编码规范书

泰山小组

目录

[1.开发工具设置 1](#_Toc25386)

[1.1一般概念 1](#_Toc6345)

[1.2示范 1](#_Toc1577)

[2.Java命名约定 1](#_Toc23616)

[2.1一般概念 1](#_Toc25733)

[2.2具体的命名应用 1](#_Toc15499)

[3.Java注释约定 2](#_Toc28843)

[3.1一般概念 2](#_Toc13214)

[3.2示范 2](#_Toc20733)

[3.3注释部分 3](#_Toc31150)

[4.Java 文件样式约定 3](#_Toc8928)

[4.2 Package/Import 3](#_Toc3174)

[4.3 Class 4](#_Toc29618)

[4.4 Class Field 4](#_Toc6306)

[4.5存取方法（类的设置与获取成员函数） 4](#_Toc1544)

[4.6构造函数 5](#_Toc8996)

[4.7.克隆方法 5](#_Toc28182)

[4.8类方法（类的普通成员函数） 5](#_Toc28644)

[4.9 toString方法 5](#_Toc9785)

[4.10.main方法 6](#_Toc25834)

[5. Java编码其它约定 6](#_Toc1276)

[5.1文档化 6](#_Toc19656)

[5.2缩进 6](#_Toc2060)

[5.3页宽 6](#_Toc15930)

[5.4 JSP文件命名 6](#_Toc17394)

[5.5 Servlet类命名 6](#_Toc22901)

[6.编程建议 6](#_Toc1686)

[6.1使用StringBuffer对象 6](#_Toc20773)

[6.2避免太多的使用synchronized关键字 7](#_Toc27345)

[6.3避免使用java.util.Vector类 7](#_Toc9265)

[6.4尽量使用接口而不是一个具体的类 7](#_Toc19866)

# 1.开发工具设置

## 1.1一般概念

（1）设置工作空间字符集为UTF-8

（2）Java Formatter使用统一格式化模板

（3）JDK大小版本统一

## 1.2示范

（1）设置工作空间字符集Window -> Preferences -> General -> Workspace

（2）Text file encoding Other选择UTF-8

# 2.Java命名约定

　　除了以下几个特例之外，命名时应始终采用完整的英文描述符。此外，一般应采用小写字母，但类名、接口名以及任何非初始单词的第一个字母要大写。

## 2.1一般概念

（1）尽量使用完整的英文描述符。

（2）采用适用于相关领域的术语。

（3）采用大小写混合使名字可读。

（4）尽量少用缩写，但如果用了，要明智地使用，且在整个工程中统一

（5）避免使用长的名字（小于 15 个字母是个好主意）

（6）避免使用类似的名字，或者仅仅是大小写不同的名字

（7）避免使用下划线（除静态常量等）

（8）一行代码仅声明一个变量

（9）方法命名与功能相符

## 2.2具体的命名应用

（1）包（Package）采用完整的英文描述符，应该都是由小写字母组成。对于全局包，将你的 Internet 域名反转并接上包名。如： java.awt、com.ambysoft等。

（2）类（Class）采用完整的英文描述符，所有单词的第一个字母大写。如： Customer,SavingsAccount等。

（3）接口（Interface）采用完整的英文描述符说明接口封装，所有单词的第一个字母大写。习惯上，名字后面加上后缀 able，ible或者er，但这不是必需的。如： Contactable, Prompter。

（4）组件/部件（Component）使用完整的英文描述来说明组件的用途，末端应接上组件类型。如：okButton, customerList,fileMenu。

（5）异常（Exception）通常采用字母 e 表示异常。

（6）类变量字段采用完整的英文描述，第一个字母小写，任何中间单词的首字大写，如：firstName,lastName。

（7）实参/参数同字段/属性的命名规则，如：

public void setFirstName(String firstName)

{

this.firstName = firstName;

}

（8）获取成员函数被访问字段名的前面加上前缀 get。如getFirstName()。

（9）布尔型的获取成员函数所有的布尔型获取函数必须用单词 is 做前缀。

如:isPersistent(),isString()。

（10）设置成员函数被访问字段名的前面加上前缀 set。如:setFirstName()

,setLastName(),setWarpSpeed()

（11）普通成员函数采用完整的英文描述说明成员函数功能，第一个单词尽可能采用一个生动的动词，首字母小写。如：openFile()，addAccount()。

（12）静态常量字段（static final）全部采用大写字母，单词之间用下划线分隔。

如:MIN\_BALANCE,DEFAULT\_DATE。

（13）循环计数器通常采用字母i，j，k 或者 counter 都可以接受。数组应该总是用下面的方式来命名：byte[] buffer;

# 3.Java注释约定

　　一个很好的可遵循的有关注释的经验法则是：问问你自己，你如果从未见过这段代码，要在合理的时间内有效地明白这段代码，你需要哪些信息。

## 3.1一般概念

（1）注释应该增加代码的清晰度

（2）保持注释的简洁

（3）在写代码之前写注释

（4）注释出为什么做了一些事，而不仅仅是做了什么

（5）注释需代码异常处理进行说明

（6）方法需有文档注释（功能、输入、返回及其他可选）

（7）复杂逻辑及算法必须注释说明逻辑

（8）对老代码进行修改时，涉及对已有注释说明变动时，需同步修改注释

## 3.2示范

1. 文档注释：

在紧靠接口、类、成员函数和字段声明的前面注释它们。如：

/\*\*

\*

\* 客户：客户是我们将服务和产品卖给的人或机构。

\* 单行

\*/

在成员函数内采用单行注释，来说明业务逻辑、代码段和暂时变量的声明。注释符"//"后必须紧跟一个空格，然后才是注释信息。如：

// 遵照Sarek的规定，给所有

// 超过 $1000 的发货单

// 打 5% 的折扣。让利活

// 动于 1995年 2 月开始.

## 3.3注释部分

　　（1）类类的目的、即类所完成的功能，注释出采用的变量。

　　（2）接口设置接口的目的、它应如何被使用以及如何不被使用。

　　（3）成员函数注释对于设置与获取成员函数，在成员变量已有说明的情况下，可以不加注释；普通成员函数要求说明完成什么功能，参数含义是什么返回什么。

　　（4）普通成员函数内部注释控制结构，代码做了些什么以及为什么这样做，处理顺序等。

　　（5）实参/参数参数含义、及其它任何约束或前提条件。

　　（6）字段/属性字段描述。

（7）局部变量无特别意义的情况下不加注释。

# 4.Java 文件样式约定

　　所有的 Java(\*.java) 文件都必须遵守如下的样式规则：

4.1.版权信息

　 版权信息必须在 java 文件的开头，比如：

/\*\*

\* Copyright @ 2012Shenzhen Oracle Co. Ltd.

\* All right reserved.

\* @author: xxx

\* date: 2012-12-12

\*/

　　其他不需要出现在javadoc的信息也可以包含在这里。

## 4.2 Package/Import

　　package 行要在 import 行之前，import 中标准的包名要在本地的包名之前，而且按照字母顺序排列。如果 import 行中包含了同一个包中的不同子目录，则应该用 \* 来处理。如：

package com.oracle;

import java.io.\*;

import java.util.Observable;

import oracle.util.Application;

这里 java.io.\* 是用来代替InputStream and OutputStream的。

## 4.3 Class

接下来的是类的注释，一般是用来解释类的。

/\*\*

\* A class representing a set of packet and byte counters

\* It is observable to allow it to be watched, but only

\* reports changes when the current set is complete

\*/

　　接下来是类定义，包含了在不同的行的 extends 和 implements

　 public class CounterSet extends Observable implements Cloneable

{

……

……

}

## 4.4 Class Field

接下来是类的成员变量：public的成员变量必须生成文档（JavaDoc）。proceted、private和 package 定义的成员变量如果名字含义明确的话，可以没有注释。如：

　/\*\*

\* Packet counters

\*/

protectedint[] packets;

## 4.5存取方法（类的设置与获取成员函数）

　　接下来是类变量的存取的方法。它只是简单的用来将类的变量赋值获取值的话，可以简单的写在一行上，如类的成员变量已经有注释，类变量的存取方法可以没有注释。

publicint[] getPackets()

{

returnthis.packets;

}

public void setPackets(int[] packets

{

this.packets = packets;

}

……

　　要求说明的是，对于集合，加入成员函数来插入和删除项；另其它的方法不要写在一行上。

## 4.6构造函数

接下来是构造函数，它应该用递增的方式写（比如：参数多的写在后面）。

publicCounterSet(int size)

{

this.size = size;

}

## 4.7.克隆方法

如果这个类是可以被克隆的，那么下一步就是 clone 方法：

public Object clone()

{

try {

……

}catch(CloneNotSupportedException e)

{

……

}

}

## 4.8类方法（类的普通成员函数）

下面开始写类的方法

/\*\*

\* Set the packet counters

\* param r1 - ……

\* param r2 - ……

\* ……

\*/

Protected final void setArray(int[] r1, int[] r2, int[] r3, int[] r4)

throws IllegalArgumentException

{

// Ensure the arrays are of equal size

……

}

## 4.9 toString方法

一般情况下，每一个类都应该定义toString方法：

public String toString() {

……

}

## 4.10.main方法

　　普通类，考虑置入一个main()方法，其中包含用于测试那个类的代码，如果包含了main() 方法, 那么它应该写在类的底部。

# 5. Java编码其它约定

## 5.1文档化

　　必须用javadoc来为类生成文档。不仅因为它是标准，这也是被各种 java 编译器都认可的方法。使用 @author 标记是不被推荐的，因为代码不应该是被个人拥有的。

## 5.2缩进

　　缩进应该是每行4个空格。不要在源文件中保存Tab字符，，在使用不同的源代码管理工具时Tab字符将因为用户设置的不同而扩展为不同的宽度。

## 5.3页宽

　　页宽应该设置为120字符。当前显示器基本使用宽屏为主，源代码一般不会超过这个宽度，并导致无法完整显示，但这一设置也可以灵活调整。在任何情况下，超长的语句应该在一个逗号或者一个操作符后折行。一条语句折行后，应该比原来的语句再缩进2个字符。

## 5.4 JSP文件命名

　　采用完整的英文描述说明JSP所完成的功能，尽可能包括一个生动的动词，第一个字母小写，如：viewMessage.jsp、editUser.jsp或者forumChooser.jsp等。

## 5.5 Servlet类命名

　　一般对应于所服务的对象加后缀Service来命名，如：UserService，TradeService等。

# 6.编程建议

## 6.1使用StringBuffer对象

　　在处理 String 的时候要尽量使用StringBuffer类，StringBuffer类是构成 String 类的基础。String 类将StringBuffer类封装了起来，（以花费更多时间为代价）为开发人员提供了一个安全的接口。当我们在构造字符串的时候，我们应该用StringBuffer来实现大部分的工作，当工作完成后将StringBuffer对象再转换为需要的 String 对象。比如：如果有一个字符串必须不断地在其后添加许多字符来完成构造，那么我们应该使用StringBuffer对象和它的 append() 方法。如果我们用 String 对象代替StringBuffer对象的话，会花费许多不必要的创建和释放对象的 CPU 时间。

## 6.2避免太多的使用synchronized关键字

　　避免不必要的使用关键字 synchronized，应该在必要的时候再使用它，这是一个避免死锁的好方法。必须使用时，也尽量控制范围，最好在块级控制。

## 6.3避免使用java.util.Vector类

　　因为"Unlike the new collection implementations, Vector is synchronized."，所以使用java.util.Vector类在性能上会有所减低。

## 6.4尽量使用接口而不是一个具体的类

　　比方如下需求，给定一个SQL语句，返回一个对象的列表，实现中用java.util.ArrayList实现，于是定义方法为：

Public java.util.ArrayListgetObjectItems(String sql)

　　上面的方法存在一个问题，当getObjectItems内改用Vector或LinkedList实现，外部类必须做相应更改。一个更好的方法是定义返回值为java.util.AbstractList更合适：publicjava.util.AbstractListgetObjectItems(String sql)。这样即使更改实现，外部类也不必做相应更改。