

The background of the slide is a blue-tinted image of architectural blueprints. A large blue rectangular box is centered on the page, containing the title text in white. The blueprints show various technical drawings, including a grid, lines, and text such as 'FIRST', 'FLOOR RAISED 21"', 'UP 15 R.', and 'DOTTED INDICATED'. A yellow ruler is visible in the top left corner, and a yellow pencil is in the bottom left corner.

# Software Design & Architecture

## Design Patterns/ Structural Design Patterns

Pengyu Nie

# Design Patterns Categories

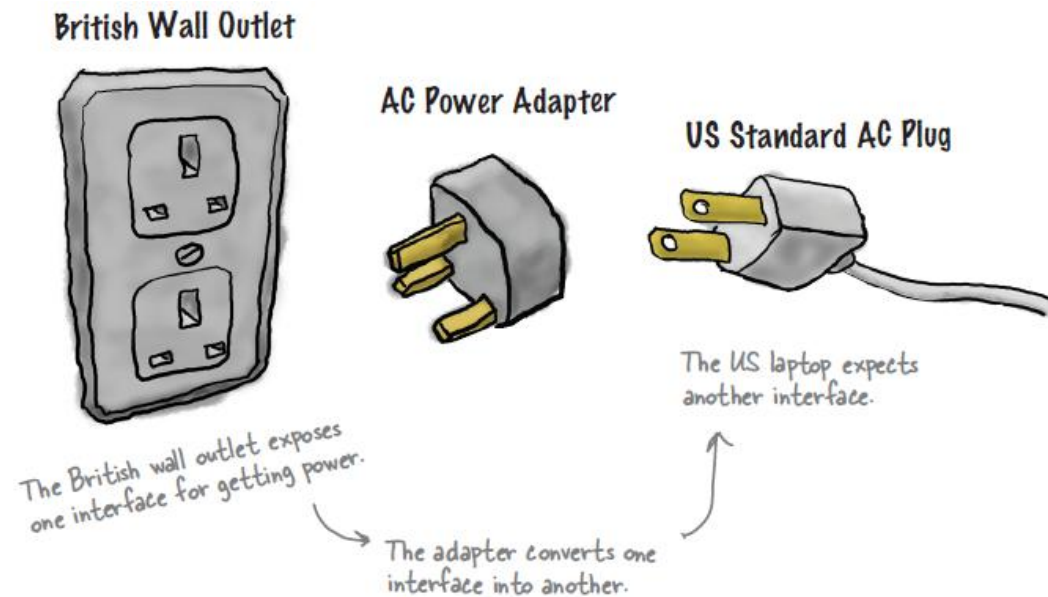
- **Creational**: concern the process of object creation
  - Singleton, Factory Method, Abstract Factory, Builder, Prototype, Object Pool *design patterns 1*
- **Structural**: concern the process of assembling objects and classes
  - Adapter, Composite, Decorator, Façade, Bridge, Flyweight, Proxy *today*
- **Behavioral**: concern the interaction between classes or objects
  - Observer, Strategy, Template Method, Iterator, State, Chain of Responsibility, Command, Mediator, Memento *design patterns 3*  
*design patterns 4 – your pick from the remaining ones*

The image shows a close-up of architectural blueprints on a light blue background. A large, semi-transparent blue rectangle is overlaid on the left side of the page. Inside this rectangle, the word "Adapter" is written in a clean, white, sans-serif font. The blueprints in the background contain various technical drawings, including lines, dimensions, and text. The word "FIRST" is printed in large, bold, capital letters across the top of the blueprints. Other visible text includes "FIRE SHUTTERS SEE SPEC.", "UPPER", "SLOPE", "DRAIN", "FLOOR DRAIN", and "DOTTED LINES INDICATED". There are also several numerical dimensions and symbols scattered throughout the drawings. A white rectangular object, possibly a piece of paper or a tool, is partially visible at the bottom left of the blueprints.

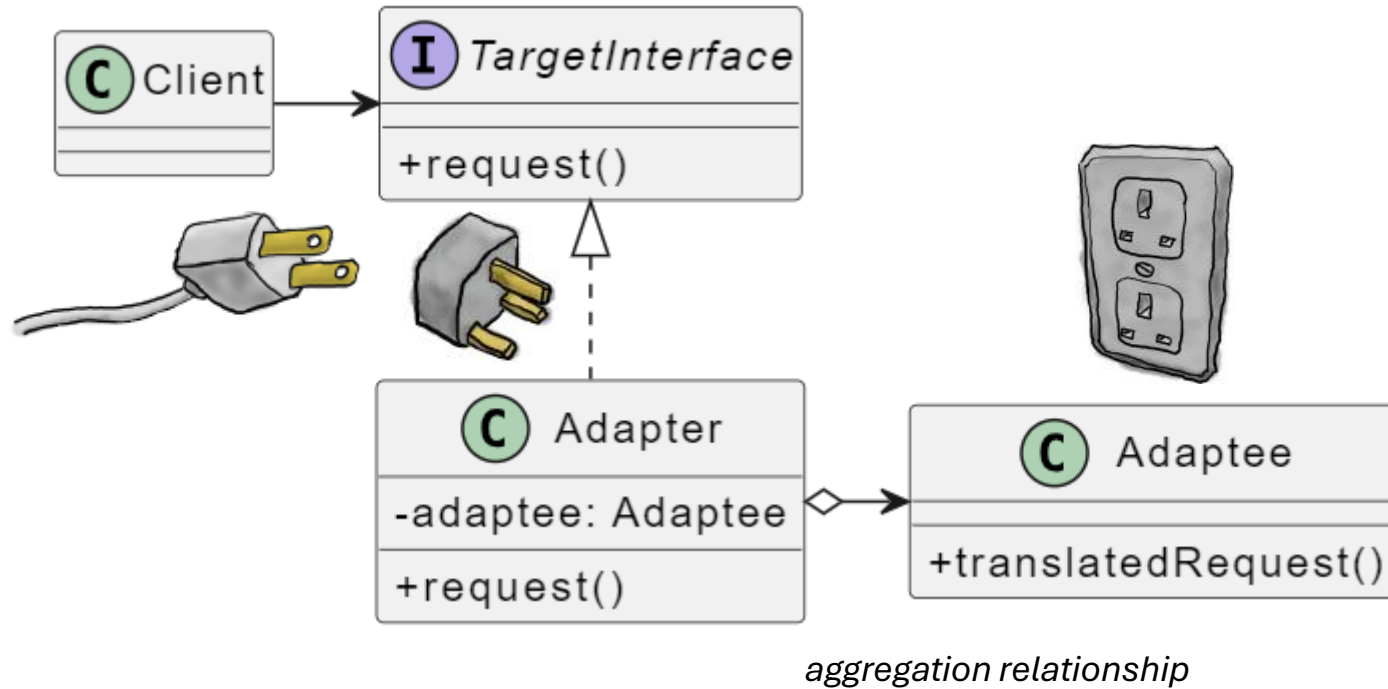
# Adapter

# Adapter: Motivation and Intent

- **Motivation:**
  - we need to interact with objects of a particular class that is incompatible with the current class defined in the client code
  - we do not want to (or more often, cannot) change the class to be used (e.g., they are defined in third-party libraries)
- **Intent:** convert the interface of a class into another interface



# Adapter: Solution



✓ Open-closed principle

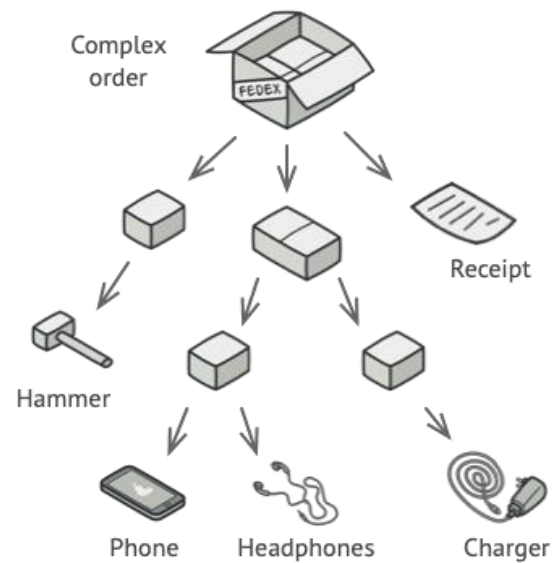


The image shows a close-up of architectural blueprints on a light blue background. A large, semi-transparent blue rectangle is overlaid on the left side of the page. The word "Composite" is written in white, sans-serif font within this rectangle. The blueprints in the background contain various technical drawings, including lines, dimensions, and text such as "FIRST", "UPPER", "SLOPE", "DRAIN", and "FLOOR DRAIN". A yellow ruler is visible in the top left corner, and a yellow pencil is in the bottom right corner.

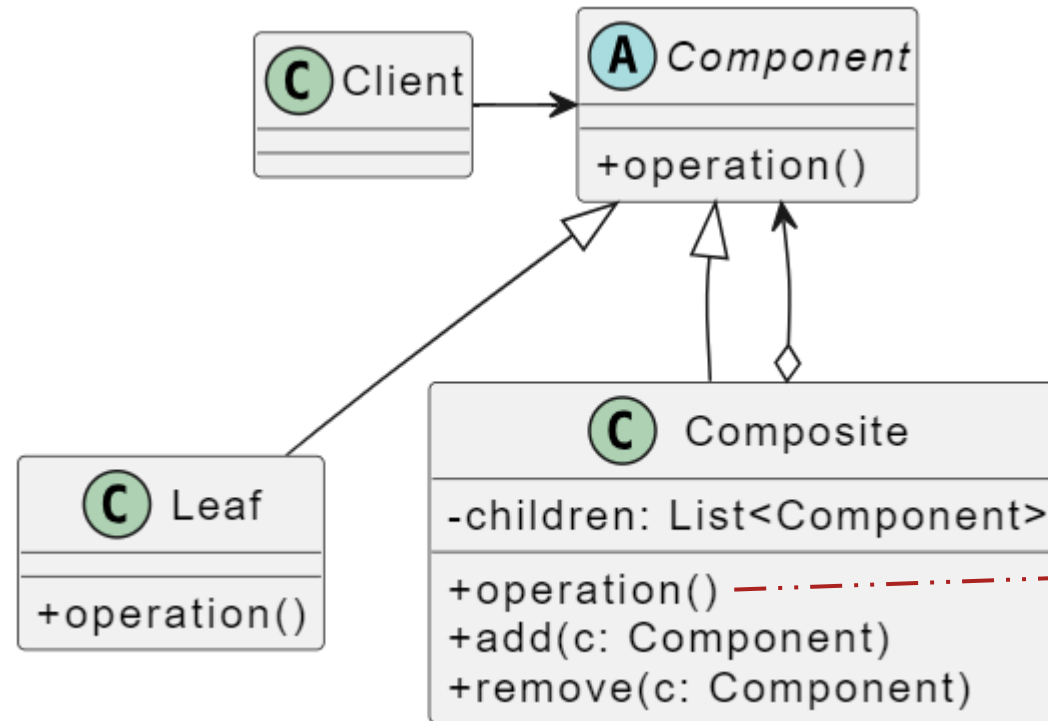
# Composite

# Composite: Motivation and Intent

- **Motivation:**
  - applications that have recursive groupings of primitives and containers (e.g., basic shapes (lines, circle, text) and compound shapes) (e.g., directories and files)
  - client treat containers and primitives in the same way
- **Intent:** compose objects into tree structures; define a shared interface



# Composite: Solution



✓ Liskov substitution principle

```
for (child in children) {
    child.operation()
}
```



The background of the slide is a close-up, slightly blurred image of architectural blueprints. A yellow ruler is visible in the top left corner, and a pencil lies diagonally across the bottom right. The blueprints feature various technical drawings, lines, and text. The word "FIRST" is printed in large, bold, capital letters across the upper middle section. Other visible text includes "UPPER", "FLOOR DRAIN", "SLOPE", "DRAIN", "FLOOR SHUTTERS SEE SPEC.", and "DOTTED LINES INDICATED". Dimensions such as "3'-10\"", "4'-6\"", "7'-6\"", and "3'-8\"" are also present. A blue rectangular box is overlaid on the left side of the image, containing the word "Decorator" in white text.

# Decorator

# Decorator: Motivation and Intent

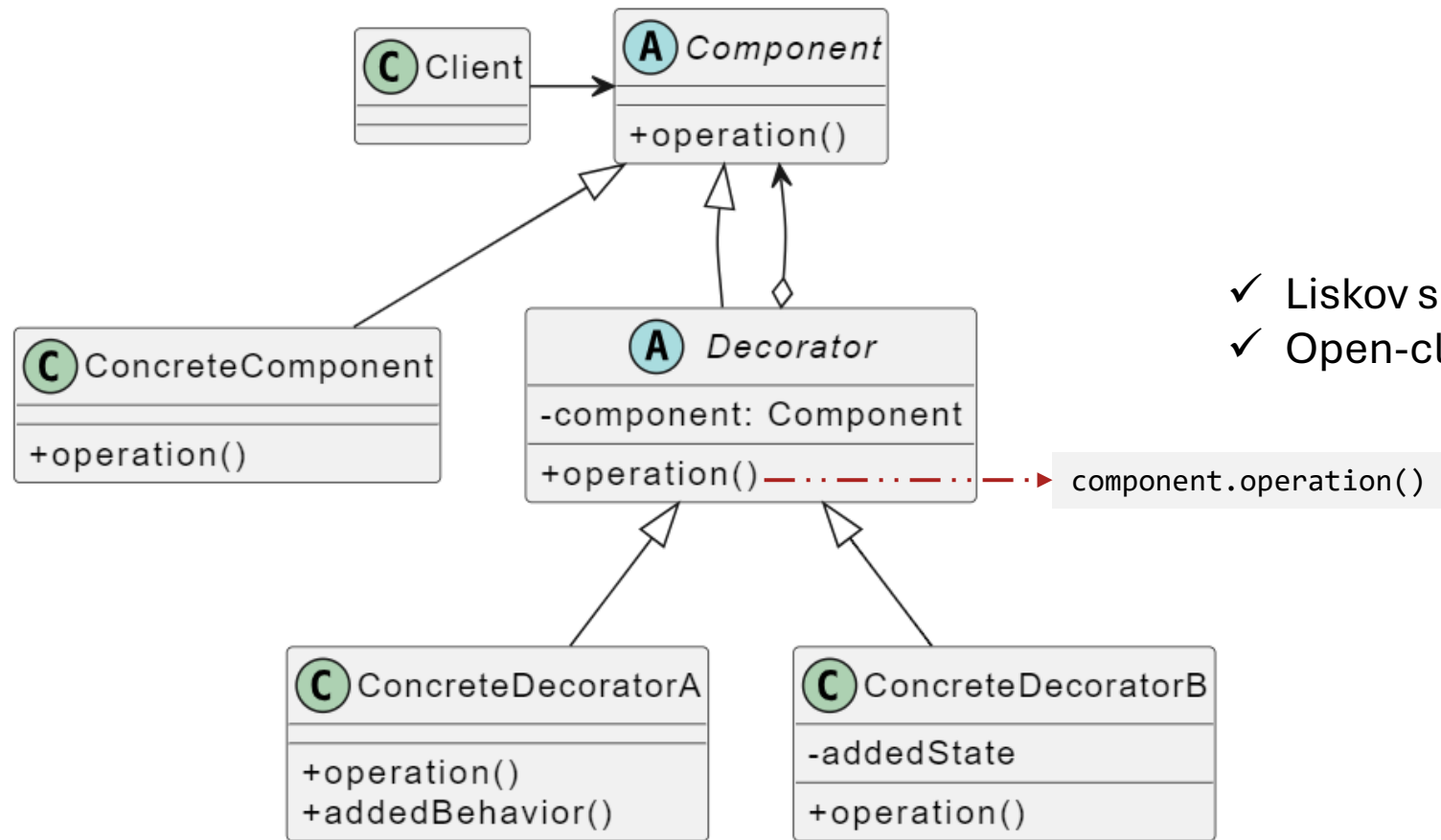
- **Motivation:** extend an object's functionality dynamically at runtime (vs. inheritance that extend functionality statically at compile time)
- **Intent:** provide a flexible alternative to inheritance; add additional responsibilities dynamically to an object



**Brewed Coffee**  
\$1.83 • 4 Cals

<b>Dairy</b>	12% Milk	^
Cream	- 0	+
2% Milk	- 1	+
Silk® Almond Beverage	- 0	+
Chobani® Oat Beverage	- 0	+
<b>Sweeteners</b>	1 Sugar	^
Sugar	- 1	+
Sweetener	- 0	+

# Decorator: Solution



- ✓ Liskov substitution principle
- ✓ Open-closed principle

# Agenda (recap)

- Structural design patterns
  - Adapter
  - Composite
  - Decorator