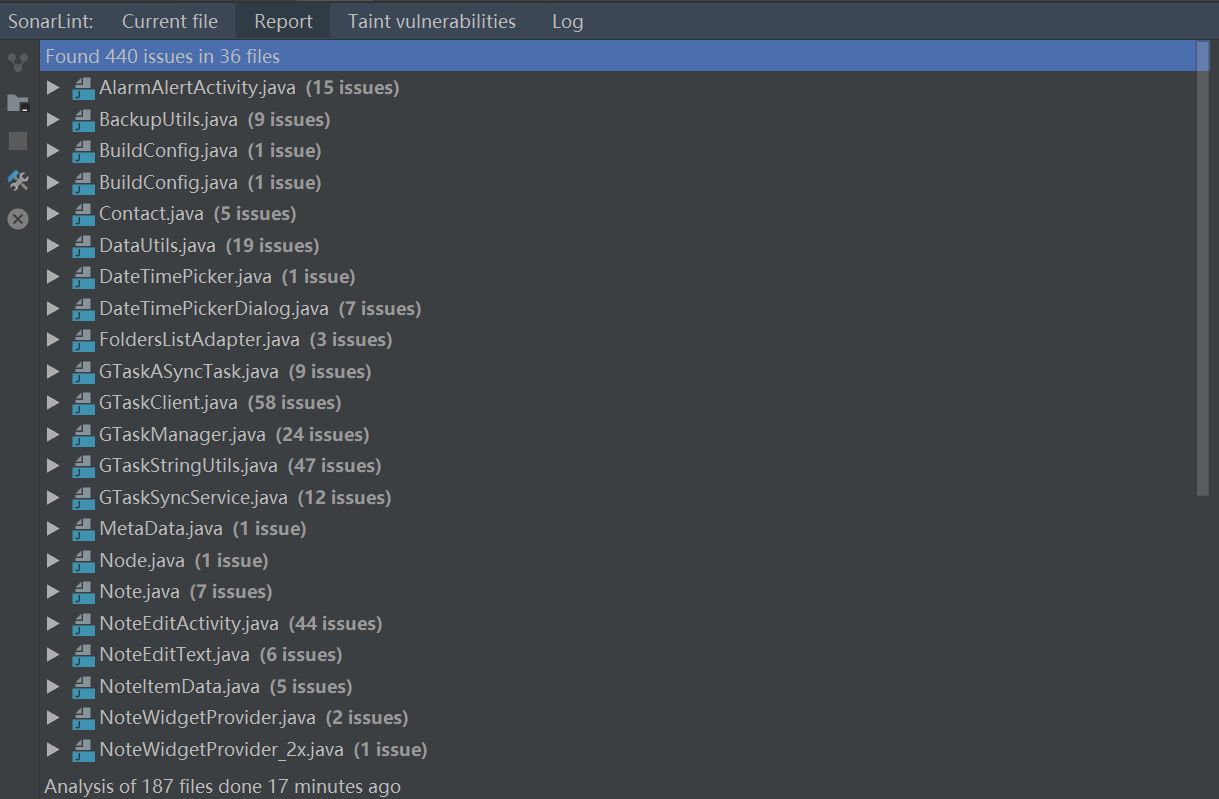
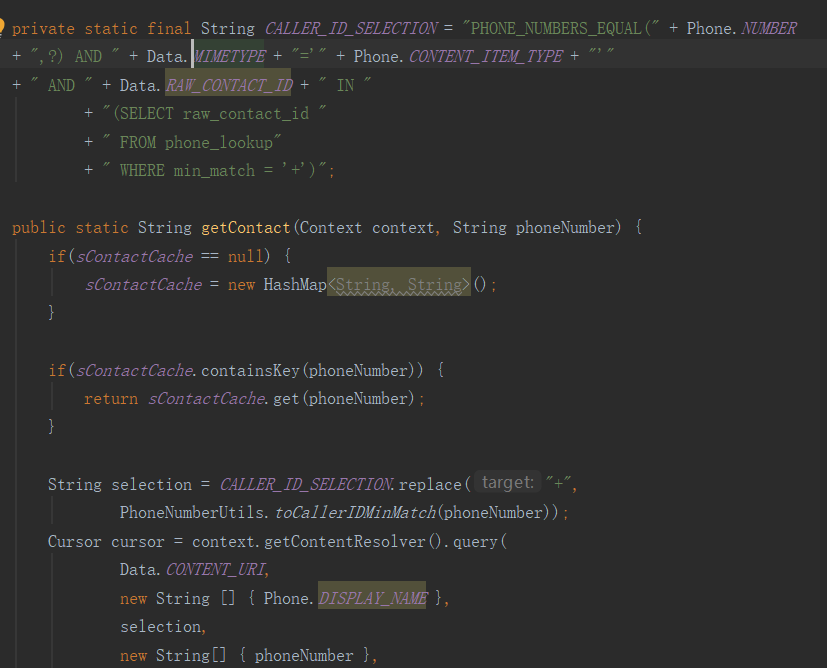
**自动质量分析：**

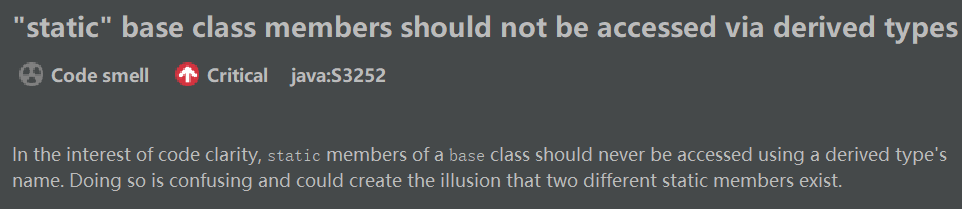


在36个文件中发现了一共440个问题。

(1)Contact.java:

Critical型问题：





问题：

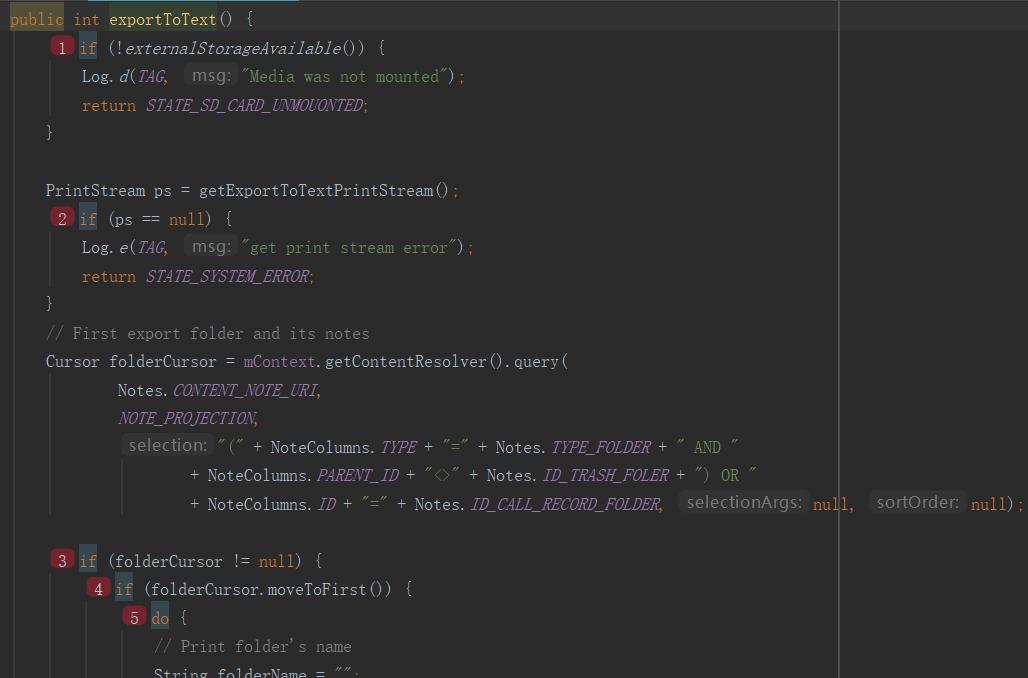
在这个函数里，使用子类的名称访问父类的静态成员。这样做会造成混淆，并且可能会导致存在两个不同的静态成员的错觉。

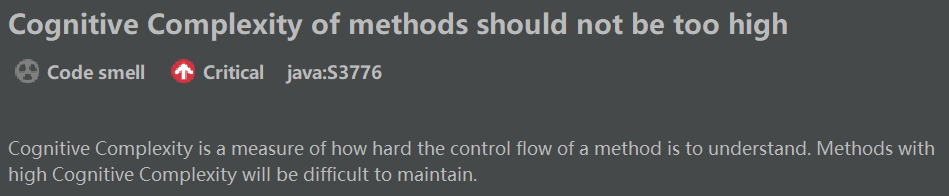
解决办法：

应该通过父类来调用

(2)BackupUtils.java:

Critical型问题：





问题：

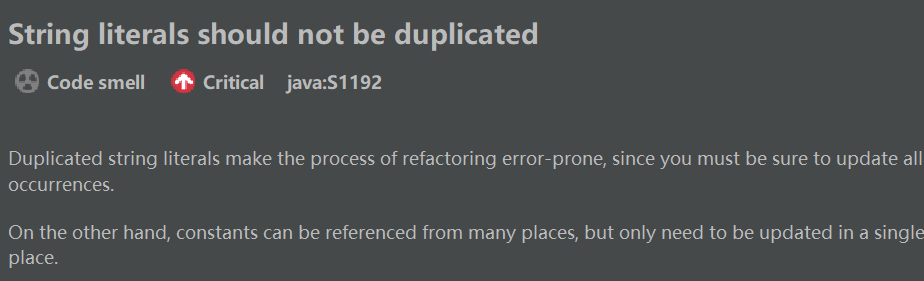
if判断分支过多，函数的认知复杂度太高，可读性较差。

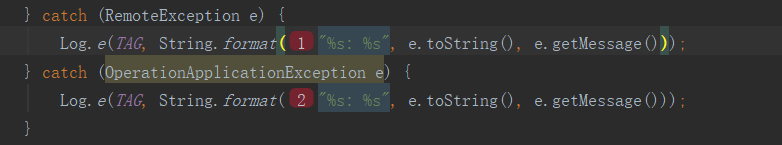
解决办法：

通过数学分析，找到最优解，减少if判断的个数，降低认知复杂度。

(3)DataUtils.java

Critical型问题：





问题：

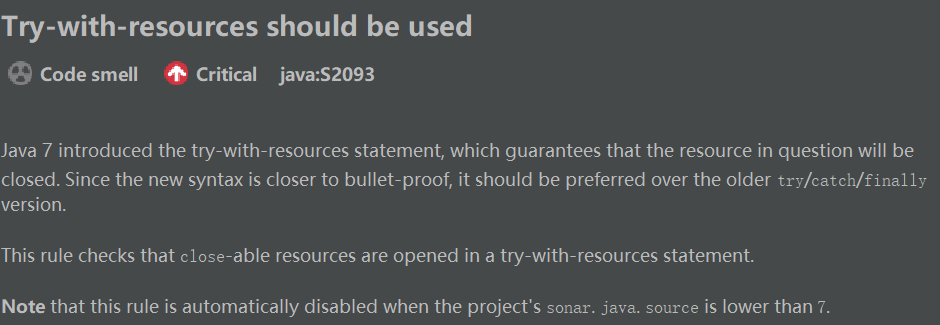
字符串重复出现，在修改时容易发生部分修改导致的数据一致性错误。

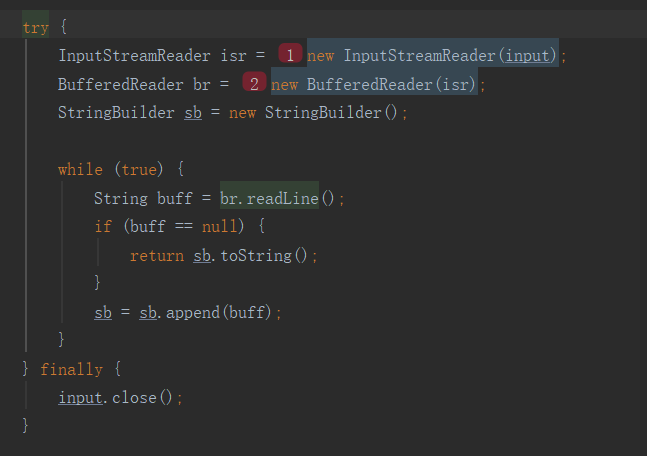
解决办法：

将其抽取定义为常量，修改时只需要在定义处更新

(4)DataUtils.java

Critical型问题：





问题：

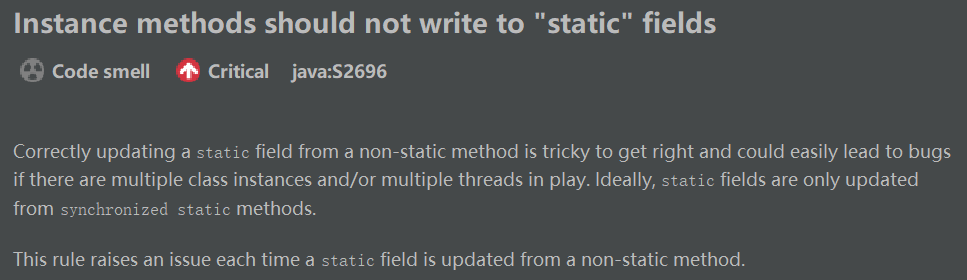
使用的是try-catch-finally版本，代码很冗余，且可能出现流通道用完后未关闭的情况。

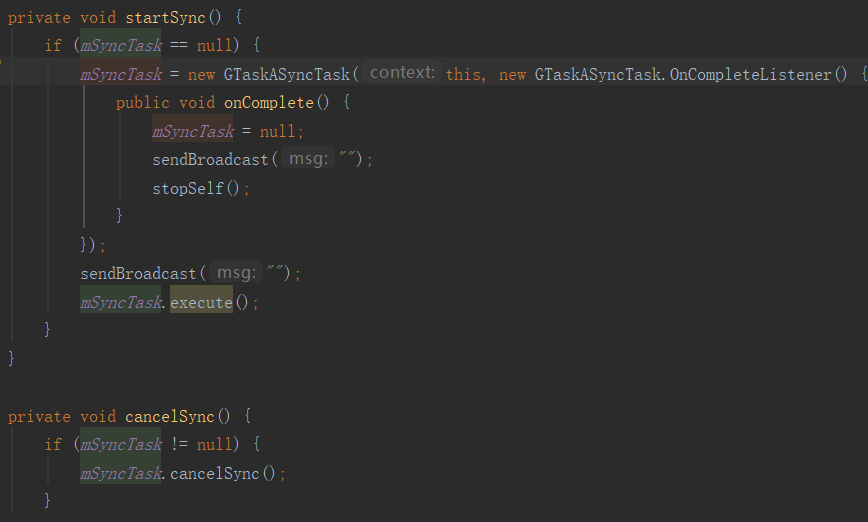
解决办法：

应使用Java7引入的try-with-resources，它可保证有问题的资源将被关闭

(5)GTaskSyncService.java

Critical型问题：





问题：

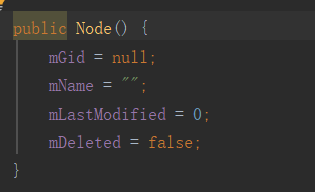
该函数从非静态方法更新了静态字段，而正确地从非静态方法更新静态字段是很棘手的, 如果在运行中有多个类实例和 / 或多个线程, 则可以很容易地导致 bug。

解决办法：

对静态属性包装一个静态方法

**手动质量分析：**

(1)Node.java：



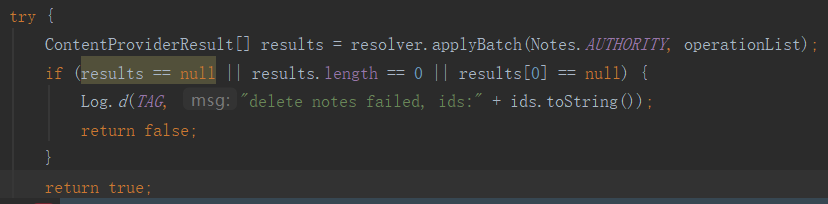
问题：

抽象类的构造函数不应该声明为public。

解决办法：

抽象类的构造函数只能在其子类的构造函数中调用，所以没有必要把它们公诸于众，通过受保护的修饰符修饰即可。

(2)DataUtils.java：



问题：布尔表达式不应该是毫无意义的。

解决办法：

如果boolean表达式的值是已定的，那么boolean表达式是没有必要的，可以移除。