Android编码规范说明文档1.0

**1.**  **命名规定**

命名规范使程序更易读，从而更易于理解。它们也可以提供一些有关标识符意图的信息，有助于开发人员理解代码。

1.1  包的命名

包的命名应该都是小写字母，单词之间用“.”分开。所有的JAVA文件必须建立在com.baidu包下，接项目名或模块名。例如：

package com.baidu.net;

package com.baidu.login;

package com.baidu.bus;

1.2  类的命名

(1)类的命名应该都是名词，第一个字母都要大写，其他每个单词的第一个字母都要大写。要用完整的单词，除非是被公认的单词缩写。例如：

class Container

class ShippingLine

(2)Android中特殊职责的类，如Activity，Service，BroadcastReceiver等，需要用“Activity”, “Service”, “Receiver”做为类命名的结尾。例如：

class IndexActivity

class DataPushService

class ScreenOnReceiver

1.3  接口的命名

接口的命名应该都是名词或形容词，第一个字母用I(即interface)来代替，都要大写，其他每个单词第一个字母都要大写。要用完整的单词，除非是被公认的单词缩写。例如：

public interface INetOperator

1.4  方法的命名

方法的命名应该都用动词或是惯用短语描述，第一个字母都要小写，其他每个单词第一个字母都要大写。例如：

run()

changeLocationTo()

getContainerId()

isAvailable()

hasLogin()

1.5  变量的命名

所有非静态成员变量名的第一个字母需要是m（即member，仿照google），其他每个单词的第一个字母都要大写。命名应尽量简单并且要有意义。变量名的选用应该易于记忆，能够指出其用途。尽量避免单个字符的变量名，除非是一次性的临时变量。

局部变量通常以第一个字母小写，其他单词首字母大写。

临时变量通常被取名为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。

变量名不应以下划线或美元符号开头，尽管这在语法上是允许的。下面是一些正确的变量命名例子：

类成员变量：

private int mWidth;

局部变量：

int numOfContainers

String containerId;

Date today;

1.6  常量命名

对于静态的final变量，在命名的时候每一个单词都要大写，单词之间用“\_”分开。例如：

final static int MIN\_WIDTH = 4;

final static int DEFAULT\_CONTAINER\_SIZE = 20;

1.7  文件的命名

java源程序文件以.java结尾，编译后的文件以.class结尾。例如：

Container.java

Container.class

1.8  推荐的命名

**1.8.1  Exception类名**

Exception类最好能用“Exception”做为类命名的结尾。例如：

DataNotFoundException

InvalidArgumentException

ServerErrorException

**1.8.2  抽象类名推荐**

抽象类最好能用“Abstract”做为类命名的开头。例如：

AbstractBeanDefinition

AbstractBeanFactory

**1.8.3  工厂类方法推荐**

工厂方法最好能把该方法做要创建的对象类型描述出来。例如：

public Container createContainer();

public Location newLocation();

**2.**  **Java文件组织(建议)**

一个文件由被空行分割而成的段落以及标识每个段落的可选注释共同组成。超过2000行的程序难以阅读，所以一个java程序文件中的代码行数不能超过2000行，除非有特殊原因。

Java源文件还遵循以下规则，这个规则规定了java程序段落的顺序：

- 开头注释

- 包和引入语句

- 类和接口声明

**3.**  **JAVA文件声明顺序（建议）**

类或接口应该按以下顺序声明：

            包的定义

            impot类（输入包的顺序、避免使用\*）

           输入包应该按照java.\*.\*，javax.\*.\*，org.\*.\* ，com.\*.\*的顺序import

           在import的时候不应该使用\* (例如: java.util.\*)

            类或接口的定义

            静态变量定义，按public，protected，private顺序

            实例变量定义，按public，protected，private顺序

            构造方法

            方法定义顺序按照public方法(类自己的方法)，实现接口的方法，重载的public方法，受保护方法，包作用域方法和私有方法。

建议：类中每个方法的代码行数不要超过200行。(不强制，如果有特殊需求)

            内部类的定义

**6.**  **JAVA文件格式缩进定义**（建议，可通过设置eclipse配置文件统一）

6.1  缩进尺寸

用4个空格做为缩进尺寸，将TAB设置为4个空格。(Eclipse中默认是4个字符)

6.2  行的尺寸

每行不要超过120个字符。(带注释除外)

6.3  行的格式定义

当一行表达式不能在一行内显示，请按下列顺序要求拆行：

            在“(”或“=”符号后拆行

            在“,”拆行

            在一个操作符后拆行

            把并发的拆行放到同一级别上的缩进

            如果在拆行中再次拆分的时候遇到“(”，应该新拆出来的行放在更远的一个缩进级别上

例如：methodWithLongName(

          expression1, expression2, expression3,

          expression4, expression5);

**7.**  **注释**

Java有两种注释方法。“/\* This is a comment \*/”或 “// This is a comment”

第一种应该被用到写JavaDoc上，并且都用“/\*\*”开头。

第二种适合于在做部分代码的注释，但只适合做非常短内容的注释。

**8.**  **声明**

8.1  变量声明

推荐每行声明一个变量，并加注释。例如：

int count;             // number of containers

int size;              // size of table

int count, size;        // 有问题！

int a = b = 1; // 有问题！

数组声明应该采用前缀方式。例如：

int[] table;

String[] args;

8.2  类或接口声明

            “{”和声明语句另起一行。（不强制，也可同一行）

            “}”符号应该独自占一行。

例如：

public class Manager extends Employee

{

    ...

}

public class ChiefExecutiveOfficer

        extends Manager  implements Person

{

    ...

}

8.3  方法声明

            “{”和声明语句独自占一行。（不强制，也可在同一行）

            “}”符号应该独自占一行。

例如：

public int myMethod(int i, int j)

{

    ...

}

或

public int myMethod(int i, int j) {

…

}

**9.**  **语句格式**

9.1  if, if-else, if-else-if-else 语句（括号不强制单独一行）

例如：

if (condition)

{

    statements;

}

if (condition)

{

    statements;

}

else

{

    statements;

}

if (condition)

{

    statements;

}

else if (*condition*)

{

    statements;

}

else if (*condition*)

{

    statements;

}

9.2  for 语句

例如：

for (initialization; condition; update)

{

    statements;

}

9.3  while语句

例如：

while (*condition*)

{

    statements;

}

9.4  do-while语句

例如：

do

{

    statements;

}

while (*condition*);

9.5  switch语句

例如：

switch (*condition*)

{

case ABC:

    statements;

case DEF:

    statements;

    break;

case XYZ:

    statements;

    break;

default:

    statements;

    break;

}

9.6  try-catch语句

例如：

try

{

    statements;

}

catch (ExceptionClass e)

{

    statements;

}

finally

{

    statements;

}

**10.**             **JavaDoc的格式定义**

10.1  类说明信息

定义文件描述，作者，版本。例如：

/\*\*

 \* Class description goes here.

 \*

 \* **@**author xxx

 \* **@**version 1.0

\* @date 2013-3-11

\* @modify {by} {date}：{description}

 \*/

修改别人的代码，以及代码的不同版本之间的较多或者重要的修改，最好都加上modify注释，便于后续维护，最后修改时间放在前面。

10.2  变量定义（不强制，public型的成员变量建议加上javadoc方便外部在调用时看到注释）

定义变量描述。例如：

/\*\*

 \* Comment for <code>${field}</code>

 \*/

10.4  方法定义

定义方法的描述，参数，返回值，参照文档，Exception。例如：

/\*\*

  \* description

  \*

  \* @param

  \* @return

  \* @exception

  \* @since （如果是框架性质的代码模块，需要加入此项版本注释）

\* @modify {by} {date}：{description}

  \*/

例：

/\*\*

\* 同步连接，不会加入队列，直接开启，获取到网络之后立刻返回

\* **@param** requestAdapter 请求适配器

\* **@return** 是否成功启动连接

\* @since 1.0

\* @modify wujian04 2013-3-11:描述2

\* @modify wujian04 2011-3-12:描述1

\*/

**public** **boolean** syncConnect(RequestAdapter requestAdapter)

{

return true;

}

**11.**             **Android编码规则**

**11.1：尽量使用接口而不是一个具体的类。**

例如有以下需求：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.ArrayList getObjectItems(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. List更合适：

|  |
| --- |
| **public** java.util. List getObjectItems(String sql){  ……  } |

这样即使更改实现，外部类也不必做相应更改。

**11.2: 对于可能发生异常的代码，都应在尽可能低的层次进行捕获。若低层不知如何处理，则将异常继续抛给高层处理，但最终必须得到处理。**

**11.3：防止下标越界，在使用数组下标之前先判断下标的合法性，比如数组类型，集合类型。**

如:

|  |
| --- |
| // 以下为不规范的写法：  String[] ary = getAry();  System.out.print(ary[2]);  // 应写为：  String[] ary = getAry();  **if** ((ary != **null**) && (ary.length > 2)){  System.out.print(ary[2]);  } |

**11.4：谨防数据类型越界。**

如:

|  |
| --- |
| **int** milsec = 25 \* 86400 \* 1000;  System.out.println(milsec);  // 打印结果是-2134967296 （超过int最大值范围2147483647） |

**11.5：确保释放数据库连接资源。**

说明：使用完Connection,Cursor后注意close(),如果使用连接池则使用后应及时归还给连接池，防止数据库连接资源被耗尽。

**11.6：确保释放网络连接、文件资源。**

说明：使用完Socket后注意close(),如果使用连接池则使用后应及时归还给连接池，及时关闭打开的文件，防止超出允许打开的最大文件句柄。

**10.7：循环编程：尽量选取for循环而不是while循环；如果使用while循环，最好能用iterator；while循环的终止条件需慎重，以避免死循环的可能。**

如:

|  |
| --- |
| /\*\* 不正确的用法：当interval取零或者负数时将可能出现死循环\* \*/  **while** (i <= max) {  i += interval;  } |

**11.8：字符串比较时，将常量置于equals之前，避免null异常。**

如:

**if** (Intent.ACTION\_SCREEN\_ON.equals(intent.getAction)) {

// do sth

}

**11.9：一般而言，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。**

如:

|  |
| --- |
| **if** (a == b && c == d) //!!!避免使用  **if** ((a == b) && (c == d)) //正确的使用方法 |

**11.10：对于简单的if-else语句，可以使用简洁的三元运算符“? :”。**

如:

|  |
| --- |
| **if** (condition) {  **return** x;  }  **return** y; |

可以写做：

|  |
| --- |
| **return** (condition ? x : y); |

**11.11：Android中撰写的XML文件根据嵌套关系，进行缩进。**

如:

**<LinearLayout>**

**<ImageView></ImageView>**

**<RelativeLayout>**

**<ImageButton></ImageButton>**

**</RelativeLayout>**

**</LinearLayout >**

**11.12：Android在写View的XML中如果遇到可复用的控件，尽量独立写一个XML，然后在其他的XML中用include引入进来，以提高复用性。**

**11.13：类的成员变量在分行时以功能类型进行划分,清晰明了。**

如:

public final static String KEY\_XXX = “key\_xxx”;

public final static String KEY\_YYY = “key\_yyy”;

private final static int MSG\_XXX = 1;

private final static int MSG\_YYY = 2;

private Button mButton = null;

private RelativeLayout mRelativeLayout;

private Handler mHandler = new Handler();

**11.14：项目/模块内部定义共有Action或Category字符串时按照”应用包名.常量字符串”形式进行定义。**

如:

public final static String ACTION\_OPEN\_BOOK = "com.baidu.bus.ACTION\_OPEN\_DATA";