题目：JDK 中的设计模式应用实例

在 JDK(Java Development Kit)类库中，开发人员使用了大量设计模式，正因为如此，我们可以在不修改 JDK 源码的前提下开发出自己的应用软件，本文列出了部分 JDK 中的模式应用实例，有兴趣的童鞋可以深入研究，看看前 Sun 公司的开发人员是如何在实际框架开发中运用设计模式的，Sunny 认为，研究 JDK 类库中的模式实例也不失为学习如何使用设计模式的一个好方式.

1. 创建型模式：

(1) 抽象工厂模式(Abstract Factory)

- java.util.Calendar#getInstance()
- java.util.Arrays#asList()
- java.util.ResourceBundle#getBundle()
- java.net.URL#openConnection()
- java.sql.DriverManager#getConnection()
- java.sql.Connection#createStatement()
- java.sql.Statement#executeQuery()
- java.text.NumberFormat#getInstance()
- java.lang.management.ManagementFactory (所有 getXXX()方法)
- java.nio.charset.Charset#forName()
- javax.xml.parsers.DocumentBuilderFactory#newInstance()
- javax.xml.transform.TransformerFactory#newInstance()
- javax.xml.xpath.XPathFactory#newInstance()

(2) 建造者模式(Builder)

• java.lang.StringBuilder#append()
• java.lang.StringBuffer#append()
• java.nio.ByteBuffer#put() (CharBuffer, ShortBuffer, IntBuffer,LongBuffer, FloatBuffer 和 DoubleBuffer 与之类似)
• javax.swing.GroupLayout.Group#addComponent()
• java.sql.PreparedStatement
• java.lang.Appendable 的所有实现类

(3) 工厂方法模式(Factory Method)

• java.lang.Object#toString() (在其子类中可以覆盖该方法)
• java.lang.Class#newInstance()
• java.lang.Integer#valueOf(String) (Boolean, Byte, Character,Short, Long, Float 和 Double 与之类似)
• java.lang.Class#forName()
• java.lang.reflect.Array#newInstance()
• java.lang.reflect.Constructor#newInstance()

(4) 原型模式(Prototype)

• java.lang.Object#clone() (支持浅克隆的类必须实现 java.lang.Cloneable 接口)

(5) 单例模式 (Singleton)

• java.lang.Runtime#getRuntime()
• java.awt.Desktop#getDesktop()

1. 结构型模式：

(1) 适配器模式(Adapter)

•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()

(2) 桥接模式(Bridge)

• AWT (提供了抽象层映射于实际的操作系统)

•JDBC

(3) 组合模式(Composite)

•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)

(4) 装饰模式(Decorator)

•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]()

(5) 外观模式(Facade)

•java.lang.Class
•javax.faces.webapp.FacesServlet

(6) 享元模式(Flyweight)

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

(7) 代理模式(Proxy)

• java.lang.reflect.Proxy
•java.rmi.*

3. 行为型模式：

(1) 职责链模式(Chain of Responsibility)

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

(2) 命令模式(Command)

• java.lang.Runnable

• javax.swing.Action

(3) 解释器模式(Interpreter)

• java.util.Pattern
• java.text.Normalizer
• java.text.Format
• javax.el.ELResolver

(4) 迭代器模式(Iterator)

• java.util.Iterator
• java.util.Enumeration

(5) 中介者模式(Mediator)

• java.util.Timer (所有 scheduleXXX()方法)

• java.util.concurrent.Executor#execute()

• java.util.concurrent.ExecutorService (invokeXXX()和 submit()方法)

• java.util.concurrent.ScheduledExecutorService (所有 scheduleXXX()方法)

•java.lang.reflect.Method#invoke()

(6) 备忘录模式(Memento)

•java.util.Date
•java.io.Serializable
•javax.faces.component.StateHolder

(7) 观察者模式(Observer)

•java.util.Observer/java.util.Observable

•java.util.EventListener (所有子类)

•javax.servlet.http.HttpSessionBindingListener
•javax.servlet.http.HttpSessionAttributeListener
•javax.faces.event.PhaseListener

(8) 状态模式(State)

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

(9) 策略模式(Strategy)

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

(10) 模板方法模式(Template Method)

•java.io.InputStream, java.io.OutputStream, java.io.Reader 和 java.io.Writer 的所有非抽象方法

•java.util.AbstractList, java.util.AbstractSet 和 java.util.AbstractMap 的所有非抽象方法

•javax.servlet.http.HttpServlet#doXXX()

(11) 访问者模式(Visitor)

•javax.lang.model.element.AnnotationValue 和 AnnotationValueVisitor

•javax.lang.model.element.Element 和 ElementVisitor

•javax.lang.model.type.TypeMirror 和 TypeVisitor