**Java语言编码规范**

信1606 20163534吕新

**1、Java源文件**

**1.1、注释**

所有的源文件都应当在开头有一个注释，此中列出类名、版本信息、日期和版本声明，具体可以参考我的博客：[Java注释模版设置](http://blog.csdn.net/u013249965/article/details/51812753)

**1.2、包和引入语句**

大部分java源文件中，第一行非注释行是包语句，在它之后可以跟引入语句，例如：

package com.org.jojo;

import java.io.Serializable;

**2、命名规范**

命名规范使程序更易读，从而更易于懂得。它们也可以提供一些有关标识符功能的信息，有助于懂得代码，无论它是一个常量，包，还是类。

2.1**包**

一个独立包名的前缀都是全部小写的ASCII字母并且是一个域名，凡是com，edu，gov，mil，net，org，或1981年 ISO 3166标准所指定的标识国度的英文双字符代码。包名的后续按照不同机构各自内部的定名规范而不同。这类定名规范可以特定名的构成来区分部门（department），项目（project），机械（machine），或注册名（login names）。

com.sun.engcom.apple.quicktime.v2 edu.cmu.cs.bovik.cheese

**2.2类**

命名规范：类名是个一名词，采取大小写混淆的体式格式，每个单词的首字母大写。尽量使你的类名简洁而富于描述。应用完全单词，避免缩写词（除非该缩写词被更广泛应用，像URL，HTML）

class Raster; class ImageSprite;

**2.3接口**

命名规范：大小写规范与类名类似

Interface0 RasterDelegate; interface Storing;

**2.4方法**

方法名是一个动词，采取大小写混淆的体式格式，第一个单词的首字母小写，之后单词的首字母大写。

run(); runFast(); getBackground();

**2.5变量**

除了变量名外，所有实例，包含类，类常量，均采取大小写混淆的体式格式，第一个单词的首字母小写，之后单词的首字母大写。变量名不该以下划线或美元符号开头，尽管这在语法上是允许的。变量名应简短且富于描述。变量名的选用应当易于记忆。尽量避免单个字符的变量名，除非是一次性的临时变量。临时变量凡是被取名为i，j，k，m和n，它们一般用于整型；c，d，e，它们一般用于字符型。

char c; int i; float myWidth;

**2.6实例变量**

大小写规范和变量名类似，除了前面须要一个下划线

int \_employeeId; String \_name; Customer \_customer;

**2.7常量**

类常量和ANSI常量的声明，应当全部大写，单词间用下划线隔开。（尽量避免ANSI常量，轻易引起错误）

static final int MIN\_WIDTH = 4; static final int MAX\_WIDTH = 999; static final int GET\_THE\_CPU = 1;

**2、缩进排版**

4个空格常被作为缩进排版的一个单位。缩进的确切声明并未具体指定（空格 vs. 制表符）。一个制表符便是8个空格。

**2.1、行长度**

尽量避免一行的长度跨越80个字符，因为很多终端和对象不能很好的处理。   
注意：用于文档中的例子应当应用更短的行长，长度一般不跨越70个字符。

**2.2、换行**

当一个表达式无法容纳在一行内时，可以根据如下一般规矩断开：

* 在一个逗号后面断开
* 在一个操纵符前面断开
* 宁可选择较高级别（higher-level）的断开，而非较初级别（lower-level）的断开
* 新的一行应当与上一行同一级别表达式的开首处对齐
* 若是以上规矩导致你的代码杂沓或者使你的代码都堆挤在右边，那就代之以缩进8个空格。

someMethod(longExpression1， longExpression2， longExpression3，

longExpression4， longExpression5);

var = someMethod1(longExpression1，

someMethod2(longExpression2，

longExpression3));

* 1

以下是两个断开算术表达式的例子。前者更好，因为断开处位于括号表达式的外边，这是个较高级别外断开。

longName1 = longName2 \* (longName3 + longName4 - longName5)

+ 4 \* longname6; //合适

longName1 = longName2 \* (longName3 + longName4

- longName5) + 4 \* longname6; //避免

以下是两个缩进办法声明的例子。前者是常规情况。后者若应用常规的缩进体式格式将会使第二行和第三行移得很靠右，所以代之以缩进8个空格

//传统的缩进

someMethod(int anArg， Object anotherArg， String yetAnotherArg，

Object andStillAnother) {

...

}

//8个缩进避免产生很深的缩进

private static synchronized horkingLongMethodName(int anArg，

Object anotherArg， String yetAnotherArg，

Object andStillAnother) {

...

}

if语句的换行凡是应用8个空格的规矩，因为常规缩进（4个空格）会使语句体看起来斗劲费劲。比如：

//不要使用这种缩进

if ((condition1&& condition2)

|| (condition3 && condition4)

||!(condition5 && condition6)) { //不合理

doSomethingAboutIt(); //导致这行容易被忽略

}

//使用这种缩进代替

if ((condition1&& condition2)

|| (condition3 && condition4)

||!(condition5 && condition6)) {

doSomethingAboutIt();

}

//或者使用这个

if ((condition1&& condition2) || (condition3 && condition4)

||!(condition5 && condition6)) {

doSomethingAboutIt();

}

这里有三种可行的办法用于处理三元运算表达式：

alpha = (aLongBooleanExpression) ? beta : gamma;

alpha = (aLongBooleanExpression) ? beta

: gamma;

alpha = (aLongBooleanExpression)

? beta

: gamma;

**3、注释**

Java程序有两类注释：实现注释和文档注释。实现注释是那些在C++中见过的，应用/*…*/和//界定的注释。文档注释（被称为”doc comments”）是Java独有的，并由/*\*…*/界定。文档注释可以经由过程javadoc对象转换成HTML文件。

注释应被用来给出代码的总括，并供给代码自身没有供给的附加信息。注释应当仅包含与浏览和懂得程序有关的信息。例如，响应的包如何被建树或位于哪个目次下之类的信息不该包含在注释中。

在注释里，对设计决定计划中首要的或者不是显而易见的地方所进行声明是可以的，但应避免给代码中已清楚表达出来的反复信息。不可标注多余的的注释。应避免那些代码更新就可能过期的注释。

**注意**：频繁的注释有时反应出代码的低质量。当你感觉被迫要加注释的时辰，推敲一下重写代码使其更清楚。

注释不该写在用星号或其他字符画出来的大框里。注释不该包含诸如制表符和回退符之类的特别字符。

**3.1、实现注释的格局**

程序可以有4种实现注释的风格：块（block）、单行（single-line）、尾端（trailing）和行末（end-of-line）。

**3.1.1、块注释**

块注释一般是对文件、办法、数据布局和算法的描述。块注释被置于每个文件的开端处以及每个方法之前。它们也可以被用于其他之处，比如方法内部。在功能和方法内部的块注释应当和它们所描述的代码具有一样的缩进格局。

块注释之前应当有一个空行，用于把块注释和代码分别开来，比如：

/\*

\* 这是一个块注释

\*/

块注释可以以/\*-开头，比如indent(1)就可以将其认为一个代码块的开端，而不会覆盖它。

/\*-

\* 这是一种比较特殊的块注释

\* 用indent(1)进行代码格式化

\*

\* one

\* two

\* three

\*/

注意：若是你不应用indent（1），就不必在代码中应用/\*-，或为他人可能对你的代码运行indent（1）作让步。

另见”文档注释”

**3.1.2、单行注释**

短注释可以显示在一行内，并与之后的代码具有一样的缩进层级。若是一个注释不能在一行内写完，就该采取块注释（见”块注释”）。单行注释之前应当有一个空行。以下是一个Java代码中单行注释的例子：

if (condition) {

/\* Handle thecondition. \*/

...

}

**3.1.3、尾端注释**

极短的注释可以与它们所要描述的代码位于同一行，当然应当有足够的空白来分隔代码和注释。若有多个短注释呈现于大段代码中，它们应当具有同样的缩进。

以下是一个Java代码中尾端注释的例子：

if (a == 2) {

return TRUE; /\* 注释 \*/

} else {

return isPrime(a); /\* 只标注这一行 \*/

}

**3.1.4、行末注释**

注释界定符”//”，可以注释掉整行或者一行中的一项目组。

if (foo > 1) {

// Do adouble-flip.

...

} else {

return false; // Explain why here.

}

//if (bar > 1){

//

// // Do a triple-flip.

// ...

//} else {

// return false;

//}

**3.2、文档注释**

**注意**：此处描述的注释格局，另见”Java源文件典范”

文档注释描述Java的类、接口、构造器，方法，以及字段。每个文档注释都被置于注释定界符/*\*…*/之中，一个注释对应一个类、接口或成员。该注释应位于声明之前：

/\*\*

\* 提供 class 示例 ...

\*/

public classExample { ...

注意外层（top-level）的类和接口是不缩进的，而其成员是缩进的。描述类和接口的文档注释的第一行（/\*\*）不需缩进；随后的文档注释每行都缩进1格（使星号纵向对齐）。成员，包含构造函数在内，其文档注释的第一行缩进4格，随后每行都缩进5格。

文档注释不可放在一个方法或构造器的定义块中，因为Java会将位于文档注释之后的第一个声明与其关联。

**4、声明**

**4.1、每行声明变量的数量**

推荐一行一个声明，因为这样有利于写注释。即，

int level; // 等级

int size; // 表格的大小

要优于：

int level， size;

不要将不同类型变量的声明放在同一行，例如：

int foo， fooarray[]; //错误！

**注意**：上面的例子中，在类型和标识符之间放了一个空格，还可以使用制表符来进行格式的调整：

int level; // 等级

int size; // 表格的大小

Object currentEntry; // 表格的当前项

**4.2、初始化**

尽量在声明局部变量的同时初始化。

**4.3、布局**

只在代码块的开始处声明变量。（一个块是指被包含在大括号”{“和”}”中心的代码。）不要在第一次用到该变量时才声明。这会把注意力不集中的程序员搞糊涂，同时会降低该代码块的可移植性。

void myMethod() {

int int1 = 0; // 方法代码块的开始

if (condition) {

int int2 = 0; // "if"代码块的开始

...

}

}

for 循环中的索引变量是一个例外：

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

避免声明的局部变量覆盖上一级声明的变量，不要在内部代码块中声明相同的变量名：

int count;

...

myMethod() {

if (condition) {

int count= 0; // 避免！

...

}

...

}

* \

**4.4、类和接口的声明**

当编写类和接口时，应当遵守以下规矩：

* 方法名和参数列表之前的左括号”（”间不要有空格
* 左大括号”{“位于声明类或方法的末尾
* 右大括号”}”另起一行，与响应的声明语句对齐，除非是一个空语句，”}”应紧跟在”{“之后
* 方法与方法之间以空行分隔

class Sample extends Object {

int ivar1;

int ivar2;

Sample(int i, int j) {

ivar1 = i;

ivar2 = j;

}

int emptyMethod() {}

...

}

**5、语句**

**5.1、简单语句**

每行最多包含一条语句，例如：

argv++; // 正确的

argc--; // 正确的

argv++; argc--; // 避免！

=**5.2、复合语句**

复合语句是包含在大括号中的语句，如”{ 语句 }”。例如：

* 括号语句的内容应在上一级语句的前提下加一个缩进
* 左大括号”{“应位于复合语句开始端行的行尾；右大括号”}”应另起一行并与复合语句首行对齐。
* 大括号可以被用于所有语句，包含单个语句，只要这些语句是诸如if-else或for一类的。以便避免添加语句的时候忘记加括号。

**5.3、返回语句**

一个带返回值的return语句不应用小括号”（）”，除非它们以某种体式格式使返回值更为显见。例如：

return;

return myDisk.size();

return (size ? size : defaultSize);

**5.4、if，if-else，if else-if else语句**

if-else语句应当具有如下格局：

if (condition) {

statements;

}

if (condition) {

statements;

} else {

statements;

}

if (condition) {

statements;

} else if (condition) {

statements;

} else {

statements;

}

**注意**：if语句老是用”{“和”}”括起来，避免引起错误的格局：

if (condition)// 避免

statement;

**5.5、for语句**

一个for语句应当具有如下格局：

for (initialization; condition; increment) {

statements;

}

一个空的for语句应当具有如下格局：

for (initialization; condition; increment);

当在for语句的初始化或更新子句中应用逗号时，避免因应用三个以上变量，而导致复杂。若须要，可以在for循环之前（为初始化子句）或for循环末尾（为更新子句）应用其它的语句。

**5.6、while语句**

一个while语句应当具有如下格局

while (condition) {

statements;

}

一个空的while语句应当具有如下格局：

while (condition);

**5.7、do-while语句**

一个do-while语句应当具有如下格局：

do {

statements;

} while (condition);

**5.8、switch语句**

一个switch语句应当具有如下格局：

switch (condition) {

case ABC:

statements;

case DEF:

statements;

break;

case XYZ:

statements;

break;

default:

statements;

break;

}

每当一个case顺着往下运行时（因为没有break语句），在break语句的地方添加注释。

**5.9、try-catch语句**

一个try-catch语句应当具有如下格局：

try {

statements;

} catch (ExceptionClass e) {

statements;

}

一个try-catch语句后面也可能跟着一个finally语句，无论try代码块是否顺利执行完，它都被执行。

try {

statements;

} catch (ExceptionClass e) {

statements;

} finally {

statements;

}

**6、空白**

**6.1、空行**

空行将逻辑相关的代码段分隔开，以提高可读性。

下列情景经常应用两个空行：

* 一个源文件的两个片段（section）之间
* 类声明和接口声明之间

下列情景经常应用一个空行：

* 两个方法之间
* 方法内的局部变量和方法的第一条语句之间
* 块注释（见”3.1.1”）或单行注释（见”3.1.2”）之前
* 一个方法内的两个逻辑块之间，用以提高可读性

**6.2、空格**

下列情景应当使用空格：

* 一个紧跟着括号的关键字应当被空格分隔，例如：

while (true) {

...

}

**注意**：空格不该置于方法名与其左括号之间。这将有助于区分关键字和办法调用。

* 空白应当位于参数列表中逗号的后面
* 所有的二元运算符，除了”.”，应当使用空格将操作符和变量分开。一元操纵符和变量之间不加空格，比如：负号（”-“）、自增（”++”）和自减（”–”）。例如：

a += c + d;

a = (a + b) /(c \* d);

while (d++ = s++) {

n++;

}

printSize("size is " + foo + "

");

* for语句中的表达式应当被空格分隔，例如：

for (expr1; expr2; expr3)

* 强制类型转换后应当跟一个空格，例如：

myMethod((byte) aNum, (Object) x);

myMethod((int) (cp + 5), ((int) (i + 3))+ 1);

**7、编程注意**

**7.1、实例以及类变量**

若没有足够的理由，不要把实例或类变量声明为公有的。实例变量无需显式的设置（set）和获取（get）。

**7.2、引用类变量和类办法**

避免用一个对象调用一个类的静态变量和静态办法。应当用类名调换。例如：

classMethod(); //OK

AClass.classMethod(); //OK

anObject.classMethod(); //避免！

**7.3、常量**

位于for循环中作为初始值的数字常量，除了-1，0和1之外，不该被直接写入代码。

**7.4、变量赋值**

避免在一个语句中给多个变量赋同样的值。它很难读懂。例如：

fooBar.fChar = barFoo.lchar = 'c'; // 避免！

不要将赋值运算符和关系运算符混用。例如：

if (c = d++) { // 避免！ （Java 不认可）

...

}

应当写成

if ((c = d++) != 0) {

...

}

不要使用内嵌赋值运算符试图提高运行的效率，这是编译器的工作。例如：

d = (a = b + c) + r; // 避免！

应当写成

a = b + c;

d = a + r;