**Creational Patterns**

[**#**](https://dofactory.com/net/design-patterns#creational)

|  |  |
| --- | --- |
| [Abstract Factory](https://dofactory.com/net/abstract-factory-design-pattern) | Creates an instance of several families of classes |
| [Builder](https://dofactory.com/net/builder-design-pattern) | Separates object construction from its representation |
| [Factory Method](https://dofactory.com/net/factory-method-design-pattern) | Creates an instance of several derived classes |
| [Prototype](https://dofactory.com/net/prototype-design-pattern) | A fully initialized instance to be copied or cloned |
| [Singleton](https://dofactory.com/net/singleton-design-pattern) | A class of which only a single instance can exist |

**Structural Patterns**

[**#**](https://dofactory.com/net/design-patterns#structural)

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

**Behavioral Patterns**

[**#**](https://dofactory.com/net/design-patterns#behavioral)

|  |  |
| --- | --- |
| [Chain of Resp.](https://dofactory.com/net/chain-of-responsibility-design-pattern) | A way of passing a request between a chain of objects |
| [Command](https://dofactory.com/net/command-design-pattern) | Encapsulate a command request as an object |
| [Interpreter](https://dofactory.com/net/interpreter-design-pattern) | A way to include language elements in a program |
| [Iterator](https://dofactory.com/net/iterator-design-pattern) | Sequentially access the elements of a collection |
| [Mediator](https://dofactory.com/net/mediator-design-pattern) | Defines simplified communication between classes |
| [Memento](https://dofactory.com/net/memento-design-pattern) | Capture and restore an object's internal state |
| [Observer](https://dofactory.com/net/observer-design-pattern) | A way of notifying change to a number of classes |
| [State](https://dofactory.com/net/state-design-pattern) | Alter an object's behavior when its state changes |
| [Strategy](https://dofactory.com/net/strategy-design-pattern) | Encapsulates an algorithm inside a class |
| [Template Method](https://dofactory.com/net/template-method-design-pattern) | Defer the exact steps of an algorithm to a subclass |
| [Visitor](https://dofactory.com/net/visitor-design-pattern) | Defines a new operation to a class without change |