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 № 6

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:**

Вивчити структуру шаблонів «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator» та навчитися застосовувати їх в реалізації програмної системи.

2. **Теоретичні відомості**

Abstract Factory: Шаблон для створення сімейств пов’язаних об’єктів без вказівки їх конкретних класів. Використовується, коли потрібно забезпечити узгодженість об’єктів одного стилю або типу. Приклад: створення об’єктів різних стилів у грі (стіни, двері, меблі).

Factory Method: Визначає інтерфейс для створення об’єктів, дозволяючи підкласам вирішувати, який саме об’єкт створювати. Підходить для розширення системи новими типами без зміни існуючого коду.

Memento: (Знімок) Дозволяє зберігати і відновлювати стан об’єкта без порушення інкапсуляції. Стан зберігається в об’єкті-знімку, доступному лише вихідному об’єкту.

Observer: (Спостерігач) Визначає залежність «один-до-багатьох»: зміна стану одного об’єкта сповіщає всіх підписаних. Приклад: підписка на канал або повідомлення про зміну даних.

Decorator: (Декоратор) Дозволяє динамічно додавати об’єктам нову функціональність без зміни їхнього коду. Декоратор «обгортає» базовий об’єкт і розширює його поведінку.

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).

Система повинна генерувати HLS-потік у багатьох варіантах якості. Логіка обробки для кожного бітрейту схожа, але параметри відрізняються. Створювати об'єкти стрімів через new LowBitrateStream(), new HighBitrateStream() прямо в бізнес-логіці призвело б до жорсткої залежності та дублювання коду. Реалізовано фабричний метод. Абстрактний клас StreamFactory визначає інтерфейс створення, а конкретна фабрика BitrateStreamFactory містить логіку вибору класу (LowBitrateStream, StandardBitrateStream тощо) залежно від переданого числа бітрейту. Клієнтський код (AudioProcessingFacade) працює лише з абстракцією IAudioStream, не знаючи деталей створення."

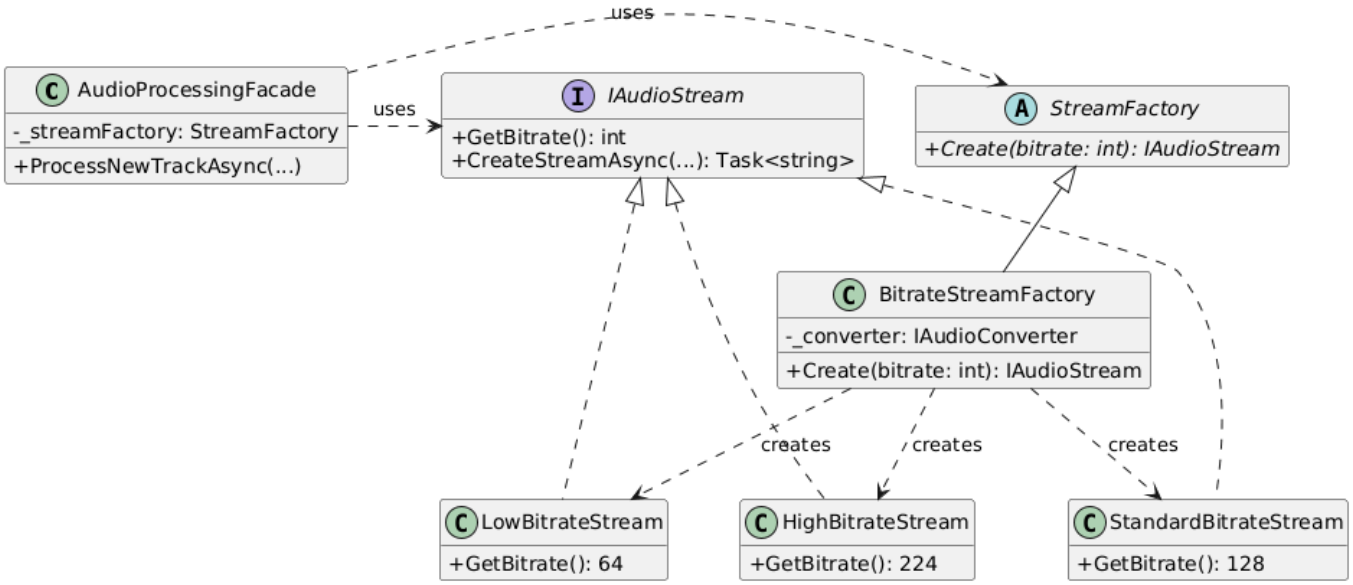


Рисунок 1 - Структура патерну Factory Method

AudioProcessingFacade (Client): Клієнт, який у циклі запитує у фабрики створення стрімів для різних бітрейтів.

StreamFactory (Creator): Абстрактний клас фабрики.

BitrateStreamFactory (ConcreteCreator): Реалізує метод Create, повертаючи потрібний клас залежно від аргументу bitrate.

IAudioStream (Product): Спільний інтерфейс для всіх стрімів.

```
namespace OnlineRadioStation.Domain
{
    5 references
    public interface IAudioStream
    {
        3 references
        int GetBitrate();
        4 references
        string StreamTrack(string title);
    }
}
```

Рисунок 2 - IAudioStream.cs

```
namespace OnlineRadioStation.Domain
{
    2 references
    public abstract class StreamFactory
    {
        2 references
        public abstract IAudioStream Create(int bitrate);
    }
}
```

Рисунок 3 - StreamFactory.cs

```
namespace OnlineRadioStation.Domain
{
    2 references
    public class BitrateStreamFactory : StreamFactory
    {
        2 references
        public override IAudioStream Create(int bitrate)
        {
            return bitrate switch
            {
                <= 64 => new LowBitrateStream(),
                <= 128 => new StandardBitrateStream(),
                _ => new HighBitrateStream()
            };
        }
    }
}
```

Рисунок 4 - BitrateStreamFactory.cs

```
namespace OnlineRadioStation.Domain
{
    1 reference
    public class LowBitrateStream : IAudioStream
    {
        1 reference
        public int GetBitrate() => 64;
        2 references
        public string StreamTrack(string title) => $"[64kb/s] Стрім: {title}";
    }
}
```

Рисунок 5 - LowBitrateStream.cs

```
private async Task<string> ConvertFileInternal(string tempFilePath)
{
    int[] targetBitrates = { 64, 92, 128, 196, 224 };
    var masterPlaylistContent = "#EXTM3U\n#EXT-X-VERSION:3\n";
    var baseFileName = Path.GetFileNameWithoutExtension(tempFilePath);
    var baseFolder = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "streams", baseFileName);
    Directory.CreateDirectory(baseFolder);

    foreach (var bitrate in targetBitrates)
    {
        var streamProduct = _streamFactory.Create(bitrate);
        await streamProduct.CreateStreamAsync(tempFilePath, bitrate.ToString());
        masterPlaylistContent += $"#EXT-X-STREAM-INF:BANDWIDTH={bitrate * 1000},CODECS=\"mp4a.40.2\"\n";
        masterPlaylistContent += $"{bitrate}/index.m3u8\n";
    }

    var masterPath = Path.Combine(baseFolder, "master.m3u8");
    await File.WriteAllTextAsync(masterPath, masterPlaylistContent);
    return $"/streams/{baseFileName}/master.m3u8";
}
```

Рисунок 6 – Використання фабрики (AudioProcessingFacade.cs)

Рисунок 8 –Результат роботи фабрики: згенеровано потоки для 5 різних бітрейтів, кожен з яких оброблено відповідним екземпляром класу стріму

4. Висновок

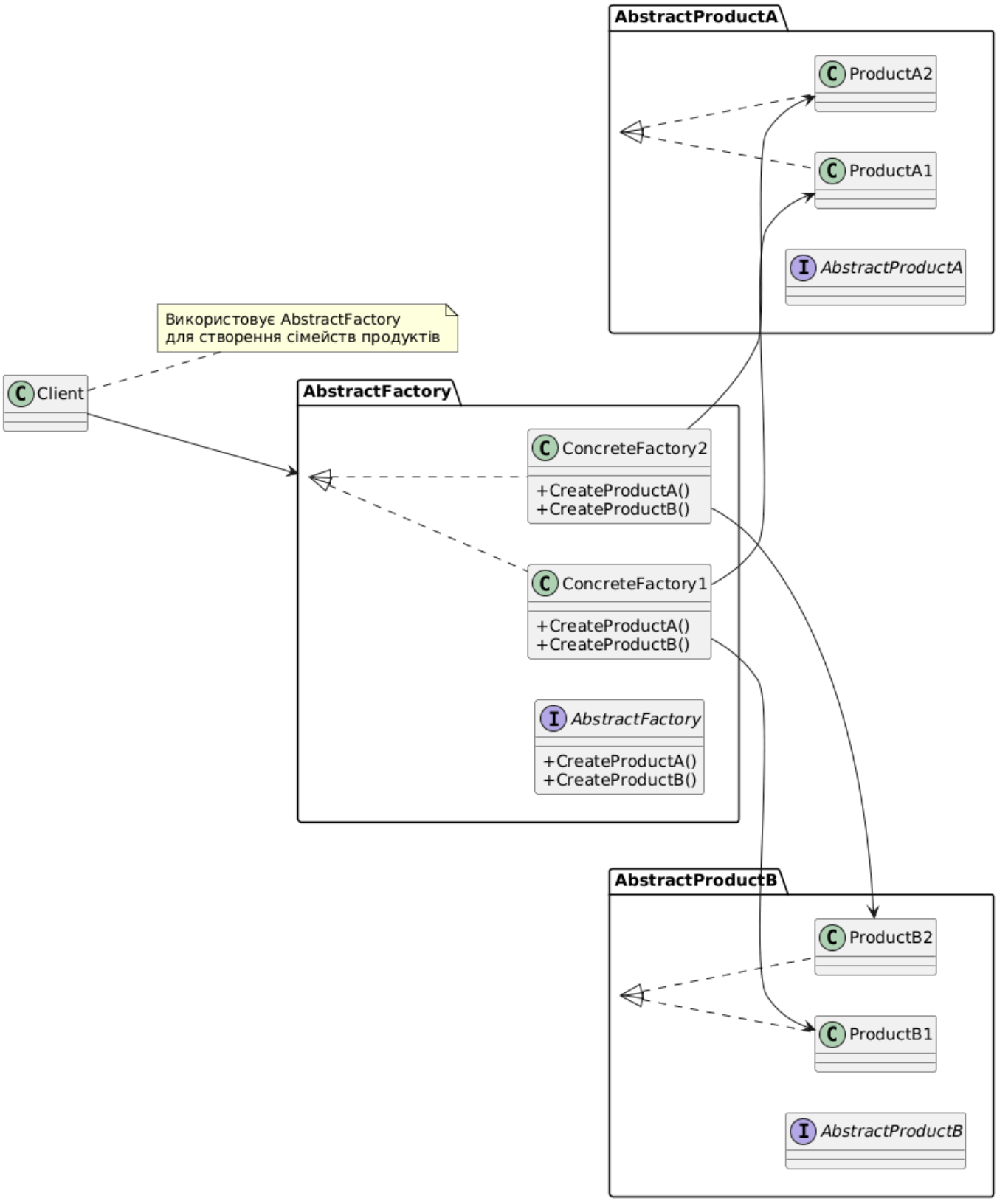
У ході виконання лабораторної роботи було розглянуто шаблони проєктування, зокрема Abstract Factory, Factory Method, Memento, Observer та Decorator. Отримані

знання допомогли зрозуміти їхню роль у створенні гнучкої, масштабованої та підтримувальної архітектури програмних систем. На прикладі веб-застосунку «Online Radio Station» було реалізовано шаблон Factory Method для організації створення об’єктів стримінгу з різними бітрейтами (64, 128, 224 kb/s). Логіка створення винесена в окремий клас-фабрику (BitrateStreamFactory), що дозволило відокремити процес інстанціювання від клієнтського коду (AudioProcessingFacade). Такий підхід спрощує керування якістю стримінгу, робить код більш читабельним і дає змогу легко розширювати функціонал — наприклад, додати підтримку нових бітрейтів (320 kb/s, VBR) або альтернативних кодеків (Opus, AAC-HE) — без зміни основної логіки сервісу.

5. Контрольні питання

1. Яке призначення шаблону «Абстрактна фабрика» Шаблон «Абстрактна фабрика» використовується для створення сімейств об'єктів без вказівки їх конкретних класів.

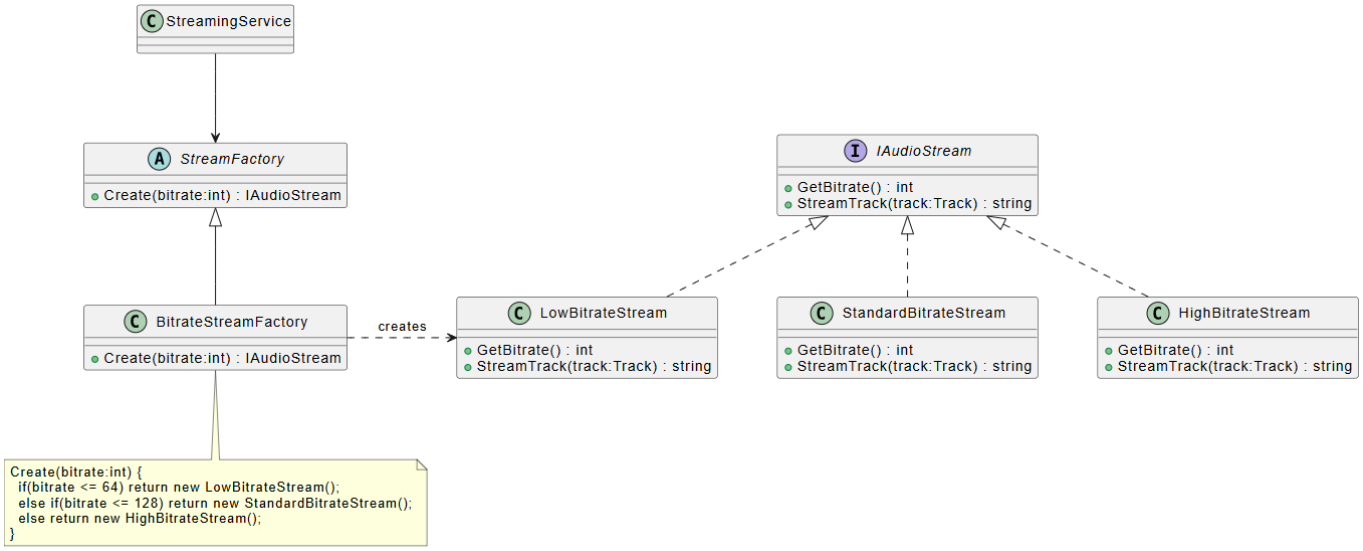
2. Нарисуйте структуру шаблону «Абстрактна фабрика»



3. Які класи входять в шаблон «Абстрактна фабрика», та яка між ними взаємодія?
Класи: AbstractFactory, ConcreteFactory, AbstractProductA/B, ConcreteProductA/B, Client. Взаємодія: Client використовує AbstractFactory для створення об'єктів через інтерфейси продуктів; ConcreteFactory реалізує методи створення конкретних продуктів одного сімейства.

4. Яке призначення шаблону «Фабричний метод»? Шаблон «Фабричний метод» визначає інтерфейс для створення об'єктів певного базового типу, залишаючи реалізацію підтипам.

5. Нарисуйте структуру шаблону «Фабричний метод»

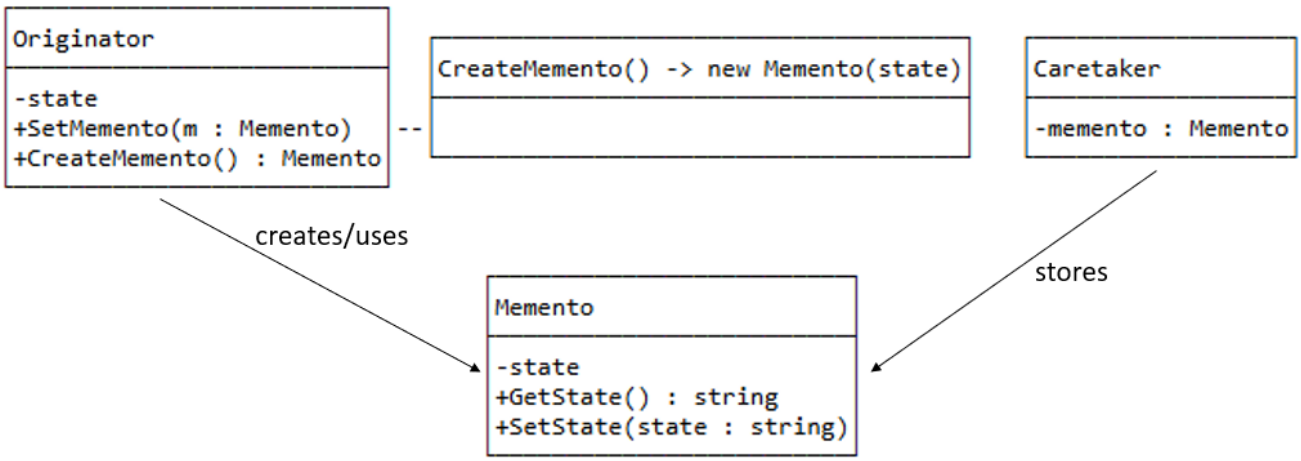


6. Які класи входять в шаблон «Фабричний метод», та яка між ними взаємодія? Класи: Creator, ConcreteCreator, Product, ConcreteProduct. Взаємодія: Creator викликає FactoryMethod() для створення об'єкта; ConcreteCreator повертає конкретний ConcreteProduct.

7. Чим відрізняється шаблон «Абстрактна фабрика» від «Фабричний метод»? «Абстрактна фабрика» створює сімейства пов’язаних об'єктів через набір методів; «Фабричний метод» створює один об'єкт через один віртуальний метод, реалізований у підкласах.

8. Яке призначення шаблону «Знімок»? Шаблон «Знімок» використовується для збереження і відновлення стану об'єктів без порушення інкапсуляції.

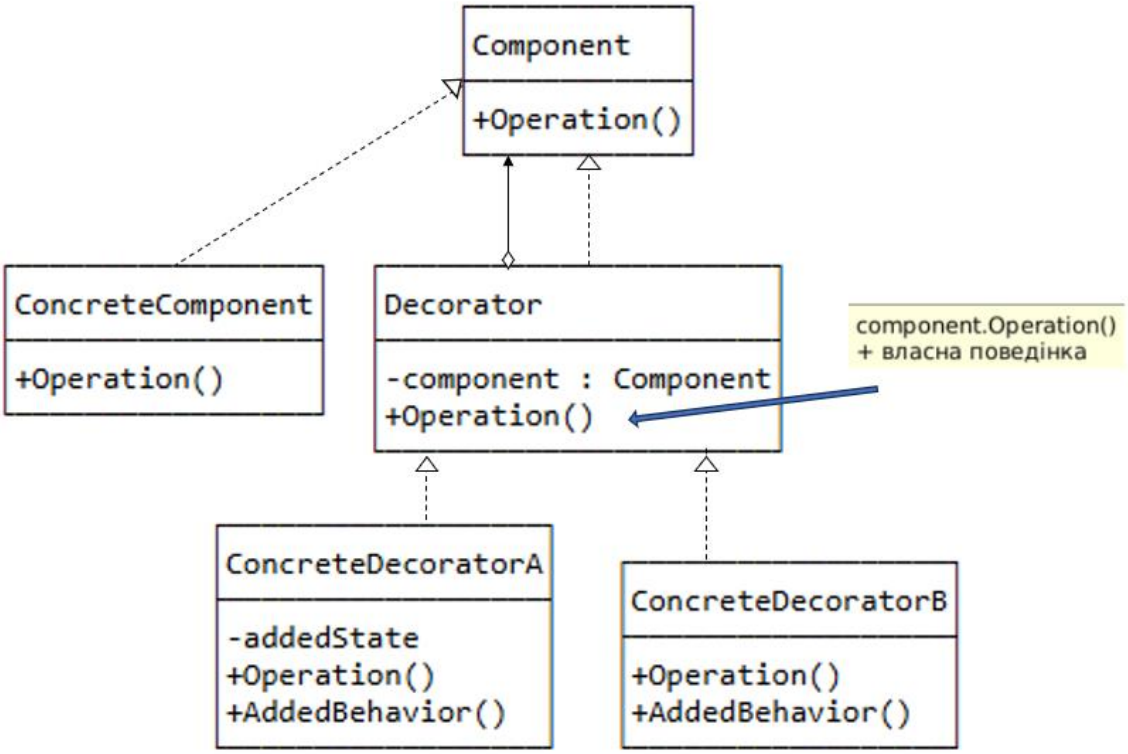
9. Нарисуйте структуру шаблону «Знімок»



10. Які класи входять в шаблон «Знімок», та яка між ними взаємодія? Класи: Originator, Memento, Caretaker. Взаємодія: Originator створює/відновлює Memento; Caretaker зберігає Memento, не маючи доступу до його вмісту.

11. Яке призначення шаблону «Декоратор»? Шаблон «Декоратор» призначений для динамічного додавання функціональних можливостей об'єкту під час роботи програми.

12. Нарисуйте структуру шаблону «Декоратор»



13. Які класи входять в шаблон «Декоратор», та яка між ними взаємодія? Класи: Component, ConcreteComponent, Decorator, ConcreteDecorator. Взаємодія: Decorator обгортає Component, викликає його Operation() і додає власну поведінку.

14. Які є обмеження використання шаблону «Декоратор»? Велика кількість крихітних класів; важко конфігурувати об’єкти, загорнуті в декілька обгорток одночасно.