**小米●互联网一部安全中心 Code Review**

**一、什么是Code Review**

Code Review代码评审是指在软件开发过程中，对[源代码](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81" \t "_blank)的系统性[检查](https://baike.baidu.com/item/%E6%A3%80%E6%9F%A5/7025234" \t "_blank)。通常的目的是查找系统缺陷，保证软件总体质量和提高开发者自身水平。 Code Review是轻量级代码评审，相对于正式代码评审，轻量级代码评审所需要的各种成本要明显低的多，如果流程正确，它可以起到更加积极的效果。正因如此，轻量级代码评审经常性得被引入到软件开发过程中。

**二、作用&目的(预防为主,防治结合)**

1. 及早发现潜在缺陷与BUG，降低事故成本

2. 避免开发人员犯一些很常见，很普通的错误

3. 提高代码质量，提升自身水平

4. 促进团队内部知识技术共享,提高团队整体水平

5. 保证项目组人员的良好沟通

**三、Code Review 事务遵循原则**

1. 持开放学习的心态对待他人对自己的Code Review

2. Code commiter 需要自己先 Review 一遍 再commit,尽量提高大家的质量和效率

3. 每次 Commit 和 Review 代码 尽量 模块化,代码少量化.

4. 尽量遵循 Code Review checklist 清单规范约束

**四、 Code Review Check List**

**1.代码风格&规范**

字段、变量、参数、方法、类的命名是否真实反映它们所代表的事物  
方法体偏长，不易管理维护，可逐步抽取成小方法来减少代码长度

避免通过一个类的对象引用访问此类的静态变量或静态方法，无谓增加编译器解析成本，直接用类名来访问即可

较为复杂的业务代码逻辑方法&类需要有一定注释，注释与功能逻辑实现一定要一致。  
不要硬编码以及使用Magic Number  
代码重复，针对功能类似的方法，可添加一个参数加以区分复用

不要再循环内进行try catch ,除非有特殊需求  
不要有嵌套太多层的条件语句或者循环语句  
限制包，类，接口，方法和域的可访问性  
当一个功能或是方法不再需要时移除相关代码和资源

如果使用到了设计模式，建议在类名中体现出具体的模式：

public class ComponentFactory

public class BufferStrategy

**2.代码健壮性**

调用对象的方法时，需要注意该对象是否可能为空

数组类结构是否做了边界校验，是否会越界

校验输入，特别是外部参数或是网络下发数据，校验数据合法性

启动第三方的Activity 、 Service时，需要校验是否存在且可用，或是加上try…catch

不要在Activity的onCreate里调用PopupWindow的showAsLoaction方法，由于Activity还没被加载完，会报错

特别需要注意多线程情况下对象的使用同步

注意数据库的升、降级操作

数据库中尽量不要使用blob对象，如确有必要需要注意数据大小校验

当你重写equals时总是重写hashCode

**3.内存泄露&清理回收**

内部类，比如Handler、Listener、Callback是否是成static class：因为非静态内部类会持有外部类的引用。

在对应的场景合理使用软引用&弱引用

（1）SoftReference(软引用):当虚拟机内存不足时，将会回收它指向的对象；需要获取对象时，可以调用get方法。

（2）WeakReference(弱引用):随时可能会被垃圾回收器回收，不一定要等到虚拟机内存不足时才强制回收。要获取对象时，同样可以调用get方法。

（3）WeakReference一般用来防止内存泄漏，要保证内存被虚拟机回收，SoftReference多用作来实现缓存机制(cache);

假如子线程持有了Activity，要用弱引用来持有： 比如Request的Activity就应该用弱引用的形式，防止内存泄漏。

要求传入Activity作为参数的函数，是否可以改用getApplicationContext()来作为参数

页面退出时，是否完成必要的清理操作

（1）是否调用Handler的removeCallbacksAndMessages(null)来清空Handler里的消息；

（2）是否取消了还没完成的请求；

（3）在页面里注册的监听，是否反注册；

（4）假如自己用到观察者模式，是否反注册；

Cursor、文件流、网络连接等使用完是否关闭

Activity中使用的WebView，在destroy时是否调用了WebView的destory()方法

避免在高频回调如onDraw或是onTouch等方法中大量创建对象

创建大的Bitmap对象时考虑是否可用RGB\_565模式

使用完大的Bitmap时，是否调用了recycle()回收

**4.Handler相关**

使用View.post()是否会有问题： 因为在View处于detached状态期间，post()里面的Runnable是不会被执行的。只有在此View处于attached状态时才会被执行。如果想改Runnable每次肯定会被执行，那么应该是用Handler.post来替代

假如程序可能多次在同一个Handler里post同一个Runnable，每次post之前都应该先清空这个Handler中还没执行的该Runnable

尽量合理使用runOnUiThread 和 HandlerThread

**5.安全&隐私**

新注册的Activity、Service或Provider，若AndroidManifest.xml中exported属性为true，要考虑是否会引发安全性问题

禁止在日志输出打印不安全或者敏感的隐私数据信息等内容

网络数据传输注意数据加密保护，URI数据保护，防止敏感数据泄露

代码混淆，签名校验

不要在客户的存放登录密码（即使你加密了），最好采用token的形式

本地数据存储（xml，数据）重要信息需做好加密保护

**6.三方包配置编译**

build.gradle远程依赖第三方包时，版本号建议写死，不要使用+号，避免由于新版本的第三方包引入了新的问题

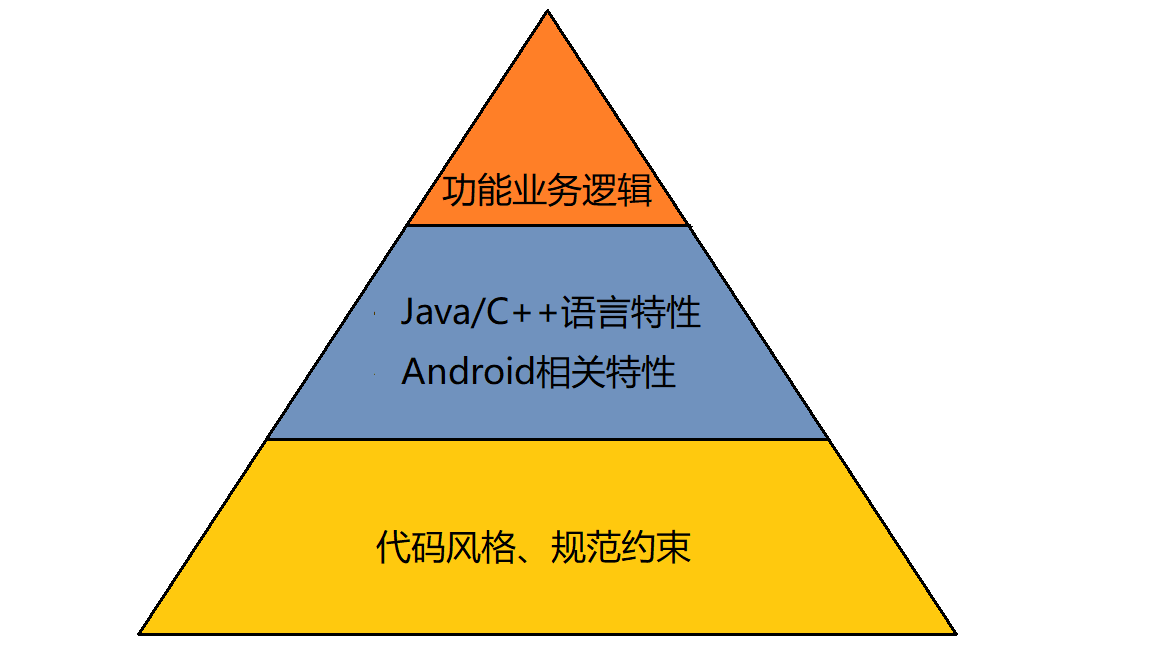
导入第三方工程时，记得把编码转换成自己工程当前是用的编码

调用第三方的包或者JDK的方法时，要跳进他们的源码，看要不要加 try-catch，否则可能会导致自己应用的崩溃

使用第三方包时，是否加上其混淆规则： 若漏掉加上第三方包的混淆规则，会导致第三方包不该混淆的代码被混淆。在Debug版本没有发现问题，但是Release版本就会出现问题

统应用添加so时，是否在固件对应的Android.mk文件上加入新增的so，否则系统可能编译不过

**五、寻找代码缺陷：分层法**



**六、推荐材料:**

1.阿里巴巴Java开发手册

2.编写高质量代码:改善Java程序的151个建议

3.代码整洁之道