

# 《编码规范》

2014 级 Java 移动智能设备方向-爆破大队

版本 2.0

文档提供：2014 级 Java 移动智能设备方向-爆破大队

文档说明：

本规范应用于项目实训-食趣项目开发中，参考《Java 编码规范》。

# 目录

|       |   |    |
|-------|---|----|
| 一、    | 类及方法命名.....   | 4  |
| 1.1   | 命名参考匈牙利命名规约.....  | 4  |
| 1.2   | 针对 Android 开发，变量名中缩写的含义.....  | 5  |
| 1.3   | XML 布局文件中，控件的 id 命名.....  | 5  |
| 1.4   | 在 XML 布局文件中的单位.....   | 5  |
| 二、    | 常量的定义.....  | 6  |
| 三、    | 注释.....   | 7  |
| 3.1   | 块注释(block Comments).....  | 7  |
| 3.2   | 单行注释(Single-Line Comments).....                                       | 7  |
| 3.3   | 行尾注释(End-Of-Line Comments).....                                       | 7  |
| 3.4   | 文档注释.....   | 8  |
| 3.5   | 类文件开始处.....   | 8  |
| 3.6   | 方法开始处.....  | 8  |
| 3.7   | 一般需要注释的内容.....  | 9  |
| 四、    | 表达式和语句.....   | 9  |
| 4.1   | if, if-else,if-elseif 语句(if, if-else, if else-elseif Statements)..... | 9  |
| 4.2   | for 语句(for Statements).....   | 10 |
| 4.3   | while 语句(while Statements).....                                       | 10 |
| 4.4   | do-while 语句(do-while Statements).....                                 | 10 |
| 4.5   | switch 语句(switch Statements).....                                     | 10 |
| 4.6   | try-catch 语句(try-catch Statements).....                               | 11 |
| 五、    | XML 文件命名规范.....   | 11 |
| 5.1   | 文件名命名规范.....  | 11 |
| 5.2   | xml 文件中的注释格式.....   | 12 |
| 5.2.1 | 文件开始时的注释.....   | 12 |
| 5.2.2 | 大的分块区域（如 RelativeLayout，LinearLayout）前的注释.....                        | 12 |
| 5.2.3 | 部件（如 Button、EditText）前的注释.....  | 12 |
| 5.3   | 代码中的缩进和对齐.....  | 13 |
| 5.3.1 | 组件.....   | 13 |
| 5.3.2 | 整体代码格式.....   | 13 |
| 六、    | Java 文件分包.....  | 16 |
| 七、    | Java 文件命名规范.....  | 16 |
| 7.1   | UI 类.....   | 16 |
| 7.2   | Adapter 类.....  | 16 |
| 7.3   | domain 类.....   | 16 |

## 卷、 类及方法命名

### 1.1 命名参考匈牙利命名规约

|      |   |   |
|------|---|---|
| 类，接口 | <ul style="list-style-type: none"><li>● 类的名字应该使用名词。</li><li>● 每个单词第一个字母应该大写。</li><li>● 避免使用单词的缩写，除非它的缩写已经广为人知，如 HTTP。</li></ul>   | <pre>Class Article;  Class ArticleType;  Interface Runner ;</pre>     |
| 方法   | <ul style="list-style-type: none"><li>● 第一个单词一般是动词。</li><li>● 第一个字母是小些，但是中间单词的第一个字母是大写。</li><li>● 如果方法返回一个成员变量的值，方法名一般为 get+成员变量名，如若返回的值是 bool 变量，一般以 is 作为前缀。</li><li>● 如果方法修改一个成员变量的值，方法名一般为：set + 成员变量名。</li></ul>   | <pre>getName();  setName(String mName);  isFirst();  setName();</pre> |
| 变量   | <ul style="list-style-type: none"><li>● 第一个字母小写，中间单词的第一个字母大写。</li><li>● 不要用_或&amp;作为第一个字母。</li><li>● 尽量使用短而且具有意义的单词。（变量最好不要写缩写）</li><li>● 单字符的变量名一般只用于生命期非常短暂的变量。i,j,k,m,n 一般用于 integers ; c,d,e 一般用于 characters。</li><li>● 如果变量是集合，则变量名应用复数。</li><li>● 类内部的属性变量一般加前缀 m 标识。</li></ul> <p>后面跟变量的属性，后继跟变量名，变量名采用驼峰命名法</p> | <pre>String myName;  int[] students;  mBtnTitle;  mTVTitle;</pre>     |

|    |                         |              |
|----|-------------------------|--------------|
| 常量 | • 所有常量名均全部大写，单词间以'_'隔开。 | int MAX_NUM; |
|----|-------------------------|--------------|

## 1.2 针对 **Android** 开发，变量名中缩写的含义

|        |                   |                |
|--------|-------------------|----------------|
| Llay : | 线性布局              | LinearLayout   |
| Rlay : | 相对布局              | RelativeLayout |
| Tlay : | 表格布局              | TableLayout    |
| Sv :   | 滚动                | ScrollView     |
|        |                   |                |
| Btn :  | 按钮                | Button         |
| Iv :   | 图像                | ImageView      |
| Tv :   | 标签                | TextView       |
| Et :   | 字段                | EditText       |
| Mp :   | 地图                | MapView        |
|        |                   |                |
| Cb :   | 复选框               | CheckBox       |
| Rg :   | 一组单选按钮            | RadioGroup     |
| Rb :   | 单选按钮              | RadioButton    |
|        |                   |                |
| Sp :   | 微调控件[下拉框] Spinner |                |
|        |                   |                |
| Gv :   | 网格控件              | GridView       |
| Lv :   | 列表控件              | ListView       |

## 1.3 XML 布局文件中，控件的 **id** 命名

针对每一个控件的 id。android:id=""

其中，首个单词为此文件的文件名，

第二个单词为此控件类型的缩写（如图像 ImageView 的缩写 lv），

第三个单词为此控件的名称。

如此图片为登陆界面 login.xml 文件中的图像型的 Logo 图标，则命名为 login\_lv\_logo。

## 1.4 在 XML 布局文件中的单位

|      |           |
|------|-----------|
| px : | 屏幕的像素点    |
| in : | 英寸        |
| mm : | 毫米        |
| pt : | 磅，1/72 英寸 |

dp : 一个基于 density 的抽象单位，如果一个 160dpi 的屏幕，1dp=1px  
dip : 等同于 dp  
sp : 同 dp 相似，但还会根据用户的字体大小偏好来缩放。

建议使用 sp 作为文本的单位，其它用 dip

## 式、 常量的定义

### 2.1 类中的常量

类中用到的一些常量，如字符串常量或整形常量等，要定义成全局变量。

如：`public static final int MENU_ABOUT = Menu.FIRST;` (定义 menu 菜单的选项 ID)

### 2.2 字符串常量

其他一些字符串常量可以定义在\res\values\string.xml 文件里，供国际化使用。

针对每一个字符串的<string name=" ">...</string> 标签。字符串名称 name，即""中的名称，首字母应小写，单词之间用下划线“\_”隔开。

如：`<string name="hello">Hello World, Activity01!</string>`  
`<string name="app_name">Examples_09_02</string>`

### 2.3 xml 文件中的控件尺寸参数

尺寸参数值可以定义在\res\values\dimens.xml 文件里，方便修改。

针对每一个字符串的<dimen name=" ">...</dimen> 标签。字符串名称 name，即""中的名称。首字母应小写，单词之间用下划线“\_”隔开。

其中，

首个单词为此控件所属 xml 文件的名称，

第二个单词为此控件对应的 id (包括类型缩写：如图像 ImageView 的缩写 Iv，和控件名称：如照片 photo)，

之后为此尺寸所属的控件属性 (由下划线隔开。如属性为 android:layout\_marginLeft，命名应为 layout\_margin\_left)。

如：

```
<dimen name="login_Iv_photo_layout_margin_left">80dip</dimen>
```

## 参、 注释

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 块注释(block Comments)

主要用来描述文件，类，方法，算法等。一般用在文档和方法的前面，也可以放在文档的任何地方。以'/\*'开头，'\*/'结尾。例：

```
.....
```

```
/*
```

```
* 注释内容
```

```
*/
```

```
.....
```

### 3.2 单行注释(Single-Line Comments)

主要用在方法内部，对代码，变量，流程等进行说明。与块注释 格式相似，但是整个注释占据一行。单行注释之前应该有一个空行。例：

```
if (condition) {
```

```
    /* 注释 */
```

```
    .....
```

```
}
```

### 3.3 行尾注释(End-Of-Line Comments)

与行注释功能相似，放在每行的最后，或者占据一行。以'//'开头。以下是所有三种风格的例子：

```

if (foo > 1) {
    // Do a double-flip. ①
    ...
} else {
    return false; // Explain why here. ②
}

//if (bar > 1) { ③
//
// // Do a triple-flip.
// ...
//}
//else {
// return false;
//}

```

### 3.4 文档注释

与块注释相似，但是可以被 javadoc 处理，生成 HTML 文件。以 `/**` 开头，`*/` 结尾。一个注释对应一个类、接口或成员。文档注释应位于声明之前，不能放在方法或程序块内。

```
/**
```

```
    注释
```

```
*/
```

```
public class Example { ...
```

### 3.5 类文件开始处

在每个类文件开始处给出整个类的文档注释。包括类名，作者，时间，版本，基本的描述，版权等信息，如：

```
/**
```

```
 *@FileName:
```

```
 *@Author:
```

```
 *@Date:
```

```
 *@Version:
```

```
 *@Description:
```

```
 *@Copyright:
```

```
*/
```

### 3.6 方法开始处

每个方法都应有一个注释标头。方法的注释标头可包含多个文字项，比如输入参数、返回值、原始作者、最后编辑该方法的程序员、上次修改日期、版权信息(这时采用/\*\* \*/这种形式，以便之后形成说明文档)

```
/**
 * #####
 * @param a
 *   @param b
 * @return
 */
public boolean setContent( String a,int b){

    return true;
}
```

### 3.7 一般需要注释的内容

1. 在每个if语句之前加上注释（采用一般注释方式即可，如// 或/\* \*/）
2. 在每个switch语句之前加上注释。
3. 在每个循环结构之前加上注释。

## 四、 表达式和语句（以编译器自动生成为主）

### 4.1 if, if-else,if-elseif 语句(if, if-else, if else-elseif

#### Statements)

任何情况下，if 语句都应该用"{"和"}"括起来。每一个 else 和 elseif 都应紧跟上一个"}"。格式如下：

```
if (condition) {
    statements;
}
```

```
if (condition) {
```



```
    statements;
} else {
    statements;
}

if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
```

## 4.2 for 语句(for Statements)

```
for (initialization; condition; update) {
    statements;
}
```

如果语句为空：

```
for (initialization; condition; update) ;
```

## 4.3 while 语句(while Statements)

一个 while 语句应该具有如下格式：

```
while (condition) {
    statements;
}
```

一个空的 while 语句应该具有如下格式：

```
while (condition);
```

## 4.4 do-while 语句(do-while Statements)

一个 do-while 语句应该具有如下格式：

```
do {
    statements;
} while (condition);
```

## 4.5 switch 语句(switch Statements)

一个 switch 语句应该具有如下格式：

```
switch (condition) {  
  case ABC:  
    statements;  
    /* falls through */  
  case DEF:  
    statements;  
    break;  
  case XYZ:  
    statements;  
    break;  
  default:  
    statements;  
    break;  
}
```

每个 switch 里都应包含 default 子语句。

每当一个 case 顺着往下执行时(没有 break 语句时)，通常应在 break 语句的位置添加注释。上面的示例代码，caseABC 中就包含注释/\* falls through \*/。

## 4.6 try-catch 语句(try-catch Statements)

一个 try-catch 语句应该具有如下格式：

```
try {  
  statements;  
} catch (ExceptionClass e) {  
  statements;  
}
```

一个 try-catch 语句后面，可能跟着一个 finally 语句，不论 try 代码块是否顺利执行完，它都会被执行。

```
try {  
  statements;  
} catch (ExceptionClass e) {  
  statements;  
} finally {
```

```
statements;  
}
```

## 伍、 XML 文件命名规范

### 5.1 文件名命名规范

1. res/layout 文件夹下的 xml：统一用小写和下划线“\_”组合命名，建议 xml 文件加前缀以便区分，如：对话框的 xml 配置文件为 dlg\_name.xml；

2. res/drawable 文件中的资源：统一用小写加下划线“\_”组合命名，同上，每个资源文件最好加前缀(如 btn)，以便区分。之后可附加此图片的编号。如：

btn\_default\_01.png, btn\_pressed\_01.png.

### 5.2 xml 文件中的注释格式

<!-- 注释内容 -->，注释内容中不能出现不成对的单个“>”符号。否则会导致注释失效。

#### 5.2.1 文件开始时的注释

每个文件开始时，对此 XML 文件的说明。包括文件，版本，时间，基本的描述，版权等信息。如：

```
<!--Filename:  
    PanvieoMainPage  
  
Version information:  
    version 1.0  
  
Date:  
    2010-05-20  
  
Description:  
    The Main Page XML  
  
Copyright notice:  
    Mark.xu-->
```

#### 5.2.2 大的分块区域（如 **RelativeLayout**，**LinearLayout**）前的注释

```
<!--  
The RelativeLayout contains login button,identification  
    button,helper button and logo.  
-->
```

### 5.2.3 部件（如 **Button**、**EditText**）前的注释

```
<!--  
Login button  
-->
```

## 5.3 代码中的缩进和对齐

### 5.3.1 组件

每个组件的开始标签和结束标签，应具有相同的缩进，左端对齐。如：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk  
    /res/android"  
    android:orientation="vertical"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent">  
  
    .....
```

```
</LinearLayout>
```

<Button/>等单个标签的组件的结束标志"/>"，不换行。如：

```
<Button  
    android:id="@+id/btn_identification"  
    android:layout_width="260px"  
    android:layout_height="180px"  
    android:layout_centerHorizontal="true"  
    android:layout_below="@id/btn_gotologinpage"  
    android:textStyle="bold"  
    android:background="@drawable/initidentify"  
    android:textSize="18px"/>
```

### 5.3.2 整体代码格式

```
<?xml version="1.0" encoding="utf-8"?>
<!--Filename:
    PanvieoMainPage

Version information:
    version 1.0

Date:
    2010-05-20

Description:
    The Main Page XML

Copyright notice :
    Mark.xu-->

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<!--
    The RelativeLayout contains login button,identification
    button,helper button and logo.
-->
    <RelativeLayout
        android:id="@+id/SongIdentification"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_centerInParent="true"
        android:layout_weight="1">
        <TextView
            android:id="@+id/tv_islogin"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#cccccc"
            android:text="您为游客, 请登录! "/>
        <Button
            android:id="@+id/btn_hint"
```

```

        android:layout_toRightOf="@id/tv_islogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#000000"
        android:textColor="#cccccc"/>
<!--
Login button
-->
<Button
    android:id="@+id/btn_gotologinpage"
    android:layout_below="@id/btn_hint"
    android:layout_alignTop="@id/btn_hint"
    android:layout_height="40px"
    android:layout_width="60px"
    android:layout_marginLeft="234px"
    android:layout_marginTop="16px"
    android:background="#000000"
    android:textColor="#cccccc"
    android:textSize="16px"
    android:text="登录"/>
<!--
Identification button
-->
<Button
    android:id="@+id/btn_identification"
    android:layout_width="260px"
    android:layout_height="180px"
    android:layout_centerHorizontal="true"
    android:layout_below="@id/btn_gotologinpage"
    android:textStyle="bold"
    android:background="@drawable/initidentify"
    android:textSize="18px"/>
<!--
Helper button
-->
<ImageButton
    android:id="@+id/imgbtn_helper"
    android:layout_height="18px"
    android:layout_width="18px"
    android:layout_below="@id/btn_identification"
    android:layout_marginLeft="270px"/>
<!--
Logo
-->

```

```

        <Button
            android:id="@+id/btn_logo"
            android:layout_height="50px"
            android:layout_width="180px"
            android:layout_centerInParent="true"
            android:layout_below="@id/imgbtn_helper"
            android:text="PanSongs"
            android:textSize="20px"
            android:gravity="center"/>
    </RelativeLayout>
    <!--
    The LinearLayout contain the four bottom button.
    They are history button,search button,recommend button
    and musicbox button.
    -->
    <LinearLayout
        android:id="@+id/bottombutton"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent"
        android:orientation="horizontal"
        android:layout_weight="5">
        <Gallery xmlns:android="http://schemas.
            android.com/apk/res/android"
            android:id="@+id/Gallery01"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:unselectedAlpha="1.2"/>
    </LinearLayout>

</LinearLayout>

```

## 六、 Java 文件分包

工程中的 Java 文件分包：

|                    |           |
|--------------------|-----------|
| com.onest.adapter: | Adapter 类 |
| com.onest.domain:  | 实体类，对象类   |
| com.onest.utils:   | 工具类       |
| com.onest.ui:      | 视图类       |

## 七、 Java 文件命名规范

## 7.1 UI 类

com.onest.ui 包中的类，作用是显示每个 UI 界面。每个类命名时为“界面名+Activity”（此界面名应和相应的 xml 文件相同）。如登陆界面的 xml 文件为 Login.xml，则对应的 Java 文件命名为：LoginActivity。

## 7.2 Adapter 类

com.onest.adapter 包中的适配器类。负责为选择部件提供数据源，也负责将单独的数据元素转换为显示在选择部件中的特定视图。（如 GridView 和 ListView 中的子条目）。由 UI 中的 Activity 调用。  
每个类命名时为“界面名+Adapter”。如 MainItemAdapter。

## 7.3 domain 类（Entity 类）

此包中主要为各个实体类。每个类要有 setter,getter 方法

例：一个文章类 Article

```
public class Article{

    private int title;

    public int getTitle() {
        return title;
    }

    public void setTitle(int title) {
        this.title = title;
    }
}
```