



Stack Overflow is a community of 4.7 million programmers, just like you, helping each other.

Join them; it only takes a minute:

Sign up

Join the Stack Overflow community to:



Ask programming questions



Answer and help your peers



Get recognized for your expertise

## Examples of GoF Design Patterns in Java's core libraries

Work on work you love. From home.



673

I am learning GoF Java Design Patterns and I want to see some real life examples of them. What are some good examples of these Design Patterns in Java's core libraries?

java oop design-patterns java-api

share

edited Apr 11 '15 at 3:40

community wiki

16 revs, 7 users 56%

unj2

2521

asked 6 years ago

viewed 345654 times

active 21 days ago

150 People Chatting

JavaScript: ECMAScript 2016 Released

56 mins ago - little pootis



PHP

57 mins ago - Wes



locked by [Shog9](#) ♦ Apr 11 '15 at 3:39

This question's answers are a collaborative effort: if you see something that can be improved, just edit the answer to improve it! *No additional answers can be added here*

comments disabled on deleted / locked posts / reviews

7 Answers

active

oldest

votes

2060

You can find an overview of a lot of design patterns in [Wikipedia](#). It also mentions which patterns are mentioned by GoF. I'll sum them up here and try to assign as many pattern implementations as possible, found in both the Java SE and Java EE APIs.

## Creational patterns

**Abstract factory** (recognizeable by creational methods returning the factory itself which in turn can be used to create another abstract/interface type)

- `javax.xml.parsers.DocumentBuilderFactory#newInstance()`
- `javax.xml.transform.TransformerFactory#newInstance()`
- `javax.xml.xpath.XPathFactory#newInstance()`

**Builder** (recognizeable by creational methods returning the instance itself)

- `java.lang.StringBuilder#append()` (unsynchronized)
- `java.lang.StringBuffer#append()` (synchronized)
- `java.nio.ByteBuffer#put()` (also on `CharBuffer`, `ShortBuffer`, `IntBuffer`, `LongBuffer`, `FloatBuffer` and `DoubleBuffer`)
- `javax.swing.GroupLayout.Group#addComponent()`
- All implementations of `java.lang.Appendable`

**Factory method** (recognizeable by creational methods returning an implementation of an abstract/interface type)

- `java.util.Calendar#getInstance()`



## Linked

15

[Examples of Design Patterns used in JDK](#)

4

[what is Gang of Four design pattern](#)

1

[Is the factory design pattern used in a Java API?](#)

0

[Java Design Patterns Examples](#)

1

[KeyAdapter inner class... What exactly is Java doing here?](#)

1

[Design patterns illustrated by Java SE/EE APIs?](#)

-3

[Where are the most common OOP concepts implemented in java swing jframes?](#)

0

[Pattern examples in Java Development Kit](#)

510

[What is an efficient way to implement a singleton pattern in Java?](#)

275

[How do the Proxy, Decorator, Adapter, and Bridge Patterns differ?](#)

[see more linked questions...](#)

## Related

83

- `java.util.ResourceBundle#getBundle()`
- `java.text.NumberFormat#getInstance()`
- `java.nio.charset.Charset#forName()`
- `java.net.URLStreamHandlerFactory#createURLStreamHandler(String)` (Returns singleton object per protocol)

**Prototype** (recognizeable by creational methods returning a *different* instance of itself with the same properties)

- `java.lang.Object#clone()` (the class has to implement `java.lang.Cloneable`)

**Singleton** (recognizeable by creational methods returning the *same* instance (usually of itself) everytime)

- `java.lang.Runtime#.getRuntime()`
- `java.awt.Desktop#getDesktop()`
- `java.lang.System#getSecurityManager()`

## Structural patterns

**Adapter** (recognizeable by creational methods taking an instance of *different* abstract/interface type and returning an implementation of own/another abstract/interface type which *decorates/overrides* the given instance)

- `java.util.Arrays#asList()`
- `java.io.InputStreamReader(InputStream)` (returns a `Reader`)
- `java.io.OutputStreamWriter(OutputStream)` (returns a `Writer`)
- `javax.xml.bind.annotation.adapters.XmlAdapter#marshal()` and `#unmarshal()`

**Bridge** (recognizeable by creational methods taking an instance of *different* abstract/interface type and returning an implementation of own abstract/interface type which *delegates/uses* the given instance)

- None comes to mind yet. A fictive example would be `new LinkedHashMap(LinkedHashSet<K>, List<V>)` which returns an unmodifiable linked map which doesn't clone the items, but *uses* them. The `java.util.Collections#newSetFromMap()` and `singletonXXX()` methods however comes close.

Learning/Implementing Design Patterns (For Newbies)

722

Does Functional Programming Replace GoF Design Patterns?

342

C++ Singleton design pattern

10

disadvantages of builder design pattern

2

C# GOF Pattern Examples

277

Design Patterns web based applications

2

Connection between GoF Design Patterns and SOLID

2

Examples of GoF design patterns in .net

41


Which GoF Design pattern will be changed or influenced by the introduction of lambdas in Java8?

-1

GoF design pattern detection

## Hot Network Questions

 Using Two LEFT JOIN or use AND with single LEFT JOIN?

 Where is the line between unit testing application logic and distrusting language constructs?

**Composite** (recognizeable by behavioral methods taking an instance of *same* abstract/interface type into a tree structure)

- `java.awt.Container#add(Component)` (practically all over Swing thus)
- `javax.faces.component.UIComponent#getChildren()` (practically all over JSF UI thus)

**Decorator** (recognizeable by creational methods taking an instance of *same* abstract/interface type which adds additional behaviour)

- All subclasses of `java.io.InputStream`, `OutputStream`, `Reader` and `Writer` have a constructor taking an instance of same type.
- `java.util.Collections`, the `checkedXXX()`, `synchronizedXXX()` and `unmodifiableXXX()` methods.
- `javax.servlet.http.HttpServletRequestWrapper` and `HttpServletResponseWrapper`

**Facade** (recognizeable by behavioral methods which internally uses instances of *different* independent abstract/interface types)

- `javax.faces.context.FacesContext`, it internally uses among others the abstract/interface types `LifeCycle`, `ViewHandler`, `NavigationHandler` and many more without that the enduser has to worry about it (which are however overrideable by injection).
- `javax.faces.context.ExternalContext`, which internally uses `ServletContext`, `HttpSession`, `HttpServletRequest`, `HttpServletResponse`, etc.


**Flyweight** (recognizeable by creational methods returning a cached instance, a bit the "multiton" idea)

- `java.lang.Integer#valueOf(int)` (also on `Boolean`, `Byte`, `Character`, `Short`, `Long` and `BigDecimal`)


**Proxy** (recognizeable by creational methods which returns an implementation of given abstract/interface type which in turn *delegates/uses* a *different* implementation of given abstract/interface type)


- `java.lang.reflect.Proxy`
- `java.rmi.*`
- `javax.ejb.EJB`
- `javax.inject.Inject`


 Is there an ideal PWM frequency for DC brush motors?

 Why we use the term "scalar" and not the common term "number" in Linear Algebra?

 Would removing spaces in a string protect against SQL injection?

 A very curious rational fraction that converges. What is the value?

 Paddington to Euston with kids: minimum walking by tube or by taxi?

 Lost battles - under which conditions does the losing party actively commemorate a battle?


 How can there be 1,000 stellar ancestors before our Sun?


 Why does Bash's source not need the execution bit?

 Count Up, Replace, Repeat!


 How to disable kill command on linux

 Math without pencil and paper

 Ok to ask random professor questions relating to hobby-project?

 Producing permutations without repetitions for a number or a string

 Normal to leave nail in tire after a flat?


 What could prevent a sentient species from going to space?

 Convert YYYYMM to MMMY

 Why did Ramsay strike him?

 Can any MAC be used as a KDF?

 Control of individual Framed edges?

 Why did Robb cross the Green Fork at The Twins?

 Can one enter Russia by train 1-1.5 hours before the visa validity period starts?

 Is this chromatic puzzle always solvable?

- `javax.persistence.PersistenceContext`
- 

## Behavioral patterns

**Chain of responsibility** (recognizable by behavioral methods which (indirectly) invokes the same method in *another* implementation of *same* abstract/interface type in a queue)

- `java.util.logging.Logger#log()`
- `javax.servlet.Filter#doFilter()`

**Command** (recognizable by behavioral methods in an abstract/interface type which invokes a method in an implementation of a *different* abstract/interface type which has been *encapsulated* by the command implementation during its creation)

- All implementations of `java.lang.Runnable`
- All implementations of `javax.swing.Action`

**Interpreter** (recognizable by behavioral methods returning a *structurally* different instance/type of the given instance/type; note that parsing/formatting is not part of the pattern, determining the pattern and how to apply it is)

- `java.util.Pattern`
- `java.text.Normalizer`
- All subclasses of `java.text.Format`
- All subclasses of `javax.el.ELResolver`

**Iterator** (recognizable by behavioral methods sequentially returning instances of a *different* type from a queue)

- All implementations of `java.util.Iterator` (thus among others also `java.util.Scanner`!).
- All implementations of `java.util.Enumeration`

**Mediator** (recognizable by behavioral methods taking an instance of different abstract/interface type (usually using the command pattern) which delegates/uses the given instance)

- `java.util.Timer` (all `scheduleXXX()` methods)

- `java.util.concurrent.Executor#execute()`
- `java.util.concurrent.ExecutorService` (the `invokeXXX()` and `submit()` methods)
- `java.util.concurrent.ScheduledExecutorService` (all `scheduleXXX()` methods)
- `java.lang.reflect.Method#invoke()`

**Memento** (recognizable by behavioral methods which internally changes the state of the *whole* instance)

- `java.util.Date` (the setter methods do that, `Date` is internally represented by a `long` value)
- All implementations of `java.io.Serializable`
- All implementations of `javax.faces.component.StateHolder`

**Observer (or Publish/Subscribe)** (recognizable by behavioral methods which invokes a method on an instance of *another* abstract/interface type, depending on own state)

- `java.util.Observer` / `java.util.Observable` (rarely used in real world though)
- All implementations of `java.util.EventListener` (practically all over Swing thus)
- `javax.servlet.http.HttpSessionBindingListener`
- `javax.servlet.http.HttpSessionAttributeListener`
- `javax.faces.event.PhaseListener`

**State** (recognizable by behavioral methods which changes its behaviour depending on the instance's state which can be controlled externally)

- `javax.faces.lifecycle.Lifecycle#execute()` (controlled by `FacesServlet`, the behaviour is dependent on current phase (state) of JSF lifecycle)

**Strategy** (recognizable by behavioral methods in an abstract/interface type which invokes a method in an implementation of a *different* abstract/interface type which has been *passed-in* as method argument into the strategy implementation)

- `java.util.Comparator#compare()`, executed by among others `Collections#sort()`.
- `javax.servlet.http.HttpServlet`, the `service()` and all `doXXX()` methods take `HttpServletRequest` and `HttpServletResponse` and the implementor has to process them (and not to get hold of them as instance variables!).

- `javax.servlet.Filter#doFilter()`

**Template method** (recognizeable by behavioral methods which already have a "default" behaviour defined by an abstract type)

- All non-abstract methods of `java.io.InputStream`, `java.io.OutputStream`, `java.io.Reader` and `java.io.Writer`.
- All non-abstract methods of `java.util.AbstractList`, `java.util.AbstractSet` and `java.util.AbstractMap`.
- `javax.servlet.http.HttpServlet`, all the `doXXX()` methods by default sends a HTTP 405 "Method Not Allowed" error to the response. You're free to implement none or any of them.

**Visitor** (recognizeable by two *different* abstract/interface types which has methods defined which takes each the other abstract/interface type; the one actually calls the method of the other and the other executes the desired strategy on it)

- `javax.lang.model.element.AnnotationValue` and `AnnotationValueVisitor`
- `javax.lang.model.element.Element` and `ElementVisitor`
- `javax.lang.model.type.TypeMirror` and `TypeVisitor`
- `java.nio.file.FileVisitor` and `SimpleFileVisitor`
- `javax.faces.component.visit.VisitContext` and `VisitCallback`

share

edited Jun 1 at 10:51

community wiki

33 revs, 9 users 80%

BalusC

4 impressive.. :) +1. `javax.lang.model.element` defines visitors ;) I'm not quite sure whether `doXXX` and `doFilter` are "strategies". – Bozho Apr 26 '10 at 13:14

4 This related blog post just appeared [briandupreez.net/2010/11/design-patterns-in-jdk.html](http://briandupreez.net/2010/11/design-patterns-in-jdk.html) – Bozho Nov 23 '10 at 19:09

8 The mentioned builders e.g. `StrinbgBuilder` are all not an example for the Builder-Pattern. It is a very common mistake however to consider them as builders (so you are not really to blame ^\_^) – Angel O'Sphere May 25 '11 at 13:41

9 @BalusC: `Object.toString()` can hardly be considered to be a factory method; the class relationship is right but the intention is wrong. It's hard to draw the line of course, but any method creating and returning another object can't be called a factory method. Maybe you can say that purpose of `toString` isn't to create a string

but the return info about the receiver, therefore it is not a factory method. – [Lii](#) Oct 20 '12 at 15:40

22 [@BalusC](#), I have a question to ask you. Did you read the **WHOLE** source code of Java and JSF? – [Tapas Bose](#) Jan 9 '13 at 21:39

[show 14 more comments](#)

Work on work you love. From home.



1. Observer pattern throughout whole swing ( `Observable` , `Observer` )
2. MVC also in swing
3. Adapter pattern: `InputStreamReader` and `OutputStreamWriter` NOTE: `ContainerAdapter` , `ComponentAdapter` , `FocusAdapter` , `KeyAdapter` , `MouseAdapter` are *not* adapters; they are actually Null Objects. Poor naming choice by Sun.
4. Decorator pattern ( `BufferedInputStream` can decorate other streams such as `FilterInputStream` )
5. AbstractFactory Pattern for the AWT Toolkit and the Swing pluggable look-and-feel classes
6. `java.lang.Runtime#getRuntime()` is Singleton
7. `ButtonGroup` for Mediator pattern
8. `Action` , `AbstractAction` may be used for different visual representations to execute same code -> Command pattern
9. Interned Strings or `CellRender` in `JTable` for Flyweight Pattern (Also think about various pools - Thread pools, connection pools, EJB object pools - Flyweight is really about management of shared resources)
10. The Java 1.0 event model is an example of Chain of Responsibility, as are Servlet Filters.
11. Iterator pattern in Collections Framework
12. Nested containers in AWT/Swing use the Composite pattern
13. Layout Managers in AWT/Swing are an example of Strategy

and many more I guess

[share](#)

[edited Oct 5 '12 at 3:00](#)

[community wiki](#)



33 java.lang.Math (6th one) is not singleton, you don't have an instance to start with, everything is static. That's not singleton – [lon Todirel](#) Apr 25 '10 at 5:11

1 Thanks for the tip on MouseAdapter. I found this explanation: [stackoverflow.com/questions/9244185/...](https://stackoverflow.com/questions/9244185/) – [Lincoln](#) May 20 '15 at 14:24

[add a comment](#)



1. **Flyweight** is used with some values of Byte, Short, Integer, Long and String.
2. **Facade** is used in many place but the most obvious is Scripting interfaces.
3. **Singleton** - java.lang.Runtime comes to mind.
4. **Abstract Factory** - Also Scripting and JDBC API.
5. **Command** - TextComponent's Undo/Redo.
6. **Interpreter** - RegEx (java.util.regex.) and SQL (java.sql.) API.
7. **Prototype** - Not 100% sure if this count, but I thinkg `clone()` method can be used for this purpose.

[share](#)

[edited Nov 4 '09 at 14:16](#)

[community wiki](#)

[2 revs](#)

[NawaMan](#)

9 You're correct about Prototype – [Scott Stanchfield](#) Nov 4 '09 at 18:49

1 Concerning **Flyweight** pattern: it could be different Layout Managers from `java.awt` and `java.swing` packages. Indeed, they share almost identical intrinsic attributes and extrinsic attributes are different UI components that they lay out in UI form. – [Vitaly](#) Oct 23 '15 at 16:48

[add a comment](#)



RMI is based on Proxy.

Should be possible to cite one for most of the 23 patterns in GoF:

1. Abstract Factory: java.sql interfaces all get their concrete implementations from JDBC JAR when

driver is registered.

2. Builder: `java.lang.StringBuilder`.
3. Factory Method: XML factories, among others.
4. Prototype: Maybe `clone()`, but I'm not sure I'm buying that.
5. Singleton: `java.lang.System`
6. Adapter: Adapter classes in `java.awt.event`, e.g., `WindowAdapter`.
7. Bridge: Collection classes in `java.util`. List implemented by `ArrayList`.
8. Composite: `java.awt.Component` + `java.awt.Container`
9. Decorator: All over the `java.io` package.
10. Facade: [ExternalContext](#) behaves as a facade for performing cookie, session scope and similar operations.
11. Flyweight: Integer, Character, etc.
12. Proxy: `java.rmi` package
13. Chain of Responsibility: Servlet filters
14. Command: Swing menu items
15. Interpreter: No directly in JDK, but JavaCC certainly uses this.
16. Iterator: `java.util.Iterator` interface; can't be clearer than that.
17. Mediator: JMS?
18. Memento:
19. Observer: `java.util.Observer/Observable` (badly done, though)
20. State:
21. Strategy:
22. Template:
23. Visitor:


I can't think of examples in Java for 10 out of the 23, but I'll see if I can do better tomorrow. That's what edit is for.


[share](#)

[edited Sep 29 '12 at 10:24](#)

community wiki  
[4 revs, 3 users](#) 85%  
[duffymo](#)

[add a comment](#)

 22 The Abstract Factory pattern is used in various places. E.g., DatagramSocketImplFactory, PreferencesFactory. There are many more---search the Javadoc for interfaces which have the word "Factory" in their name.


 Also there are quite a few instances of the Factory pattern, too.

[share](#)

answered [Nov 4 '09 at 13:59](#)

community wiki  
[uckelman](#)

[add a comment](#)

 16 Even though I'm sort of a broken clock with this one, Java XML API uses Factory a lot. I mean just look at this:

```
Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(sou
String title = XPathFactory.newInstance().newXPath().evaluate("//title", doc);
```

...and so on and so forth.

Additionally various Buffers (StringBuffer, ByteBuffer, StringBuilder) use Builder.


[share](#)

answered [Nov 4 '09 at 14:07](#)

community wiki  
[Esko](#)

[add a comment](#)

#### • [Factory method](#)

 14 java.util.Collection#Iterator is a good example of a Factory Method. Depending on the concrete subclass of Collection you use, it will create an Iterator implementation. Because both the Factory superclass (Collection) and the Iterator created are interfaces, it is sometimes confused with AbstractFactory. Most of the examples for AbstractFactory in the the accepted answer (BalusC) are examples of [Factory](#), a simplified version of Factory Method, which is not part of the original GoF patterns. In Factory the Factory class hierarchy is collapsed and the factory uses other means to choose the product to be returned.

- Abstract Factory

An abstract factory has multiple factory methods, each creating a different product. The products produced by one factory are intended to be used together (your printer and cartridges better be from the same (abstract) factory). As mentioned in answers above the families of AWT GUI components, differing from platform to platform, are an example of this (although its implementation differs from the structure described in Gof).

share

edited May 22 '11 at 8:44

community wiki

3 revs

Catweazle

add a comment

Not the answer you're looking for? Browse other questions tagged [java](#) [oop](#) [design-patterns](#) [java-api](#) or [ask your own question](#).

[about us](#) [tour](#) [help](#) [blog](#) [chat](#) [data](#) [legal](#) [privacy policy](#) [work here](#) [advertising info](#) [mobile](#) [contact us](#) [feedback](#)

TECHNOLOGY

Stack Overflow  
Server Fault  
Super User  
Web Applications  
Ask Ubuntu  
Webmasters  
Game Development  
TeX - LaTeX

Programmers  
Unix & Linux  
Ask Different (Apple)  
WordPress Development  
Geographic Information Systems  
Electrical Engineering  
Android Enthusiasts  
Information Security

Database Administrators  
Drupal Answers  
SharePoint  
User Experience  
Mathematica  
Salesforce  
ExpressionEngine® Answers  
**more (13)**

LIFE / ARTS

Photography  
Science Fiction & Fantasy  
Graphic Design  
Movies & TV  
Seasoned Advice (cooking)  
Home Improvement  
Personal Finance & Money  
Academia  
**more (9)**

CULTURE / RECREATION

English Language & Usage  
Skeptics  
Mi Yodeya (Judaism)  
Travel  
Christianity  
Arqade (gaming)  
Bicycles  
Role-playing Games  
**more (21)**

SCIENCE

Mathematics  
Cross Validated (stats)  
Theoretical Computer Science  
Physics  
MathOverflow  
Chemistry  
Biology  
**more (5)**

OTHER

Stack Apps  
Meta Stack Exchange  
Area 51  
Stack Overflow Careers

site design / logo © 2016 Stack Exchange Inc; user contributions licensed under [cc by-sa 3.0](#) with [attribution required](#)

rev 2016.6.23.3701