Head First Design Patterns Poster

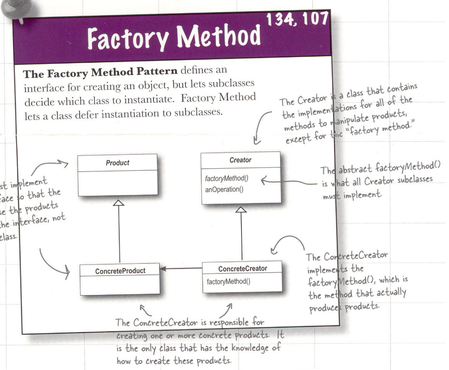
Head First Design Patterns and this companion poster are a great combination that will load patterns into your brain in a way that sticks.

Eric Freeman & Elisabeth Freeman

O'REILLY®
ISBN 0-596-10214-3 US $9.95 CAN $13.95
9 780596 102142 6 36920 10214 4

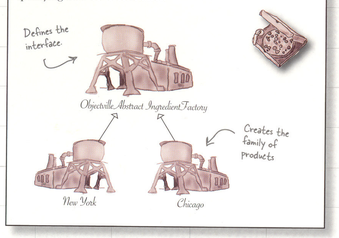
Factory Method 134, 107

The Factory Method Pattern defines an interface for creating an object, but lets subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.



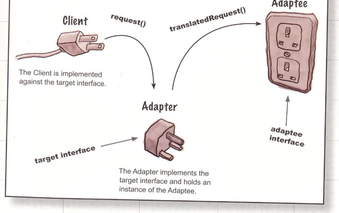
Abstract Factory 156, 87

The Abstract Factory Pattern provides an interface for creating families of related or dependent objects without specifying their concrete classes.



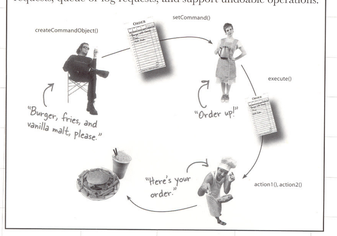
Adapter 243, 139

The Adapter Pattern 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.



Command 206, 233

The Command Pattern encapsulates a request as an object, thereby letting you parameterize other objects with different requests, queue or log requests, and support undoable operations.

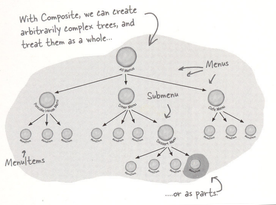


Page number in Head First Design Patterns.

Page number in the Gang of Four's Design Patterns.

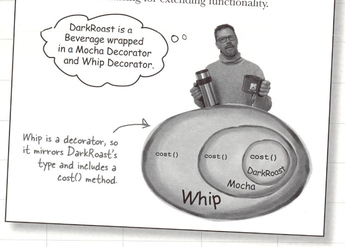
Composite 356, 163

The Composite Pattern allows you to compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.



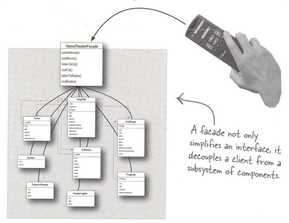
Decorator 91, 175

The Decorator Pattern attaches additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.



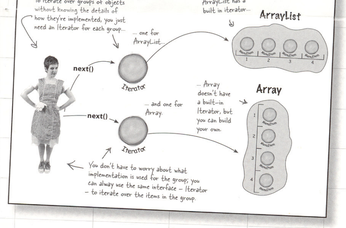
Façade 264, 185

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



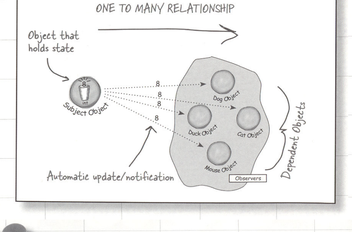
Iterator 336, 257

The Iterator Pattern provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation.



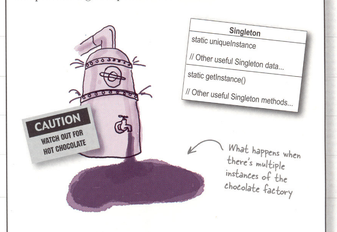
Observer 51, 293

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



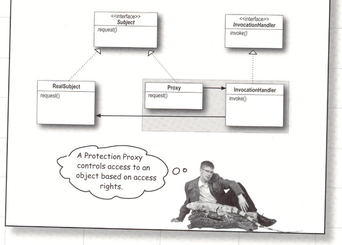
Singleton 177, 127

The Singleton Pattern ensures a class has only one instance and provides a global point of access to it.



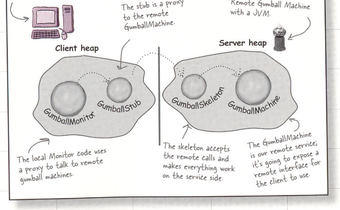
Proxy (Protection) 474, 207

The Protection Proxy provides a surrogate or placeholder for another object to control access to it.



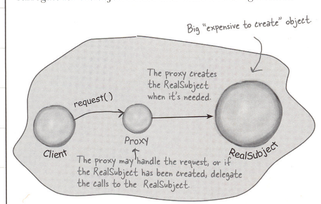
Proxy (Remote) 434, 207

The Remote Proxy acts as a local representative for a remote object.



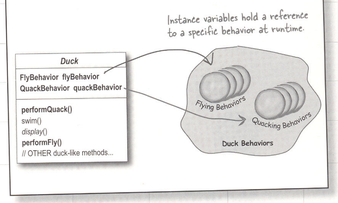
Proxy (Virtual) 462, 207

The Virtual Proxy acts as a representative for an object that may be expensive to create. The Virtual Proxy often defers the creation of the object until it is needed, and also acts as a surrogate for the object before and while it is being created.



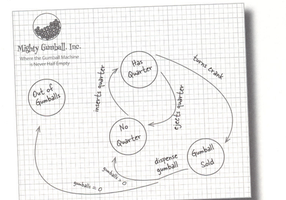
Strategy 24, 315

The Strategy Pattern defines a family of algorithms, encapsulates each one, and makes them interchangeable. Strategy lets the algorithm vary independently from clients that use it.



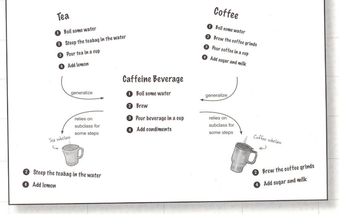
State 410, 305

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



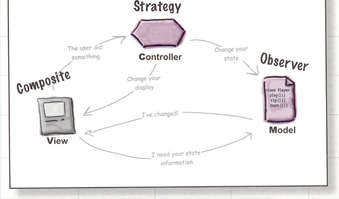
Template Method 289, 325

The Template Method Pattern defines the skeleton of an algorithm in a method, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.



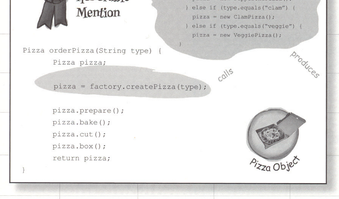
MVC 526, 4

A Compound Pattern, the model-view-controller pattern (MVC) separates an application into three distinct components: the model, view, and controller.



Simple Factory 117, n/a

Not a true Pattern, but like Abstract Factory and Factory Method, commonly used to encapsulate object creation.



WARNING: Overuse of design patterns can lead to code that is downright over-engineered. Always go with the simplest solution that does the job and introduce patterns where the need emerges.

How to read a Patterns Catalog

All patterns in a catalog start with a name. The name is a vital part of a pattern -- without a good name, a pattern can't become part of the vocabulary that you share with other developers.

The motivation gives you a concrete scenario that describes the problem and how the solution solves the problem.

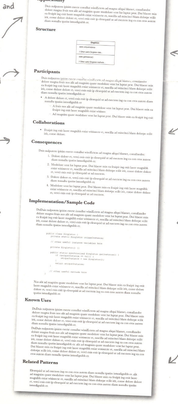
The applicability describes situations in which the pattern can be applied.

The participants are the classes and objects in the design. This section describes their responsibilities and roles in the pattern.

The consequences describe the effects that using this pattern may have: good and bad.

Implementation provides techniques you need to use when implementing this pattern, and issues you should watch out for.

Known uses describes examples of this pattern found in real systems.



This is the pattern's classification or category.

The intent describes what the pattern does in a short statement. You can also think of this as the pattern's definition.

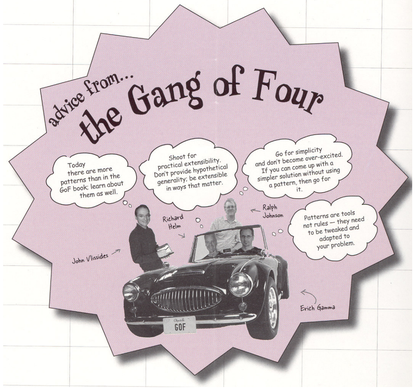
The structure provides a diagram illustrating the relationships among the classes that participate in the pattern.

Collaborations tells us how the participants work together in the pattern.

Sample code provides code fragments that might help with your implementation.

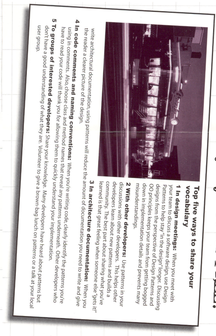
Related patterns describes the relationship between this pattern and others.

advice from... the Gang of Four



Annihilating evil with Anti-Patterns

Act now, and get the power of the shared vocabulary... yours for FREE!



What's Inside...

The poster in your hand is a companion to Head First Design Patterns book. It summarizes visually 18 of the most common design patterns including Adapter, Command, Composite Decorator, Observer, Proxy, State, Singleton, Strategy, and more. Each pattern includes a handy page reference to both Head First Design Patterns and the "Gang of Four" text, the canonical description of the pattern, and a visual guide designed (and inspired by the examples in Head First Design Patterns) to jog your memory of the objects, classes and their relationships.

Don't forget the Book...

Count on Head First Design Patterns to load patterns into your brain in a way that sticks. In a way.that lets you put them to work immediately. In a way that makes you better at solving software design problems, and better at speaking the language of patterns with others on your team.

Every copy of *Head First Design Patterns* contains the *Guide to Better Living with Patterns*. A handbook even Martha Stewart would be proud to own, this guide has all the tips, tricks, and guidelines you'll need to get the most out of patterns. You'll find a few things from the guide in this poster that you might find useful on a daily basis.