Structural Design Patterns

Structural Design Patterns are design patterns that focus on how classes and objects are composed to form larger structures. These patterns help ensure that if one part of a system changes, the entire system does not need to change with it. They emphasize the way objects and classes are composed, promoting flexible and efficient ways of object composition.

Key Aspects of Structural Design Patterns:

- Composition: Structural patterns deal with the composition of classes or objects. Instead of creating new functionality through inheritance, structural patterns compose objects to create more complex structures.
- Interface and Implementation: Structural patterns often involve working with interfaces and their implementations, allowing different parts of a system to interact without being tightly coupled.
- Adaptability: Structural patterns often make systems more adaptable and flexible. They allow for objects to be interconnected in ways that allow systems to grow and evolve without significant changes to the existing codebase.
- Reusability: These patterns promote code reuse by enabling existing classes to work together even if they were not designed to do so from the beginning.

Examples of Structural Design Patterns:

1. Adapter: Allows incompatible interfaces to work together by creating an adapter that translates between them.
2. Bridge: Decouples an abstraction from its implementation so that the two can vary independently.
3. Composite: Composes objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.
4. Decorator: Adds responsibilities to objects dynamically without modifying the object itself.
5. Facade: Provides a simplified interface to a complex subsystem.
6. Flyweight: Reduces the cost of creating and manipulating a large number of similar objects.
7. Proxy: Provides a surrogate or placeholder for another object to control access to it.

Benefits of Structural Design Patterns:

- Flexibility: They allow systems to be more flexible by enabling different components to be connected or disconnected easily.

- Scalability: Structural patterns make it easier to add new functionality to a system without disrupting the existing structure.

- Maintainability: By promoting a clear separation of concerns, these patterns make systems easier to maintain and modify.

When to Use Structural Design Patterns:

- When you need to create complex structures from simpler components.

- When you want to extend a class functionality without using inheritance.

- When you need to simplify complex subsystems for client use.

- When you want to create a system where components can be replaced or added independently of each other.