# Design Patterns

## Creational

- Abstract Factory
  - ★ Provide an interface for creating families of related or dependent objects without specifying their concrete classes.
- Factory Method
  - ★ Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.
- Builder
  - ★ Separate the construction of a complex object from its representation so that the same construction process can create different representations.
- Lazy Initialization
- Object Pool
  - ★ used to manage the object caching
- Prototype
  - ★ Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.
- Singleton
  - Highlander
    - ★ One instance only
  - Borg
    - ★ Multiple instances sharing state
  - Multiton
    - One instance per key
- Utility

## Structural

- Adapter
  - ★ Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.
- Bridge
  - ★ Decouple an abstraction from its implementation so that the two can vary independently.
- Composite
  - ★ Compose objects into tree structures to represent whole-part hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.
- Decorator
  - ★ Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.
- Facade
  - ★ Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.
  - Simple Interface
  - Hide complexity
- Flyweight
  - ★ Use sharing to support large numbers of fine-grained objects efficiently.
- Proxy
  - ★ Provide a surrogate or placeholder for another object to control access to it.

## Behavioral

- Chain Of Responsibility
  - ★ Avoid 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.
- Command
  - ★ Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.
- Interpreter
  - ★ Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.
- Iterator
  - ★ Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.
- Mediator
  - ★ Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.
- Memento
  - ★ Without violating encapsulation, capture and externalize an object's internal state so that the object can be returned to this state later.
- Observer
  - ★ Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.
- State
  - ★ Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.
- Strategy
  - ★ Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from the clients that use it.
- Specification
- Template Method
  - ★ Define the skeleton of an algorithm in an operation, deferring some steps to client subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.
  - Concrete Methods
  - Organizing Methods
- Visitor
  - ★ Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates.
  - Single-Serving Visitor
  - Hierarchical Visitor
- Null Object
  - ★ The intent of a Null Object is to encapsulate the absence of an object by providing a substitutable alternative that offers suitable default do nothing behavior.
- Sentinel

## Categories

- Creational
- Structural
- Behavioral

# Design Patterns

http://en.wikipedia.org/wiki/Design_pattern_(computer_science)

# Abstract Factory

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

# Adapter

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

# Borg

Shared Data across object instance

# Bridge

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

# Builder

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

# Chain Of Responsibility

Avoid 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.

# Command

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

# Composite

Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.

# Decorator

Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.

# Facade

Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.

# Factory Method

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

# Flyweight

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

# Hierarchical Visitor

Provide a way to visit every node in a hierarchical data structure such as a tree.

# Highlander

Really only one object instance

# Interpreter

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

# Iterator

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

# Lazy Initialization

Tactic of delaying the creation of an object, the calculation of a value, or some other expensive process until the first time it is needed.

## Mediator

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

## Memento

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

## Object Pool

Avoid expensive acquisition and release of resources by recycling objects that are no longer in use

## Observer

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

## Prototype

Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.

## Proxy

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

## Sentinel

The SentinelPattern occurs when a unique value, outside the set of "domain" values, is used to communicate information to and/or from a function or subsystem.

## Single-Serving Visitor

Optimize the implementation of a visitor that is allocated, used only once, and then deleted N

## Singleton

Ensure a class only has one instance, and provide a global point of access to it.

## Specification

Recombinable Business logic in a boolean fashion

## State

Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.

## Strategy

Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

## Template Method

Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.

## Utility

A class with a private constructor that contains only static methods.

## Visitor

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