前言

以后代码提交后会有人为的Review。事实证明，这样能有效提高自己的代码质量和功能的稳定性。下面列出参考的标准，希望大家以后每次提交代码前，都可以看下这份Review清单哈。此外，欢迎大家补充和修正

清理操作

1.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；
5. 假如用了RxJava的话，是否解除订阅；

2.数据库的游标是否已经关闭

这个点一般人都知道，出问题一般在于，没有考虑到多线程并发时的情况下，Cursor没有被释放。

3.打开过的文件流是否关闭

4.Android 3.0以下的版本，

使用完的Bitmap是否调用recycle()，否则会一直占用内存而Android 3.0及以上的版本不需要调用recycle()，因为这些版本的Bitmap全部放到虚拟机的堆内存中，让GC自动回收。

5.WebView使用完是否调用了其destory()函数

是否能进一步优化自己的代码

1. 保存在内存中的图片，是否做过压缩处理再保存在内存里否则可能由于图片质量太高，导致OOM
2. Intent传递的数据太大，会导致页面跳转过慢。太大的数据可以通过持久化的形式传递，例如读写文件
3. 频繁地操作同一个文件或者执行同一个数据库操作，是否考虑把它用静态变量或者局部变量的形式缓存在内存里。用空间换时间
4. 放在主页面的控件，是否可以考虑用ViewStub来优化启动速度

要小心第三方包

1. 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

2. 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
3. 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
4. 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
5. 系统应用添加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

```
    @lib/armeabi/libcommon.so
    @lib/armeabi/libabcdefg.so
```

注意要成对出现的地方

1. 系统的、自己写的，注册和反注册的方法，是否成对出现
2. 在生命周期的回调里，创建和销毁的代码是否对应起来比如：onCreate()里面创建了Adapter，那么对应Adapter的退出处理操作(比如清空Image缓存)，一般就要写在onDestory()，而不能写在onDestoryView()。
   类似的生命周期对应的代码有：onStart()、onStop();onCreate()、onDestory();onResume()、onPause();onCreateView()、onDestoryView()
3. 若ListView的item复用了，对Item里View的操作是否成对出现比如：switch (type) {

```
case ArticleListItem.TYPE_AD:
    mTitleView.setText(tencentAdBean.title);
    mGreenLabelView.setVisibility(VISIBLE);
    mRedLabelView.setText("");
    mRedLabelView.setVisibility(GONE);
    break;
case ArticleListItem.TYPE_ARTICLE:
    mTitleView.setText(mzAdBean.adData.getTitle());
    mGreenLabelView.setVisibility(GONE);
    mRedLabelView.setText("ABC");
    mRedLabelView.setVisibility(VISIBLE);
    break;
```

   比如以上对mTitleView、mGreenLabelView和mRedLabelView的操作，都是成对出现。否则ListView可能会由于Item复用，导致Item显示错乱问题

防内存泄漏

1. 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。
2. 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。

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

Handler相关

1. 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
2. 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如：

其他

1. 多思考某些情况下，某变量是否会为空而且在函数体内，处理参数前，必须加上判空语句
2. 回调函数是否处理好回调函数很容易出问题。比如网络请求的回调，需要判断此时的Aciivity等是否还存在，再进行调用。因为异步操作回来，Activity可能就消失不存在了。而且还要对一些可能被回收的变量进行判空。
3. 修改数据库后，是否把数据库的版本号+1
4. 启动第三方的Activity时，是否判断了该Intent能否被解析

```
Intent sendIntent = new Intent(mContext, Demo.class);
// 这种方式判断是否存在
if (sendIntent.resolveActivity(getPackageManager()) != null) {
startActivity(sendIntent);
}
```

若Activity不存在，会出现ActivityNotFoundException的异常

5. 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

```
<activity android:name="com.inkenka.DemoActivity"
     android:exported="true"/>
```

因为exported属性为true时，外部应用就可以直接调用起该Activity。可能导致的问题：1）若外部应用直接启动详情页，从而让某些验证页面直接被绕过2）若外部应用给该Activity传递乱七八糟的Intent，可能让该应用崩溃。也就是Android中的拒绝服务漏洞

6. 除数是否做了非0判断
7. 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

功能完成后，自测时的检查点

1. 思考某些情况下，某个变量是否会造成空指针问题
2. 把手机横屏，检查布局是否有Bug
3. 在不同分辨率的机型上，检查布局是否有Bug
4. 切换到英文等外文字体下，检查外文是否能完整显示
5. 从低版本升级上来，会不会有问题比如可能会出现数据库不兼容的问题
6. 按下Home再返回是否正常
7. 熄灭屏幕再打开是否正常
8. 切换成其它应用再切换回来会怎样
9. 利用手机的开发者选项中的 “调试GPU过度绘制” ，“GPU呈现模式分析” 和 “显示FPS和功耗” 功能，看自己的新功能是否会导致过度绘制、是否会掉帧
10. 测试看是否影响启动速度adb shell am start -W 包名/Activity
11. 对比看APK大小是否有增大
12. 跑1小时Monkey，测试其稳定性