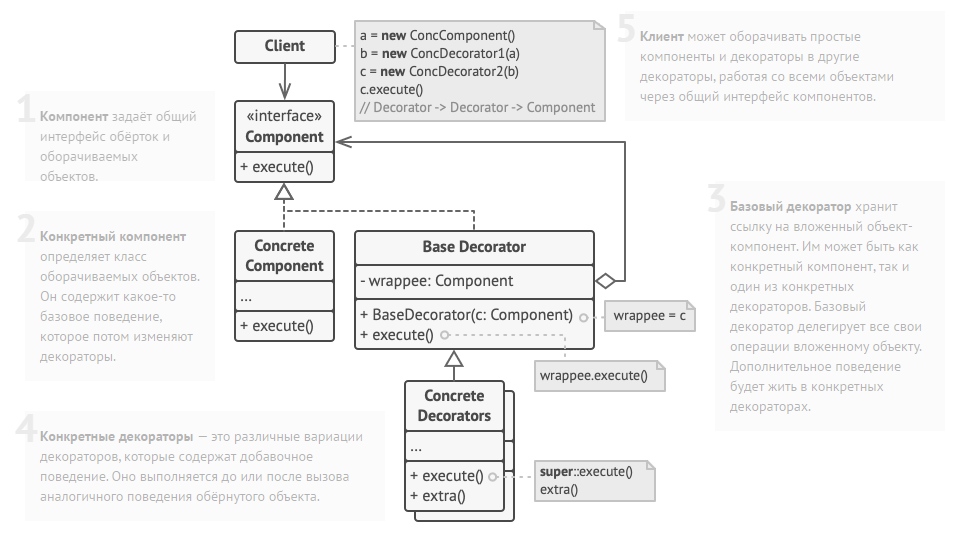
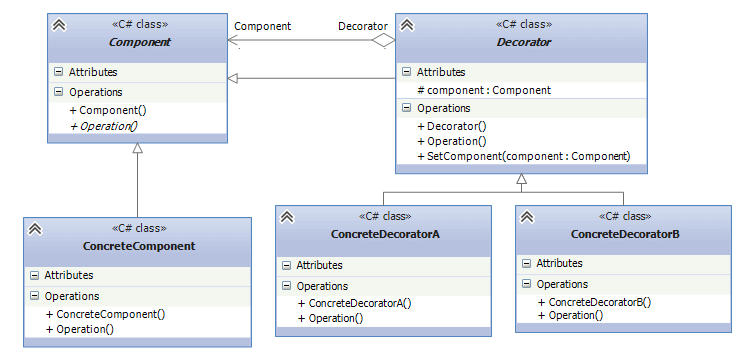
<https://refactoring.guru/ru/design-patterns/decorator>

**Декоратор** — это структурный паттерн проектирования, который позволяет динамически добавлять объектам новую функциональность, оборачивая их в полезные «обёртки».



<https://metanit.com/sharp/patterns/4.1.php>



Декоратор (Decorator) представляет структурный шаблон проектирования, который позволяет динамически подключать к объекту дополнительную функциональность.

Когда следует использовать декораторы?

* Когда надо динамически добавлять к объекту новые функциональные возможности. При этом данные возможности могут быть сняты с объекта
* Когда применение наследования неприемлемо. Например, если нам надо определить множество различных функциональностей и для каждой функциональности наследовать отдельный класс, то структура классов может очень сильно разрастись. Еще больше она может разрастись, если нам необходимо создать классы, реализующие все возможные сочетания добавляемых функциональностей.

Component: абстрактный класс, который определяет интерфейс для наследуемых объектов

ConcreteComponent: конкретная реализация компонента, в которую с помощью декоратора добавляется новая функциональность

Decorator: собственно декоратор, реализуется в виде абстрактного класса и имеет тот же базовый класс, что и декорируемые объекты. Поэтому базовый класс Component должен быть по возможности легким и определять только базовый интерфейс.

Класс декоратора также хранит ссылку на декорируемый объект в виде объекта базового класса Component и реализует связь с базовым классом как через наследование, так и через отношение агрегации.

Классы ConcreteDecoratorA и ConcreteDecoratorB представляют дополнительные функциональности, которыми должен быть расширен объект ConcreteComponent.