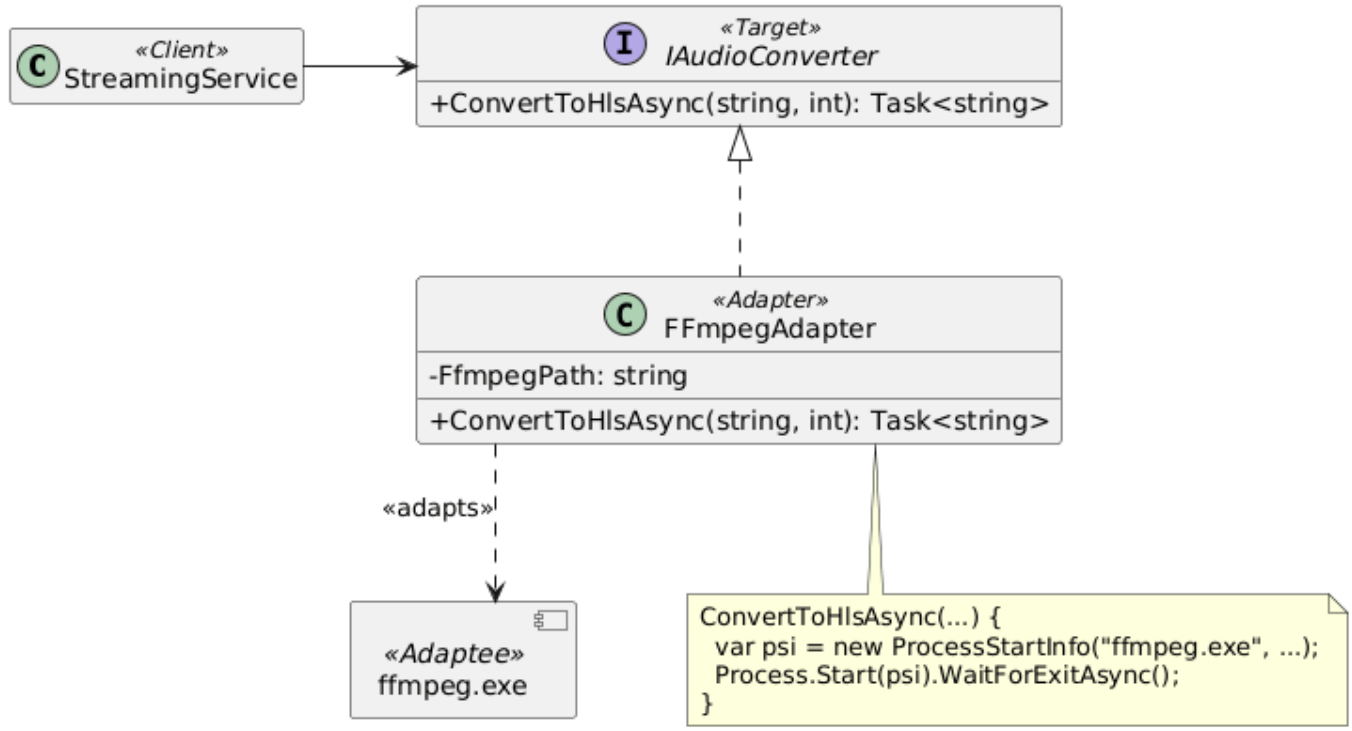


Рисунок 1 - Структура патерну Adapter

У даному застосунку реалізовано структурний шаблон проектування Adapter, який дозволяє узгодити інтерфейс зовнішнього інструменту з вимогами внутрішнього коду.
Інтерфейс IAudioConverter виступає контрактом для перетворення аудіо — він визначає метод ConvertToHlsAsync(...), який має повертати шлях до згенерованого HLS-плейлиста.
Клас FFmpegAdapter реалізує цей інтерфейс і адаптує консольну утиліту ffmpeg.exe до внутрішнього контракту. Адаптер формує та запускає процес ffmpeg із потрібними параметрами, очікуючи завершення конвертації, після чого повертає результат.
Клас StreamingService виступає клієнтом, який працює лише з інтерфейсом IAudioConverter і не знає про існування ffmpeg.exe чи особливості його виклику.
Такий підхід забезпечує гнучкість і розширюваність системи — у майбутньому можна легко замінити FFmpegAdapter на іншу реалізацію (наприклад, LibAvAdapter), не змінюючи бізнес-логіку клієнта.

```
public class FFmpegAdapter : IAudioConverter
    private const string FfmpegPath = "wwwroot/ffmpeg/ffmpeg.exe";

    2 references
    public async Task<string> ConvertToHlsAsync(string inputPath, int bitrate)
    {
        var outputFolder = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "streams", Path.GetFileNameWithoutExtension(ir
        Directory.CreateDirectory(outputFolder);
        var playlist = Path.Combine(outputFolder, "stream.m3u8");

        var args = $"-i \"{inputPath}\" " +
                   $"-c:a aac -b:a {bitrate}k " +
                   $"-f hls -hls_time 10 -hls_list_size 0 " +
                   $"-hls_segment_filename \"{outputFolder}/seg%03d.ts\" " +
                   $"\"{playlist}\"";

        var psi = new ProcessStartInfo
        {
            FileName = FfmpegPath,
            Arguments = args,
            UseShellExecute = false,
            RedirectStandardOutput = true,
            RedirectStandardError = true,
            CreateNoWindow = true
        };

        using var process = Process.Start(psi)!;
        await process.WaitForExitAsync();

        if (process.ExitCode != 0)
            throw new System.Exception(await process.StandardError.ReadToEndAsync());

        return playlist;
```

Рисунок 2 – FfmpegAdapter.cs

```
using System.Threading.Tasks;

namespace OnlineRadioStation.Domain
{
    4 references
    public interface IAudioConverter
    {
        2 references
        Task<string> ConvertToHlsAsync(string inputPath, int bitrate);
    }
}
```

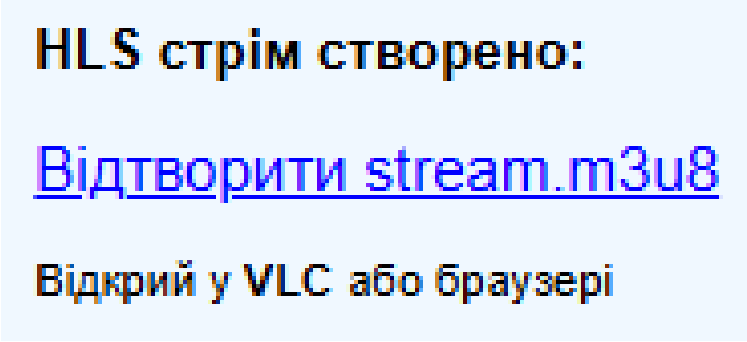
Рисунок 3 – IAudioProcessor.cs

```
public async Task<string> CreateHlsStreamAsync(string mp3Path, int bitrate = 128)
{
    if (_converter == null)
        throw new InvalidOperationException("IAudioConverter не зареєстровано в DI");

    return await _converter.ConvertToHlsAsync(mp3Path, bitrate);
}
```

Рисунок 4 – StreamingService.cs

```
HttpGet("adapter")]
 references
ublic async Task<IActionResult> AdapterTest()

    var mp3Path = Path.Combine(_webRootPath, "samples", "demo.mp3");

    if (!System.IO.File.Exists(mp3Path))
    {
        return new ContentResult
        {
            Content = "<p style='color: red; font-weight: bold;'>ПОМИЛКА:</p>" +
                      "<p>Файл <code>wwwroot/samples/demo.mp3</code> не знайдено!</p>" +
                      "<p>Додай будь-який .mp3 файл (5–10 сек).</p>",
            ContentType = "text/html; charset=utf-8"
        };
    }

    try
    {
        var hlsUrl = await _streamingService.CreateHlsStreamAsync(mp3Path, 128);


        var webPath = "/" + Path.GetRelativePath(_webRootPath, hlsUrl).Replace("\\", "/");

        return new ContentResult
        {
            Content = "<div style='font-family: Arial; padding: 20px; background: #f0f8ff; border-radius: 10px;'>" +
                      "<h3 style='color: #006400;'>ЛР5: Adapter (FFmpeg) — УСПІХ!</h3>" +
                      "<p><strong>HLS стрім створено:</strong></p>" +
                      $"<p><a href='{webPath}' target='_blank' style='font-size: 18px; color: #0000ff; text-decoration: unde
                      "<p><small>Відкрий у <strong>VLC</strong> або браузері</small></p>" +
                      "<hr><p><em>Файли: <code>wwwroot/streams/demo/</code></em></p>" +
                      "</div>",
            ContentType = "text/html; charset=utf-8"
```

Рисунок 4 – TestController.cs



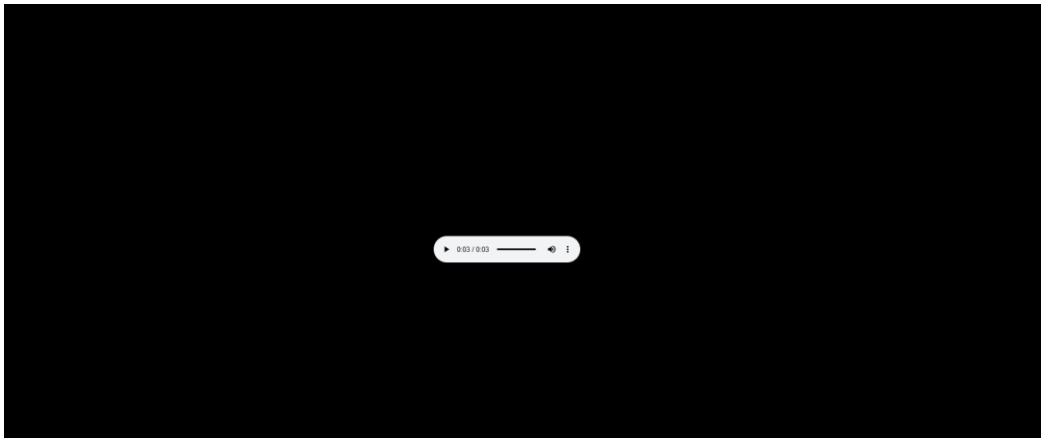


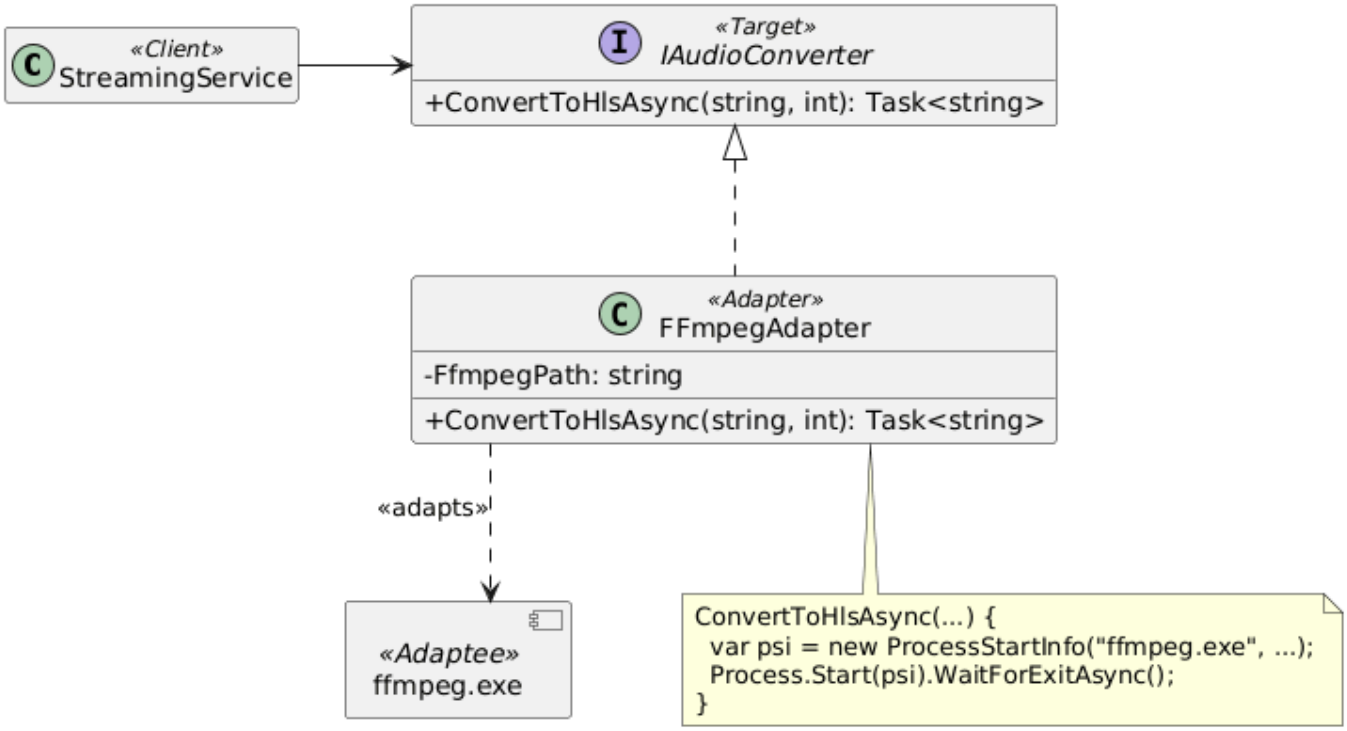
Рисунок 5 – Успішно інтегровано FFmpeg

4. Висновок

У ході виконання лабораторної роботи було розглянуто базові шаблони проєктування, зокрема Adapter, Builder, Command та інші. Отримані знання дали можливість зрозуміти роль патернів у побудові гнучких програмних систем. На прикладі веб-застосунку «Online Radio Station» було реалізовано шаблон Adapter для інтеграції зовнішньої утиліти ffmpeg.exe. Було створено цільовий інтерфейс IAudioConverter, а клас FFmpegAdapter виступив "перехідником", що приховує складну логіку запуску процесу. Це дало змогу відокремити клієнта (клас StreamingService) від виконавця (ffmpeg.exe). Такий підхід робить систему гнучкою, оскільки StreamingService залежить лише від абстракції, що дозволить у майбутньому легко замінити конвертер без зміни бізнес-логіки.

5. Контрольні питання:

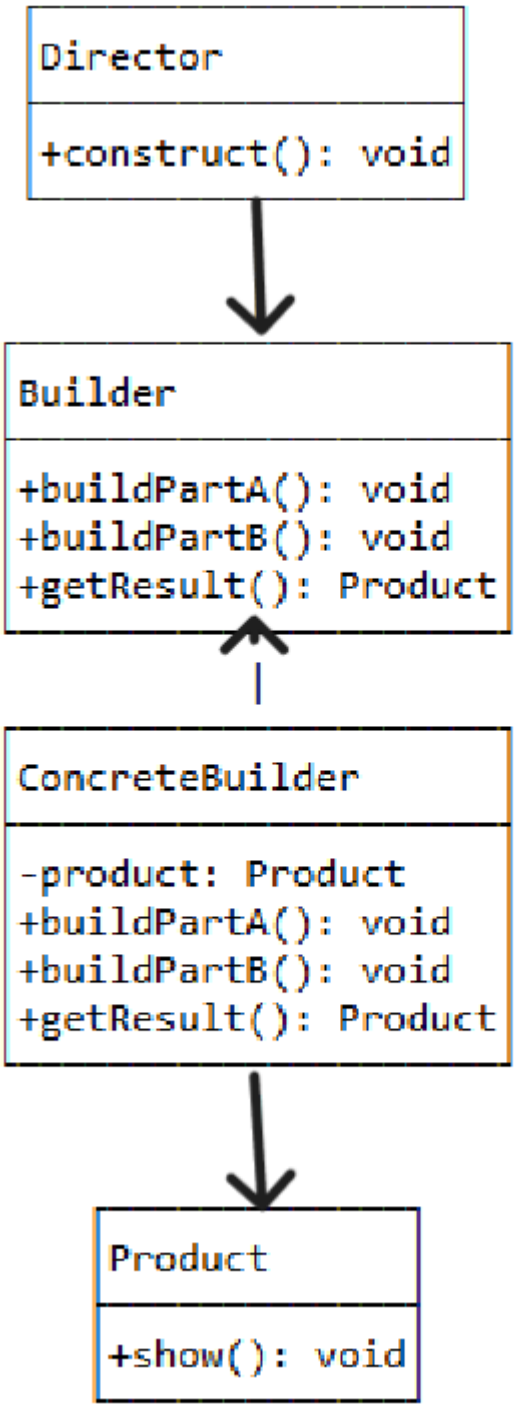
1) Яке призначення шаблону «Адаптер»? Адаптує інтерфейс одного об’єкта до іншого, дозволяючи несумісним компонентам працювати разом (наприклад, уніфікація бібліотек принтерів).
2) Нарисуйте структуру шаблону «Адаптер».



3) Які класи входять в шаблон «Адаптер», та яка між ними взаємодія?

Client, Target, Adapter, Adaptee. Client працює з Target через адаптований інтерфейс, Adapter перенаправляє виклики до Adaptee.

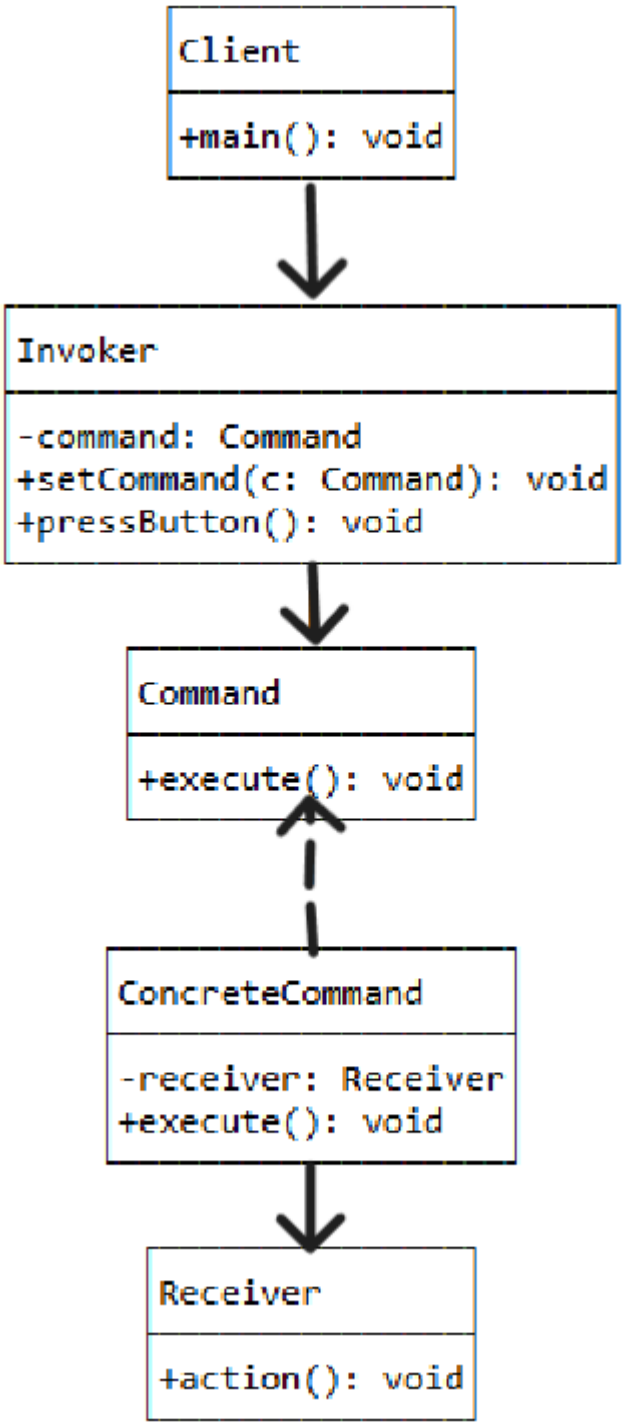
4) Яка різниця між реалізацією «Адаптера» на рівні об’єктів та на рівні класів? Об’єктний адаптер використовує композицію, класовим — успадкування.
5) Яке призначення шаблону «Будівельник»? Відокремлює процес створення об’єкта від його представлення, придатний для складних або багатоформних конструкцій.
6) Нарисуйте структуру шаблону «Будівельник».



7) Які класи входять в шаблон «Будівельник», та яка між ними взаємодія?

Director, Builder, ConcreteBuilder, Product. Director керує будівництвом, Builder визначає інтерфейс, ConcreteBuilder реалізує його, Product — результат.

8) У яких випадках варто застосовувати шаблон «Будівельник»? При складному процесі створення або необхідності різних форм об’єкта.
9) Яке призначення шаблону «Команда»? Перетворює виклик методу в об’єкт для гнучкої системи команд із відміною, логуванням чи плануванням.
10) Нарисуйте структуру шаблону «Команда».



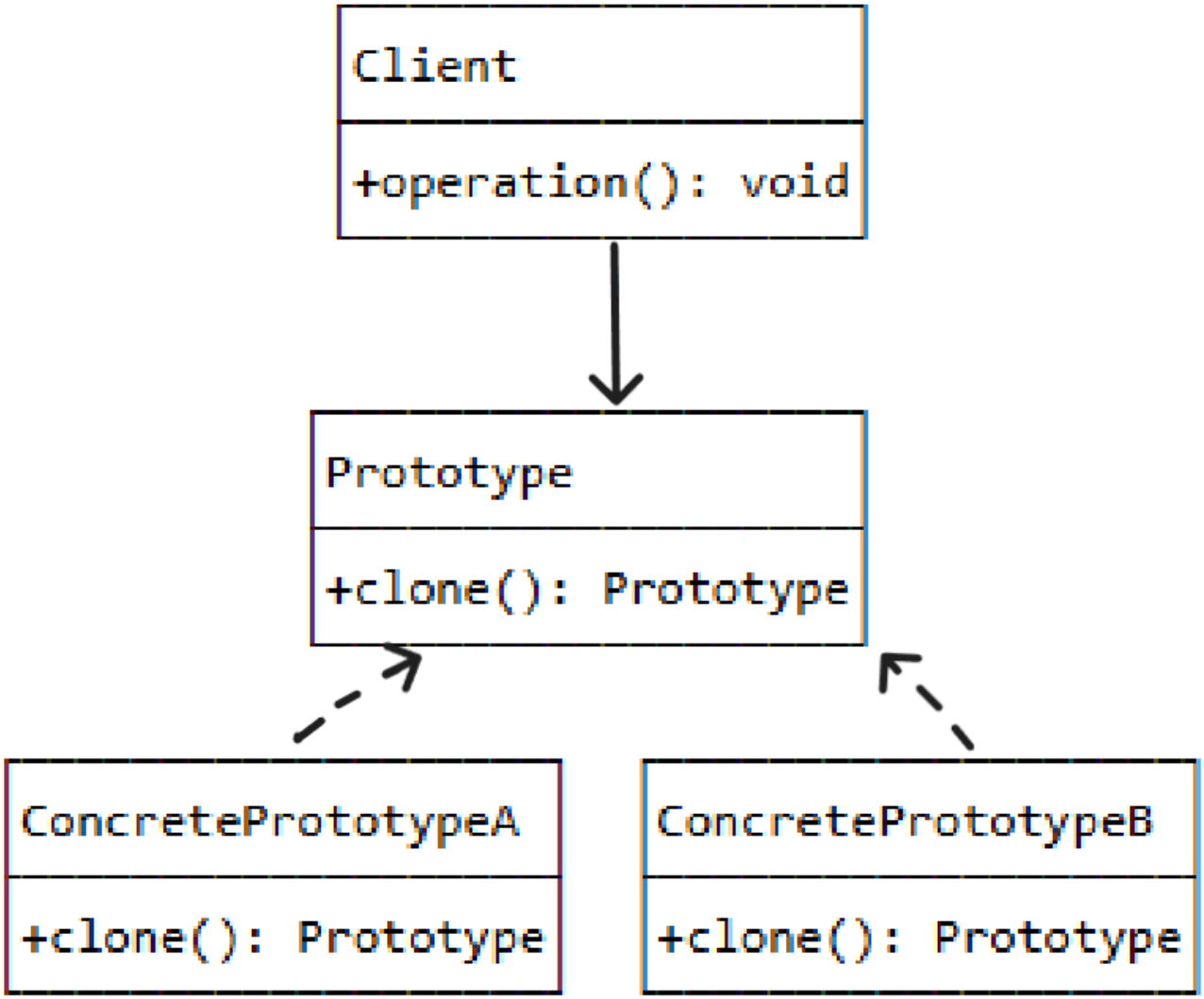
11) Які класи входять в шаблон «Команда», та яка між ними взаємодія?

Client, Invoker, Command, ConcreteCommand, Receiver. Client створює команду, Invoker виконує її, Receiver реалізує дію.

12) Розкажіть як працює шаблон «Команда». Команда інкапсулює запит як об’єкт, який передається Invoker до Receiver для виконання, підтримуючи додаткові функції (скасування, логування).

13) Яке призначення шаблону «Прототип»? Створює об’єкти клонуванням прототипу, зменшуючи ієрархію спадкування.

14) Нарисуйте структуру шаблону «Прототип».



15) Які класи входять в шаблон «Прототип», та яка між ними взаємодія?

Client, Prototype. Client клонувати об’єкт через метод Prototype.

16) Які можна привести приклади використання шаблону «Ланцюжок відповідальності»? Формування контекстного меню в UI, обробка запитів у ієрархії документів.