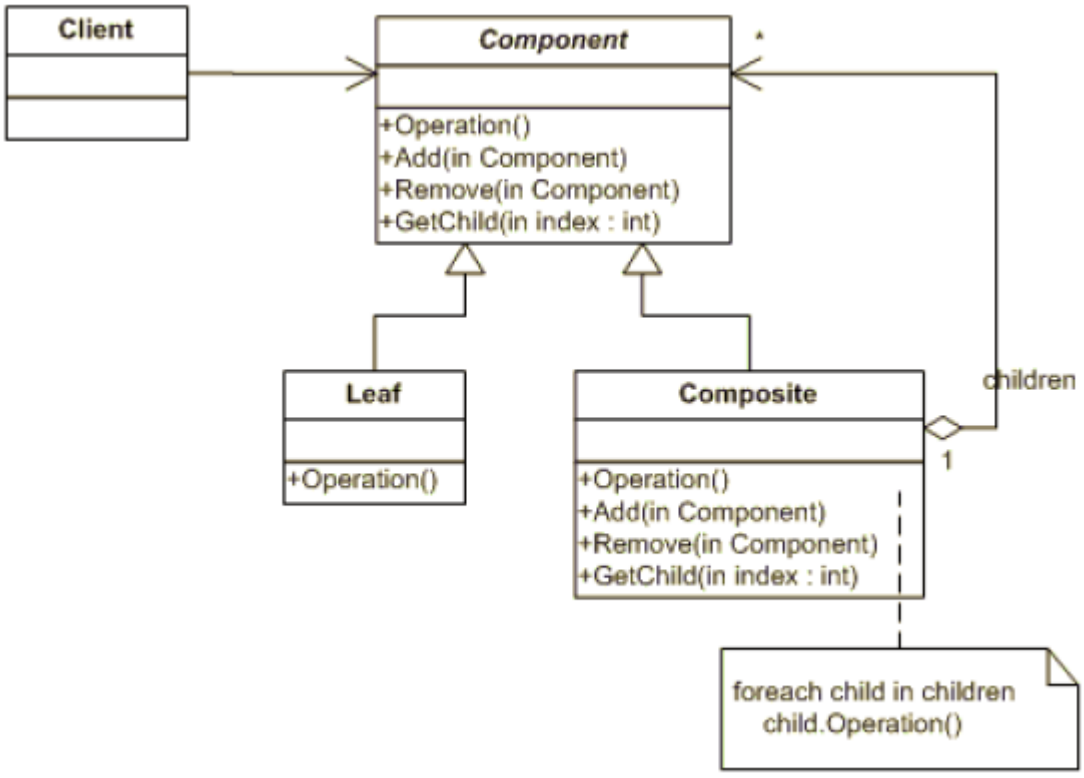
Лабораторна робота №6

Компонувальник (англ. **Composite pattern**) - шаблон проектування, відноситься до структурних патернів, об'єднує об'єкти в деревовидну структуру для представлення ієрархії від приватного до цілого. Компонувальник дозволяє клієнтам звертатися до окремих об'єктів і до груп об'єктів однаково. По іншому це можна сказати так: - компонувальник дозволяє нам зберігати деревовидну структуру і працювати однаково із батьками та синами у дереві.

Мета(Ціль)

Патерн визначає ієрархію класів, які одночасно можуть складатися з примітивних і складних об'єктів, спрощує архітектуру клієнта, робить процес додавання нових видів об'єкта більш простим. Іншими словами Компонує об’єкти в деревовидну структуру для представлення частково-цілої ієрархії. Компонувальник дозволяє розглядати окремі об’єкти і їхні композиції єдиним способом.

UML діаграма



Пристосуванець(легковаговик)(англ. **Flyweight pattern**) - це об'єкт, що представляє себе як унікальний екземпляр в різних місцях програми, але по факту не є таким. Він забезпечує підтримку великої кількості об’єктів шляхом виокремлення спільної інформації для збереження в одному екземплярі

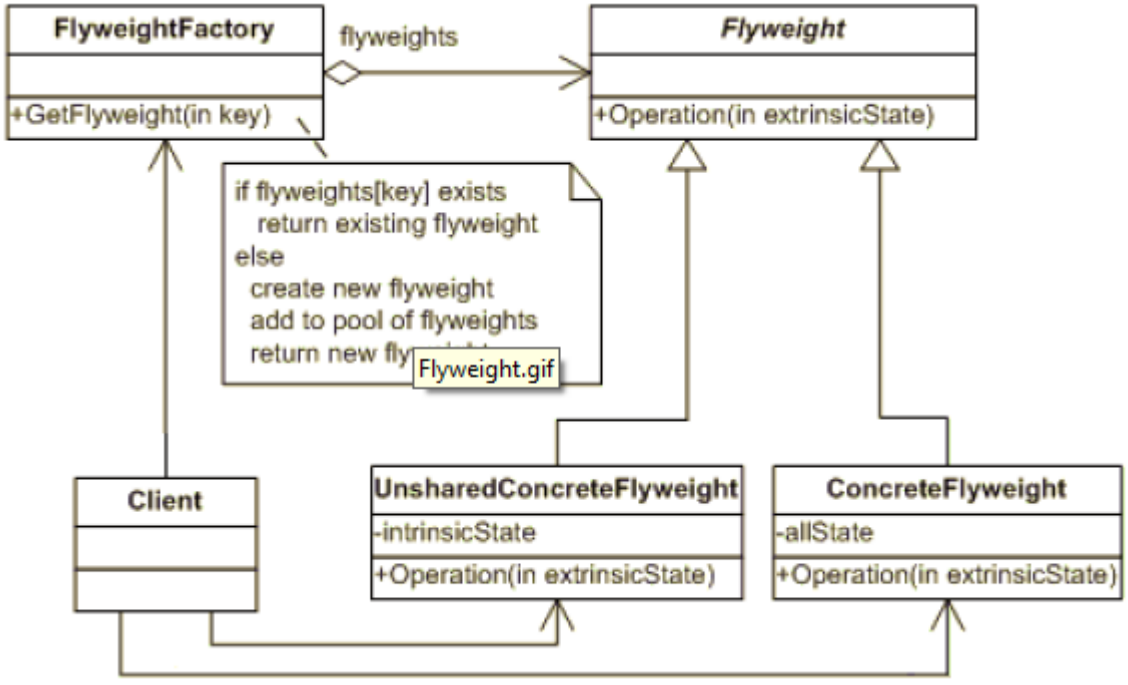
Мета(Ціль)

Оптимізація роботи з пам'яттю, шляхом запобігання створення екземплярів елементів, що мають загальну сутність.

Опис

Flyweight використовується для зменшення витрат при роботі з великою кількістю дрібних об'єктів. При проектуванні пристосуванця(легковаговика) необхідно розділити його властивості на зовнішні і внутрішні. Внутрішні властивості завжди незмінні, тоді як зовнішні можуть відрізнятися в залежності від місця і контексту застосування і повинні бути винесені за межі пристосуванця(легковаговика). Flyweight доповнює патерн Factory таким чином, що Factory при зверненні до неї клієнта для створення нового об'єкта шукає вже створений об'єкт з такими ж параметрами, що і у необхідного, і повертає його клієнту. Якщо такого об'єкта немає, то Factory створить новий.

UML діаграма



Шаблон **Proxy** (визначає об'єкт-заступник англ. Surrogate інакше замінник англ. Placeholder) - шаблон проектування, який надає об'єкт, який контролює доступ до іншого об'єкта, перехоплюючи всі виклики (виконує функцію контейнера). Проксі підміняє реальний об'єкт та надсилає запити до нього тоді, коли це потрібно. Проксі також може ініціалізувати реальний об'єкт, якщо він до того не існував.

Призначення

Надає замінника або утримувача для іншого об’єкту, щоб керувати ним.

Шаблон proxy буває декількох видів, а саме:

- Віддалений заступник (англ. Remote proxies): забезпечує зв'язок з «Суб'єктом», який знаходиться в іншому адресному просторі або на віддаленій машині. Так само може відповідати за кодування запиту та його аргументів і відправку закодованого запиту реальному «Суб'єкту»,
- Віртуальний заступник (англ. Virtual proxies): забезпечує створення реального «Суб'єкта» тільки тоді, коли він дійсно знадобиться. Так само може кешувати частина інформації про реальний «Суб'єкт», щоб відкласти його створення,
- Копіювати-прі-записи: забезпечує копіювання «суб'єкта» при виконанні клієнтом певних дій (окремий випадок «віртуального проксі»).
- Захищаючий заступник (англ. Protection proxies): може перевіряти, чи має викликає об'єкт необхідні для виконання запиту права.
- Кешуючий проксі: забезпечує тимчасове зберігання результатів розрахунку до віддачі їх множинним клієнтам, які можуть розділити ці результати.
- Екранувальний проксі: захищає «Суб'єкт» від небезпечних клієнтів (або навпаки).
- Синхронізуючий проксі: виробляє синхронізований контроль доступу до «Суб'єкту» в асинхронного багатопотокового середовища.
- Smart reference proxy: виробляє додаткові дії, коли на «Суб'єкт» створюється посилання, наприклад, розраховує кількість активних посилань на «Суб'єкт».

Переваги

- Віддалений заступник;

- Віртуальний заступник може виконувати оптимізацію;

- Захищаючий заступник;

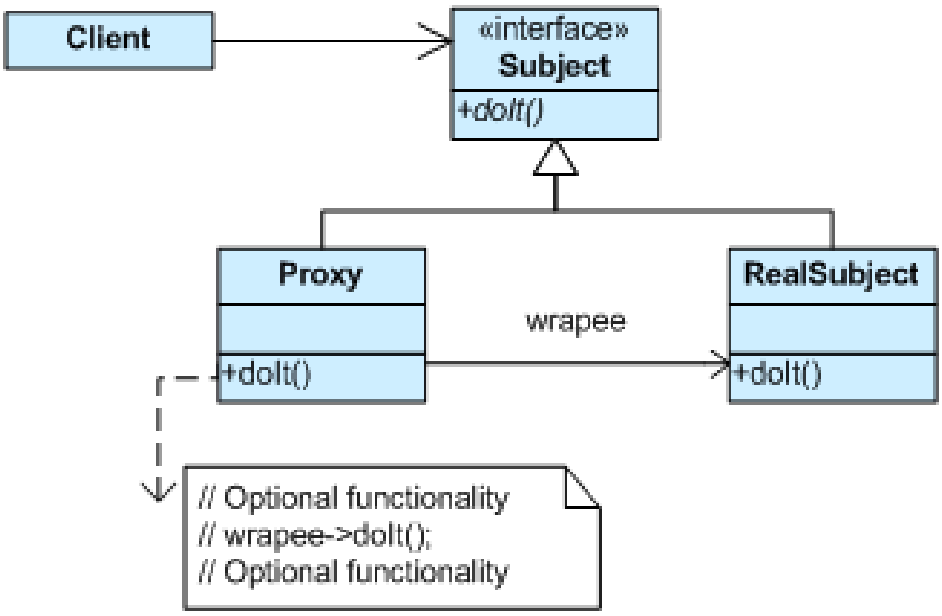
Недоліки

- Різке збільшення часу відгуку.

Сфера застосування

Шаблон Proxy може застосовуватися у випадках роботи з мережевим з'єднанням, з величезним об'єктом у пам'яті (або на диску) або з будь-яким іншим ресурсом, який складно або важко копіювати. Добре відомий приклад застосування - об'єкт, що підраховує число посилань.

UML діаграма



Завдання до Лабораторної роботи

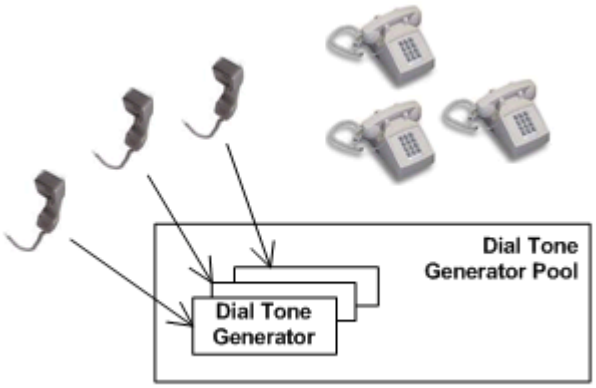
1. Написати програму з використанням шаблону **Proxy** для обліку співробітників фірми. Розробити частину, що відповідає за роботу з локальним джерелом даних. Будемо вважати, що додаток взаємодіє з ним ексклюзивно. Інтерфейс IEmployeeDataSource містить методи для читання і запису даних про співробітника. Його реалізує клас EmployeeDataSource, для породження примірників якого існує Фабричний метод DataSourceFactory.CreateEmployeeDataSource(). Для спрощення прикладу, замість бази даних будемо використовувати static поле _database. Повільну швидкість виконання запиту отримання даних емулюємо викликом методу Sleep (). Крім того, з метою демонстрації додамо висновок на форму повідомлень про поточні операції.
2. Написати програму з використанням шаблону **Composite** , а саме написати код виведення карти міста. Зрозуміло, спростимо його і відкинемо все, що не потрібно для розуміння реалізації шаблону. Почнемо розробку з загального інтерфейсу, який включає посилання на батька, ім'я, а так само методи для відображення на карті і пошуку нащадка по імені. Будемо вважати, що компонент зберігає положення щодо свого батьківського елементу, координати лівого нижнього кута якого передаються йому в параметрах x і y методу Draw (). Створимо базову реалізацію даного інтерфейсу. Оскільки в загальному випадку не відомо як виводити компонент, то зробимо метод Draw () абстрактним. Додамо два поля для збереження відносних координат об'єкту: _x і _y. Крім того, в загальному випадку немає даних про нащадків, тому FindChild () буде просто перевіряти відповідність імені

даного екземпляра шуканому. В якості бази для реалізації складеного об'єкта будемо використовувати MapComponent. У цьому випадку необхідно тільки перевизначити методи Draw () і FindChild (), а так само реалізувати AddComponent (). З останнім все просто: будемо використовувати List в ролі сховища об'єктів і просто додамо новий елемент у його список.

3. Написати програму з використанням шаблону **Flyweight.** Уявіть що нам потрібно поставити п'єсу. Однак за сценарієм в цій п'єсі задіяні кілька десятків людей, які по своїй суті виконують однакові дії, наприклад беруть участь у масовках різних сцен в різні проміжки часу, але між ними все ж є якісь відмінності (наприклад костюми). Нам би коштувало величезних грошей наймати для кожної ролі окремого актора, тому ми використовуємо патерн «пристосуванець». Ми створимо всі потрібні нам костюми, але для кожної масовки будемо переодягати невелику групу акторів в необхідні для цієї сцени костюми. У результаті ми маємо можливість ціною малих ресурсів створювати видимість управління великою кількістю здавалося б різних об'єктів.
4. Написати програму з використанням шаблону **Proxy.** Співробітникам одного з підрозділів фірми регулярно потрібно отримувати інформацію про те, якого числа бухгалтерія планує виплатити зарплату. З одного боку кожен з них може індивідуально і регулярно їздити в бухгалтерію для з'ясування цього питання (гадаю така ситуація нерідко зустрічається у багатьох організаціях). З іншого боку, при наближенні запланованої дати підрозділ може вибрати одну людину, яка буде з'ясовувати цю інформацію у бухгалтерії, а надалі вже всі в підрозділі можуть з'ясувати цю інформацію у нього (що значно швидше). Ось саме ця людина і буде реалізованим «проксі» патерном, який надаватиме спеціальний механізм доступу до інформації з бухгалтерії.
5. Написати програму з використанням шаблону **Composite.** Для прикладу розглянемо керування солдатами в строю. Існує стройової статут, який визначає як управляти строєм і згідно цього статуту абсолютно неважливо кому віддається наказ (наприклад «кроком руш») одному солдатові або цілому взводу. Відповідно до статуту (якщо його в чистому вигляді вважати патерном «компоновщик») не можна включити команду, яку може виконати тільки один солдат, але не може виконати група, або навпаки.
6. Написати програму з використанням шаблону **Composite.** Написати найпростіший суматор виразів із застосуванням патерну «Composite». Суматор не повинен займатися розбором виразів, він повинен лише описувати і реалізовувати деревоподібну структуру для більш зручного обчислення виразів.

Вираз 45-(18-17)-(11+8)
Привести до 20 - a - b

7. Написати програму з використанням шаблону **Flyweight**, а саме телефонну мережу загального користування. Такі ресурси як генератори тональних сигналів (Зайнято і т.д.), приймачі цифр номера абонента, що набирається в тоновом наборі, є загальними для всіх абонентів.



Коли абонент піднімає трубку, щоб подзвонити, йому надається доступ до всіх потрібних ресурсів, що розділяються.

8. Написати програму з використанням шаблону **Proxy.** Банківський чек по суті є замісником грошей на банківському рахунку. Чек може бути використаний замість грошей для покупок і по суті керує доступом до грошей на рахунку власника. Реалізувати дану систему
9. Написати програму з використанням шаблону **Flyweight**, Розглянемо завдання проектування векторного редактора. Причому, для отримання максимальної гнучкості редактора, необхідно спроектувати його аж до найнижчих рівнів - до елементарного примітиву - точки. Очевидно, що при прямому підході в одному малюнку може бути величезна кількість точок, працювати з якими і зберігати їх в пам'яті редактору буде не просто. Тому, спроектуємо редактор використовуючи патерн «Пристосуванець». Нехай векторний редактор, вміє працювати з обмеженим набором примітивів - Квадрат (Square), Окружність (Circle), Точка (Point) та Зображення (Picture). Всі ці примітиви є пристосуванцями. Причому, лише зображення не є розділяються пристосуванцем, зважаючи на свою складовою природи. Варто відзначити, що зображення в даному випадку, приклад патерну «Компоновщик». Решта ж примітиви повинні бути розділені на підставі свого внутрішнього стану. Для квадрата - розміри сторін (height, width), для окружності - радіус (radius). Точка, хоч і є розділяються пристосуванцем, не має внутрішнього стану, з огляду на

те, що вся необхідна для її відтворення інформація віддається клієнтом в т.зв. контексті малювання (Context).