# 命名规范：

# Package 的命名（com.monitor.xxx）

　　Package 的名字应该都是由一个小写单词组成。

# Class 的命名

　　Class 的名字必须由大写字母开头而其他字母都小写的单词组成

Class 变量的命名

　　变量的名字必须用一个小写字母开头。后面的单词用大写字母开头。

Static Final 变量的命名

　　Static Final 变量的名字应该都大写，并且指出完整含义。

参数的命名

　　参数的名字必须和变量的命名规范一致。

数组的命名

　　数组应该总是用下面的方式来命名:

　　byte[] buffer;

　　而不是:

　　byte buffer[];

方法的参数

　　使用有意义的参数命名，如果可能的话，使用和要赋值的字段一样的名字:

　　setCounter(int size){

　　this.size = size;

　　}

Java 文件样式

　　所有的 Java(\*.java) 文件都必须遵守如下的样式规则

版权信息

　　版权信息必须在 java 文件的开头，比如:

　　/\*\*

\* User:开发者名称

\* mail：开发者邮箱

\* All right reserved.

　　\*/

　　其他不需要出现在 javadoc 的信息也可以包含在这里。

Package/Imports

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。

　　package com.monitor.xxx;

　　import java.io.\*;

　　import java.util.Observable;

　　import hotlava.util.Application;

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

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](http://www.yesky.com/key/4979/4979.html" \t "_blank) 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

　　Class Fields

　　接下来是类的成员变量:

　　/\*\*

　　\* Packet counters

　　\*/

　　protected int[] packets;

　　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。

存取方法

　　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。

　　/\*\*

　　\* Get the counters

　　\* @return an array containing the statistical data. This array has been

　　\* freshly allocated and can be modified by the caller.

　　\*/

　　public int[] getPackets() { return copyArray(packets, offset); }

　　public int[] getBytes() { return copyArray(bytes, offset); }

　　public int[] getPackets() { return packets; }

　　public void setPackets(int[] packets) { this.packets = packets; }

　　其它的方法不要写在一行上

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)。

访问类型 ("public", "private" 等.) 和 任何 "static", "final" 或 "synchronized" 应该在一行中，并且方法和参数另写一行，这样可以使方法和参数更易读。

　　public

　　CounterSet(int size){

　　this.size = size;

　　}

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 {

　　CounterSet obj = (CounterSet)super.clone();

　　obj.packets = (int[])packets.clone();

　　obj.size = size;

　　return obj;

　　}catch(CloneNotSupportedException e) {

　　throw new InternalError("Unexpected CloneNotSUpportedException: " + e.getMessage());

　　}

　　}

类方法

　　下面开始写类的方法:

　　/\*\*

　　\* Set the packet counters

　　\* (such as when restoring from a database)

　　\*/

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

　　throws IllegalArgumentException

　　{

　　//

　　// Ensure the arrays are of equal size

　　//

　　if (r1.length != r2.length || r1.length != r3.length || r1.length != r4.length)

　　throw new IllegalArgumentException("Arrays must be of the same size");

　　System.arraycopy(r1, 0, r3, 0, r1.length);

　　System.arraycopy(r2, 0, r4, 0, r1.length);

　　}

toString 方法

　　无论如何，每一个类都应该定义 toString 方法:

　　public

　　String toString() {

　　String retval = "CounterSet: ";

　　for (int i = 0; i < data.length(); i++) {

　　retval += data.bytes.toString();

　　retval += data.packets.toString();

　　}

　　return retval;

　　}

　　}

main 方法

　　如果main(String[]) 方法已经定义了, 那么它应该写在类的底部.

**代码编写格式**

代码样式

　　代码应该用 unix 的格式，而不是 windows 的(比如:回车变成回车+换行)

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。

缩进

缩进应该是每行2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.

如果你使用 UltrEdit 作为你的 Java 源代码编辑器的话，你可以通过如下操作来禁止保存Tab字符, 方法是通过 UltrEdit中先设定 Tab 使用的长度室2个空格，然后用 [Format](http://www.yesky.com/key/4780/534780.html" \t "_blank)|Tabs to Spaces 菜单将 Tab 转换为空格。

页宽

页宽应该设置为80字符. 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.

　　{} 对

　　{} 中的语句应该单独作为一行. 例如, 下面的第1行是错误的, 第2行是正确的:

　　if (i>0) { i ++ }; // 错误, { 和 } 在同一行

　　if (i>0) {

　　i ++

　　}; // 正确, { 单独作为一行

　　} 语句永远单独作为一行.

　　如果 } 语句应该缩进到与其相对应的 { 那一行相对齐的位置。

括号

左括号和后一个字符之间不应该出现空格, 同样, 右括号和前一个字符之间也不应该出现空格. 下面的例子说明括号和空格的错误及正确使用:

　　CallProc( AParameter ); // 错误

　　CallProc(AParameter); // 正确

不要在语句中使用无意义的括号. 括号只应该为达到某种目的而出现在源代码中。下面的例子说明错误和正确的用法:

　　if ((I) = 42) { // 错误 - 括号毫无意义

　　if (I == 42) or (J == 42) then // 正确 - 的确需要括号