**开发平台**

**编码规范**

V1.0

长天科技

2014年07月

**文档历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **作者** | **版本** | **注释** |
| 2014-11-28 | 任明辉 | V1.0 | 编写文档初稿。 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**版权说明**

亿众公司拥有本解决方案及相关应用软件、文档的全部版权。未经本公司书面授权许可，任何单位及个人不得以任何形式对应用软件及相关文档的任何部分进行复制、抄录、传播或将技术文档翻译成他国语言。

**信息反馈**

欢迎您通过各种渠道与我们联系，您的问题和建议都会得到我们的重视和妥善处理。

目 录

[1. 引言 6](#_Toc394335708)

[1.1. 编写目的 6](#_Toc394335709)

[1.2. 阅读对象 6](#_Toc394335710)

[1.3. 参考资料 6](#_Toc394335711)

[2. Java编码规范 6](#_Toc394335712)

[2.1. 行长度 6](#_Toc394335713)

[2.2. 命名规范 8](#_Toc394335714)

[2.2.1. Package的命名 8](#_Toc394335715)

[2.2.2. Class的命名 8](#_Toc394335716)

[2.2.3. Class 成员属性及变量的命名 9](#_Toc394335717)

[2.2.4. 变量的命名 9](#_Toc394335718)

[2.2.5. 常量的命名 9](#_Toc394335719)

[2.2.6. 数组的命名 9](#_Toc394335720)

[2.2.7. 方法的参数 9](#_Toc394335721)

[2.2.8. 方法命名 9](#_Toc394335722)

[2.2.9. 一般命名注意事项 9](#_Toc394335723)

[2.3. 编程惯例 10](#_Toc394335724)

[2.3.1. 变量赋值 10](#_Toc394335725)

[2.3.2. 常量赋值 10](#_Toc394335726)

[2.4. 注释规范 11](#_Toc394335727)

[2.4.1. 程序注释 11](#_Toc394335728)

[2.4.2. 文档注释 12](#_Toc394335729)

[2.5. 构造方法规范 12](#_Toc394335730)

[2.6. 修饰符规范 13](#_Toc394335731)

[2.7. 声明规范 13](#_Toc394335732)

[2.7.1. 变量的声明 13](#_Toc394335733)

[2.7.2. 类和接口声明 14](#_Toc394335734)

[2.8. 语句规范 14](#_Toc394335735)

[2.9. 代码长度 15](#_Toc394335736)

[3. 文档编写规范 15](#_Toc394335737)

# 引言

编码规范对于程序员而言尤为重要，有以下几个原因：

* 一个软件的生命周期中，80%的花费在于维护。
* 几乎没有任何一个软件，在其整个生命同期中，均由最初的开发人员来维护。
* 编码规范可以改善软件的可读性，可以让程序员尽快而彻底地理解新代码。
* 如果你将源码作为产品发布，就需要确认它是否被很好的打包并且清晰无误，一如你已构建的其它任何产品。

为了执行规范，每个软件开发人员必须一致遵守编码规范。每个人！！！

## 编写目的

本文档主要介绍长天快速开发平台的编码规范，

本编程规范建立在标准的Java编程规范的基础上，如和标准的Java编程规范有冲突，以本编程规范为准。

本文档涉及Java编码规范、文档编写规范。

## 阅读对象

文档预期读者群为Java开发人员。

## 参考资料

《sun推荐的Java编码规范.doc》

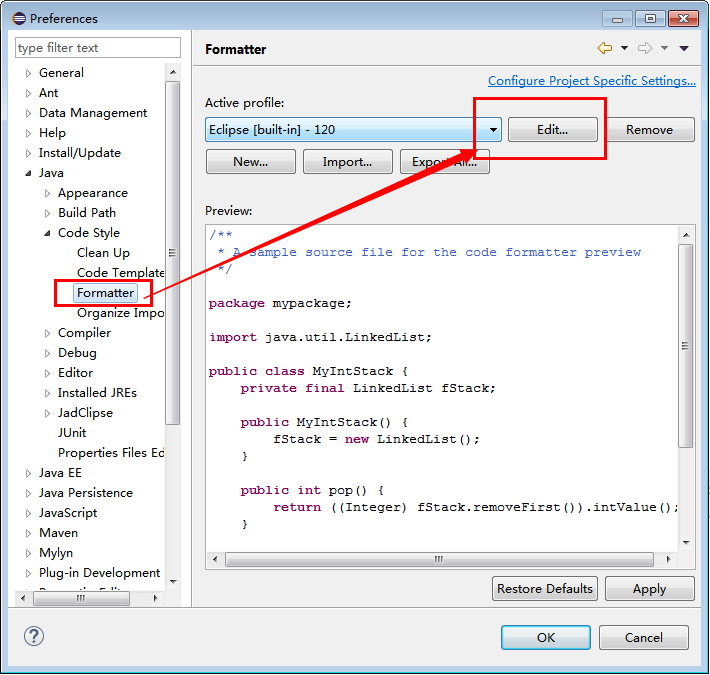
《java编程规范》

# Java编码规范

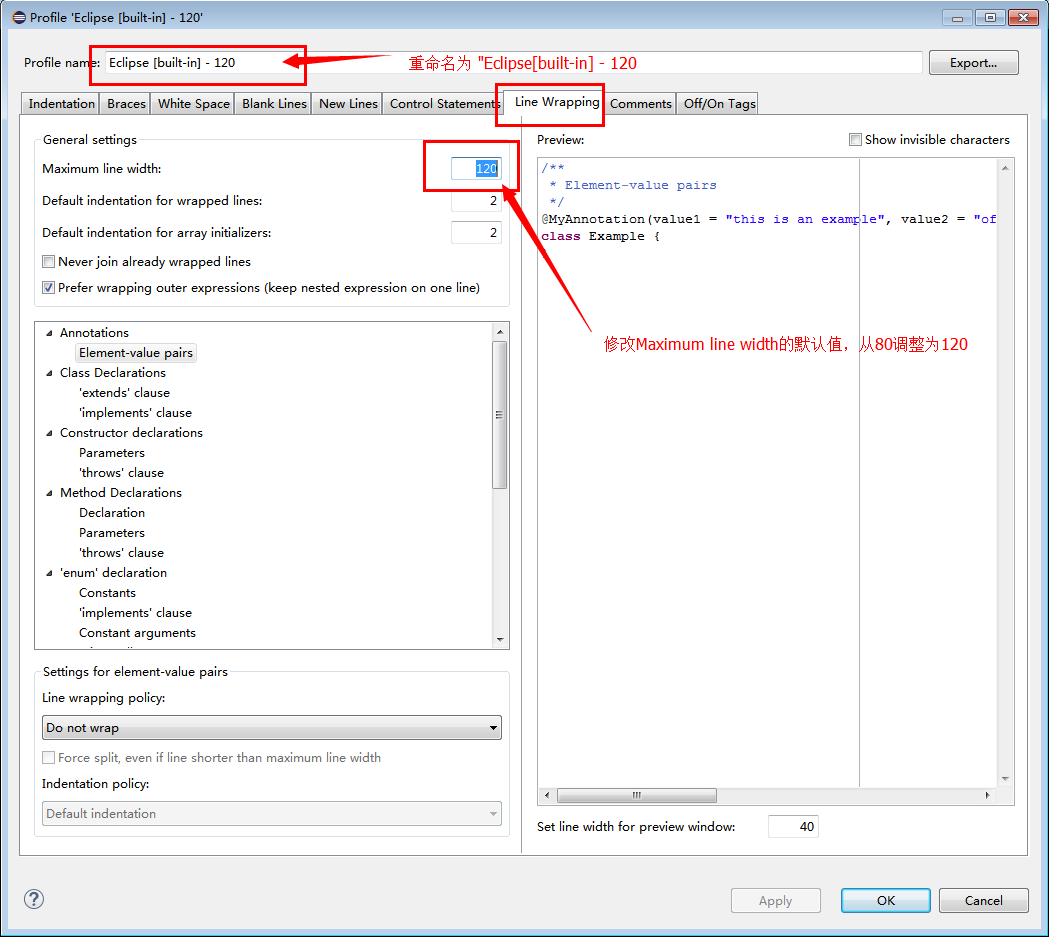
## 行长度

一行代码的长度不能超过120个字节，这里要求大家使用Eclipse或MyEclipse中的工具进行格式化。模板修改方法如下：

打开Java->Code Style->Formatter编辑当前的”active Profile”属性。



切换到“line wrapping”页签，将”Maximum line width”的默认值从80修改为120，ProfileName修改为我们自定义的名称，后点击OK保存。



经过如上操作后，再对java代码进行Ctrl+F格式化操作，即可保证代码120个字符才会换行。

## 命名规范

### Package的命名

Package的名字应由一个小写单词组成。为了保证包的唯一性，一般以公司国际域名开头。

例如：com.pansky.integration.\*\*。

### Class的命名

Class的名字必须由大写字母开头，并且其它单词的首写字母也要大写。

例如：

DbAccess

### Class 成员属性及变量的命名

变量/成员属性的名字用一个小写字母开头，后面的单词以大写字母开头。特别指出不要使用两个名词组成一个变量名。

例如：

private String lastName;

### 变量的命名

应简短且富于描述。变量名的选用应该易于记忆，便于他人理解，即：能够指出其用途，避免单个字符的变量名。

### 常量的命名

Java 里的常量，是用static final 修饰的，应该用全大写加下划线命名，并且常量名应尽量指出完整含义。如：

public static final String ORACLE\_DB\_DRIVER = "oracle.jdbc.driver.OracleDriver" ;

### 数组的命名

数组应该总是用下面的形式来命名：

byte[] buffs;

而不是：

byte buffs[];

### 方法的参数

和变量的命名规范一致，且应使用有意义的参数命名，如果可能的话，使用和要赋值的属性一样的名字，如：

void setCounter(int size){

this.size = size;

}

### 方法的命名

用一个小写字母开头，后面的单词以大写字母开头。应意义明确，对于属性的存取方法，应使用getXXX()和setXXX()名称，

以isXXX()，hasXXX()来命名返回值为boolean 类型的方法。

这里建议采用Eclipse自动生成get、set方法。

### 一般命名注意事项

#### 用有意义的名字命名变量

首先，用完整的英语单词或约定俗成的简写命名变量，鼓励多用约定俗成的简写命名变量。

例如：

public String zipCode

public int errNo

#### 用复数命名Collection 类变量

Collection，包括数组，Vector 等。命名时使用复数：

例如：

public Collection customers

## 编程惯例

### 变量赋值

避免在一个语句中给多个变量赋相同的值。它很难读懂，切容易发生对象引用问题。例如：

String fooBar =lchar = “abc”;

不要使用内嵌(embedded)赋值运算符试图提高运行时效率，这是编译器的工作。例如：

d = (a = b +c) + r;

应该写成

a = b + c;

d = a + r;

### 常量赋值

Java代码中位于for循环中作为计数器值的数字常量，除了-1，0和1之外，其他数值不应被直接写入代码，而应该以变量或常量的形式出现。

例如：

static final int MIN\_WIDTH = 4;

static final int MAX\_WIDTH = 999;

static final int GET\_THE\_CPU = 1;

## 注释规范

Java 提供了两种类型的注释：程序注释和文档注释。

注释应该是代码的概括性描述，提供不易直接从代码中得到的信息，并且只包含对阅读和理解程序有用的信息。

例如注释中包含相应的包如何编译或者在哪个目录下，而不应该包括这个包驻留在哪儿的信息。

注释中可以描述一些精妙的算法和一些不易理解的设计思想，但应该避免那些在程序代码中很清楚的表达出来的信息。

尽可能的避免过时的信息。错误的注释比没有注释更有害。

### 程序注释

由分隔符/\*…\*/，和// 隔开的部分，这些注释和C++ 中的注释一样。主要是对程序的某部分具体实现方式的注释。

程序注释有四种格式：块注释格式，单行注释，跟随注释，行尾注释

#### 块注释格式

块注释主要用于描述：文件、方法、数据结构和算法。一般在文件或者方法定义的之前使用。也可以用在方法定义里面，如果块注释放在函数或者方法定义里，它必须与它所描述的代码具有相同的缩进形式。

块注释应该用一个空行开头，以便于代码部分区分开来。

块注释举例：

/\*

\* Here is a block comment.

\*/

#### 单行注释

比较短的注释可以放在一行中，但必须与它所跟随的代码有相同的缩进。如果注释不可以放在一行，那么必须按照块注释的格式来写。单行的注释会被解释为一空行。

单行注释举例：

if (condition) {

/\* Handle the condition. \*/

...

}

#### 跟随注释

非常短的注释可以和它所描述的代码放在同一行。但要保证代码和注释之间有足够的间隔。在同一块代码中不止一个这样的注释时它们应该对齐。

跟随注释举例：

if (a == 2) {

return TRUE; /\* special case \*/

} else {

return isPrime(a); /\* works only for odd a \*/

}

#### 行尾注释

注释标记“//”能够注释一行或者该行由“//”开始直到行尾的部分。行尾注释不能用在多行文本注释中。但它可以将多行代码注释掉。这三种注释方法举例如下。

if (foo > 1) {

// Do a double-flip.

...

}else{

return false; // Explain why here.

}

//if (bar > 1) {

//

// // Do a triple-flip.

// ...

//}

//else{

// return false;

//}

### 文档注释

文档注释（即“doc 注释”）是Java 独有的。由分隔符/\*\*…\*/隔开。使用javadoc工具能够将文档注释抽取出来形成HTML文件。

/\*\*

\* 根据文件路径和文件名，查询对象！如果有多个，按照id倒叙，返回第一个（即返回最后一条数据）。

\* **@param** path

\* **@param** name

\* **@return**

\*/

**public** DocAttach findByPathAndName(String path, String name) {

**return null;**

}

文档注释是对程序的描述性注释，主要是提供给不需要了解程序具体实现的开发者使用。

## 构造方法规范

工具类不允许有public或default构造方法。这可以避免不必要的创建实例。

一个类里面的所有public方法都是static的，则这个类就可以作为工具类。其它类使用工具类的方法时，无需创建其实例，直接使用其方法即可。

例：在sysframework包中存在着的一个典型的工具类的例子：

org.springframework.util.StringUtils

它的所有public方法都是static的。

## 修饰符规范

按照Java语言规范，修饰符按如下顺序组织：

public

protected

private

abstract

static

final

transient

volatile

synchronized

native

strictfp

例：

public static final int NOT\_FOUND = 100; //正确的顺序

final static private String policyNo; //不正确的顺序。

## 声明规范

### 变量的声明

#### 每行定义变量数目

每行定义的变量数目必须有且只有一个。

例如：

int level; // indentation level

int size; // size of table

#### 变量初始化

在声明局部变量的时候就要初始化变量。

#### 变量定义位置

在for循环里的循环变量可以在for语句里面定义。

for (int i = 0; i < maxLoops; i++) { ... }

#### 数组的定义

数组的[]应该放在类型名的后面，而不是变量名的后面。

例：

String[] dangerNo; //正确的顺序

String[][][] bigArray; //正确的顺序

String dangerAddress[] ;//错误的顺序

### 类和接口声明

类和接口的声明应该遵循以下规范，建议大家使用Eclipse中的格式化快捷键进行格式化（Ctrl+F）：

1. 在方法名和参数列表的圆括号以及括号后的第一个参数间都没有空格。

2. 开括号“{”必须与声明语句放在同一行。

3. 闭括号“}”必须与声明语句有相同的缩进格式。

4. 如果类或者接口实现内容为空，则可以将“}”放在“{”后面。

5. 方法之间要用一个空行隔开。

class Sample extends Object {

int ivar1;

int ivar2;

Sample(int i, int j) {

ivar1 = i;

ivar2 = j;

}

int emptyMethod() {

. ..

}

}

## 语句规范

不允许出现空语句的情况，典型的错误案例是：

* 一行里只有一个分号
* 一个语句后面连续出现几个分号

不允许出现无意义语句，如

policyNo = policyNo;

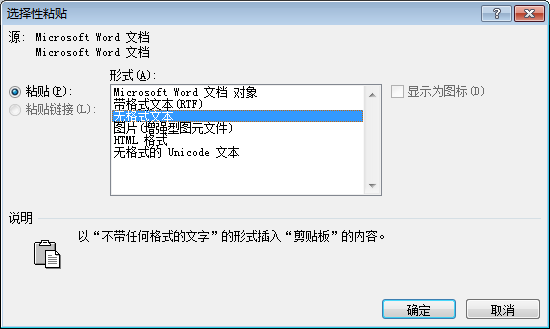
## 代码长度

|  |  |
| --- | --- |
| **检查点** | **最大字符数或行数** |
| 可执行语句的最大行数 | 120 |
| Java源文件最大行数 | 2000 |
| 方法、构造方法最多字符数 | 120 |
| 匿名内部类的最大行数 | 120 |
| 参数的最大个数 | 7 |

# 文档编写规范

请使用同一个文档编写工具进行文档编写，不要在word和wps之间切换，会导致文档格式丢失。

文档编写时，如果需要粘贴文档，不允许直接粘贴，而应该使用Ctrl+Alt+V的形式，选择粘贴“无格式文档”。



# 平台自有编码规范

## 开发前

Java代码在开发前，如果界面业务逻辑复杂超过三步，必须绘制VISIO流程图存档，存档前评审，评审通过后方能进入代码开发阶段。

## 代码中

Java代码开始编写时，首先写明类的注释，然后写接口注释，代码中的业务逻辑超过10行，必须先写业务处理注释，写过后在每个注释下补充代码实现。