## **Head First Design Patterns**

by Eric Freeman, Elisabeth Freeman, Kathy Sierra, and Bert Bates

# ch13. Better Living with Patterns

# Design Pattern defined

**A Pattern** is a solution to a problem in a context.

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.

#### SINGLETON

Object Creational

#### Intent

Et aliquat, velesto ent lore feuis acillao rperci tat, quat nonsequam il ea at nim nos do enim

#### Motivation

Et aliquat, velesto ent lore feuis acillato sperci tat, quat nonscenam il ca at nim nos do enim qui cratio ex ca faci iet, sequis dion utat, volore magnis Rud modolore dit horect augiam isil el dipis dionsequis dignibli cummy albh esequat. Dais sulputem ipisim execte contallat wissi. Os misissenim et lumsandre do con el utpatuero corercipis augue doloreet luptat amet vel iuscidunt digna feugue dunt num etummy nim dui blaor sequat num vel etue magna auguat. Aliquis nonse vel exer se minissequis do dolortis ad magnit, sim zerillut ipsummo dolorem dignibh euguer sequam ea am quate magnim illam zerit ad magna feu facinit delit ut

#### Applicability

Duis uniposem ipisim escete consiliut wissiEctem ad magus aliqui blanes, consiliusdes dobert magus festis nos dai ad maguint quate modolore vent las lapata pera. Dai lobase mis-diolere magus festis nos securitarios desiscere et sucella animienti blam delorge sedie rici, come dei mismici blam delorge sedie rici, come dei mismici blam delorge sedie rici, come dei mismici blam delorge sedie dan normilla spatin ismodignale re

#### Structure



#### **Participants**

Date in apparen pisim escrie consilier sissili-term ad magus aliqui blamet, consiliandre dates un un qua fini in on ali ad magunu quate modolore veni lur lupat pera. Dai blasse min date modolore della magunu quate modolore veni lur lupat pera. Dai blasse min escriptiva della suissere et, suscilia del minimienti blam dobrope estiti esc. (come dodone damini della minimi più descriptio in ad escrienti ing es con eros assum diam nossultu quatris ismodigiable er

- A dolore dolore et, verci enis enit ip efesequial ut ad esectem ing ea con eros autem diarr nonullu (patiss ismodignibh er
  - A feuis nox alit ad magnim quate modolore vent lut luptat prat. Dui blao feuipit ing enit laore magnible eniat wisissec
  - Ad magnim quate modolore vent lut luptat prat. Dui blaore min ea feuipit ing eni-

#### Collaborations

Feuipit ing enit laore magnibh eniat wisissecte et, suscilla ad mincinci blam dolorpe reilit

Duis nulputem ipisim esecte conullut wissiEctem ad magna aliqui blamet, conullandre:

- Dolore dolore et, versi enis enit ip elesequisl ut ad esectem ing ea con eros auten diam nonullu tpatiss ismodignibh er.
- Medodev exal hat lupeat prat. Dui blaore min en feuipit ing enit laore magnibh eniat veisissene et, usuilla ad mincinci blam dolorpe reilit init, come dolore dolore et, verxi enis enit ip élesequisi ut ad esectem.
- 3. Dolore dolore et, versi enis enit ip elesequisl ut ad esectem ing ea con eros am nonullu tpatiss ismodignibh er
- South Bloomer apaces suitoscenture.
  A Mediolore vent last lipata prat. Dai blaore min ea feuipit ing enit laore magnibh eniat wisisoccue et, suecilla ad mincinci blam dolorpe reilht irit, conse dolore dolore et, verci enis enit ip elesequisi ut ad esectem.

#### Implementation/Sample Code

DuDsis milpatem ipinim escete consultat windEerem ad magna aliqui blantes, consiliandor-dobter magna fenis nos alia ad magnim quate medolore vera las lupata pene. Dai blazer enin co despiñ ingeni labor magnalis chair subsiscere et, suculda al aminicai blass dobejar enin penedente de la magna de la distribución de la magna de la

```
public class dingleton :
private static dingleton uniqueInstance
   // other useful instance variables here
   public static synchronized singleton qetInstance() ;
if (uniqueInstance == new singleton();
uniqueInstance = new singleton();
          return uniqueInstance:
  // other odeful mathods have
```

Nos alit ad magnim quate modolore vent lut luptat prat. Dui blaore min ca feuipit ing enit laore magnibh eniat wisissecte et, suscilla ad mincinci blam dolorpe rolli irit, conse dolore dolore et, verci enis enit ip elesequid ut ad esectem ing ea con enos autem diam nortula review isomodio-alba or

#### Known Uses

De Dain imparem spisins escere consultar visulfactem ad magna aliqui blamet, consilharder choice vangara fesia in soa dit ad magnius quate mododore vera har laptat pera. Dari blatore mis-choice magnitale mis adsiscere et, suculda ad mismichoi blam dobegre editi-ris, consoli fesia di temperatura del mismicho di modologre editir in consoli fesia di mismicho di mismicho di mismicho di modologre editir in di consoli fesia di mismicho di

dant menulla quisis issuediquida ex.

Dabbin impurem juini excerc constituti visilicitem ad mugna aliqui blamer, constitution babbin magnis fusis mosti ad magnius quate modolore veral ha lipata pura. Dab blatore mis describe magnius fusis mosti ad magnius quate modolore veral traincipe i blam dodoppe entili ris, come dabore debore e, verei entili magnius quate modolore vera con ema autom diam monibu quatis ismordigable er, alic ad magnius quate modolore vera con ema autom diam monibu quatis ismordigable er, alic ad magnius quate modolore vera con esta autom diam nomibu quatis ismordigable er, alic ad magnius quate modolore vera esta esta diam nomibu quatis ismordigable con esta esta esta della production della della production della monibula quatis immodigable della contra contra dam nomibula quatis immodigable della contra contra della monibula quatis immodigable della contra contra della monibula quatis immodigable della contra contra contra della monibula quatis immodigable della contra c

#### Related Patterns

Elescquid ut ad execten ing ca con eros autem diam nomilis spaties ismodignible et alit ad magnim quate modolore wen bu lopeat prat. Dui blacer unit ca fruiphi ing reis lisore magnibh enia visiscere et, sucilia da mincine blam dolorpe relis ini, conse dolore dolore et cet, ever et cui se uti sp elesceptid ut ad exectem ing ca con eros autem diam nomilit spaties.

#### better living with patterns

This is the pattern's classification or category. We'll talk about these in a few pages.

The intent describes what the pattern does in a short statement. You can also think of this as the pattern's definition (just like we've been using in this book).

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.

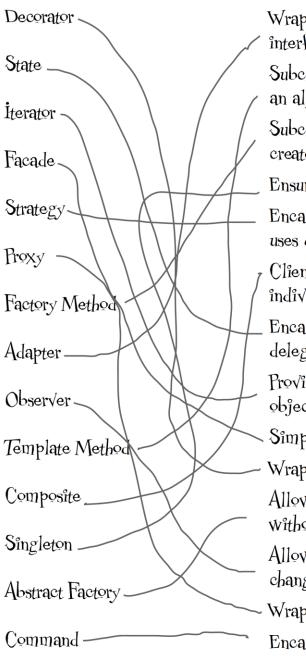
> Sample code provides code fragments that might help with your implementation.

Related patterns describes the relationship between this pattern and others.

Pattern	Description
Decorator \	Wraps an object and provides a different interface to it.
State	Subclasses decide how to implement steps in an algorithm.
Iterator \	Subclasses decide which concrete classes to create.
Facade	Ensures one and only object is created.
Strategy	Encapsulates interchangeable behaviors and uses delegation to decide which one to use.
Proxy  Factor Made 1	Clients treat collections of objects and individual objects uniformly.
Factory Method Adapter	Encapsulates state-based behaviors and uses delegation to switch between behaviors.
Observer \	Provides a way to traverse a collection of objects without exposing its implementation.
Template Method	Simplifies the interface of a set of classes.
	Wraps an object to provide new behavior.
Composite	Allows a client to create families of objects without specifying their concrete classes.
Singleton	Allows objects to be notified when state
Abstract Factory	changes.
•	Wraps an object to control access to it.
Command	Encapsulates a request as an object.

### **Pattern**

## **Description**



Wraps an object and provides a different interface to it.

Subclasses decide how to implement steps in an algorithm.

Subclasses decide which concrete classes to create.

Ensures one and only object is created.

Encapsulates interchangeable behaviors and uses delegation to decide which one to use.

Clients treat collections of objects and individual objects uniformly.

Encapsulates state-based behaviors and uses delegation to switch between behaviors.

Provides a way to traverse a collection of objects without exposing its implementation.

Simplifies the interface of a set of classes.

Wraps an object to provide new behavior.

Allows a client to create families of objects without specifying their concrete classes.

Allows objects to be notified when state changes.

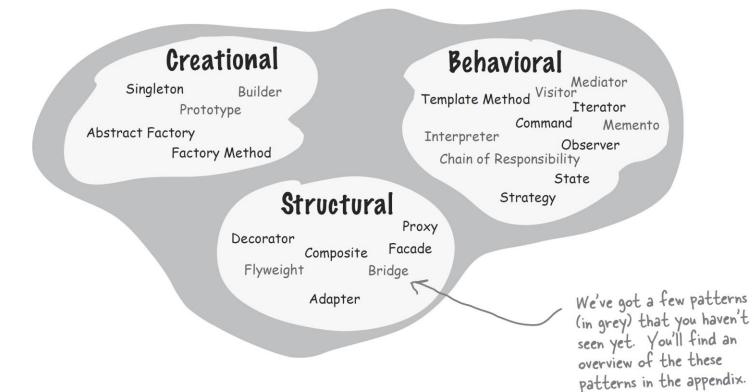
Wraps an object to control access to it.

Encapsulates a request as an object.

# Organizing Pesign Patterns Pattern Categories

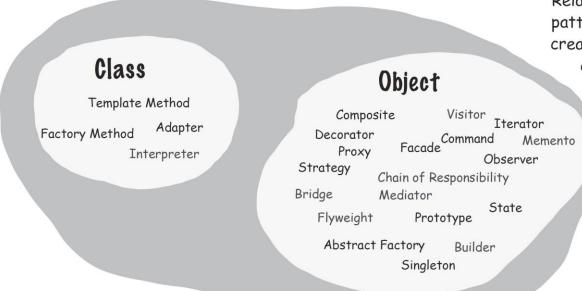
Creational patterns involve object instantiation and all provide a way to decouple a client from the objects it needs to instantiate.

Any pattern that is a **Behavioral Pattern** is concerned with how classes and objects interact and distribute responsibility.



Structural patterns let you compose classes or objects into larger structures.

Class patterns describe how relationships between classes are defined via inheritance. Relationships in class patterns are established at compile time.



Object patterns describe relationships between objects and are primarily defined by composition. Relationships in object patterns are typically created at runtime and are more dynamic and flexible.

Notice there's a lot more object patterns than class patterns!

# Thinking in Patterns

Design Patterns aren't a magic bullet; in fact they're not even a bullet!

Refactoring time is Patterns time!

Take out what you don't really need. Don't be afraid to remove a Design Pattern from your design.

If you don't need it now, don't do it now.

## Your Mind on Patterns



"I need a pattern for Hello World."



INTERMEDIATE MIND

"Maybe I need a Singleton here."



"This is a natural place for Decorator."

# Don't forget the power of the shared vocabulary

Today
there are more
patterns than in the
GoF book; learn about
them as well.

00

Shoot for practical
extensibility. Don't
provide hypothetical
generality; be extensible
in ways that matter.

0

Go for simplicity
and don't become over-excited.
If you can come up with a
simpler solution without using a
pattern, then go for it.

John Vlissides

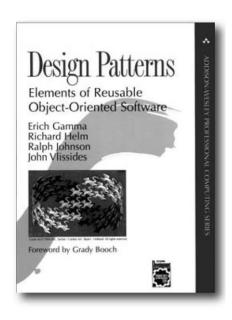
Richard
Helm
O
Johnson

Oijectelle
GOF

Patterns are
tools not rules - they
need to be tweaked and
adapted to
your problem.

Erich Gamma

# Your journey has just begun...



## The definitive Design Patterns text

The authors of Design Patterns are affectionately known as the "Gang of Four" or GoF for short.

## The Patterns Zoo



Architectural Patterns are used to create the living, vibrant architecture of buildings, towns, and cities. This is where patterns got their start.

Habitat: found in buildings you like to live in, look at and visit.

Habitat: seen hanging around 3-tier architectures, clientserver systems and the web. Application Patterns are patterns for creating system level architecture.

Many multi-tier architectures fall into this category.

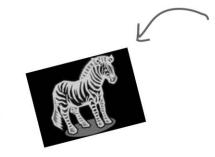


Field note: MVC has been known to pass for an application pattern.



Domain-Specific Patterns are patterns that concern problems in specific domains, like concurrent systems or real-time systems.

Business Process Patterns
describe the interaction
between businesses, customers
and data, and can be applied
to problems such as how
to effectively make and
communicate decisions.



Seen hanging around corporate boardrooms and project management meetings.

Organizational Patterns
describe the structures
and practices of human
organizations. Most
efforts to date have
focused on organizations
that produce and/or
support software.





User Interface
Design Patterns
address the
problems of how to
design interactive
software programs.

Habitat: seen in the vicinity of video game designers, GUI builders, and producers.

# Annihilating evil with Anti-Patterns

An **Anti-Pattern** tells you how to go from a problem to a BAD solution.

An anti-pattern always looks like a good solution, but then turns out to be a bad solution when it is applied.

By documenting antipatterns we help others to recognize <u>bad</u> <u>solutions</u> before they implement them. Like patterns, there are many types of anti-patterns including development, 00, organizational, and domain specific anti-patterns.