23个经典JDK设计模式

2010-11-29 09:59 | 1363次阅读 | 【已有40条评论】发表评论

来源：酷壳 | *收藏到我的网摘*

酷壳版主陈皓近日发表博文《JDK里的设计模式》，文中他列出了JDK中有关23个经典设计模式的示例。现把全文转载于此，全文如下：

下面是JDK中有关23个经典设计模式的示例：

Structural（结构模式）

Adapter:

把一个接口或是类变成另外一种。

- ● java.util.Arrays#asList()
- ● javax.swing.JTable(TableModel)
- ● java.io.InputStreamReader(InputStream)
- ● java.io.OutputStreamWriter(OutputStream)
- ● javax.xml.bind.annotation.adapters.XmlAdapter#marshal()
- ● javax.xml.bind.annotation.adapters.XmlAdapter#unmarshal()

Bridge:

把抽象和实现解藕，于是接口和实现可在完全独立开来。

- ● AWT (提供了抽象层映射于实际的操作系统)
- ● JDBC

Composite:

让使用者把单独的对象和组合对象混用。

- ● javax.swing.JComponent#add(Component)
- ● java.awt.Container#add(Component)
- ● java.util.Map#putAll(Map)
- ● java.util.List#addAll(Collection)
- ● java.util.Set#addAll(Collection)

Decorator:

为一个对象动态的加上一系列的动作，而不需要因为这些动作的不同而产生大量的继承类。这个模式在JDK中几乎无处不在，所以，下面的列表只是一些典型的。

- ● java.io.BufferedInputStream(InputStream)

- ● java.io.DataInputStream(InputStream)
- ● java.io.BufferedOutputStream(OutputStream)
- ● java.util.zip.ZipOutputStream(OutputStream)
- ● java.util.Collections#checked[List|Map|Set|SortedSet|SortedMap]()

Facade:

用一个简单的接口包状一组组件，接口，抽象或是子系统。

- ● java.lang.Class
- ● javax.faces.webapp.FacesServlet

Flyweight:

有效率地存储大量的小的对象。

- ● java.lang.Integer#valueOf(int)
- ● java.lang.Boolean#valueOf(boolean)
- ● java.lang.Byte#valueOf(byte)
- ● java.lang.Character#valueOf(char)

Proxy:

用一个简单的对象来代替一个复杂的对象。

- ● java.lang.reflect.Proxy
- ● RMI

Creational（创建模式）

Abstract factory:

创建一组有关联的对象实例。这个模式在JDK中也是相当的常见，还有很多的framework例如Spring。我们很容易找到这样的实例。

- ● java.util.Calendar#getInstance()
- ● java.util.Arrays#asList()
- ● java.util.ResourceBundle#getBundle()
- ● java.sql.DriverManager#getConnection()
- ● java.sql.Connection#createStatement()
- ● java.sql.Statement#executeQuery()
- ● java.text.NumberFormat#getInstance()
- ● javax.xml.transform.TransformerFactory#newInstance()

Builder:

主要用来简化一个复杂的对象的创建。这个模式也可以用来实现一个 Fluent

Interface。

- ● java.lang.StringBuilder#append()
- ● java.lang.StringBuffer#append()
- ● java.sql.PreparedStatement
- ● javax.swing.GroupLayout.Group#addComponent()

Factory:

简单来说，按照需求返回一个类型的实例。

- ● java.lang.Proxy#newProxyInstance()
- ● java.lang.Object#toString()
- ● java.lang.Class#newInstance()
- ● java.lang.reflect.Array#newInstance()
- ● java.lang.reflect.Constructor#newInstance()
- ● java.lang.Boolean#valueOf(String)
- ● java.lang.Class#forName()

Prototype:

使用自己的实例创建另一个实例。有时候，创建一个实例然后再把已有实例的值拷贝过去，是一个很复杂的动作。所以，使用这个模式可以避免这样的复杂性。

- ● java.lang.Object#clone()
- ● java.lang.Cloneable

Singleton:

只允许一个实例。在 Effective Java中建议使用Emun.

- ● java.lang.Runtime#getRuntime()
- ● java.awt.Toolkit#getDefaultToolkit()
- ● java.awt.GraphicsEnvironment#getLocalGraphicsEnvironment()
- ● java.awt.Desktop#getDesktop()

Behavioral(行为模式)

Chain of responsibility:

把一个对象在一个链接传递直到被处理。在这个链上的所有的对象有相同的接口（抽象类）但却有不同的实现。

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

Command:

把一个或一些命令封装到一个对象中。

- ● java.lang.Runnable
- ● javax.swing.Action

Interpreter:

一个语法解释器的模式。

- ● java.util.Pattern
- ● java.text.Normalizer
- ● java.text.Format

Iterator:

提供一种一致的方法来顺序遍历一个容器中的所有元素。

- ● java.util.Iterator
- ● java.util.Enumeration

Mediator:

用来减少对象单的直接通讯的依赖关系。使用一个中间类来管理消息的方向。

- ● java.util.Timer
- ● java.util.concurrent.Executor#execute()
- ● java.util.concurrent.ExecutorService#submit()
- ● java.lang.reflect.Method#invoke()

Memento:

给一个对象的状态做一个快照。Date类在内部使用了一个long型来做这个快照。

- ● java.util.Date
- ● java.io.Serializable

Null Object:

这个模式用来解决如果一个Collection中没有元素的情况。

- ● java.util.Collections#emptyList()
- ● java.util.Collections#emptyMap()
- ● java.util.Collections#emptySet()

Observer:

允许一个对象向所有的侦听的对象广播自己的消息或事件。

- ● java.util.EventListener
- ● javax.servlet.http.HttpSessionBindingListener
- ● javax.servlet.http.HttpSessionAttributeListener
- ● javax.faces.event.PhaseListener

State:

这个模式允许你可以在运行时很容易地根据自身内部的状态改变对象的行为。

- ● java.util.Iterator
- ● javax.faces.lifecycle.LifeCycle#execute()

Strategy:

定义一组算法，并把其封装到一个对象中。然后在运行时，可以灵活的使用其中的一个算法。

- ● java.util.Comparator#compare()
- ● javax.servlet.http.HttpServlet
- ● javax.servlet.Filter#doFilter()

Template method:

允许子类重载部分父类而不需要完全重写。

- ● java.util.Collections#sort()
- ● java.io.InputStream#skip()
- ● java.io.InputStream#read()
- ● java.util.AbstractList#indexOf()

Visitor:

作用于某个对象群中各个对象的操作. 它可以使你在不改变这些对象本身的情况下,定义作用于这些对象的新操作.

- ● javax.lang.model.element.Element 和 javax.lang.model.element.ElementVisitor
- ● javax.lang.model.type.TypeMirror 和 javax.lang.model.type.TypeVisitor

在stakeoverflow也有相应的讨论：

http://stackoverflow.com/questions/1673841/examples-of-gof-design-patterns