

name	type	design principle	problem	synopsis
Adapter	structural		Incompatible interfaces and you cannot change the interface you want to use	Converts the interface of a class into another interface the clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.
Facade	structural	Principle of Least Knowledge: talk only to your immediate friends	lots of different interfaces, and too much low level complexity	Provides a unified interface to a set of interfaces in a subsystem. Facade defines a higher level interface that makes the system easier to use.
template method	behavioral	The Hollywood principle: Don't call us, we'll call you	duplicated code is difficult to change, maintain or extend	defines the skeleton of an algorithm in a method, deferring some steps to subclasses.
iterator	behavioral		Aggregate objects such as a list should allow a way to traverse its elements without exposing its internal structure	Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation
composite	behavioral	A class should have only one reason to change.	When dealing with tree-structured data, programmers often have to discriminate between a leaf-node and a branch. This makes code more complex, and therefore, error prone.	Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.
State	behavioral		Methods that have large, multipart conditional statements that depend on the object's state are difficult to maintain and extend	Allows an object to alter its behavior when its internal state changes. The object will appear to change its class.
Proxy	structural		Objects talk to each other using an interface that has been overburdened with the needs of networking, security, access coherence, or historic versions of the interface.	provides a surrogate or placeholder for another object to control access to it.