# Java编码规范

1.0

# 版本历史

| 日期 | 版本 | 描述 | 作者 |
| --- | --- | --- | --- |
| 2016-10-8 | 1.0 | | |

# 1. 简介

## 1.1. 文档说明

本文档描述项目代码开发工作基本编码规范，主要针对Java语言的代码开发提供一套代码的标准、约定和指导，使代码易于理解、维护和增强。通过遵循和改进这些程序设计标准，使各项目产生的代码有更好的一致性，并提高软件开发团队的生产和维护效率

## 1.2. 定义，首字母缩写词和缩略语

## 1.3. 参考资料

## 1.4. 适用范围

本文档适用范围包括 Java EE 项目开发组全体开发人员：技术管理人员、系统设计人员、系统开发人员、系统测试人员、系统维护人员、推广培训人员及其他相关人员。

# 2. Java 编码规范

## 2.1. 命名规范

### 2.1.1. 包 Package

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 所有包顶级层级统一公司域名 | 必须 | net.cn.rainbow |
| Package 命名全部小写，不允许出现除小写字母以外的字符 | 必须 | |
| package 的名称必须为单数 | 必须 | 使用 net.cn.rainbow.service<br>不使用 net.cn.rainbow.services |

### 2.1.2. 类 Class

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 类名第一个字母大写和单词首字母大写或小写 | 必须 | ThisIsTest.java |

### 2.1.3. 接口 Interface

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 接口类的默认常规实现类命名加 Impl 后缀 | 必须 | BaseDaoImpl |
| 接口类的其他类型实现命名加实现方式的后缀 | 必须 | BaseDaoHibernate |

### 2.1.4. 方法 Method

| 命名规范 | 是否必须 | 范例 |
|---|---|---|

| | | |
|---|---|---|
| 第一个英文单词首字母小写 | 必须 | createCard()　getString() |
| 类的域变量 get 方法规范 | 必须 | getFirstName()　getLastName() |
| 类的域变量 set 方法规范 | 必须 | setFirstName()　setLastName() |
| 类的域变量是布尔型获取规范 | 必须 | isString()　isActive(); |
| 类的普通方法采用完整的英文描述说明成员方法功能，并尽可能是动+名形式。 | 必须 | openFile()<br>addRole() |

### 2.1.5. 变量 Variable

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 变量的名字以小写字母开头，后面的单词用大写字母开头 | 必须 | userName　thisIsAClassVariable |
| 循环记数变量通常采用 i，j，k 或者 counter | 建议 | i，j，k　counter |

### 2.1.6. 参数 Parameter

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 参数的名字必须和变量的命名规范一致 | 必须 | userName　thisIsAClassVariable |
| 使用有意义的命名，尽量使用和要赋值的字段一样 | 必须 | setCounter(int size) |

### 2.1.7. 常量 Constant

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 常量的名字必须都大写，下划线分隔 | 必须 | final static int DEFAULT_PORT=8080 |

### 2.1.8. 数组 Array

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 数组的名字必须参照示例命名 | 必须 | 使用：byte[] buffer<br>不使用：byte buffer[] |

## 2.2. 格式规范

### 2.2.1. 版权信息 Copyright Information

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 每个源码文件版权信息命名必须参照示例命名，注意两个时间，第一时间表示当前代码被创建编写时间，第二个时间表示该代码版本被正式发布时间 | 必须 | /**<br>* TH APPLIANCE CHAINS.<br>* Copyright (c) 2010-2010 All Rights Reserved.<br>*/ |

### 2.2.2. 注释 Comment

| 命名规范 | 是否必须 | 范例 |
|---|---|---|

| | | |
|---|---|---|
| 类/接口注释：<br>描述 Java 类、接口、每个文档注释设在注释分割符/**…*/中，这种注释应该放在声明之前，“作者”如果使用为天虹员工，采用中文名称加天虹内部统一邮箱作为标识；若为外部人员，则使用中文名称加公司邮箱作为标识。版本中必须包含编写/最新修订日期 | 必须 | /**<br>*<br>* 功能描述： XXXXXX<br>* @author 张 zhangsan@rainbowcn.com<br>* @created 2010-8-7 下午 01:42:58<br>* @version 1.0.0<br>* @date 2010-8-7 下午 01:42:58<br>*/<br>public class Example { . . . |
| 方法注释:<br>描述方法。每个文档注释设在注释分割符/**…*/中，必须明确的描述方法的出入参数、异常、以及方法的含义。对于实现接口的方法，其注释的内容信息不得少于原接口中的注释内容 | 必须 | /**<br>* 功能描述：<br>* 输入参数：<按照参数定义顺序><br>* @param 参数说明<br>* 返回值： 类型 <说明><br>* @return 返回值<br>* @throw 异常描述<br>* @see 需要参见的其它内容<br>*/ |
| 成员注释:<br>所有类成员必须进行注释；其中公共成员必须进行 javaDoc 注释；对于方法内部的私有成员，缺省的需要注释,对于临时的，直观含义明确的可以不加注释； | 必须 | /**<br>* XXXXXX<br>*/ |
| 块注释：<br>块注释用于给文件、方法、数据结构和算法提供描述。必须利用块注释描述程序的要点。写在函数或方法中的块注释必须与它所描述的代码的缩进一致。一个块注释之前应该由一个空行将它与代码隔离开 | 必须 | /**<br>* XXXXXX<br>*/ |
| 单行注释：<br>单行的短注释与它所跟着的代码的缩进一致。如果一个注释不能在一行中写完，就应该使用块注释的格式。单行注释之前应该有一个空行。 | 必须 | If (type) {<br><br>// 当类型是… |

注意：

- ✧ 注释要简单明了。
- ✧ 编写代码边注释，修改代码同时修改相应的注释，以保证注释与代码的一致性。
- ✧ 在必要的地方注释，注释量要适中。注释的内容要清楚、明了，含义准确，防止注释二义性。保持注释与其描述的代码相邻，即注释的就近原则。
- ✧ 对代码的注释应放在其上方相邻位置，不可放在下面。对数据结构的注释应放在其上方相邻位置，不可放在下面；对结构中的每个域的注释应放在此域的右方；同一结构中不同域的注释要对齐。
- ✧ 变量、常量的注释应放在其上方相邻位置或右方。
- ✧ 全局变量要有较详细的注释，包括对其功能、取值范围、哪些函数或过程存取它以及存取时注意事项等的说明。
- ✧ 在每个函数或过程的前面要有必要的注释信息，包括：函数或过程名称；功能描述；输入、输出及返回值说明；调用关系及被调用关系说明等。

### 2.2.3. 缩进 Indentation

| 命名规范 | 是否必须 | 范例 |
| --- | --- | --- |
| 子功能块应在其父功能块后缩进 | 必须 | |

### 2.2.4. 空格 Space

| 命名规范 | 是否必须 | 范例 |
| --- | --- | --- |
| 操作符前后使用空格 | 建议 | if (x == 3) |
| 关键字与紧跟着的括号间使用空格分开 | 建议 | while (true){<br>...<br>} |
| for 语句中的表达式建议使用空格分开 | 建议 | for (expr1; expr2; expr3) |

注意：

空格不应该置于方法名与其左括号之间，这将有助于区分关键字和方法调用。

### 2.2.5. 空行 Blank Line

| 命名规范 | 是否必须 | 范例 |
| --- | --- | --- |
| 两个方法之间使用一个空行 | 建议 | |
| 方法内的局部变量和方法的第一条语句之间使用一个空行 | 建议 | |
| 块注释或单行注释之前使用一个空行 | 建议 | |
| 一个方法内的两个逻辑段之间使用一个空行，以提高可读性 | 建议 | |

注意：

空行将逻辑相关的代码段分隔开，以提高可读性。

### 2.2.6. 行长度 Line Length

| 命名规范 | 是否必须 | 范例 |
| --- | --- | --- |
| 单个方法的代码行(不包含注释行)建议少于 100 行 | 建议 | |
| 单个类的代码行(包含注释行) 建议少于 1500 行 | 建议 | |
| 一行的长度尽量少于 100 个字符，因为很多的终端和工具不能很好的处理，可能导致无法正确显示 | 建议 | |

### 2.2.7. 换行 New Line

| 命名规范 | 是否必须 | 范例 |
| --- | --- | --- |
| else if 和 else 不要另起一行 catch 需要另起一行 | 建议 | if () {<br>…<br>}else if () {<br>…<br>}<br>try{<br>…<br>} |

| | | |
|---|---|---|
| | | catch(Exception e){<br>… |

### 2.2.8. 圆括号 Parentheses

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 不应该在语句中使用无意义的括号 | 必须 | 使用：if(I=42)<br>不使用：if((I)=42) |
| 左括号和后一个字符之间不建议出现空格 | 建议 | 建议：callProc(aParameter)<br>不建议：callProc( aParameter) |
| 右括号和前一个字符之间不建议出现空格 | 建议 | 建议：callProc(aParameter)<br>不建议：callProc(aParameter ) |

### 2.2.9. 大括号 Big Brackets

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| {}中的语句应该单独作为一行 | 必须 | 使用：<br>if(i>0) {<br>i++;<br>}<br>不使用：<br>if(i>0) {i++;<br>} |
| {}在使用时，左括号在代码块起始行行尾，右括号与代码块起始行上下对齐。大括号里的代码缩进一个缩进位。 | 必须 | if(submit==null){<br>clear();<br>}<br>else{<br>update();<br>} |

## 2.3. 语句规范

### 2.3.1. 简单语句 Simple Statement

| 命名规范 | 是否必须 | 范例 |
|---|---|---|

| | | |
|---|---|---|
| 每一行只能包含一个语句 | 必须 | 使用：<br>argv++; // 正确<br>argc--; // 正确<br>不使用：<br>argv++; argc--; |

### 2.3.2. 条件语句 Condition Statement

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 不要在条件语句内用’=’，这会与”==”混乱(同时适用于 while 语句内条件说明) | 必须 | |
| 在混合运算表达式中使用括号避免运算优先级问题，即使对运算的优先顺序非常清晰,也应该这么做。(同时适用于 while 语句内条件说明) | 建议 | 建议:<br>if ((a == b) && (c == d))<br>不建议:<br>If (a == b && c == d) |

### 2.3.3. 分支语句 Switch Statement

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 每一个 Switch 的 case 建议有 default 做为最后出口 | 必须 | switch (x) {<br>case 1 : {<br>expr;break;<br>} // end case<br>case 2 : {<br>expr;break;<br>} // end case<br>default : {<br>expr;break;<br>} // end default<br>} // end switch |

### 2.3.4. 循环语句 Loop Statement

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 如果 for 循环中表达式很长，建议将多行进行分割，保证代码的可读性 | 建议 | |
| for 语句的初始化或更新语句中使用逗号时，避免复杂性，最多使用三个变量。必要的话，初始化可以在 for 循环之前使用单独的语句，或更新语句可以在循环的底部执行 | 建议 | |

### 2.3.5. 包和引入语句 Packet and Import Statement

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| import 语句不要使用通配符“ *” | 必须 | 使用：import java.io.File 不使用：import java.io.* |
| 程序中不允许出现未使用的导入包 | 必须 | |
| import 中标准的包名要在本地的包名之前 | 建议 | import java.io.File;<br>import java.util.Observable;<br>import com.rainbow.framework |

### 2.3.6. 声明语句 Declare Statement

| 命名规范 | 是否必须 | 范例 |
| --- | --- | --- |
| 不要将变量的声明放在同一行。 | 必须 | 使用：<br>int foo;<br>int [] fooarray;<br>不使用：<br>int foo, fooarray[]; |
| 尽量在声明局部变量的同时初始化。唯一不这么做的理由是变量的初始值依赖于某些先前发生的计算。 | 必须 | |
| 避免声明的局部变量与上一级变量重名。 | 必须 | int count;<br>...<br>myMethod() {<br>if (condition) {<br>int count = 0;<br>// AVOID!<br>...<br>}<br>...<br>} |

### 2.3.7. 例外控制语句 Exception Control Statement

| 命名规范 | 是否必须 | 范例 |
| --- | --- | --- |
| 不要有空的 catch 语句出现，至少打印出在哪里抛出例外和相关信息 | 必须 | |
| 在捕获一场时尽可能使用具体异常,最好不要使用 Exception 这类通用异常类型，除非有明确的要求需要捕获所有异常。 | 建议 | |

### 2.3.8. 类和接口 Class and Interface

| 命名规范 | 是否必须 | 范例 |
| --- | --- | --- |
| 类、接口定义之前应先进行注释。注释包括类、接口的目的、作用、功能，实现的算法、使用方法、示例程序等，还可以包括期望改进工作的地方和不希望改变的地方 | 必须 | |
| 左括号”{”置于声明行尾，右括号”}”另起一行，缩进匹配相应开始语句 | 必须 | class Sample {<br>int ivar1;<br>int ivar2;<br><br>Sample(int i, int j) {<br>ivar1 = i;<br>ivar2 = j;<br>}<br>int emptyMethod() {}<br>} |

### 2.3.9. 构造/析构 Constructor/Finalizer

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 要初始化父类的构造函数 | 建议 | |
| 不要在构造函数中初始化静态数据，应在声明中初始化 | 建议 | |
| 用构造函数做简单初始化，复杂的功能应留到其他的方法在构造函数后执行 | 建议 | |
| 使用 Finalize() 释放资源，不过不要完全依赖 Finalize，资源要尽可能早的进行释放 | 建议 | |

### 2.3.10. 方法 Method

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| public 方法必须写注释，且注释中至少要包含@param 和 return 标签 getter/setter 方法可以不用写注释 private 方法必须写注释，但注释格式不做要求 | 必须 | |
| 尽可能缩小方法的能够被访问的范围。如果不需要类以外的方法访问的方法则使用 private ， 而只需要 package 范围内访问的方法使用 default，若即要 package 内的代码能够访问又要求其继承类能够访问则使用 protected,最后若需要所有的代码都可以访问则再使用 public | 建议 | |
| 应尽可能验证所有传入参数，不能假定非空对象，验证有错误时返回明确的错误信息 | 建议 | If(Parameter1!=NULL){<br>…}<br>// endIf<br>if(Parameter2!=NULL) {<br>…}<br>// endIf |
| 尽量保证每个方法只有一个出口，否则可能出现丢失返回的情况 | 建议 | boolean check() {<br>boolean result = false;<br>if (...) {<br>result = true;<br>}<br>return result;<br>} |

### 2.3.11. 属性 Attribute

| 命名规范 | 是否必须 | 范例 |
|---|---|---|
| 当某一些数据为常量时不要直接在代码中使用这些数据，而应该先将这些数据定义为 Java 的常量然后引用常量的标识符来完成预定义功能 | 必须 | |
| 在子类中不要重复定义父类中的成员变量 | 必须 | |
| 建议使用 getter 和 setter 访问同一个类中的成员变量，或者用关键字 this 应 避免出现 public 的成员变量,原则上都是<br>private ， 用 getter 和 setter 进行访问，保证封装性 | 建议 | this.getInternalName();<br>或者 this.internalName; |

| | | |
|---|---|---|
| 避免内联(in-line) 声明临时变量，除非是循环计数 | 建议 | for (int i = 0; i < 10; i++){<br>...<br>} |
| 建议不要在一个语句中为多个变量赋相同的值 | 建议 | 建议:<br>fooBar.fChar = 'c' ;<br>barFoo.lchar = 'c';<br>不建议:<br>fooBar.fChar = barFoo.lchar =<br>'c'; |
| 不要把赋值号用在容易被误认为等号的地方 | 建议 | |
| 不要为了提高运行时的性能而使用嵌入式的赋值 | 建议 | 建议:<br>a = b + c;<br>d = a + r<br>不建议:<br>d = (a = b + c) + r |

## 2.4. 页面规范

普通的详情页采取替换当前页、跳转到详情页的形式。只有复杂业务操作，如商品、订单时，因新窗口对原来信息依赖较大，才会弹出 div 窗口，而不是替换原有页。
所有 input 页面的 input 域都用 struts2 标签，而不要直接用 html 标签。否则无法回到 input 页面。

div 弹出窗口的实现示例：

- 窗口的 div 定义
- 添加表单验证和 ajax submit 及响应。所有 div 弹出窗口都用 ajax 来提交 form。
- 初始化 jqmodal
- 显示窗口

## 2.5. Action 开发规范

每个 Action 仅有一个操作方法。如果 action 方法对应的 input 页面不是默认的 input 页面，则需要配置 action 方法的名为"result"的 input。
Action 里不能直接操作 dao，Action 层只能通过 Service 操作 DAO。Action 层更不可以直接操作 Hiberntae。

## 2.6. Service 开发规范

Service 层不能直接操作数据库，必须通过 DAO 操作数据库。
为了事务控制，service 层的方法命名需要满足以下规范：

- save*:可写，必须要事务
- delete*:可写，必须要事务
- update*:可写，必须要事务
- get*：只读load*：只读

- find*:只读
- *：只读

对于需要缓存的内容可以采取以下方法：
在获取对象方法里加上以下注解以缓存对象：

```
@Cacheable(modelId = "caching")
```

在对象更新方法里加上以下注解以更新缓存：

```
@CacheFlush(modelId = "flushing")
```

## 2.7. DAO 开发规范

DAO 里尽量不要有业务逻辑，仅包括数据库操作。

## 2.8. Entity 开发规范

所有对象关联关系都采取 lazy-load 形式。

- One-to-one：对象两边都设置关联对象。
- One-to-many：如果从逻辑上两个对象是一种包含关系，则需要在父子两端都设置关联对象。
- Many-to-many：一般不要设置关联对象。除非两者关联很紧密。

Entity 里不能调用 DAO，也不能调用 service。Entity 里除了一般的 get 和 set 外，也可以有业务方法，但业务方法里仅能调用该 Entity 拥有的属性值或关联 Entity 的属性，而不能调用其它 Entity 的属性。对于仅与 Entity 直接的属性值或关联 Entity 的属性相关的业务方法，且从业务含义上是 Entity 的方法的话，最好将方法写在 Entity 里，而不是写在 Service 里。

异常处理、日志记录、数据验证、操作结果、国际化等规范参考“详细设计”文档。