Abstract Factory

Builder

Factory Method

Prototype

Singleton

Adapter

Bridge

Composite

Decorator

Facade

Separates the construction of a complex object from its representation so that the same construction process can create different representations.

Provides an interface for creating families of related or dependent objects without specifying their concrete classes.

Specifies the kinds of objects to create using an instance, and creates new objects by copying this instance.

Defines an interface for creating an object, but lets subclasses decide which class to instantiate. Lets a class defer instantiation to subclasses.

Converts the interface of a class into another interface clients expect. Lets classes work together that couldn't otherwise because of incompatible interfaces.

Ensures a class has only one instance, and provides a global point of access to it.

Organises objects into tree structures to represent whole-part hierarchies. This pattern lets clients treat individual objects and object compositions uniformly.

Decouples an abstraction from its implementation so that the two can vary independently.

Provides a unified interface to a set of interfaces in a subsystem. Defines a higher-level interface that makes the subsystem easier to use.

Attaches additional responsibilities to an object dynamically. Provides a flexible alternative to subclassing for extending functionality.

Flyweight

Proxy

Chain of Responsibility

Command

Interpreter

Iterator

Mediator

Memento

Observer

State

Provides a surrogate or placeholder for another object to control access to it.

Uses sharing to support large numbers of fine-grained objects efficiently.

Encapsulates a request as an object, thereby letting you parameterise clients with different requests, queue or log requests, and support undoable operations.

Avoids coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.

Provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation.

Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.

Without violating encapsulation, capture and externalise an object's internal state so that the object can be restored to this state later.

Defines an object that encapsulates how a set of objects interact. Promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.

Allows an object to alter its behaviour when its internal state changes. The object will appear to change its class.

Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

Strategy

Template Method

Visitor

Defines the skeleton of an algorithm in an operation, deferring some steps to subclasses. Lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.

Defines a family of algorithms, encapsulates each one, and makes them interchangeable. Lets the algorithm vary independently from the clients that use it.

Represents an operation to be performed on the elements of an object structure. Lets you define a new operation without changing the classes of the elements on which it operates.