ONE PAGE SUMMARY

IT & Software › Design Patterns › Structural Patterns

Introduction to Structural Patterns

Mastering Structural Patterns for Better Software Design

DEFINITION

> Structural patterns help organize classes and objects to form larger structures. They simplify relationships and enhance code flexibility and reusability.

KEY CONCEPTS

Adapter Pattern

Allows incompatible interfaces to work together by wrapping an existing class with a new interface.

Bridge Pattern

Separates abstraction from implementation, allowing both to evolve independently.

Composite Pattern

Enables treating individual objects and compositions of objects uniformly, simplifying client code.

Decorator Pattern

Adds new functionality to an object dynamically without altering its structure.

Facade Pattern

Provides a simplified interface to a complex subsystem, making it easier to use.

EXAMPLES

→ Using an adapter to connect a legacy system with a new application.

→ Implementing a decorator to add logging functionality to an existing service.

→ Creating a facade for a complex library to simplify user interactions.

MEMORY TIPS

★ Remember 'A Bridge Connects' for the Bridge Pattern.

★ Think 'Decorate for Functionality' for the Decorator Pattern.

★ Use 'Composite = Parts and Whole' to recall the Composite Pattern.

COMMON MISTAKES

✗ Confusing Adapter with Decorator; they serve different purposes.

✗ Overusing patterns when simpler solutions exist.

✗ Neglecting to consider the impact on performance with complex structures.

QUICK RECAP

Structural patterns are essential for organizing code and enhancing flexibility. They help manage relationships between objects effectively, leading to cleaner and more maintainable software design.