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 *№4*

із дисципліни ***«Технології розробки програмного забезпечення»***

Тема: ***«Вступ до паттернів проектування»***

Виконав:

Студент групи ІА-34

Ковальчук Станіслав

Перевірив:

асистент кафедри ІСТ

Мягкий Михайло Юрійович

Київ — 2025

Тема: Вступ до паттернів проектування.

Мета:Вивчити принципи побудови гнучкої архітектури програмного забезпечення. Ознайомитися з класифікацією патернів проектування. Навчитися реалізовувати поведінковий патерн Strategy (Стратегія) на прикладі системи "Office Communicator".

Застосунок: Office communicator (strategy, adapter, abstract factory, bridge, composite, client-server)

Мережевий комунікатор для офісу повинен нагадувати функціонал програми Skype з можливостями голосового / відео / конференц-зв'язку, відправки текстових повідомлень і файлів (можливо, оффлайн), веденням організованого списку груп / контактів.

Короткі теоретичні відомості:

Патерн проєктування — це перевірене часом та формалізоване рішення типових задач, що виникають під час розробки програмного забезпечення. Він включає опис проблеми, концептуальне рішення та рекомендації щодо його адаптації до різних умов.

Застосування шаблонів надає розробнику вагомі переваги:

- Чітка структура: Модель системи базується на логічному виокремленні ключових елементів та зв'язків.
- Прозорість: Проєкти, побудовані на патернах, легше розуміти іншим розробникам, оскільки вони використовують "спільну мову".
- Гнучкість: Система стає стійкішою до змін вимог, а її подальша підтримка та масштабування значно спрощуються.

Приклади популярних патернів

Singleton (Одинак)

Гарантує, що клас має лише один екземпляр, і надає глобальну точку доступу до нього. Зазвичай цей об'єкт зберігається у статичному полі самого класу, що запобігає створенню дублікатів.

Iterator (Ітератор)

Надає спосіб послідовного доступу до елементів колекції (агрегата), не розкриваючи її внутрішню структуру.

Ключова ідея: Розподіл обов'язків. Колекція відповідає за зберігання даних, а ітератор — за логіку обходу цих даних.

Proxy (Замісник)

Об'єкт, який виступає в ролі сурогату або "заглушки" для іншого об'єкта. Проксі дозволяє контролювати доступ до реального об'єкта, додаючи допоміжну логіку (наприклад, кешування, логування або перевірку прав доступу), не змінюючи код основного класу.

State (Стан)

Дозволяє об'єкту змінювати свою поведінку залежно від внутрішнього стану. З боку здається, ніби об'єкт змінив свій клас.

- Приклад: Відсоток нарахувань на банківську картку змінюється залежно від її статусу (Silver, Gold, Platinum).
- Реалізація: Стан виноситься в окремий інтерфейс, а кожен конкретний стан реалізується як окремий клас.

Strategy (Стратегія)

Визначає сімейство схожих алгоритмів, інкапсулює кожен з них і робить їх взаємозамінними. Це дозволяє обирати потрібний алгоритм безпосередньо під час виконання програми.

- Приклад: Вибір різних методів сортування (швидке, бульбашкою тощо) або різних способів оплати в інтернет-магазині.

Хід роботи:

1. Діаграма класів.

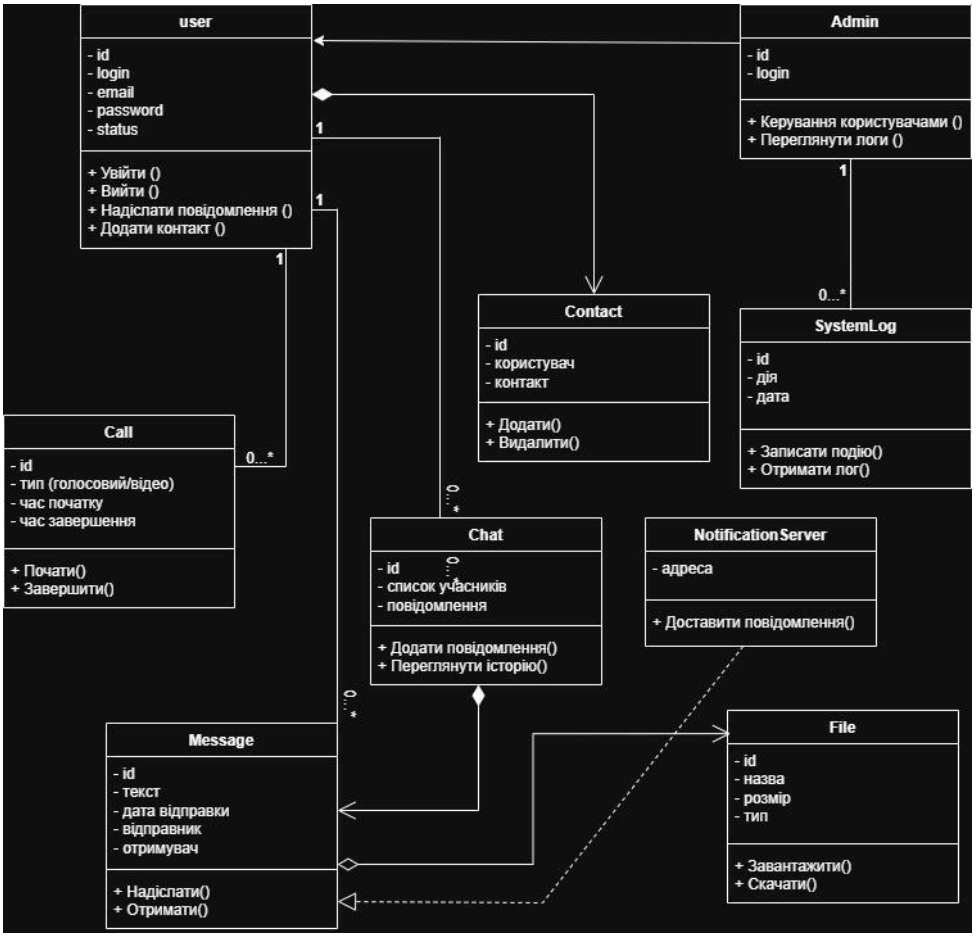


Рис. 4.1 – Діаграма класів системи

2. Діаграма класів з використанням шаблону «Strategy»:

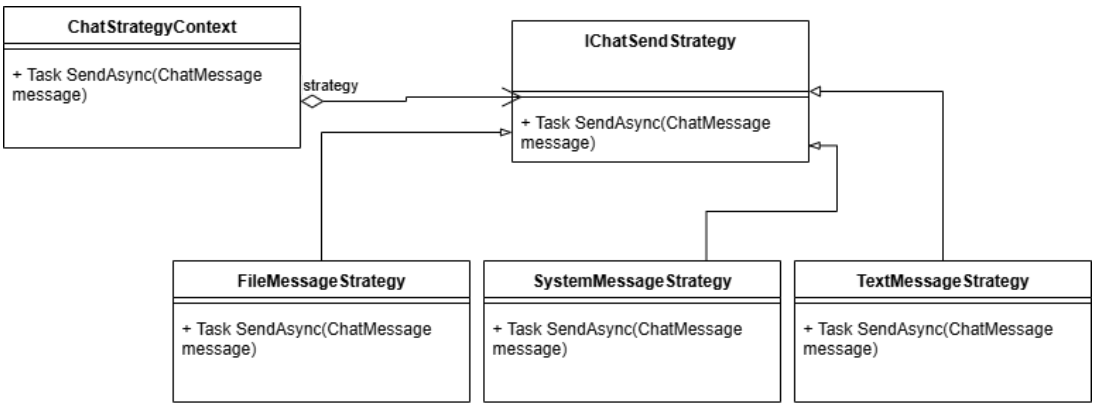


Рис. 4.2 – Діаграма класів системи з використанням шаблону «Strategy»

Визначимо інтерфейс IChatSendStrategy, який має метод асинхронного надсиланнння повідомлення SendAsync(ChatMessage message).

Крім того визначимо клас контексту ChatStrategyContext, який і буде обирати стратегію.

Далі визначимо класи TextMessageStrategy, SystemMessageStrategy та FileMessageStrategy, які визначають стратегію надсилання текстових, системних та файлових повідомлень відповідно.

3ПРОГРАМНА РЕАЛІЗАЦІЯ

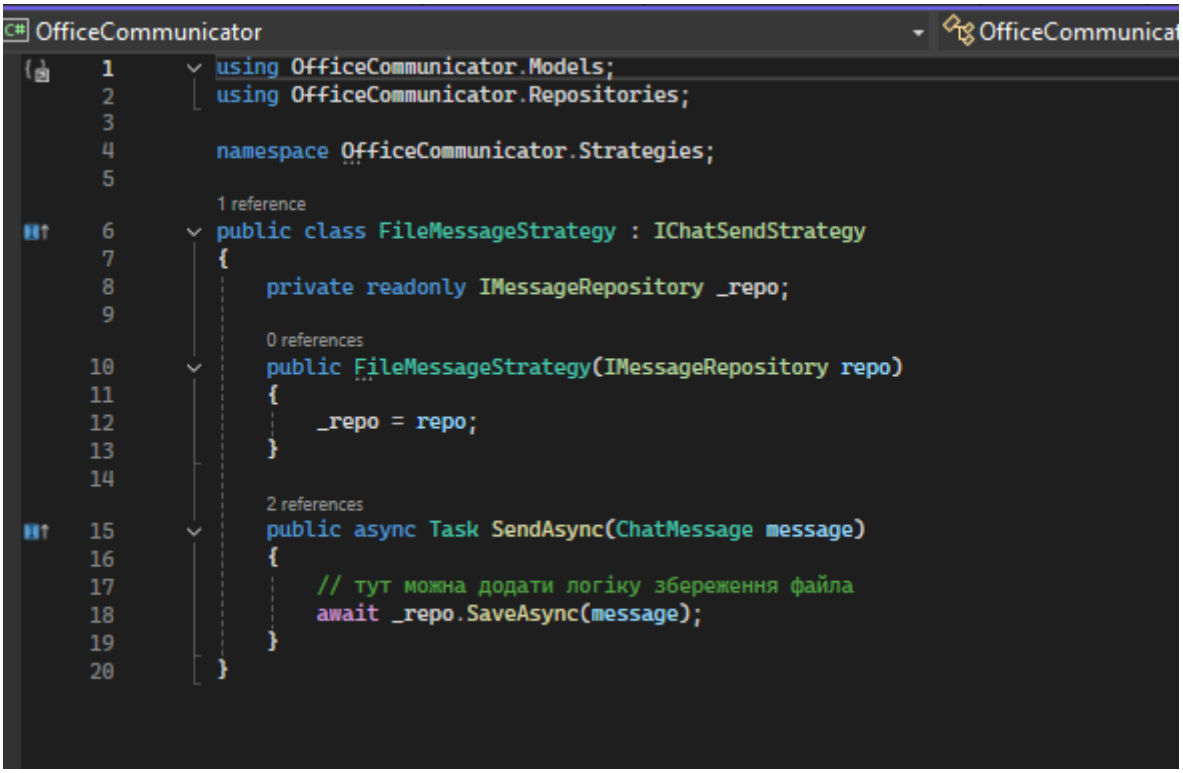
Для забезпечення гнучкості вибору алгоритму відправки повідомлень у системі "Office Communicator" було реалізовано патерн **Strategy**. Це дозволяє системі обробляти різні типи повідомлень (текстові, файлові, системні) через єдиний інтерфейс, делегуючи логіку збереження відповідним класам стратегій.Нижче наведено лістинг програмного коду реалізації.

Інтерфейс стратегії (IChatSendStrategy.cs) Цей інтерфейс визначає контракт для всіх алгоритмів відправки повідомлень. Він приймає об'єкт моделі ChatMessage.

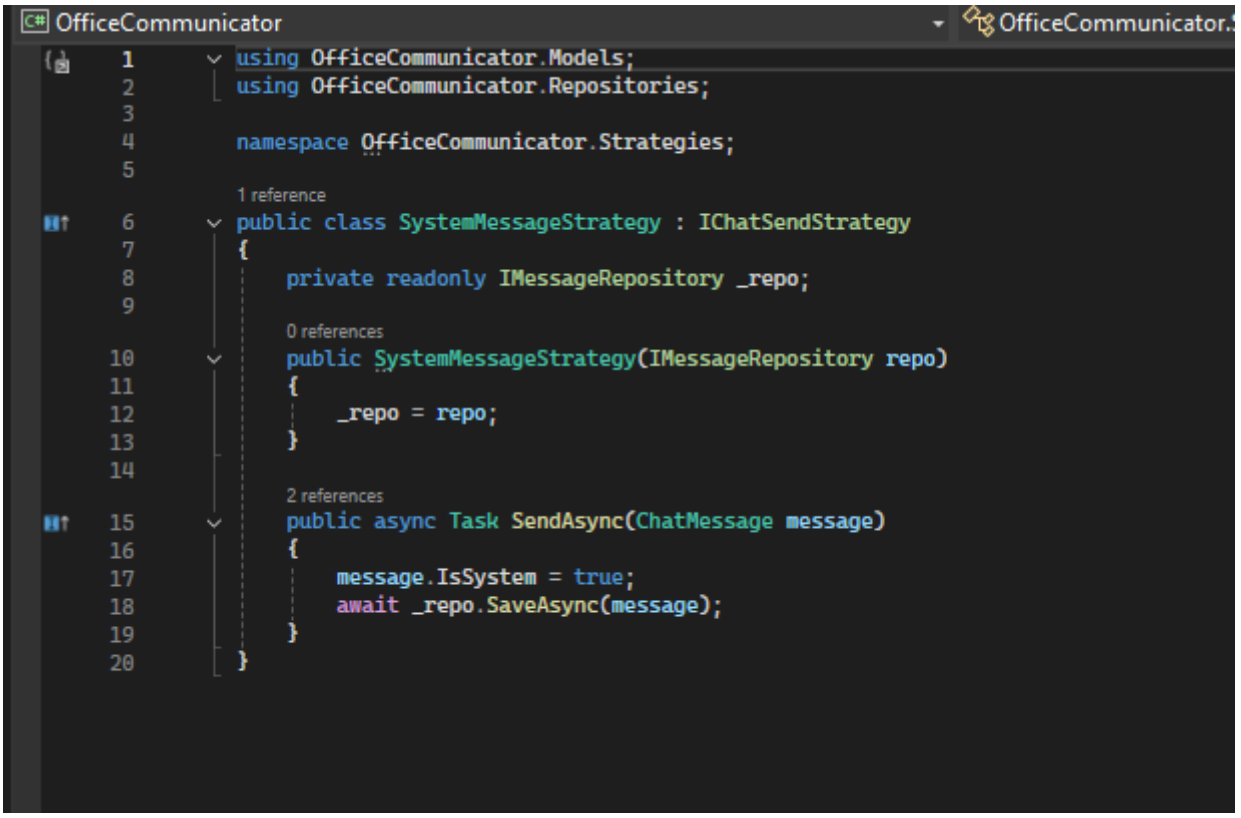
```
1   using OfficeCommunicator.Models;
2
3   namespace OfficeCommunicator.Strategies;
4
    5 references
5   public interface IChatSendStrategy
6   {
        4 references
7       Task SendAsync(ChatMessage message);
8   }
```

Конкретні стратегії (Strategies.cs) Реалізація стратегій використовує репозиторій IMessageRepository для збереження даних у базу.

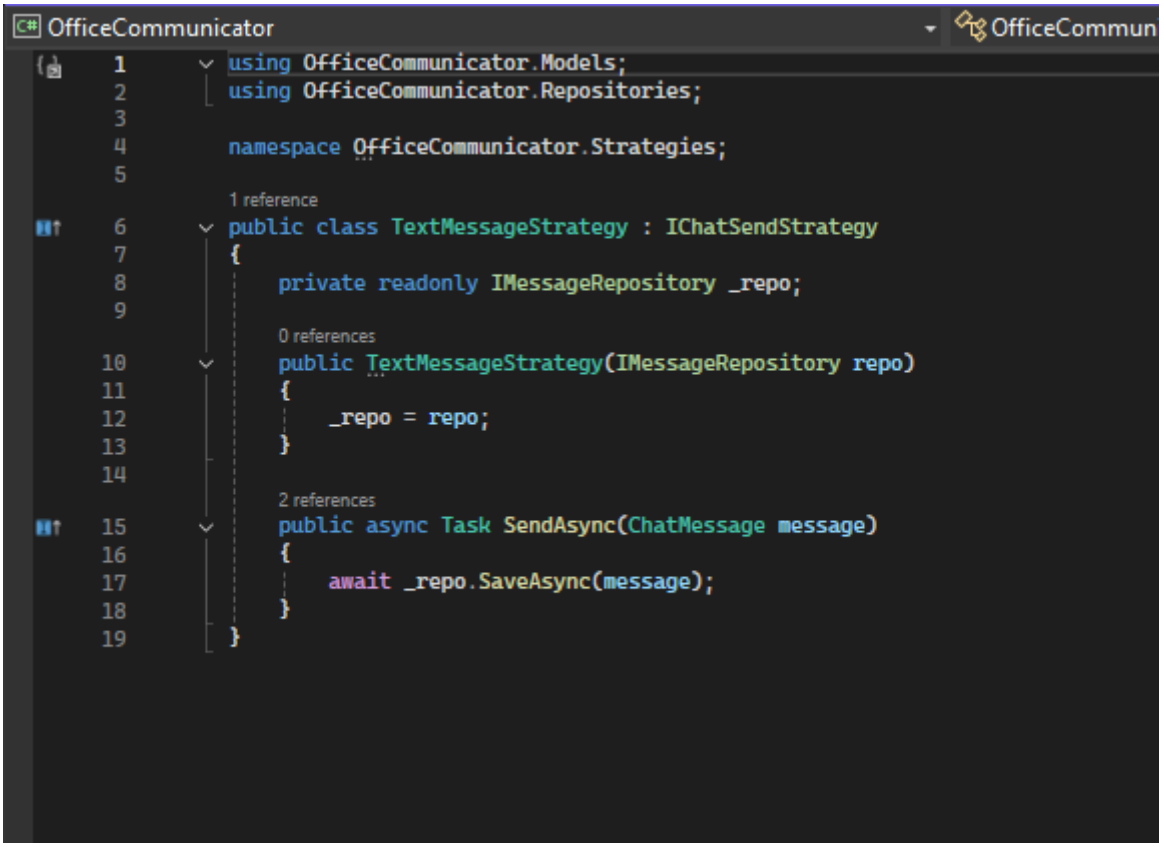
Стратегія для файлових повідомлень:



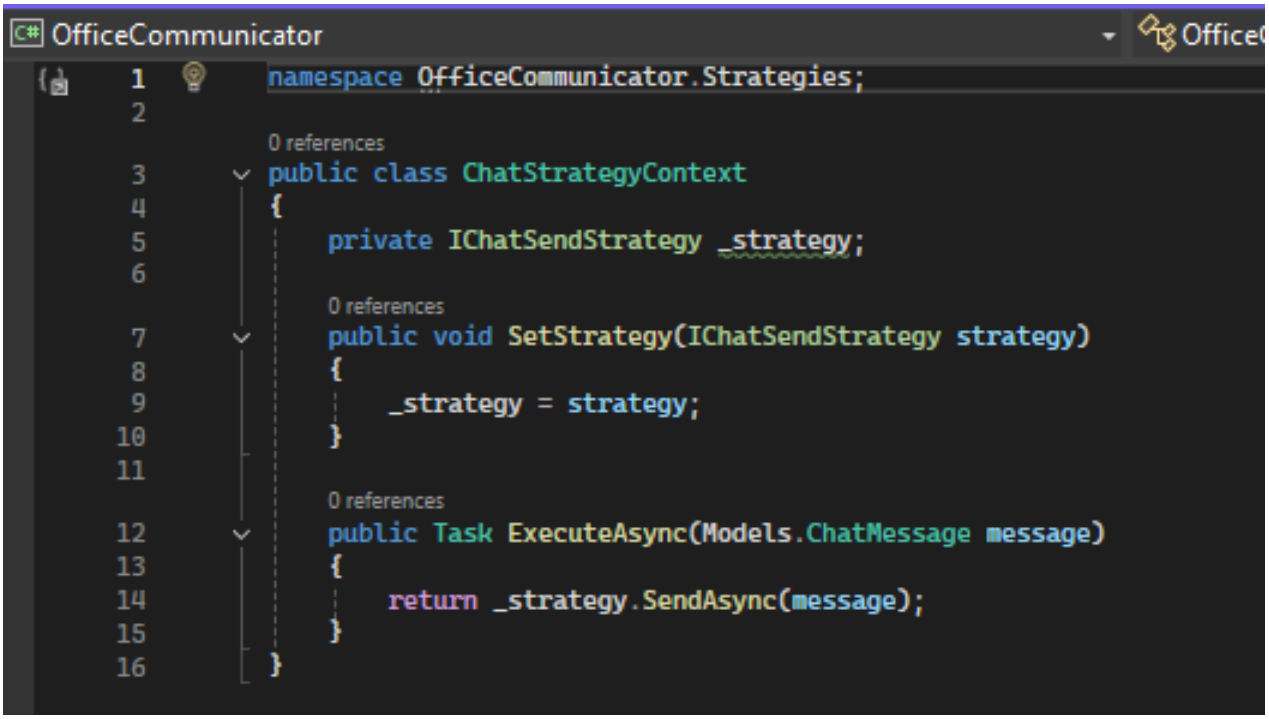
Стратегія для системних повідомлень:



Стратегія для звичайних текстових повідомлень:



Контекст (ChatStrategyContext.cs) Клас-контекст, який керує вибором стратегії та делегує їй виконання операції. Це дозволяє динамічно змінювати поведінку (наприклад, перемикатися між відправкою тексту та файлу).



Висновок:Під час виконання лабораторної роботи я опанував концепцію патернів проєктування, вивчив їхню класифікацію, а також проаналізував переваги та недоліки їх застосування.Для практичної реалізації було обрано патерн **«Strategy»** (Стратегія), який я імплементував у модулі надсилання повідомлень. Реалізація включає контекст ChatStrategyContext та конкретні стратегії (TextMessageStrategy, FileMessageStrategy, SystemMessageStrategy), що взаємодіють із репозиторієм даних.Вибір цього шаблону зумовлений його гнучкістю: він дозволяє легко змінювати алгоритми обробки повідомлень, відокремлюючи їхню логіку від основного коду (контексту).Крім того, застосування «Стратегії» дозволило значно спростити архітектуру, мінімізувавши кількість умовних операторів (if, switch) при перевірці типу повідомлення та дотримуючись принципу відкритості/закритості (Open/Closed Principle).

Відповіді на контрольні питання:

1. Що таке шаблон проєктування?

   Патерн проєктування — це перевірене часом та формалізоване рішення типових задач, що виникають під час розробки програмного забезпечення. Він включає опис проблеми, концептуальне рішення та рекомендації щодо його адаптації до різних умов.
2. Навіщо використовувати шаблони проєктування?
   - Чітка структура: Модель системи базується на логічному виокремленні ключових елементів та зв'язків.
   - Прозорість: Проєкти, побудовані на патернах, легше розуміти іншим розробникам, оскільки вони використовують "спільну мову".

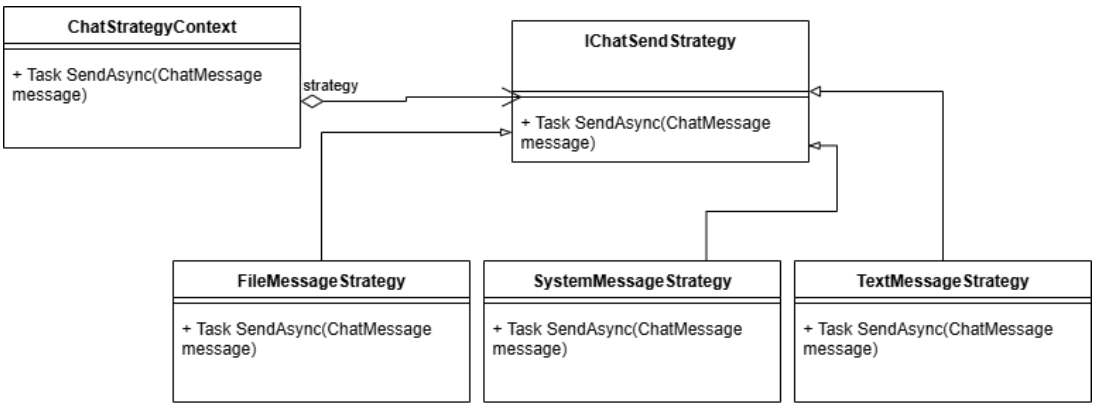
- Гнучкість: Система стає стійкішою до змін вимог, а її подальша підтримка та масштабування значно спрощуються.

3. Яке призначення шаблону «Стратегія»?

   Шаблон «Strategy» (Стратегія) дозволяє змінювати деякий алгоритм поведінки об'єкта іншим алгоритмом, що досягає ту ж мету іншим способом.

   Прикладом можуть служити алгоритми сортування: кожен алгоритм має власну реалізацію і визначений в окремому класі.

4. Нарисуйте структуру шаблону «Стратегія».



5. Які класи входять в шаблон «Стратегія», та яка між ними взаємодія?

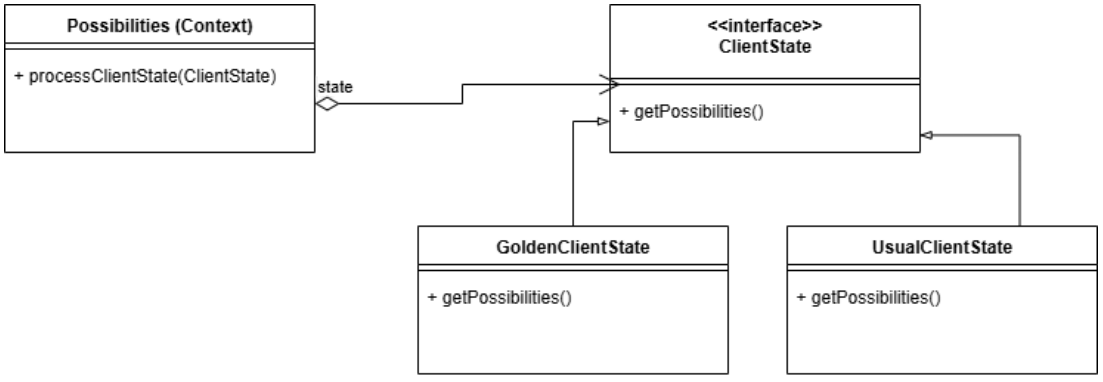
   Context – визначає метод, в якому один з параметрів – стратегія

   Strategy – інтерфейс або абстрактний клас, який містить спільний метод для всіх інших класів-стратегій, які реалізують цей інтерфейс.

6. Яке призначення шаблону «Стан»?

   Шаблон «State» (Стан) дозволяє змінювати логіку роботи об'єктів у випадку зміни їх внутрішнього стану.

7. Нарисуйте структуру шаблону «Стан».



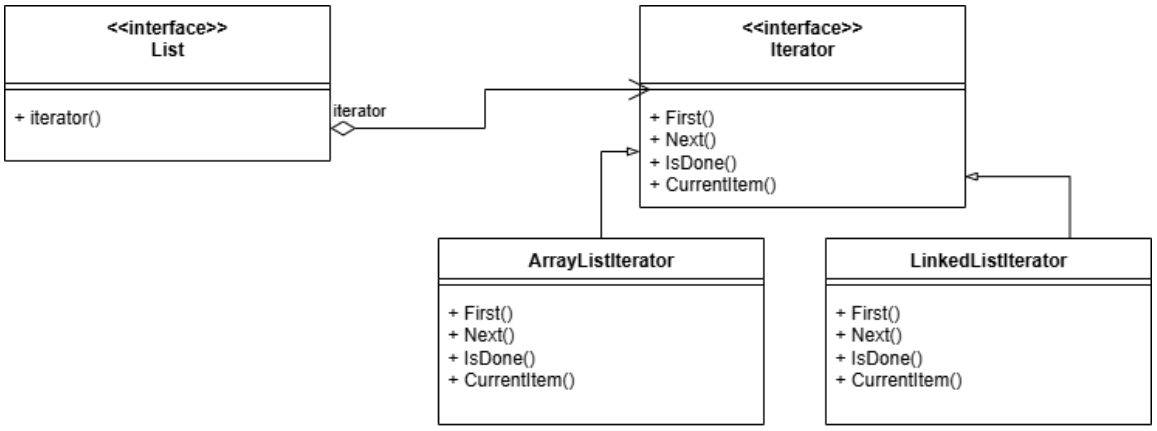
8. Які класи входять в шаблон «Стан», та яка між ними взаємодія?

   Об’єкти, що мають стан (Context), при зміні стану просто записують новий об'єкт в поле state, що призводить до повної зміни поведінки об'єкта, а пов'язані зі станом поля, властивості, методи і дії виносяться в окремий загальний інтерфейс (State); кожен стан являє собою окремий клас (ConcreteStateA, ConcreteStateB), які реалізують загальний інтерфейс.

9. Яке призначення шаблону «Ітератор»?

   «Iterator» (Ітератор) являє собою шаблон реалізації об'єкта доступу до набору (колекції, агрегату) елементів без розкриття внутрішніх механізмів реалізації.

10. Нарисуйте структуру шаблону «Ітератор».



11. Які класи входять в шаблон «Ітератор», та яка між ними взаємодія?

    Шаблонний ітератор містить:

    - First() – установка покажчика перебору на перший елемент колекції;
    - Next() – установка покажчика перебору на наступний елемент колекції;
    - IsDone – булевське поле, яке встановлюється як true коли покажчик перебору досяг кінця колекції;
    - CurrentItem – поточний об'єкт колекції.

12. В чому полягає ідея шаблона «Одинак»?

«Singleton» (Одинак) являє собою клас в термінах ООП, який може мати не більше одного об'єкта. Даний об'єкт найчастіше зберігається як статичне поле в самому класі.

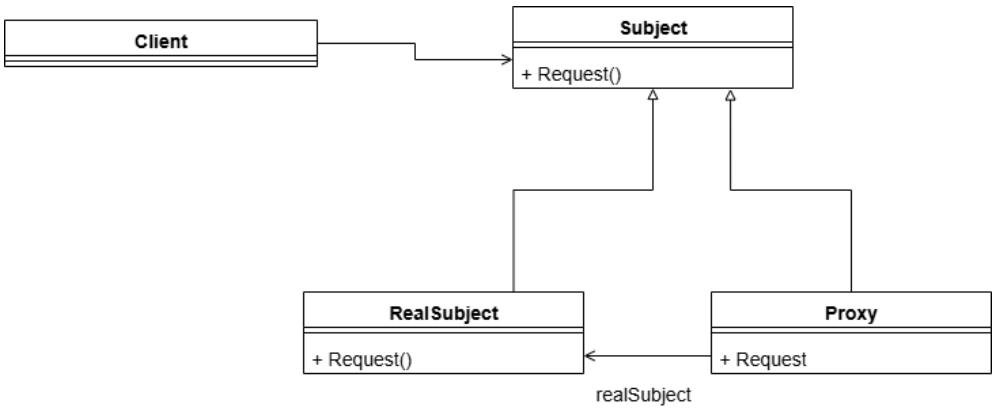
13. Чому шаблон «Одинак» вважають «анти-шаблоном»?

    Це пов'язано з тим, що «одинаки» представляють собою глобальні дані (як глобальна змінна), що мають стан. Стан глобальних об'єктів важко відслідковувати і підтримувати коректно.

14. Яке призначення шаблону «Проксі»?

    «Proxy» (Проксі) – об'єкти є об'єктами-заглушками або двійниками/замінниками для об'єктів конкретного типу.

15. Нарисуйте структуру шаблону «Проксі».



16. Які класи входять в шаблон «Проксі», та яка між ними взаємодія?

    Зовнішня система Client – приймає об’єкт з предмету інтерфейсу Subject, а вже справжній предмет RealSubject та проксі Proxy реалізовують його.