链滴

《码出高效》系列笔记（二）：代码风格

作者：matthewhan

原文链接：https://ld246.com/article/1579164661433

来源网站：链滴

<blockquote>
良好的编码风格和完善统一的规约是最高效的方式。
</blockquote>
<h2 id="前言">前言</h2>
本篇汲取了本书中较为精华的知识要点和实践经验加上读者整理，作为本系列里的第二篇章：代风格篇。
本系列目录：

《码高效》系列笔记（一）：面向对象中的类
《码出效》系列笔记（一）：面向对象中的方法
《码出效》系列笔记（一）：面向对象中的其他知识点
《码出效》系列笔记（二）：代码风格
《出高效》系列笔记（三）：异常与日志
《码高效》系列笔记（四）：数据结构与集合的框架
《码高效》系列笔记（四）：数据结构与集合的数组和泛型
《码高效》系列笔记（四）：元素的比较

<h2 id="命名规约">命名规约</h2>
代码风格一般不会影响程序运行，通常与数据结构、逻辑表达无关。往往指代不可见字符的展示式、代码元素的命名方式和代码注释风格等，但是却会隐藏潜在风险。虽然编码习惯不存在明显优劣分，但是在团队开发效率上也许会是一个巨大的内耗，牺牲小我，成就大我，提升效能也许来的更关。
但是我们都是独一无二有灵魂有思想的个体，不是 clone 出来的机器人，难免在理解和习惯有所偏差如何去统一部分习惯呢？这本《码出高效》最早就是以规约而出名，本人也遵循了本书的中的大部分则用于日常的开发中，书中有很多点因为阿里大量业务经验的存在而比我们考虑规范周全得多，所以接采用本书的一些“规定”往往会便捷的多。
<h2 id="命名符合本语言特性">命名符合本语言特性</h2>
每种语言都有自己独特的命名风格，有些语言在定义时提倡以前缀来区分局部变量、全局变量、件类型。比如 li_count 表示 local int 局部整型变量，dw_report 表示 data window 用于暂时报表据的控件，有些语言规定以下划线为前缀来进行命名。在 Java 中，所有代码元素的命名均能以下划线或美元符号开始或结束。
<h2 id="命名体现代码元素特征">命名体现代码元素特征</h2>

类名采用大驼峰形式（UpperCamelCase），一般为名词，例如：Object、StringBuffer、FileInutStream 等。

方法名采用小驼峰形式（lowerCamelCase），一般为动词，与参数组成动宾结构，例如 Object 的 wait()、StringBuffer 的 append(String)、FileInputStream 的 read()等。
变量包括参数、成员变量、局部变量等，也采用小驼峰形式。
常量的命名方式比较特殊，除了局部常量外字母全部大写，单词之间用下划线连接。

在 Java 命名时，以下列方式体现元素特征：

包名统一使用小写，点分隔符之间有且仅有一个自然语义的英文单词。包名统一使用单数形式，是类名如果有复数含义，则可以使用复数形式。
抽象类命名是用 Abstract 或 Base 开头；异常类明明采用 Exception 结尾；测试类命名以它要试的类名开始，以 Test 结尾。
类型与中括号紧挨着相连来定义数组，例如： <code>String[] args</code> 。
枚举类名带上 Enum 后缀，枚举成员名称参考常量的命名方式。

<h2 id="命名最好望文知义">命名最好望文知义</h2>
从名称上就能理解某个词句的确切含义是坠吼滴，带到自解释的目的。

所以要避免不规范的缩写，比如 condition 缩写成 condi、consumer 缩写成 cons，类似随意缩写会严重降低代码的可理解性。
避免中文拼音、中英混合的方式，比如 DaZePromotion（打折促销类）、PfmxBuilder（评分型抽闲工厂类）。alibaba、baidu、taobao 这类国际通用的名称，视为英文。

<h2 id="常量">常量</h2>
作为在作用域内保持不变的值，一般用 final 关键字进行修饰，根据作用域划分成：全局常量、内常量、局部常量。

全局常量：指类的公开静态属性，使用 <code>public static final</code> 修饰。
类内常量：私有静态属性，使用 <code>private static final</code> 修饰，
局部常量分为方法常量和参数常量，前者是在方法或代码内定义的常量，后者是定义形式参数是增加 final 标识，表示此参数值不能被修改。

<pre><code class="language-java highlight-chroma"><span c
ass="highlight-cl">public c
ass EasyCoding {</
pan>
    <span class="h
ghlight-kd">public static <span class="highlig
t-kd">final String G
OBAL_CONSTANT =
shared in global";
    <span class="h
ghlight-kd">public static <span class="highlig
t-kd">final String C
ASS_CONSTANT = "
hared in class";

    <span class="h
ghlight-kd">public void <span class="highlight
nf">f(String
a) <span class="high
ight-o">{
        <span class=
highlight-kd">final String <span class="highligh
-n">methodConstant = <span class="highlight-

">"shared in method";
    <span class="h
ghlight-o">}
    <span class="h
ghlight-kd">public void <span class="highlight
nf">g(final
int b<span class="h
ghlight-o">) {
        <span class=
highlight-c1">// 编译出错，不允许对常量参数进行重新赋值
<span cla
s="highlight-c1">   b <span class="highlight
o">= 3;
    <span class="h
ghlight-o">}
<span class="high
ight-o">}
</code></pre>
常量在代码中具有穿透性，使用甚广，所以必须是一个恰当的命名并且保证长期使用。常量是为
干掉一些可能会在迭代中改变的魔法值，比如某业务中，12345 五种代表着课程的审核状态，在团队
模小时口口相传加上注释可以保证不出错，但是在业务扩展的过程中会变得更加复杂（课程的等级状
也有 12345 五种，很容易混淆），则需要一套枚举类和全局常量类提高可读性来管理这些状态。</p

书中认为，系统成长到某个阶段后，重构是一种必然选择。优秀的架构设计不是去阻止未来一切
构的可能性，毕竟技术栈、业务方向和规模都在不断变化，二是尽量让重构来得晚一些，幅度小一些

<h2 id="变量">变量</h2>
广义来说，在程序中变量是一切通过分配内存并赋值的量，分为不可变量（常量）和可变变量。
br>
变量命名需要满足小驼峰形式，体现业务含义即可。重点强调：在 POJO 类中，针对布尔类型的变量
命名不可以加 is 前缀。例如 ORM 映射关系中，数据库 <code>is_deleted</code> 字段，在类中
可以这样声明 <code>Boolean isDeleted;</code> ，因为 getter 方法也是 <code>isDeleted()</
ode> ，因为框架反向解析时会误以为对应的属性是 <code>deleted</code> ，导致获取不到属性
进而抛出异常。我们可以通过 中添加下映射就 vans 辣！
<h2 id="代码展示风格">代码展示风格</h2>
<h2 id="缩进-空格和空行">缩进、空格和空行</h2>
像 Python 这种没有大括号的语言，对缩进的使用非常严格，Java 虽然没有这么严格，但是井
有序的风格让 review 变得更加高效。

缩进：缩进表示层次对应关系，由于不同编辑器对 Tab 的解析不一致，而空格在编辑器中往往
兼容的，所以一般规定采用 4 个空格作为默认的缩进方式，当然现在 IDE 这么
能，早就不需要手打 4 个空格了，可以再 Tab 键和空格之间实现快速转换。其中 IDEA 设置 Tab 键为
4 个空格时，请勿勾选 Use tab character；而在 eclipse 中，必须勾选 Inset spaces for tabs。高
本的 IDEA 好像已经是默认 4 个空格代替 tab 缩进。

空格：空格用于分隔不同的的编程元素，空格使得各元素之间错落有致，方便定位。一般有如下
定：

任何二目、三目运算符的左右两边都必须加一个空格。
注释的双斜线与注释内容之间有且仅有一个空格。
方法参数在定义和传入时，多个参数逗号后面必须加空格。
没有必要增加若干空格使变量的赋值等号遇上一行对应位置的等号对齐。
如果是大括号内为空，那么简洁的写成{}即可，大括号中间无需换行和空格。

左右小括号与括号内部的相邻字符之间不要出现空格。
左大括号前需要加空格。

实例：
<pre><code class="language-java highlight-chroma"><span c
ass="highlight-cl">public c
ass EasyCoding {</
pan>
    <span class="h
ghlight-cm">/**
<span cla
s="highlight-cm">   * 不需要等号的位置一致
<span cla
s="highlight-cm">   */
    <span class="h
ghlight-kd">public static <span class="highlig
t-n">String one =</
pan> "1";
    <span class="h
ghlight-kd">public static <span class="highlig
t-n">String two =</
pan> "2";
    <span class="h
ghlight-kd">public static <span class="highlig
t-n">Long three =</
pan> 2L;
    <span class="h
ghlight-kd">public static <span class="highlig
t-n">String weChat
 "weChat";

    <span class="h
ghlight-kd">public static <span class="highlig
t-kt">void main(</
pan>String[] <span c
ass="highlight-n">args) <span class="highlight
o">{
        <span class=
highlight-c1">// 缩进4个空格，并且在try关键字与左大括号之间保留一个空格
<span cla
s="highlight-c1">   try <span class="highligh
-o">{
            <span clas
="highlight-c1">// 二目运算符的左右必须有一个空格
<span cla
s="highlight-c1">   int <span class="highl
ght-n">count = 0<
span>;

            <span clas
="highlight-c1">// 三目运算符的左右两边必须有一个空格，小括号相邻字符无需空格。
<span cla

```
s="highlight-c1"></span>            <span class="highlight-kt">boolean</span> <span class=
highlight-n">condition</span> <span class="highlight-o">=</span> <span class="highligh
-o">(</span><span class="highlight-n">count</span> <span class="highlight-o">==</sp
n> <span class="highlight-mi">1</span><span class="highlight-o">)</span> <span class=
highlight-o">?</span> <span class="highlight-kc">true</span> <span class="highlight-o">
</span> <span class="highlight-kc">false</span><span class="highlight-o">;</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-kt">int</span> <span class="highlight-n">num</span> <span class="highlight
o">=</span> <span class="highlight-o">(</span><span class="highlight-n">count</span>
<span class="highlight-o">==</span> <span class="highlight-mi">0</span><span class="
ighlight-o">)</span> <span class="highlight-o">?</span> <span class="highlight-mi">99<
span> <span class="highlight-o">:</span> <span class="highlight-o">-</span><span clas
="highlight-mi">1</span><span class="highlight-o">;</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-cm">/**
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">             * if关键字与小括号之间保留一个空格
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">             * 小括号与大括号之间保留一个空格
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">             * 建议使用IDE的自动补全
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">             */</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-k">if</span> <span class="highlight-o">(</span><span class="highlight-n">c
ndition</span><span class="highlight-o">)</span> <span class="highlight-o">{</span>
</span></span><span class="highlight-line"><span class="highlight-cl">                <span c
ass="highlight-n">System</span><span class="highlight-o">.</span><span class="highlig
t-na">out</span><span class="highlight-o">.</span><span class="highlight-na">println</
pan><span class="highlight-o">(</span><span class="highlight-s">"996"</span><span cla
s="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-cm">/**
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">             * if-else后无论逻辑复杂与否，都需要加上大括号，并且之间保留一个
格
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">             * else不用换行
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">             */</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-o">}</span> <span class="highlight-k">else</span> <span class="highlight-o"
{</span>
</span></span><span class="highlight-line"><span class="highlight-cl">                <span c
ass="highlight-n">System</span><span class="highlight-o">.</span><span class="highlig
t-na">out</span><span class="highlight-o">.</span><span class="highlight-na">println</
pan><span class="highlight-o">(</span><span class="highlight-s">"965"</span><span cla
s="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-o">}</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-c1">// 多个实参逗号后面必须有一个空格
```

```
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>            <span class="highlight-n">String</span> <span class="hi
hlight-n">fuckTheWorld</span> <span class="highlight-o">=</span> <span class="highlig
t-n">getStr</span><span class="highlight-o">(</span><span class="highlight-n">one</sp
n><span class="highlight-o">,</span> <span class="highlight-n">two</span><span class=
highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-n">System</span><span class="highlight-o">.</span><span class="highlight-
a">out</span><span class="highlight-o">.</span><span class="highlight-na">println</sp
n><span class="highlight-o">(</span><span class="highlight-n">fuckTheWorld</span><s
an class="highlight-o">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// catch体是不应该出现空内容的，但是这里为了讲解需要。{}中无需换行和空格。
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-o">}</span> <span class="highlight-
">catch</span> <span class="highlight-o">(</span><span class="highlight-n">Exception<
span> <span class="highlight-n">e</span><span class="highlight-o">)</span> <span clas
="highlight-o">{}</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-o">}</span>
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-cm">/**
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     * 多个形参，逗号后面保留一个空格
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     * @param one
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     * @param two
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     * @return
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-cm">     */</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-kd">private</span> <span class="highlight-kd">static</span> <span class="highli
ht-n">String</span> <span class="highlight-nf">getStr</span><span class="highlight-o">
</span><span class="highlight-n">String</span> <span class="highlight-n">one</span><
pan class="highlight-o">,</span> <span class="highlight-n">String</span> <span class="h
ghlight-n">two</span><span class="highlight-o">)</span> <span class="highlight-o">{</
pan>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-c1">// 任何二目运算符的左右必须有一个空格，包括赋值运算符，加号运算符。
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>        <span class="highlight-k">return</span> <span class="highl
ght-n">one</span> <span class="highlight-o">+</span> <span class="highlight-n">two</
pan><span class="highlight-o">;</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-o">}</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-o">}</span>
</span></span></code></pre>
<ul>
<li>空行：空行用来分隔功能相似、逻辑内聚、意思相近的代码片段，是的代码布局更加清晰。一般
```

如下地方可以添加空行：

 方法定义
 属性定义结束
 不同逻辑
 不同于一
 不同业务

<h2 id="换行与高度">换行与高度</h2>

换行：单行字符不超过 120 个，超过必须要换行，换行遵循以下原则：

 第二行相对第一行缩进 4 个空格，从第三行开始，不再继续缩进，参考示例
 运算符与下文一起换行
 方法调用的点符号与下文一起换行
 方法调用中的多个参数需要换行时，在逗号后面换行
 在括号前不要换行

<pre><code class="language-java highlight-chroma"><span c
ass="highlight-cl">StringBuffer <span class="highlight-n
>sb = new
span class="highlight-n">StringBuffer();
<span class="high
ight-n">sb.append
/span>("我"<span cl
ss="highlight-o">).append<span class="highli
ht-o">("要")...</span

    <span class="h
ghlight-o">.append
"正")...
    <span class="h
ghlight-o">.append
"能")...
    <span class="h
ghlight-o">.append
"量");
</code></pre>

方法行数限制：
方法是执行单位，也是阅读代码逻辑的最高粒度模块。代码逻辑要分为主次、个性与共性，抽取次要
辑作为独立方法，共性逻辑抽取陈共性方法（日期、参数校验、权限判断）。
约定单个方法的总行数不超过 80 行。

<h2 id="控制语句">控制语句</h2>
控制语句遵循如下约定：

在 if、else、for、while、do-while 等语句中必须使用大括号。即使只有一行代码，也要加上大
号。
在条件表达式中不允许有赋值操作，也不允许在判断表达式中出现复杂的逻辑组合。

<pre><code class="language-java highlight-chroma"><span c
ass="highlight-cl">// 反例：如下。
<span cla
s="highlight-c1">if (
file.<span cl
ss="highlight-na">open(<span class="highlight
n">fileName, "w"</s
an>) != <span class
"highlight-kc">null) <span class="highlight-o"
&& (...) ||<
span> (...)) {
    <span class="h
ghlight-o">...
<span class="high
ight-o">}
</code></pre>
<pre><code class="language-java highlight-chroma"><span c
ass="highlight-cl">// 正例：而是应该赋值给一个布尔变量。
<span cla
s="highlight-c1">final <span class="highlight-
t">boolean existed =
 (file<span c
ass="highlight-o">.open<span class="highligh
-o">(fileName,</sp
n> "w") <span class=
highlight-o">!= null
 && (...)</
pan> || (...);
<span class="high
ight-k">if (existed</
pan>) {
    <span class="h
ghlight-o">...
<span class="high
ight-o">}
</code></pre>

多层嵌套不要超过 3 层。还记得太吾绘卷的一个 if-else 走天下吗？如果确实比较复杂的判断逻
，可以采用卫语句、策略模式、状态模式来实现。其中卫语句即代码逻辑先考虑失败、异常、中断、
出等直接返回的情况，以方法多个出口的方式，解决代码中判断 分支嵌套的问题，这是逆向思维的体
。

<pre><code class="language-java highlight-chroma"><span c
ass="highlight-cl">public v
id func {
    <span class="h
ghlight-k">if (condit
on1) {
        <span class=
highlight-o">...
    <span class="h
ghlight-o">}
    <span class="h

ghlight-k">if (condit
on2) {
  <span class=
highlight-o">...
  <span class="h
ghlight-o">}
  <span class="h
ghlight-k">if (condit
on3) {
  <span class=
highlight-o">...
  <span class="h
ghlight-o">}
  <span class="h
ghlight-o">...
  <span class="h
ghlight-k">return;
<span class="high
ight-o">}
</code></pre>

避免采用取反逻辑运算符。判断 x 是否小于 1，应该采用 <code>if (x < 1)</code> 而不是 <
ode>if (!(x >= 1))</code> 。

<h2 id="代码注释">代码注释</h2>
<h3 id="Javadoc-规范">Javadoc 规范</h3>
类、类属性和类方法的注释必须遵循 Javadoc 规范，使用文档注释（/** */）的格式。规范编写
注释，可以生成规范的 JavaAPI 文档，为外部用户提供有效支持。IDE 也会自动提示所用到的类、方
的注释。

 枚举类十分特殊，他的代码极为稳定。 （我以前就有这个疑问，枚举类的一般的 description
性加上 name 已经可以描述该枚举，为什么还要再加上注释。但是枚举类和全局常量一样，穿透性极
，而且在定义前就要深思熟虑，因为影响较大，所以注释是必须。）
 注释的内容不仅限于解释属性值的含义，还可以包括注意事项、业务逻辑。 （修改代码，可以
上修改和创建时间。）
 枚举类的删除或者修改都存在很大的风险。 （一般需要标注为过时属性，不可直接删除。）</li

<h3 id="简单注释">简单注释</h3>
包括单行和多行注释，特别强调此类注释不可写在代码后方，必须写在代码上方</stro
g>。双划线的注释与注释内容保留一个空格。