# 小米便签代码质量分析和异味、bug处理

# 一、代码质量问题

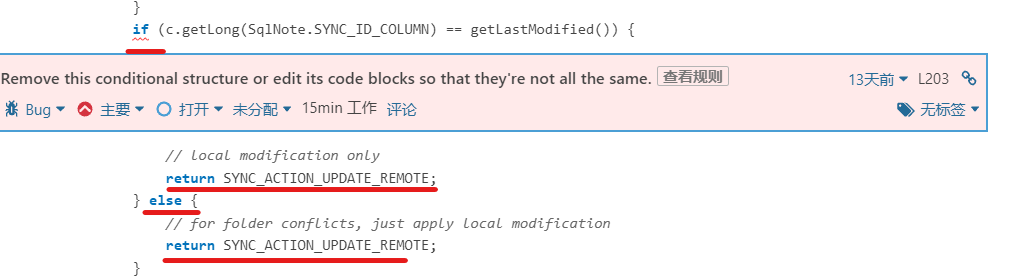
1、注释不清楚

在小米便签代码中，注释基本没有，导致在阅读一些较为复杂的方法实现时较为困难。对于变量代表含义，方法实现的算法都不够清晰，给阅读代码带来了不小挑战。



标注不清

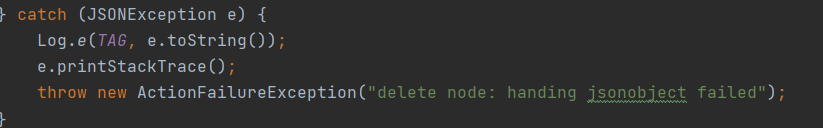
2、存在分支语句重复冗余



分支语句冗余

3、日志写入不够明确

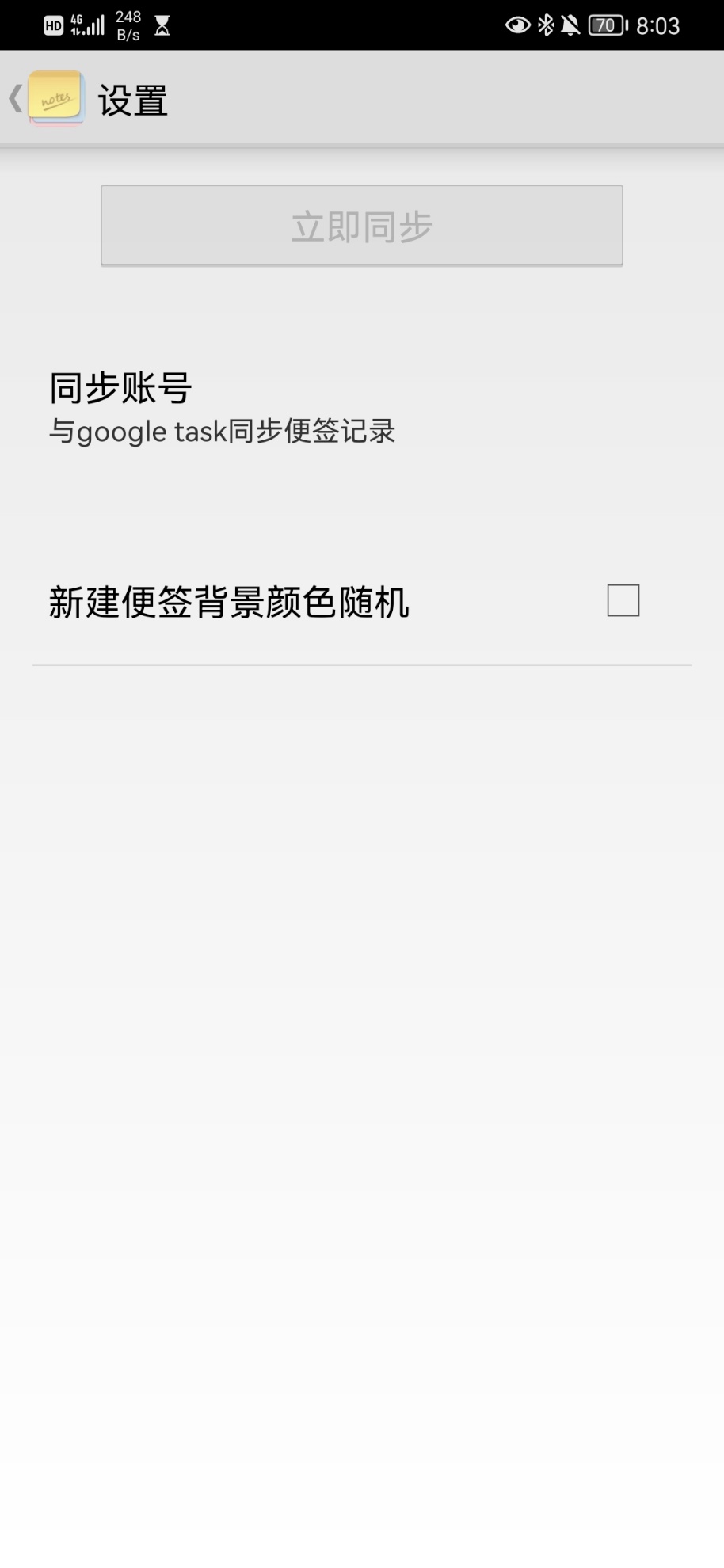
当发生删除异常时，会产生log，并写入日志信息，这是一个很好的编程规范，不过日志信息的写入应当尽可能简明扼要，并且准确可靠。



日志写入不明确

1. 存在功能重复

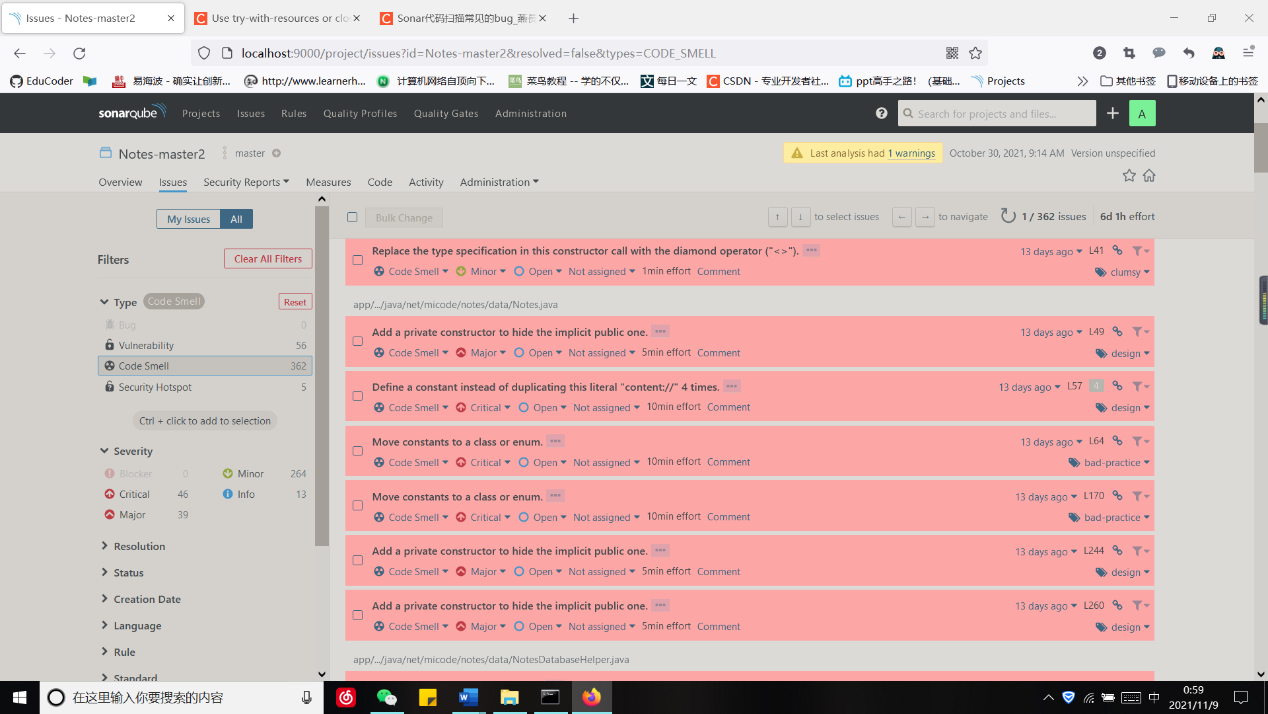
在小米便签中，设置选项的“设置”选项和“同步”选项实现的功能相同。

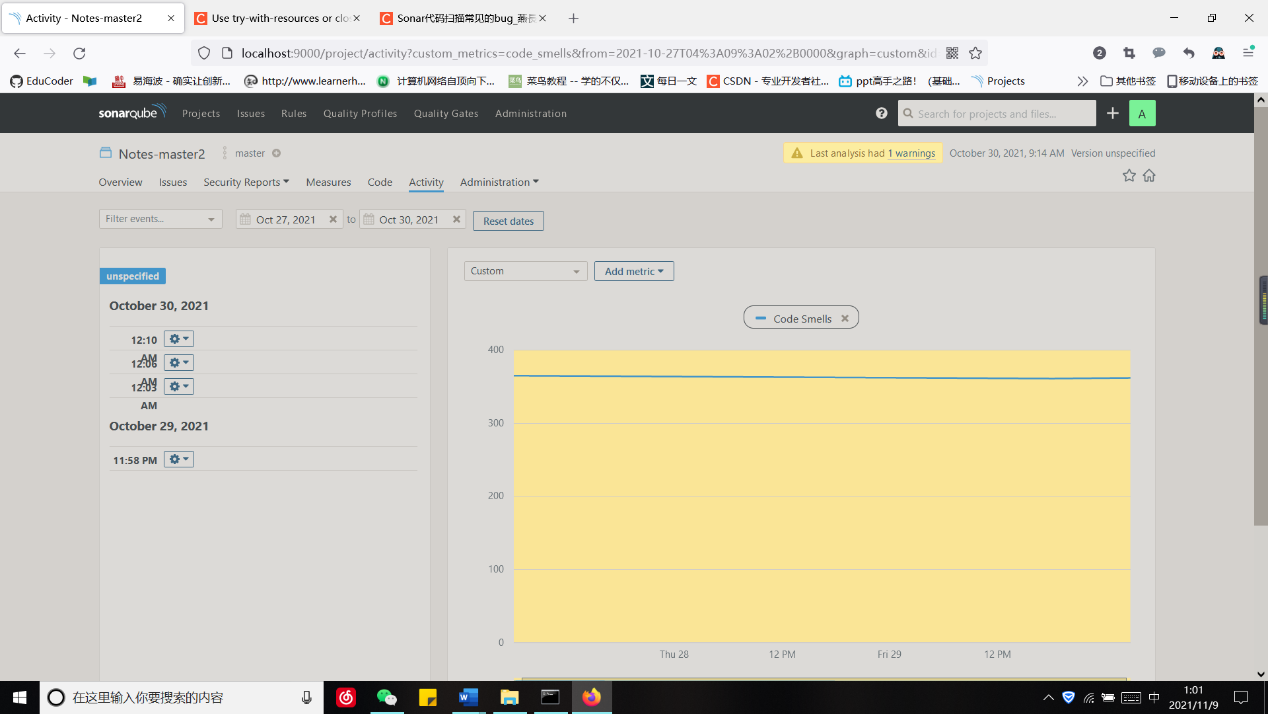


功能重

**二、异味和bug**

运行sonarqube软件，对小米便签代码进行评测，生成代码质量分析报告如下：





可以发现在每个包的代码中均存在较多的异味，异味总计362处。同时存在一些Bug和漏洞。

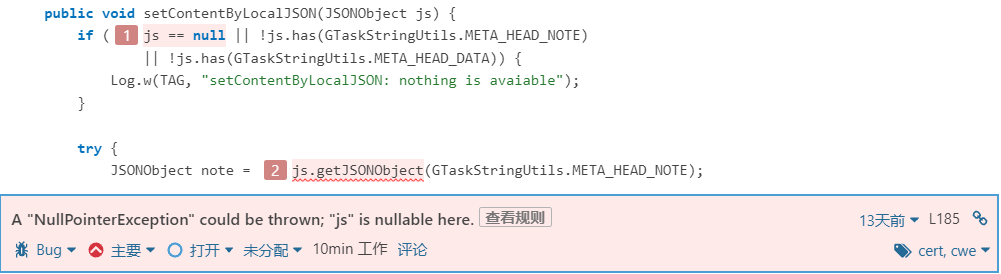
下面就主要问题进行分析：



Sonarqube报告中12处bug

**下面就部分bug进行分析：**

1. **返回值null导致报错**





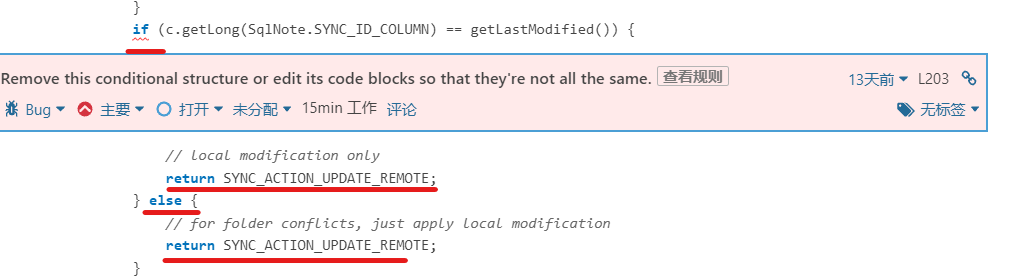
返回值null报错

跟踪到报错代码并进行分析，发现上面两个报错属于同一类型，均为对象值不为null但是方法的返回值却有可能为Null。

尽管在代码中通过if(js==null)进行判断，但是在getJSONObject的返回值依旧可能为null，继续运行，程序可能因为js的null值而停止运行。

解决方法：利用 JSR-305 中定义的 @CheckForNull 和@Nonnull 注释来理解哪些值可以为空，哪些值不能为空，除非在参数 toequals 上使用@Nonnull，根据约定，这应该始终与 null 一起工作。

**2、条件分支冗余**



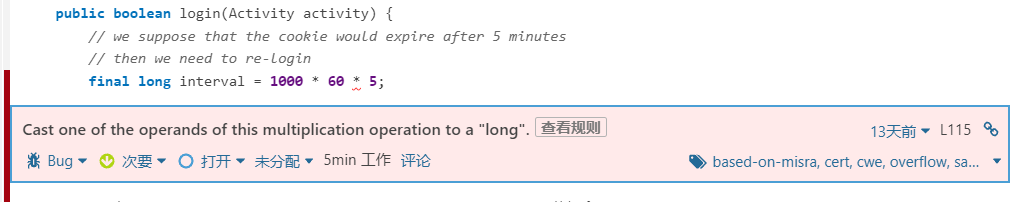
条件分支冗余

跟踪到该bug代码处，显而易见，条件分支语句的内容相同。

存在这样Bug，在读代码时会使读者一头雾水，不知道是逻辑错误还是在写条件分支语句时的执行指令错误。

解决方法：如果执行语句错误，那么更改if、else的执行语句，如果没有错误，则删去该分区。

**3、数字类型的运算中的溢出风险、强制类型转换问题**



数字类型错误

该bug也比较简单，在数字的计算、数字的赋值时的强制类型转换。

将乘法结果赋值给 long类型，可能在进行乘法运算时结果已经溢出，因此赋值时便会将溢出的值赋给long类型。

例如：

Final long interval = 1000000000\*1000000000\*100000000

解决方法：在这里，乘法结果并没有超过整形范围。如果要解决，可以采用如下思想：

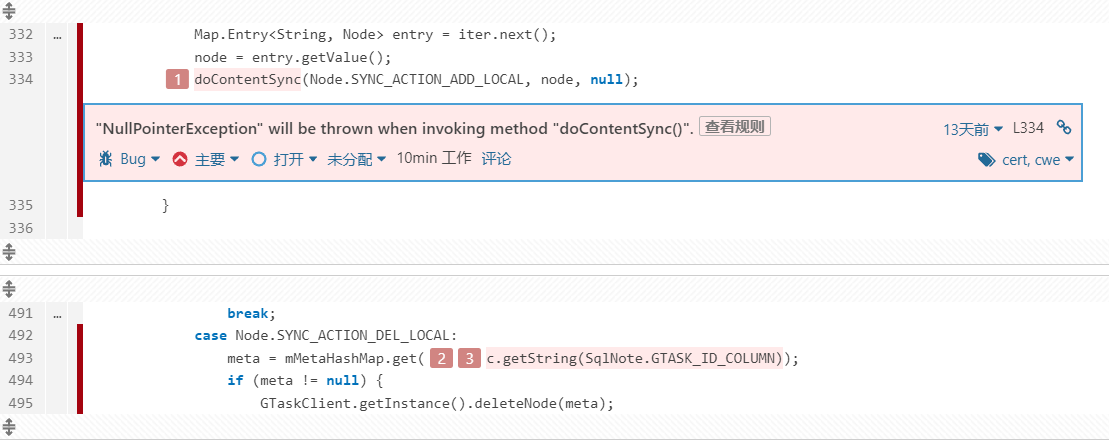
Final long a = 1000

Final long b = 60

Final long c =5

Final long interval = a \* b \* c

**4、参数null的方法传递了非null的参数**

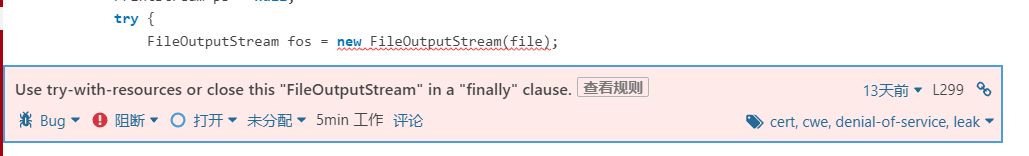


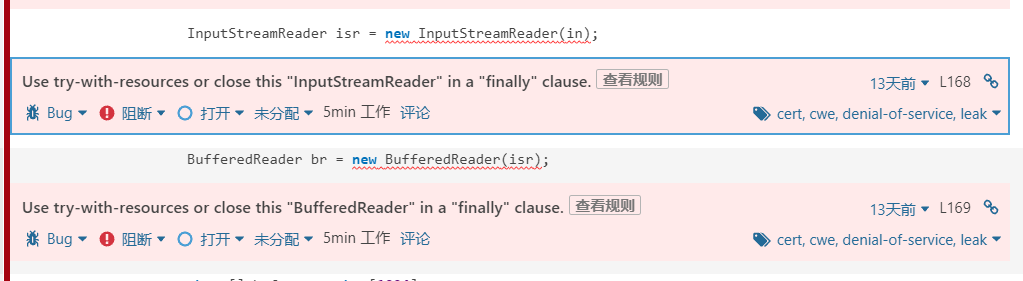


上述两个bug属于同一类型：方法的参数需要为null 而得到的参数值可能非null。

解决方法：在传递参数时加上判断是否为null语句，确保该方法得到的参数为null.

**5、文件流关闭问题**





上面三个属于同一类bug，在try catch之后没有关闭相应的FileOutputStream() InputStreamReader() BufferedReader()，也就时文件流没有正常关闭。

解决方法：对于后面两个bug，可以采取将两个文件放在不同try中处理。在catch之后加上相应判断函数，即使关闭。

如可以加上：

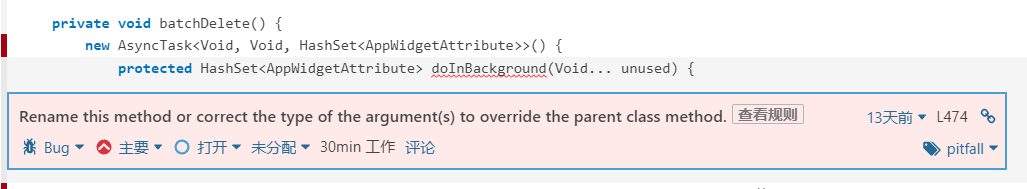
finally{

if(null != printwriter) {

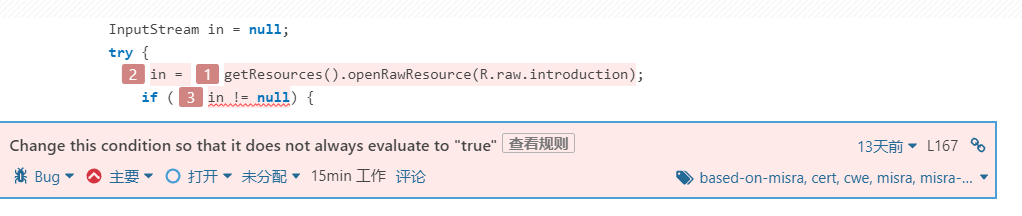
FileOutputStream.close();

}

}



重命名此方法或更正参数的类型以重写父类方法。对于权限修饰符，在重写时要注意。



在进行条件判断后，返回的结果一直为true。其中的openRawResource不能返回非null，因此in一直为非null值，正因此，if语句一直成立。

解决方法，因为openRawResourse是Android的方法，因此可以尝试将if（in!=null）{…..}删去。