**Structural Design Patterns:**

Structural design patterns are concerned about how classes and objects can be composed to form larger structures.

Types of creational patterns:

* Adapter
* Bridge
* Compositer
* Decorator

**Adapter:**

Adapter is a structural design pattern that allows objects with incompatible interfaces to collaborate

Ex1: In Remote procedure calls, different clients can communicate with different servers, which are having different protocols, using adapters.

Ex2: In networking, different devices and systems may use various communication protocols. Network protocol adapters are used to convert data between different protocols.

**Bridge:**

It aims to decouple an abstraction from its implementation so that both can vary independently. It allows you to create two separate hierarchies: one for the abstraction and another for the implementation. It avoids the formation of complex hierarchies.

Ex1: A messaging app can have multiple communication protocols (like TCP, UDP) and different messaging formats (like JSON, XML). The Bridge pattern can be used to separate the communication protocols from the messaging formats.

Ex2: In a payment processing system, there can be various payment gateways (like paytm, phonepay) and different payment methods (like upi or bank transfer). The Bridge pattern can be used to separate the payment gateway logic from the payment method implementation

**Composite:**

The Composite design pattern is a structural design pattern that allows you to compose objects into tree-like structures which enables clients to treat individual objects and compositions of objects uniformly.

Ex1: In a music playlist, playlists can contain individual songs and other playlists, forming a hierarchical structure. The Composite pattern allows you to represent the music playlist as a composite, enabling easy organization and management of playlists and songs.

Ex2: In e-commerce websites, product categories can have subcategories, forming a tree-like structure. The Composite pattern allows you to represent the product categories and subcategories as composites, enabling easy navigation and management of product hierarchies.

**Decorator:**

Decorator is a structural design pattern that allows behavior to be added to individual objects, either statically or dynamically, without affecting the behavior of other objects from the same class.

Ex1: Electronic gadgets like smartphones or laptops often offer various accessories and add-ons that can be considered as decorators. Customers can customize their gadgets with features like protective cases, extra storage

Ex2: In word processors, text formatting options like bold, italic, underline, and font size can be considered as decorators. The text content is the base component, and decorators are used to modify or enhance the appearance of the text