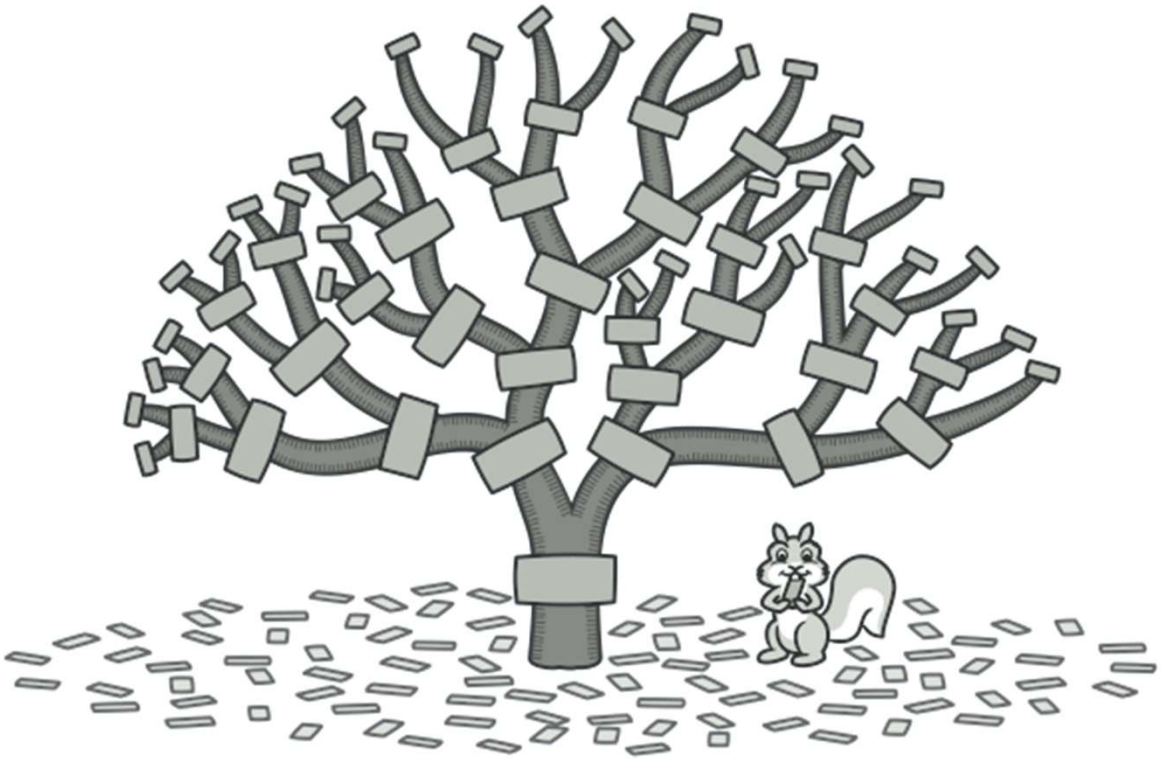
Composite Pattern

- **Компонувальник** (також відомий як Дерево або Композит) — це структурний патерн проектування, що дає змогу згрупувати декілька об'єктів у деревоподібну структуру, а потім працювати з нею так, ніби це одиничний об'єкт.
- Армії більшості країн можуть бути представлені у вигляді перевернутих дерев. На нижньому рівні у вас солдати, далі взводи, далі полки, а далі цілі армії. Накази віддаються зверху вниз структурою командування до тих пір, поки вони не доходять до конкретного солдата.



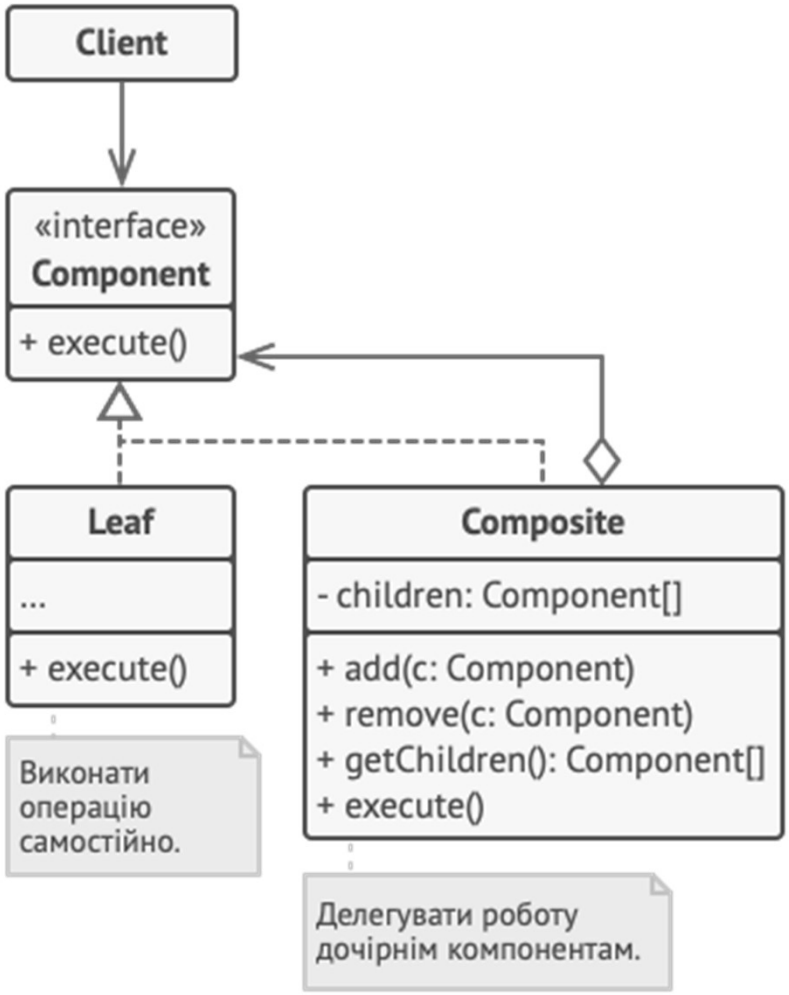
Структура

Компонент описує загальний інтерфейс для простих і складових компонентів дерева.

Лист — це простий компонент дерева, який не має відгалужень. Класи листя міститимуть більшу частину корисного коду, тому що їм нікому передавати його виконання.

Контейнер (або *композит*) — це складовий компонент дерева. Він містить набір дочірніх компонентів, але нічого не знає про їхні типи. Це можуть бути як прості компоненти-листя, так і інші компоненти-контейнери. Проте, це не проблема, якщо усі дочірні компоненти дотримуються єдиного інтерфейсу. Методи контейнера переадресовують основну роботу своїм дочірнім компонентам, хоча можуть додавати щось своє до результату.

Клієнт працює з деревом через загальний інтерфейс компонентів.



Життєвий приклад для кращого розуміння

- Ми пакуємо валізу у подорож. У нас всередині нашої валізи можуть бути абсолютно різні речі: одяг, засоби гігієни тощо. Давайте візьмемо зубну щітку (лист) та мило (лист). Ми їх покладемо в окрему сумочку для засобів гігієни(контейнер). Цю сумочку ми можемо покласти в більш велику сумку і потім вже покласти у саму валізу. Тобто у валізі буде масив інших «валіз».

Застосування

1. Якщо вам потрібно представити деревоподібну структуру об'єктів.

- Патерн Компонувальник пропонує зберігати в складових об'єктах посилання на інші прості або складові об'єкти. Вони, у свою чергу, теж можуть зберігати свої вкладені об'єкти і так далі. У підсумку, ви можете будувати складну деревоподібну структуру даних, використовуючи всього два основних різновида об'єктів.

2. Якщо клієнти повинні однаково трактувати прості та складові об'єкти.

- Завдяки тому, що прості та складові об'єкти реалізують спільний інтерфейс, клієнту байдуже, з яким саме об'єктом він працюватиме.

Взаємодія Composite з іншими патернами

Паттерн Composit дозволяє об'єднувати об'єкти в деревоподібні структури, що дозволяє обробляти окремі об'єкти та групи об'єктів однаково. Це дозволяє створювати складні структури, які можуть бути опрацьовані в одному блоку коду. Паттерн Composit може бути поєднаний з іншими паттернами, такими як Iterator або Visitor, щоб здійснювати ітерацію чи візитування вузлів дерева.

Використання патерну Composite

1. Adobe використовує паттерн Composite у своєму програмному забезпеченні для створення складних об'єктів зі складових частин, таких як веб-сайти та мультимедійні презентації.
2. Google використовує паттерн Composite в своїх програмних продуктах для створення складних ієрархій об'єктів, наприклад, вікон та інтерфейсів користувача.

Кроки реалізації

1. Переконайтеся, що вашу бізнес-логіку можна представити як деревоподібну структуру. Спробуйте розбити її на прості компоненти й контейнери. Пам'ятайте, що контейнери можуть містити як прості компоненти, так і інші вкладені контейнери.
2. Створіть загальний інтерфейс компонентів, який об'єднає операції контейнерів та простих компонентів дерева. Інтерфейс буде вдалим, якщо ви зможете використовувати його, щоб взаємозаміняти прості й складові компоненти без втрати сенсу.
3. Створіть клас компонентів-листя, які не мають подальших відгалужень. Майте на увазі, що програма може містити декілька таких класів.
4. Створіть клас компонентів-контейнерів і додайте до нього масив для зберігання посилань на вкладені компоненти. Цей масив повинен бути здатен містити як прості, так і складові компоненти, тому переконайтеся, що його оголошено з типом інтерфейсу компонентів.

   Реалізуйте в контейнері методи інтерфейсу компонентів, пам'ятаючи про те, що контейнери повинні делегувати основну роботу своїм дочірнім компонентам.

5. Додайте операції додавання й видалення дочірніх компонентів до класу контейнерів.

Майте на увазі, що методи додавання/видалення дочірніх компонентів можна оголосити також і в інтерфейсі компонентів. Так, це порушить *принцип розділення інтерфейсу*, тому що реалізації методів будуть порожніми в компонентах-листях. Проте усі компоненти дерева стануть дійсно однаковими для клієнта.

Переваги та недоліки

Переваги:

- Спрощує архітектуру клієнта при роботі зі складним деревом компонентів.
- Полегшує додавання нових видів компонентів.

Недоліки:

- Створює занадто загальний дизайн класів (порушує Interface Segregation) .