

四川信息职业技术学院

众议

-让会议动起来

编

码

规

范

2.0

修改历史

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 文档标识： | |  | | | | 当前版本： | | **2.0** | | |
| 当前状态： | | 草稿 | | |  | 发布日期： | |  | |
| 发布 | | | ✓ |
| **日期** | **版本** | | **作者** | **修改内容** | | | **评审号** | | **变更控制号** |
| 5.13 | 1.0 | | 范子才 | 规范制定 | | |  | |  |
| 8.11 | 2.0 | | 范子才 | 补充及格式修订 | | |  | |  |
|  |  | |  |  | | |  | |  |

目录

[1 概述 4](#_Toc427157173)

[1.1 编写目的 4](#_Toc427157174)

[1.2 预期读者 4](#_Toc427157175)

[1.3 适用范围 4](#_Toc427157176)

[2 命名规范 4](#_Toc427157177)

[2.1 包命名 4](#_Toc427157178)

[2.2 类，接口命名 4](#_Toc427157179)

[2.3 方法名 4](#_Toc427157180)

[2.4 Layout命名 5](#_Toc427157181)

[2.5 Id命名 5](#_Toc427157182)

[2.6 资源命名 5](#_Toc427157183)

[3 变量名 5](#_Toc427157184)

[3.1 数组 5](#_Toc427157185)

[3.2 集合 5](#_Toc427157186)

[3.3 泛型 6](#_Toc427157187)

[4 代码风格 6](#_Toc427157188)

[4.1 花括号 6](#_Toc427157189)

[4.2 缩进 6](#_Toc427157190)

[4.3 空行 6](#_Toc427157191)

[4.4 方法 7](#_Toc427157192)

[4.5 控制语句 7](#_Toc427157193)

[4.6 异常的捕捉处理 7](#_Toc427157194)

[4.7 访问控制 7](#_Toc427157195)

[4.8 变量赋值 7](#_Toc427157196)

[4.9 圆括号 8](#_Toc427157197)

[4.10 返回值 8](#_Toc427157198)

[4.11 条件运算符“？”前的表达式， 8](#_Toc427157199)

[5 注释 9](#_Toc427157200)

[5.1 文件注释 9](#_Toc427157201)

[5.2 类注释 9](#_Toc427157202)

[5.3 方法注释 9](#_Toc427157203)

[5.4 类成员变量和常量注释 10](#_Toc427157204)

[5.5 其他注释 10](#_Toc427157205)

[5.6 XML注释 10](#_Toc427157206)

# 概述

## 编写目的

本文描述了JAVA开发中的有关包、类、接口、方法、实例变量、变量和常量的命名规则，用于规范JAVA编程过程中的命名和代码书写规范。

## 预期读者

“会议移动白板”开发人员

## 适用范围

适用于参与本APP开发或者二次开发的人员阅读，以及遵循的规则。

# 命名规范

## 包命名

1. 包名均为小写，如果包名称中包含单词缩写，则这个所写词的每个字母均应大写，如：HTTPpackage。
2. 根据功能模块划分包。

例:com.qihang<.projectname><.modulename1>[.modulename2][.modulename3]

## 类，接口命名

* 1. 类的名字必须由大写字母开头而单词中的其他字母均为小写;如果类名称由多个单词组成，则每个单词的首字母均应为大写例如TestPage;如果类名称中包含单词缩写，则这个所写词的每个字母均应大写，如：XMLExample,还有一点命名技巧就是由于类是设计用来代表对象的，所以在命名类时应尽量选择名词。
  2. 接口命名应该以大写i开头。

例:People TestPageXMLExample

## 方法名

方法的名字的第一个单词应以小写字母作为开头，后面的单词则用大写字母开头。可以为动词或动词+名词组合。

1. 设置/获取某个值的Method，应该遵循setV/getV规范
2. 返回长度的Method，应该命名为length
3. 测试某个布尔值的Method，应该命名为isV
4. 将对象转换为某个特定类型的Mehod应该命名为toF

getDate(); length(); isReady(); toOracleFormat();

## Layout命名

layout xml 的命名必须以全部单词小写，单词间以下划线分割，并且使用名词或名词词组，即使用模块名\_功能名称来命名。

例：knowledge\_gained\_main.xml正确

例：list\_book.xml错误！

## Id命名

layout 中所使用的id必须以全部单词小写，单词间以下划线分割，并且使用名词或名词词组，并且要求能够通过id直接理解当前组件要实现的功能。

例：某TextView @+id/textbookname 错误 !应为 @+id/book\_name\_show

例：某EditText @+id/textbookname  错误 !应为 @+id/book\_name\_edit

## 资源命名

layout中所使用的所有资源（如drawable,style等）命名必须以全部单词小写，单词间以下划线分割，并且尽可能的使用名词或名词组，即使用 模块名\_用途 来命名。如果为公共资源，如分割线等，则直接用用途来命名 。

例：menu\_icon\_navigate.png   正确

例：某分割线：line.png  或 separator.png  正确

# 变量名

1. 普通变量命名应该采用首字母小写，其他字母首字母大写的方式。
2. final static变量的名字应该都大写，并且指出完整含义。如果一个常量名称由多个单词组成，则应该用下划线来分割这些单词如：

NUM\_DAYS\_IN\_WEEK MAX\_VALU

1. 如果需要对变量名进行缩写时，一定要注意整个代码中缩写规则的一致性

Context=ctxmessage=msg

* 1. 无论什么时候，均提倡应用常量取代数字、固定字符串。也就是说，程序中除0、1以外，尽量不应该出现其他数字。

1. 索引变量：i、j、k等只作为小型循环的循环索引变量。
2. 逻辑变量：避免用flag来命名状态变量，用is来命名逻辑变量。

例：if(isClosed){ dosomeworks; return; }

## 数组

总是使用以下方式定义数组：

int[] arr = new int[10];

禁止：

intarr[] = new int[10];

## 集合

数组或者容器推荐命名方式为名词+s的方式，例如：

List persons = getPerson();

for(Person person : persons){

dosomeworks;

}

## 泛型

1. 应该尽量简明扼要(最好是一个字母)，以利于与普通的class或interface区分。
2. Container中的Element应该用E表示，Map里的key用K表示，value用V，Type用T表示，异常用X表示。
3. 如果需要接收多个Type类型的参数，应该用邻接T的大写字母——例如S——来依次表示，当然也可以用T1, T2这样的方式。

例：

public class HashSet extends AbstractSet {…}

public class HashMap extends AbstractMap {…}

public class ThreadLocal {…}

public interface Functor { T val() throws X; }

# 代码风格

## 花括号

1. 花括号统一采用以下格式：

if(boolexperssion){

dosomework;

}

1. 除非花括号中为空，不然任何情况下不能省略花括号，并且花括号必须换行，例如：

if(i==0){

return;

}

while(true) {}

禁止：

if(i != 0) return;

禁止：

if(i !=0) {return;}

## 缩进

强烈要求代码要在需要缩进的地方就得使用缩进，缩进方法可以使用Tab，也可以使用空格进行缩进，已使得代码阅读起来清晰明了。

## 空行

空行将把逻辑相关的代码给分隔开，以提高可读性。

以下为要用空行分割的地方

1. 一个方法里面的两个或多个逻辑之间，用以提高程序的可读性
2. 局部变量和方法的第一条语句之间
3. 变量声明区域之后要有空行分割
4. 常量声明区域之后要有空行分割
5. 方法声明之前要有空行分割

## 方法

1. 一个方法尽量不要超过20行，如果方法太长，说明其业务逻辑过于复杂，那么就需要进行方法拆分，尽量一个方法做一件事。
2. 一个方法尽量不要超过6个参数。

## 控制语句

1. 判断中如果有常量，则应将常量放在判断表达式的左边。如：

if(true == isLogin())…..

2. 尽量不要使用三目运算符的嵌套

所有if语句都应该使用花括号包裹，哪怕只有一行代码。如：

if(true){

//Do something…………

}

if(true)

i = 0;//不要这样使用

3. 对于循环：

while(index <objects.getCount()){

//每次循环都得调用getCount()

//如果这个方法很耗时，则会很影响效率。

//而且可能带来同步问题，如果有同步需求，请使用同步块或同步方法

}

//将操作结果存放到一个临时变量里面，这样可减少方法的调用次数。

finalint count = objects.getCount();

while(index < count){

//do something...

}

## 异常的捕捉处理

最小化从一个给定的抽象类中导出的异常的个数。对于经常发生的可预计事件不要采用异常。不要使用异常实现控制结构。

若有finally 子句，则不要在try 块中直接返回，亦不要在finally 中直接返回。

## 访问控制

若没有足够理由，不要把实例或类变量声明为公有。

## 变量赋值

避免在一个赋值语句中给多个变量赋相同的值，它会很难读懂。

例：ObjectA.m\_char= object.m\_char = ‘c’;

不要将赋值运算符用在容易与相等关系运算符混淆的地方。

例：

if(c++ = d++){//不要这样

……..

}

if((c++ = d++) != 0){//建议这样

….

}

不要使用内嵌赋值符来试图提高运行时的效率，那是编译器干的事。

例：

d = (a = b + c) + r; //不要写成这样

应该写成：

a = b + c;

d = a + r;

## 圆括号

一般而言，在含有多种运算符的表达式中使用圆括号来避免运算符优先级问题，是个好方法。

即使运算符的优先级对你而言可能很清楚，但对其他人未必如此。你不能假设别的程序员和你一样清楚运算符的优先级。

If(a == b && a == c){….}//不要这样写

If((a == b) && (a ==c)){….}//应该这样写

## 返回值

设法让你的程序结构符合目的。例如：

if(booleanExpression){

return true;

}else{

return false;

}

应该代之为如下代码：

returnbooleanExpression;

类似的

if(condition){

return x;

}else{

return y;

}

应该代之为：

return(condition?x:y);

## 条件运算符“？”前的表达式，

如果一个包含二元运算符的表达式出现在三元运算符" ? : "的"?"之前，那么应该给表达式添上一对圆括号。例如

 (x >= 0) ? x : -x;

# 注释

Java 程序有两类注释：实现注释(implementation comments)和文档注释(document comments)。实现注释是使用/\*...\*/和//界定的注释。文档注释(被称为"doc comments")由/\*\*...\*/界定。文档注释可以通过javadoc 工具转换成HTML 文件。

## 文件注释

所有的源文件都应该在开头有一个注释，其中列出类名、版本信息、日期和版权声明。如下：

/\*

文件名

包含的类名列表

版本信息，版本号

创建日期

版权声明

\*/

## 类注释

每一个类都要包含如下格式的注释，以说明当前类的功能等。

/\*\*

\*类名

\*@author作者<br />

\*实现的主要功能

\* 创建日期

\* 修改者，修改日期，修改内容

\*/

## 方法注释

每一个方法都要包含 如下格式的注释 包括当前方法的用途，当前方法参数的含义，当前方法返回值的内容和抛出异常的列表

/\*\*

\*

\* 方法的一句话的描述

\* <p>方法的详述（简单方法不用详述）</p>

\* @param s 说明参数含义

  \* @return 说明返回值含义

  \* @throws IOException 说明发生此异常的条件

  \* @throws NullPointerException 说明发生此异常的条件

\*/

## 类成员变量和常量注释

成员变量和常量需要使用java doc形式的注释，以说明当前变量或常量的含义

/\*\*

\*xxxx含义

\*/

## 其他注释

方法内部的注释 如果需要多行 使用/\*„„ \*/形式，如果为单行是用//„„形式的注释。不要再方法内部使用 java doc 形式的注释“/\*\*„„\*\*/”，简单的区分方法是，java doc形式的注释在 eclipse中为蓝色，普通注释为绿色。

## XML注释

如果当前layout 或资源需要被多处调用，或为公共使用的layout（若list\_item），则需要在xml写明注释。要求注释清晰易懂。