**Structural Design Pattern**

Structural design patterns explain how to assemble objects and classes into larger structures, while keeping these structures flexible and efficient.

**Creational Design Pattern**

Creational design patterns provide various object creation mechanisms, which increase flexibility and reuse of existing code.

­­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Strategy Design Pattern**

* The Strategy Design Pattern is a behavioral design pattern. It allows you to dynamically change the behavior of an object by encapsulating it into different strategies.
* This pattern enables an object to choose from multiple algorithms and behaviors at runtime, rather than statically choosing a single one.
* It is based on the principle of composition over inheritance. It defines a family of algorithms, encapsulates each one, and makes them interchangeable at runtime. The core idea behind this pattern is to separate the algorithms from the main object. This allows the object to delegate the algorithm's behavior to one of its contained strategies.

**Observer Design Pattern**

* Observer Pattern is one of the behavioral design patterns. Observer design pattern is useful when you are interested in the state of an object and want to get notified whenever there is any change. In observer pattern, the object that watch on the state of another object are called Observer and the object that is being watched is called Subject.
* Observers Design Pattern’s intent is “Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically”.

**Decorator Design Pattern**

* Decorator patterns allow a user to add new functionality to an existing object without altering its structure. So, there is no change to the original class.
* The decorator design pattern is a structural pattern, which provides a wrapper to the existing class.
* The decorator design pattern uses abstract classes or interfaces with the composition to implement the wrapper.
* Decorator design patterns create decorator classes, which wrap the original class and supply additional functionality by keeping the class methods’ signature unchanged.
* Decorator design patterns are most frequently used for applying single responsibility principles since we divide the functionality into classes with unique areas of concern.

**Factory Design Pattern**

Factory Method is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.