**DECORATOR DESIGN PATTERN** (Important)

When to use:

* Dynamically add or modify behavior of objects without altering their structure.
* Avoid subclass explosion by providing a flexible alternative.
* Extend functionality without changing existing code.
* Compose objects with different behavior variants at runtime.

Further Explanation: <https://www.youtube.com/watch?v=w6a9MXUwcfY&list=PL6W8uoQQ2c61X_9e6Net0WdYZidm7zooW&index=6>

The Decorator Design Pattern is like adding layers of wrapping paper around a gift box. Each layer adds something extra, but the gift box remains the same underneath.

In programming, it's used to add new features to objects dynamically without changing their original structure. Let's say you have a simple object, like a plain cake. With the Decorator Pattern, you can "decorate" this cake with different toppings or decorations without modifying its core.

For example, you can have a basic Cake object representing the plain cake. Then, you can create decorator classes for each additional feature you want to add, like frosting, sprinkles, or fruit. These decorators wrap around the original cake object, adding their own functionality.

So, the Decorator Pattern allows you to build objects with flexible combinations of features, just like how you can customize a cake with various toppings.