🧠 Why Are They Called Structural Design Patterns?

✅ Definition:

> **Structural patterns** focus on **how classes and objects are composed** to form larger structures, enabling flexibility and reusability.

💡 Key Insight:

- They **emphasize composition** (i.e., using object references inside another object) rather than **inheritance** (i.e., extending behavior via subclassing).
- This allows **runtime flexibility**, **dynamic wiring**, and **loose coupling**.

🔁 Structural vs Creational Patterns – Interview Differences

Feature	Structural Patterns	Creational Patterns
🎯 **Primary Focus**	Structure and relationships between objects	Object creation mechanisms
🔧 **Technique Used**	**Composition** (has-a relationship)	Often use **Inheritance** or **new/clone/instance**
🔁 **Reuse Mechanism**	Reuse existing objects by combining or wrapping	Reuse creation logic by centralizing or hiding constructors
🔀 **Runtime Flexibility**	High – can rewire objects dynamically	Moderate – creation logic fixed at config or init
🧬 **Change Impact**	Low coupling → changes in components are isolated	Creation logic is decoupled, but structure may vary
📦 **Examples**	Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy	Singleton, Factory, Abstract Factory, Builder, Prototype
🔄 **Composition vs Inheritance**	Mostly **composition-based**	Often **inheritance-based**, especially in Factory Method

---

🧱 Inheritance vs Composition – Real Differences

Aspect	Inheritance (Creational)	Composition (Structural)
Type of relationship	is-a (e.g., `Dog extends Animal`)	has-a (e.g., `Car has Engine`)
Compile-time vs Runtime	Mostly compile-time	Mostly runtime wiring
Flexibility	Rigid – fixed hierarchy	Flexible – pluggable components

Aspect	Inheritance (Creational)	Composition (Structural)
Substitution	Can override behavior	Can delegate behavior dynamically
Encapsulation	Can break encapsulation (inherits internals)	Preserves encapsulation (wraps internally)

🧩 Quick Examples of Composition in Structural Patterns:

Pattern	Composition Example
Adapter	Wraps incompatible class (target ← adaptee)
Bridge	Has a reference to another interface (decouples abstraction from implementation)
Composite	A container holds a list of child components
Decorator	Has a reference to the component it enhances
Facade	Composes multiple subsystems into a single interface
Flyweight	Shares common parts using composition of shared state
Proxy	Wraps a real object and adds control logic

✅ Interview-Ready Answer:

"Structural patterns are called 'structural' because they focus on **how objects are composed** to form larger structures. Unlike creational patterns that often use inheritance to manage object creation, structural patterns rely on **composition** to reuse, extend, and organize code without tight coupling. This promotes flexibility and runtime adaptability."
