**[Creational design patterns](https://sourcemaking.com/design_patterns/creational_patterns)**

These design patterns are all about class instantiation. This pattern can be further divided into class-creation patterns and object-creational patterns. While class-creation patterns use inheritance effectively in the instantiation process, object-creation patterns use delegation effectively to get the job done.

* [Abstract Factory](https://sourcemaking.com/design_patterns/abstract_factory)  
  Creates an instance of several families of classes
* [Builder](https://sourcemaking.com/design_patterns/builder)  
  Separates object construction from its representation
* [Factory Method](https://sourcemaking.com/design_patterns/factory_method)  
  Creates an instance of several derived classes
* [Object Pool](https://sourcemaking.com/design_patterns/object_pool)  
  Avoid expensive acquisition and release of resources by recycling objects that are no longer in use
* [Prototype](https://sourcemaking.com/design_patterns/prototype)  
  A fully initialized instance to be copied or cloned
* [Singleton](https://sourcemaking.com/design_patterns/singleton)  
  A class of which only a single instance can exist

[**Structural design patterns**](https://sourcemaking.com/design_patterns/structural_patterns)

These design patterns are all about Class and Object composition. Structural class-creation patterns use inheritance to compose interfaces. Structural object-patterns define ways to compose objects to obtain new functionality.

* [Adapter](https://sourcemaking.com/design_patterns/adapter)  
  Match interfaces of different classes
* [Bridge](https://sourcemaking.com/design_patterns/bridge)  
  Separates an object’s interface from its implementation
* [Composite](https://sourcemaking.com/design_patterns/composite)  
  A tree structure of simple and composite objects
* [Decorator](https://sourcemaking.com/design_patterns/decorator)  
  Add responsibilities to objects dynamically
* [Facade](https://sourcemaking.com/design_patterns/facade)  
  A single class that represents an entire subsystem
* [Flyweight](https://sourcemaking.com/design_patterns/flyweight)  
  A fine-grained instance used for efficient sharing

[Private Class Data](https://sourcemaking.com/design_patterns/private_class_data)  
Restricts accessor/mutator access

* [Proxy](https://sourcemaking.com/design_patterns/proxy)  
  An object representing another object

[**Behavioral design patterns**](https://sourcemaking.com/design_patterns/behavioral_patterns)

These design patterns are all about Class's objects communication. Behavioral patterns are those patterns that are most specifically concerned with communication between objects.

* [Chain of responsibility](https://sourcemaking.com/design_patterns/chain_of_responsibility)  
  A way of passing a request between a chain of objects
* [Command](https://sourcemaking.com/design_patterns/command)  
  Encapsulate a command request as an object
* [Interpreter](https://sourcemaking.com/design_patterns/interpreter)  
  A way to include language elements in a program
* [Iterator](https://sourcemaking.com/design_patterns/iterator)  
  Sequentially access the elements of a collection
* [Mediator](https://sourcemaking.com/design_patterns/mediator)  
  Defines simplified communication between classes
* [Memento](https://sourcemaking.com/design_patterns/memento)  
  Capture and restore an object's internal state
* [Null Object](https://sourcemaking.com/design_patterns/null_object)  
  Designed to act as a default value of an object
* [Observer](https://sourcemaking.com/design_patterns/observer)  
  A way of notifying change to a number of classes

[State](https://sourcemaking.com/design_patterns/state)  
Alter an object's behavior when its state changes

* [Strategy](https://sourcemaking.com/design_patterns/strategy)  
  Encapsulates an algorithm inside a class
* [Template method](https://sourcemaking.com/design_patterns/template_method)  
  Defer the exact steps of an algorithm to a subclass
* [Visitor](https://sourcemaking.com/design_patterns/visitor)  
  Defines a new operation to a class without change