Android 开发规范

适用于中原集团驻京 Android 开发团队

作者：燕文强

Android 开发规范

前言

一个项目转手两个人，历时半年时间，如若没有开发规范，到头来你作为亲爹亲妈都不一定还认得出来，这货怎么长成了这个样子！因此，一个好的程序，不仅仅是能得出正确的运行结果，而且还应该在其内部内部保持清晰的代码逻辑和语义。

作为开发人员，我们需要清楚的认识到，我们辛辛苦苦撸出来的代码，并非是要人类去执行的，执行者是机器，它们对于执行代码的效率是非常高的。你需要看一年的代码，在机器执行起来，可能只要几秒。

高级开发工程师的目标不应该仅仅是实现业务，他们写出来的代码是一种美，一种艺术，除了巧妙的架构，代码看上去应该是一种美的享受，处处流露一种艺术感，且易读，易改。没有规矩不成方圆，这就要求我们在写代码时，有着严格的规范。

1. 换行的使用

1) 区分代码逻辑

逻辑较为紧密的代码放在一起，方便理解代码逻辑。

2) 参数换行

如有参数较长的方法，建议做参数换行。

2. 命名规范

1) Pascal 规则

a) 类名命名

b) 枚举命名

c) 接口命名

2) Camel 规则

a) 变量命名

b) 内部方法命名

3) 命名要能准确表达含义

杜绝拼音；不可以项目中即出现英文，又出现拼音，这会让其他人看不懂，看代码的人需要首先弄清楚这个名称是拼音还是英文，给理解代码的人，又增加了一层迷雾，多一层理解过程。同时代码看上去很 low，像刚毕业的同学写出来的。

4) 事件方法和其它方法命名

为清晰区分方法属于一个事件还是内部方法，我们需要规范事件命名方法和内部方。事件命名要用 “事件名称_方法类型”，以下为事件命名和内部方法命名示例：

```
private void button_click(){};
private void getPropLists(){};
```

```
private void initView(){
    lv_remind_list.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            // do something
        }
    });
    footView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // do something
        }
    });
    aet_remin_search.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {
            // do something
        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            // do something
        }

        @Override
        public void afterTextChanged(Editable s) {
            // do something
        }
    });

}
```

这样的编码完全可以正常实行，而且有大量开发这样做，但是编码风格会影响代码理解。因此，推荐一下这种方式：

```
@Override
protected void appendEvents() {
    atv_prospecting_fulltext.setOnClickListener((v) → {
            viewFullText_click();
    });
    abtn_prospecting_pass.setOnClickListener((v) → {
            prospectingPass_click();
    });
    abtn_prospecting_refuse.setOnClickListener((v) → {
            prospectingRefuse_click();
    });

    gv_prospecting_photo.setOnItemClickListener((parent, view, position, id) → {
            photosGridView_itemClick(position);
    });

    aRequest(prospectingPhotoDetailApi);
}

/**...*/
private void photosGridView_itemClick(int position) {...}

/**...*/
private void prospectingRefuse_click() {...}

/**...*/
private void prospectingPass_click() {...}

/**...*/
private void viewFullText_click() {...}
```

5) 类，抽象类，接口，方法

a) 类命名遵守 Pascal 规则。

b) 抽象类命名遵守 Pascal 规则，名称以“Abs”开头。

c) 接口命名遵守 Pascal 都以大写的“I”开头，设置方法要用“setXxxxxxListener()”为命名模版。

d) 以上命名长度以 30 为分水岭，不得以需要超过 30 的，我们可以使用缩写或专有名词，以上两种解决方法必须要基于方便理解的基础。

6)

3. 注释

1) 文件注释

以下为标准文件注释，不符合的，请修改自己的 File Template

```
/**
 * @author yanwq
 * @description 提醒人/部门
 * @date 2015/5/29.
```

2) 文档注释

以下为标准文档注释，设置快捷键从 Keymap 中查找“fix doc comment”来自定义。

```
/**
 * @param startDate
 * @param endDate
 * @return
 */
private String getPropLists(String startDate,String endDate){
    // do something...
}
```

3) 变量注释

4. 编码习惯及效率优化

1) 提出变量

不可出现天文数字，天文字符串，这样会将代码变得非常诡异。这种代码杜绝出现。

```
private void test(String name,int tag){
    if(name == "s"){
        // do something...
    }

    if(tag == 6){
        // do something...
    }
}
```

2) **for 循环避免不必要的劳动**

for 循环体，以下代码每次循环，都要执行 entity.getCount();而当 entity.getCount()越大，循环体要计算的次数就会越多，更要命的是，entity.getCount()方法的逻辑可能很复杂，很耗费时间，结果大家可想而知;

```
private void test(String name,int tag,Entity entity){

    for (int i = 0; i < entity.getCount(); i++) {

    }
}
```

3) **使用快捷键**

移除不需要的 import

展开 / 折叠代码块

变量重命名

提交 svn 前格式化

5. **删除已注释的代码**
6. **如有建议可随时联系我**