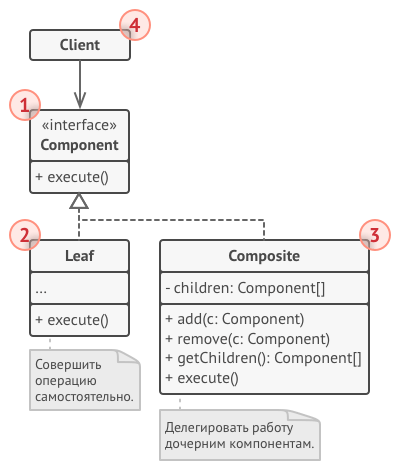
**КОМПОНОВЩИК** - объединяет объекты в древовидную структуру для представлений иерархий "часть/целое". Позволяет клиенту выполнять однородные операции с отдельными объектами и их совокупностями. Все компоненты должны реализовывать один интерфейс, но листья и узлы обычно имеют разные роли, поэтому какие-то методы будут общими, а какие-то необходимо будет переопределять в реализациях. Следовательно, часть методов будет не релевантной для определенной реализации, поэтому методы, различные для листьев и узлов, могут по дефолту выбрасывать исключение типа UnsupportedOperationException или возвращать false или null (можно также выделить их в отдельные интерфейсы). В любом из вариантов нужно будет делать проверку с помощью try/catch, boolean, null или опрератора instanceof (если все же выделять их в интерфейсы)

ПРИМЕНЕНИЕ:

- когда нужно представить древовидную структуру объектов

- когда клиенты должны единообразно трактовать простые и составные объекты

РЕАЛИЗАЦИЯ:

**1. АБСТРАКЦИЯ КОМПОНЕНТА**: определяет общий интерфейс для простых и составных компонентов дерева. Для методов, которые могут быть нерелевантными для какой-то из реализаций, по дефолту при его вызове стоит вызывать UnsupportedOperationException.

**2. ЛИСТ**: простой компонент дерева, не имеющий ответвлений. Из-за того, что им некому больше передавать выполнение, классы листьев будут содержать большую часть полезного кода. Может хранить ссылку на свой узел (например, при удалении листа, необходимо будет подняться на уровень узла).

**3. КОНТЕЙНЕР (УЗЕЛ, КОМПОЗИТ)**: составной компонент дерева. Содержит набор дочерних компонентов, но ничего не знает об их типах. Это могут быть как простые компоненты-листья, так и другие компоненты-контейнеры. Не является проблемой, если все дочерние компоненты следуют единому интерфейсу. Методы контейнера переадресуют основную работу своим дочерним компонентам, хотя и могут добавлять что-то своё к результату.

**4. КЛИЕНТ**: работает с деревом через общий интерфейс компонентов. Клиенту не важно, что перед ним находится — простой или составной компонент дерева.

ПЛЮСЫ:

- упрощает архитектуру клиента при работе со сложным деревом компонентов

- облегчает добавление новых видов компонентов

МИНУСЫ:

- создаёт слишком общий дизайн классов