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

Лабораторна робота №4
«Технології розроблення програмного забезпечення»
Тема: « E-mail клієнт»

Виконав:
Студент групи ІА-34
Марченко А.О.
Перевірив:
Мягкий Михайло Юрійович

Київ 2025

Тема: Вступ до паттернів проектування.

Мета: Вивчити структуру шаблонів «Singleton», «Iterator», «Proxy», «State», «Strategy» та навчитися застосовувати їх в реалізації програмної системи.

Завдання

• Ознайомитись з короткими теоретичними відомостями.

• Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.

• Реалізувати один з розглянутих шаблонів за обраною темою.

• Реалізувати не менше 3-х класів відповідно до обраної теми.

• 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.

Тема:

15. **E-mail клієнт** (singleton, builder, decorator, template method, interpreter, client-server)

Поштовий клієнт повинен нагадувати функціонал поштових програм Mozilla Thunderbird, The Bat і т.д. Він повинен сприймати і коректно обробляти pop3/smtp/imap протоколи, мати функції автонастройки основних поштових провайдерів для України (gmail, ukr.net, i.ua), розділяти повідомлення на папки/категорії/важливість, зберігати чернетки незавершених повідомлень, прикріплювати і обробляти прикріплені файли.

Теоретичні відомості

Поняття шаблону проєктування

Будь-який патерн проєктування, використовуваний при розробці інформаційних систем, являє собою формалізований опис, який часто зустрічається в завданнях проєктування, вдале рішення даної задачі, а також рекомендації по застосуванню цього рішення в різних ситуаціях [5]. Крім того, патерн проєктування обов’язково має загальновживане найменування.

Правильно сформульований патерн проєктування дозволяє, відшукавши одного разу вдале рішення, користуватися ним знову і знову. Варто підкреслити, що важливим початковим етапом при роботі з патернами є адекватне моделювання розглянутої предметної області. Це є необхідним як для отримання належним чином формалізованої постановки задачі, так і для вибору відповідних патернів проєктування.

Відповідне використання патернів проєктування дає розробнику ряд незаперечних переваг. Наведемо деякі з них. Модель системи, побудована в межах патернів проєктування, фактично є структурованим виокремленням тих елементів і зв'язків, які значимі при вирішенні поставленого завдання.

Крім цього, модель, побудована з використанням патернів проєктування, більш проста і наочна у вивченні, ніж стандартна модель. Проте, не дивлячись на простоту і наочність, вона дозволяє глибоко і всебічно опрацювати архітектуру розроблюваної системи з використанням спеціальної мови. Застосування патернів проєктування підвищує стійкість системи до зміни вимог та спрощує неминуче подальше доопрацювання системи.

Крім того, важко переоцінити роль використання патернів при інтеграції інформаційних систем організації. Також слід зазначити, що сукупність патернів проєктування, по суті, являє собою єдиний словник проєктування, який, будучи уніфікованим засобом, незамінний для спілкування розробників один одним.

Таким чином шаблони представляють собою, підтверджені роками розробок в різних компаніях і на різних проєктах, «ескізи» архітектурних рішень, які зручно застосовувати у відповідних обставинах.

Шаблон «Singleton»

Призначення : «Singleton» (Одинак) являє собою клас в термінах ООП, який може мати не більше одного об'єкта (звідси і назва «одинак») [6].

Насправді, кількість об'єктів можна задати (тобто не можна створити більш n об'єктів даного класу). Даний об'єкт найчастіше зберігається як статичне поле в самому класі.

Переваги та недоліки: Однак слід зазначити, що в даний час патерн «Одинак» багато хто вважає т.зв. «анти-шаблоном», тобто поганою практикою проєктування. 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; також глобальні об'єкти важко тестуються і вносять складність в програмний код (у всіх ділянках коду виклик в одне єдине місце з «одинаком»; при зміні підходу доведеться змінювати масу коду).

При цьому реалізація контролю доступу можлива за допомогою статичних змінних, замикань, мютексов та інших спеціальних структур.

+ Гарантує наявність єдиного екземпляра класу.
+ Надає до нього глобальну точку доступу.
- Порушує принцип єдиної відповідальності класу.
- Маскує поганий дизайн.

Шаблон «Iterator»

Призначення: «Iterator» (Ітератор) являє собою шаблон реалізації об'єкта доступу до набору (колекції, агрегату) елементів без розкриття внутрішніх механізмів реалізації. Ітератор виносить функціональність перебору колекції елементів з самої колекції, таким чином досягається розподіл обов'язків: колекція відповідає за зберігання даних, ітератор – за прохід по колекції [6]. При цьому алгоритм ітератора може змінюватися – при необхідності пройти в зворотньому порядку використовується інший ітератор. Можливо також написання такого ітератора, який проходить список спочатку по парних позиціях (2,4,6-й елементи і т.д.), потім по непарних. Тобто, шаблон ітератор дозволяє реалізовувати різноманітні

способи проходження по колекції незалежно від виду і способу представлення даних в колекції.

Переваги та недоліки: Цей шаблон дозволяє уніфікувати операції проходження по наборам об'єктів для всіх наборів. Тобто, незалежно від реалізації (масив, зв'язаний список, незв'язаний список, дерево та ін.), кожен з наборів може використовувати будь-який з реалізованих ітераторів.

+ Дозволяє реалізувати різні способи обходу структури даних.
+ Спрощує класи зберігання даних.
- Не виправданий, якщо можна обійтися простим циклом.

Шаблон «Proxy»

Призначення: «Proxy» (Проксі) – об'єкти є об'єктами-заглушками або двійниками/замінниками для об'єктів конкретного типу. Зазвичай, проксі об'єкти вносять додатковий функціонал або спрощують взаємодію з реальними об'єктами [5]. Проксі об'єкти використовувалися в більш ранніх версіях інтернет- браузерів, наприклад, для відображення картинки: поки картинка завантажується, користувачеві відображається «заглушка» картинки.

Переваги та недоліки:

+ Легкість впровадження проміжного рівня без переробки клієнтського коду.
+ Додаткові можливості по керуванню життєвим циклом об’єкту.
- Існує ризик падіння швидкості роботи через впровадження додаткових операцій.
- Існує ризик неадекватної заміни відповіді клієнтському коду

Шаблон «State»

Призначення: Шаблон «State» (Стан) дозволяє змінювати логіку роботи об'єктів у випадку зміни їх внутрішнього стану [6]. Наприклад, відсоток нарахованих на картковий рахунок грошей залежить від стану картки: Visa Electron, Classic, Platinum і т.д. Або обсяг послуг, які надані хостинг компанією, змінюється в залежності від обраного тарифного плану (стану членства – бронзовий, срібний або золотий клієнт). Реалізація даного шаблону полягає в наступному: 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 [6, 8].

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.

Це дозволяє легко додавати в майбутньому і обробляти нові стани, відокремлювати залежні від стану елементи об'єкта в інших об'єктах, і відкрито проводити заміну стану (що має сенс у багатьох випадках).

Переваги та недоліки:

+ Код специфічний для окремого стану реалізується в класі стану.
+ Класи та об’єкти станів можна використовувати з різними контекстами, за рахунок чого збільшується гнучкість системи.
+ Код контексту простіше читати, тому що вся залежна від станів логіка винесена в інші класи.
+ Відносно легко додавати нові стани, головне правильно змінити переходи між станами.
- Клас контекст стає складніше через ускладнений механізм переключення станів.

Шаблон «Strategy»

Призначення: Шаблон «Strategy» (Стратегія) дозволяє змінювати деякий

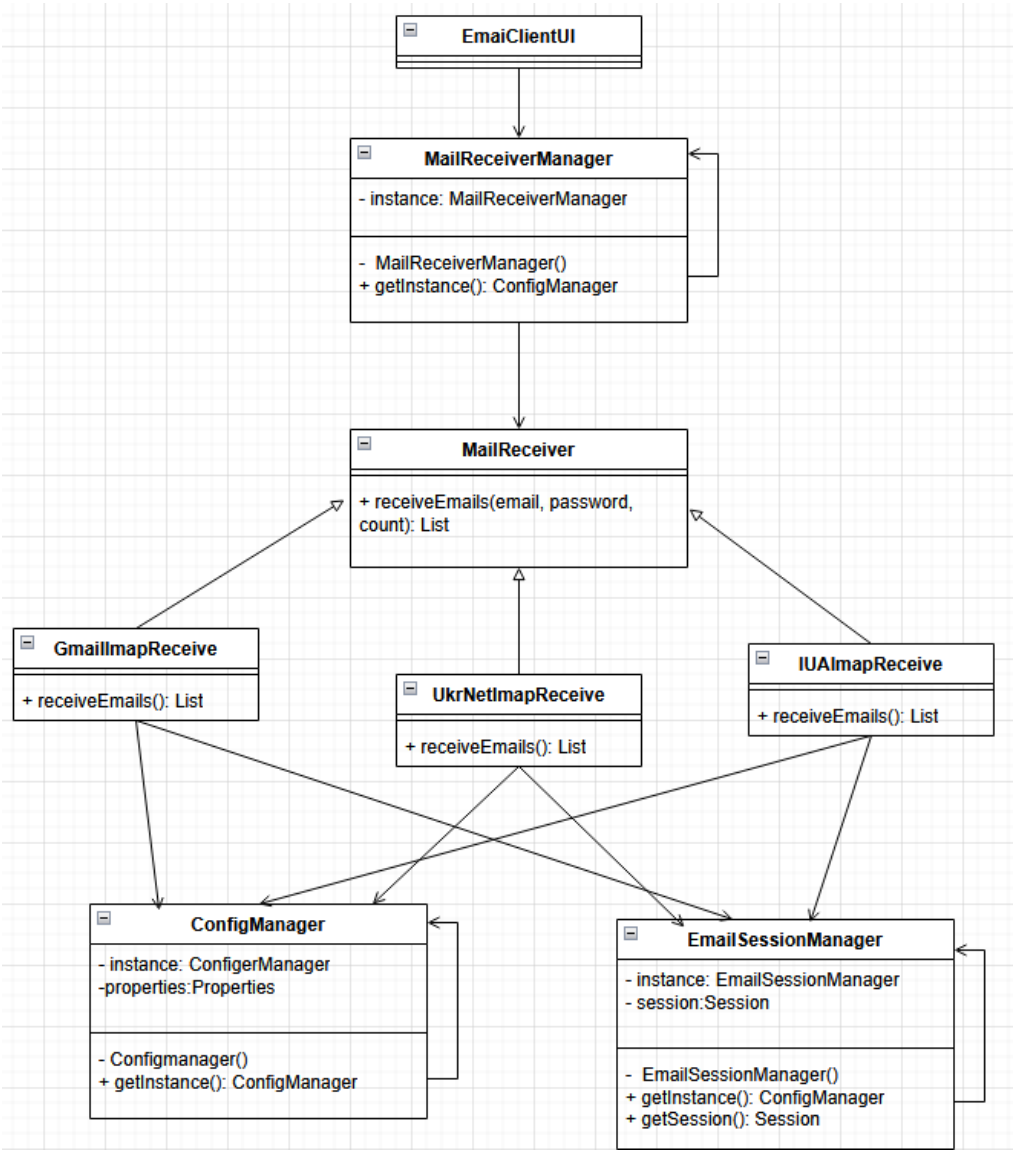
алгоритм поведінки об'єкта іншим алгоритмом, що досягає ту ж мету іншим способом. Прикладом можуть служити алгоритми сортування: кожен алгоритм має власну реалізацію і визначений в окремому класі; вони можуть бути взаємозамінними в об'єкті, який їх використовує [6].

Даний шаблон дуже зручний у випадках, коли існують різні «політики» обробки даних. По суті, він дуже схожий на шаблон «State» (Стан), проте використовується в абсолютно інших цілях – незалежно від стану об'єкта відобразити різні можливі поведінки об'єкта (якими досягаються одні й ті самі або схожі цілі).

Переваги та недоліки:

+ Використовувані алгоритми можна змінювати під час виконання.
+ Реалізація алгоритмів відокремлюється від коду, що його використовує.
+ Зменшує кількість умовних операторів типу switch та if в контексті.
- Надмірна складність, якщо у вас лише кілька невеликих алгоритмів.
- Під час виклику алгоритму, клієнтський код має враховувати різницю між стратегіями.

Хід роботи



Представлена діаграма класів відображає структуру програмної системи клієнта електронної пошти, призначеного для отримання листів з різних поштових серверів через протокол IMAP. Центральним елементом є клас MailReceiverManager, який реалізує патерн Singleton і відповідає за керування процесом отримання електронних листів. Клас взаємодіє з інтерфейсом користувача EmailClientUI, який ініціює запит на отримання пошти.

Основна логіка обробки електронних повідомлень зосереджена в абстрактному класі MailReceiver, який визначає метод receiveEmails(email, password, count). Цей метод реалізується у похідних класах — GmailImapReceive, UkrNetImapReceive та IUAIMapReceive, кожен з яких відповідає за отримання листів із конкретного поштового сервісу. Такий підхід

забезпечує гнучкість і розширюваність системи, дозволяючи легко додавати нові типи поштових серверів без зміни базової логіки.

Для забезпечення належного функціонування програми використовуються допоміжні класи ConfigManager та EmailSessionManager, які також реалізують патерн Singleton. Клас ConfigManager зберігає конфігураційні параметри, такі як адреси серверів, порти, параметри безпеки та інші властивості, необхідні для підключення. Клас EmailSessionManager відповідає за створення та підтримку поштової сесії, використовуючи об’єкт типу Session. Обидва класи забезпечують централізований доступ до налаштувань і ресурсів програми, що спрощує керування залежностями між компонентами.

Питання до лабораторної роботи

1. Що таке шаблон проєктування?

   Шаблон проєктування — це перевірене, багаторазово використовуване рішення типових проблем, що виникають у програмуванні. Це не готовий код, а загальний опис архітектурного підходу, який допомагає створювати більш гнучкий, масштабований та підтримуваний код.

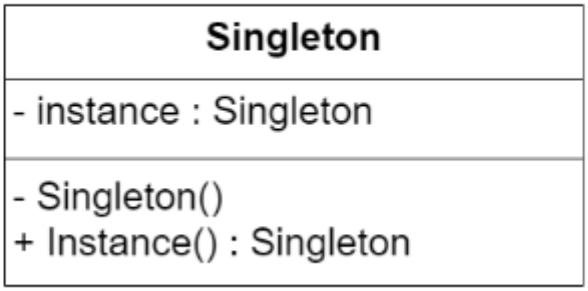
2. Навіщо використовувати шаблони проєктування?

   Шаблони проєктування використовуються для створення більш надійного, гнучкого та масштабованого програмного забезпечення, оскільки вони надають перевірені рішення для типових завдань, економлять час розробників та покращують спілкування між ними.

3. Яке призначення шаблону «Стратегія»?

   «Singleton» (Одинак) являє собою клас в термінах ООП, який може мати не більше одного об'єкта (звідси і назва «одинак»). Насправді, кількість об'єктів можна задати (тобто не можна створити більш n об'єктів даного класу). Даний об'єкт найчастіше зберігається як статичне поле в самому класі.

4. Нарисуйте структуру шаблону «Стратегія».



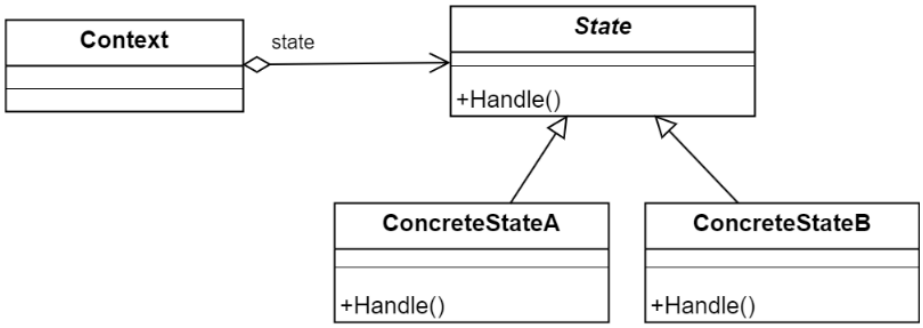
5. Які класи входять в шаблон «Стратегія», та яка між ними взаємодія?

   В основі шаблону "Стратегія" в програмуванні лежать три класи: Context, Strategy та ConcreteStrategy. Context – це клас, що використовує стратегію, Strategy – це інтерфейс або абстрактний клас, що визначає загальний метод стратегії, а ConcreteStrategy – це конкретні класи, які реалізують цей інтерфейс, надаючи власну логіку. Взаємодія між ними полягає в тому, що Context делегує виконання певної роботи об'єктам ConcreteStrategy, які він містить, а Context може динамічно змінювати стратегію, яку він використовує, залежно від умов

6. Яке призначення шаблону «Стан»?

   Шаблон «State» (Стан) дозволяє змінювати логіку роботи об'єктів у випадку зміни їх внутрішнього стану. Наприклад, відсоток нарахованих на картковий рахунок грошей залежить від стану картки: Visa Electron, Classic, Platinum і т.д. Або обсяг послуг, які надані хостинг компанією, змінюється в залежності від обраного тарифного плану (стану членства – бронзовий, срібний або золотий клієнт).

7. Нарисуйте структуру шаблону «Стан».



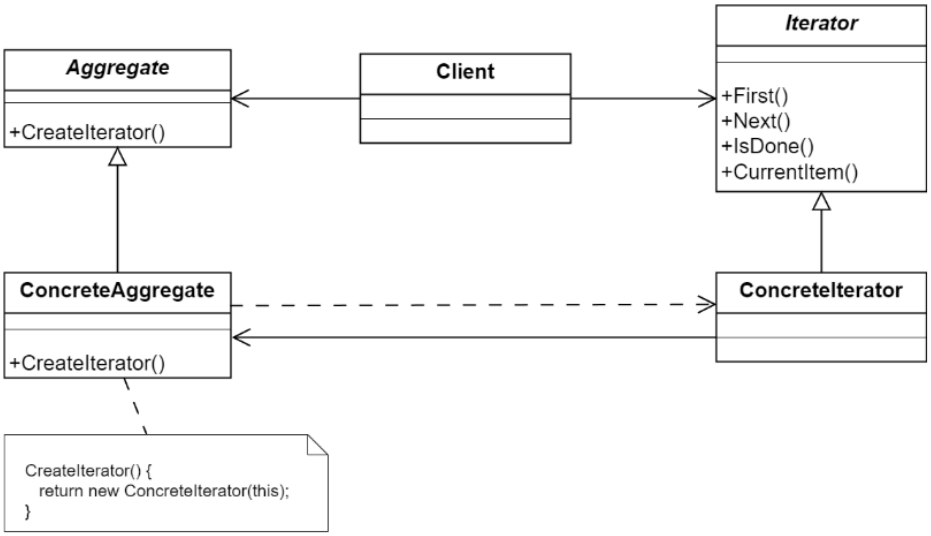
8. Які класи входять в шаблон «Стан», та яка між ними взаємодія?

Реалізація даного шаблону полягає в наступному: 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. 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.

9. Яке призначення шаблону «Ітератор»?

«Iterator» (Ітератор) являє собою шаблон реалізації об'єкта доступу до набору (колекції, агрегату) елементів без розкриття внутрішніх механізмів реалізації. Ітератор виносить функціональність перебору колекції елементів з самої колекції, таким чином досягається розподіл обов'язків: колекція відповідає за зберігання даних, ітератор – за прохід по колекції.

10. Нарисуйте структуру шаблону «Ітератор».



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

Клас Aggregate визначає інтерфейс для створення ітератора за допомогою методу CreateIterator(). Його конкретна реалізація — клас ConcreteAggregate — містить колекцію елементів і реалізує метод CreateIterator(), який створює об’єкт типу ConcreteIterator, передаючи йому посилання на себе.

Клас Iterator задає інтерфейс для обходу колекції та містить методи First(), Next(), IsDone() і CurrentItem(). Ці методи дозволяють отримати перший елемент, перейти до наступного, перевірити, чи досягнуто кінця колекції, та отримати поточний елемент. Клас ConcreteIterator реалізує цей інтерфейс і зберігає інформацію про поточну позицію в колекції, отримуючи доступ до елементів через посилання на об’єкт ConcreteAggregate.

Клас Client є користувачем цього механізму. Він створює об’єкт конкретного агрегату (ConcreteAggregate) і через метод CreateIterator() отримує відповідний ітератор. Далі клієнт використовує методи ітератора (First(), Next(), IsDone(), CurrentItem()) для послідовного перегляду всіх елементів колекції, не маючи уявлення про те, як ці елементи зберігаються всередині.

12. В чому полягає ідея шаблона «Одинак»?

Ідея шаблону «Одинак» полягає в тому, щоб гарантувати, що клас матиме тільки один екземпляр, і забезпечити глобальну точку доступу до цього єдиного екземпляра.

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

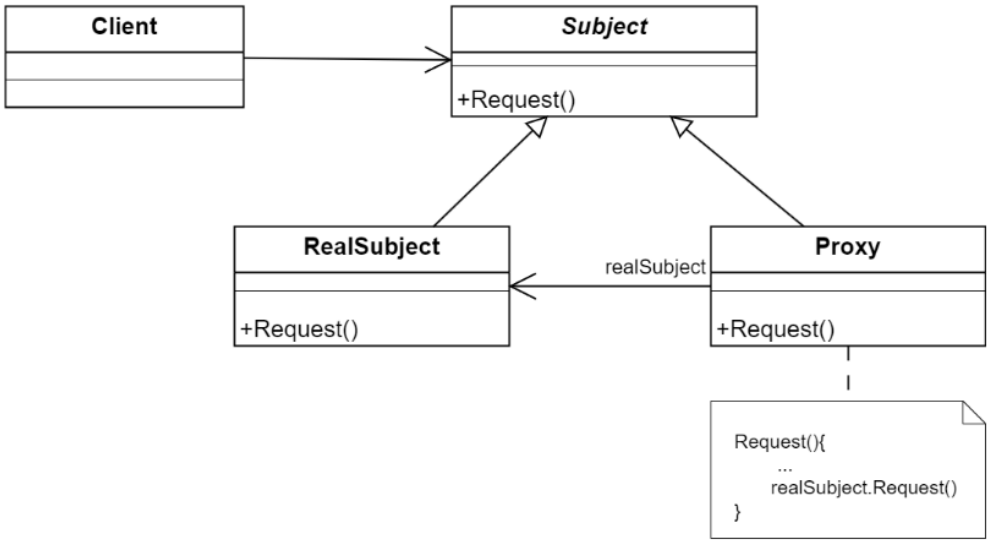
В даний час патерн «Одинак» багато хто вважає т.зв. «анти-шаблоном», тобто поганою практикою проєктування. Це пов'язано з тим, що «одинаки» представляють собою глобальні дані (як глобальна змінна), що мають стан. Стан глобальних об'єктів важко відслідковувати і підтримувати коректно; також глобальні об'єкти важко тестуються і вносять складність в програмний код (у всіх ділянках коду виклик в одне єдине місце з «одинаком»; при зміні підходу доведеться змінювати масу коду).

14. Яке призначення шаблону «Проксі»?

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

проксі об'єкти вносять додатковий функціонал або спрощують взаємодію з реальними об'єктами.

15. Нарисуйте структуру шаблону «Проксі».



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

Клас Subject визначає спільний інтерфейс, який містить метод Request(). Він є базовим для двох інших класів — RealSubject і Proxy. Клас RealSubject виконує основну роботу та реалізує метод Request(), у якому міститься реальна логіка виконання запиту, наприклад, обробка даних чи доступ до зовнішнього ресурсу.

Клас Proxy також реалізує інтерфейс Subject і містить посилання на об’єкт типу RealSubject. У методі Request() він може виконувати додаткові дії перед або після звернення до реального об’єкта. Наприклад, проксі може перевіряти права доступу, ініціалізувати реальний об’єкт лише за потреби або вести журнал викликів. Після цього він викликає realSubject.Request(), делегуючи виконання основної операції справжньому об’єкту.

Клас Client звертається до інтерфейсу Subject, не знаючи, чи має справу з реальним об’єктом (RealSubject), чи з його замісником (Proxy). Це робить роботу клієнта повністю прозорою — поведінка об’єкта не змінюється, навіть якщо між ним і клієнтом стоїть проксі.