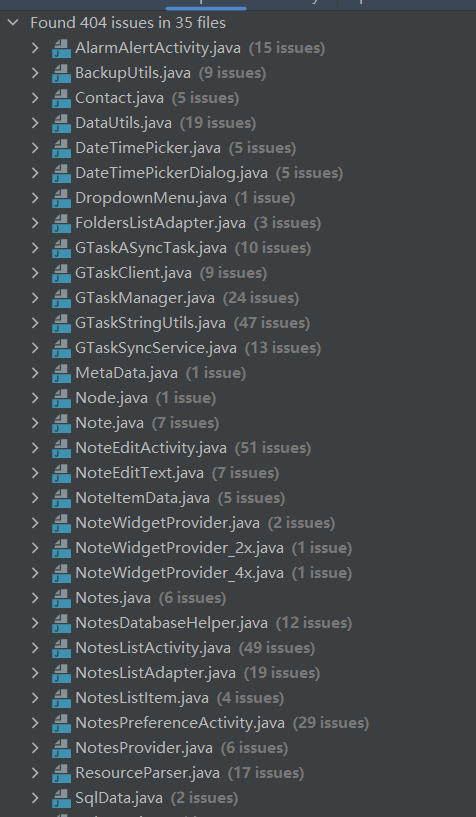
## 自动分析



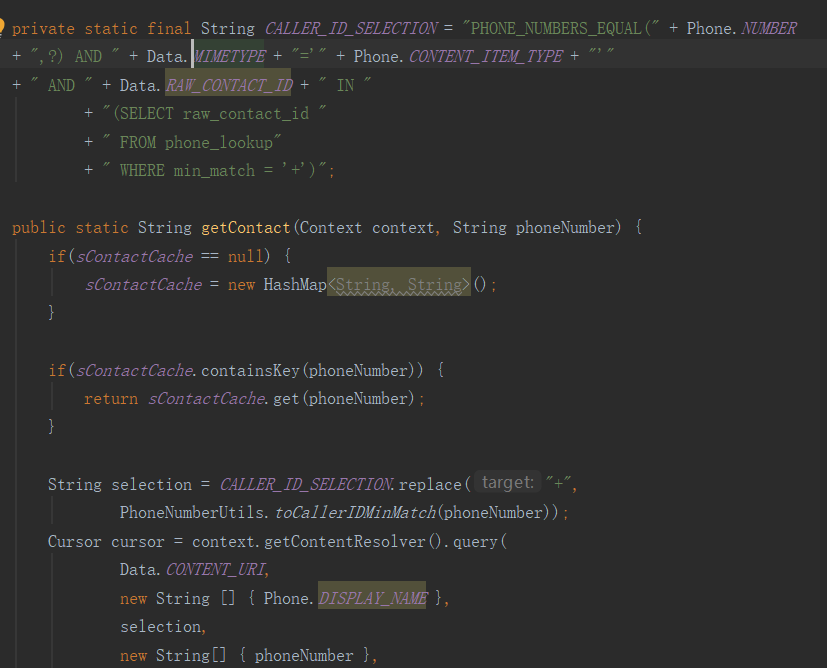
共发现404处问题

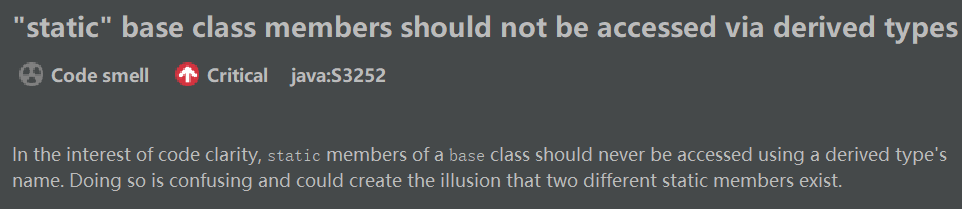
人工复查后可以发现，其中大部分问题是命名规范问题或因开发时间原因导致的部分函数已被弃用的问题。

在SonarQube中，代码缺陷按照从错误的严重程度从重到轻可以分为下面5个等级：  
1. Blocker: 极有可能影响应用程序表现的错误;  
2. Critical: 可能影响应用程序表现的错误和表示安全缺陷的问题;  
3. Major:严重影响开发者效率的质量缺陷:.  
4. Minor:轻微影响开发者效率的质量缺陷;.  
5. Info:不是错误或者质量缺陷。

### 维护前问题

#### (1)*Contact.java* Critical型问题





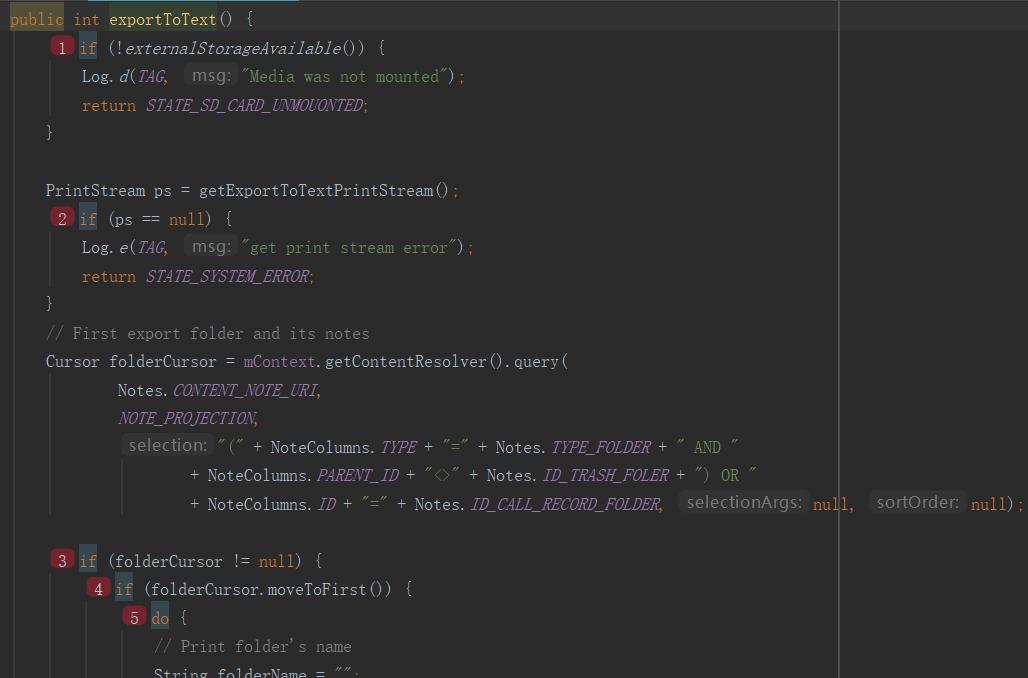
**问题：**

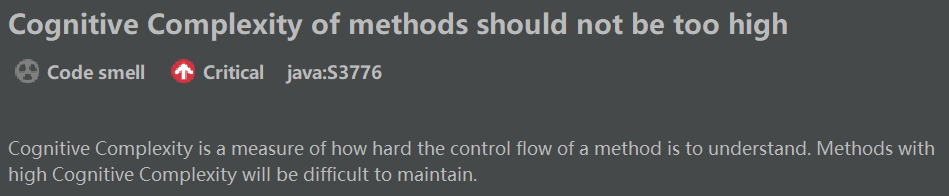
在这个函数里，使用子类的名称访问父类的静态成员。这样做会造成混淆，并且可能会导致存在两个不同的静态成员的错觉。

**解决办法：**

应该通过父类来调用

#### (2)*BackupUtils.java* Critical型问题





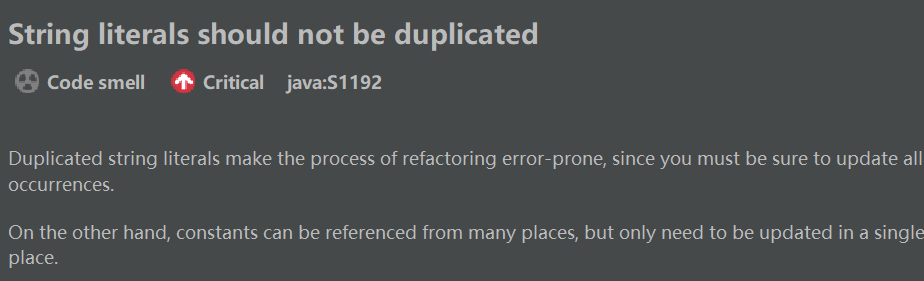
**问题：**

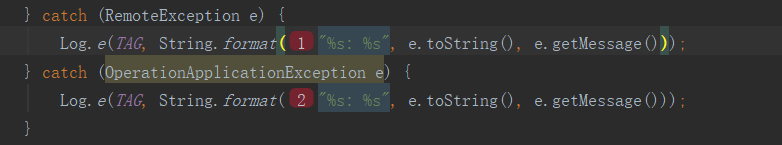
if判断分支过多，函数的认知复杂度太高，可读性较差。

**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





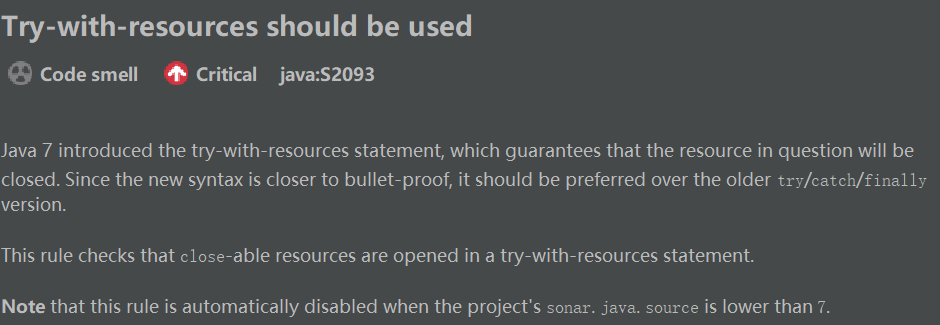
**问题：**

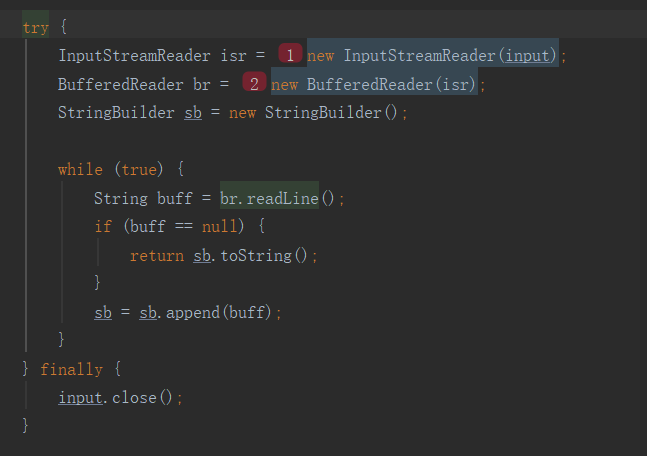
字符串重复出现，在修改时容易发生部分修改导致的数据一致性错误。

**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





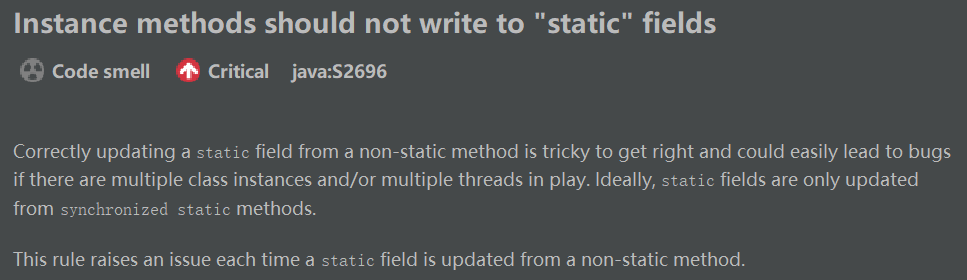
**问题：**

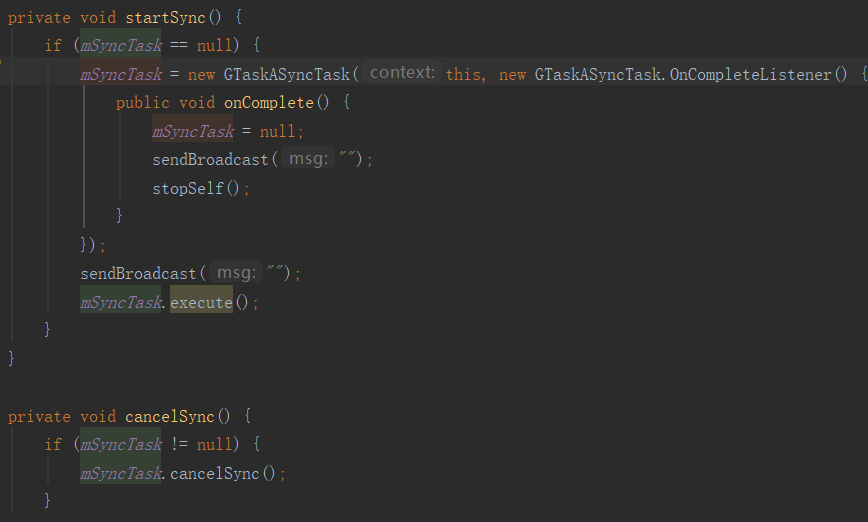
使用的是try-catch-finally版本，代码很冗余，且可能出现流通道用完后未关闭的情况。

**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：**

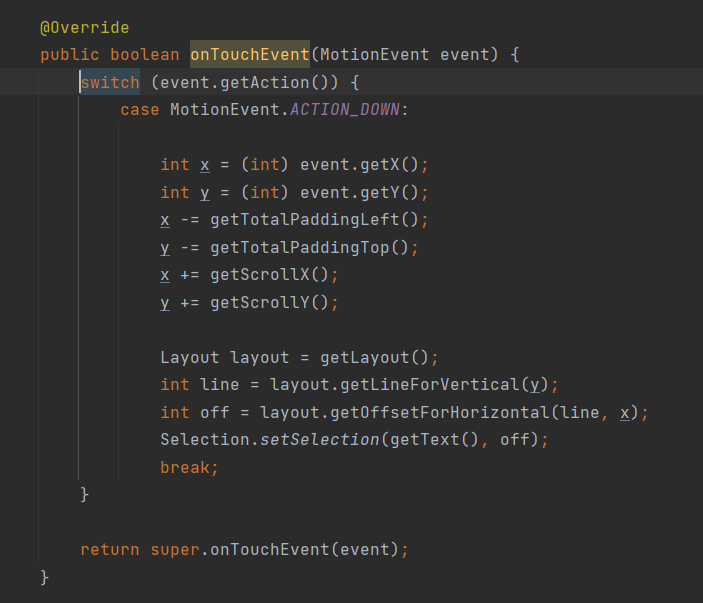
该函数从非静态方法更新了静态字段，而正确地从非静态方法更新静态字段是很棘手的, 如果在运行中有多个类实例和/或多个线程, 则可以很容易地导致=bug。

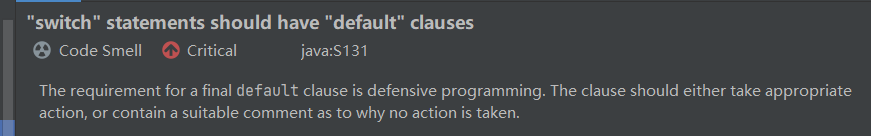
**解决办法：**

对静态属性包装一个静态方法

### 维护后问题

#### (1)*ui/NoteEditText.java* Critical型问题





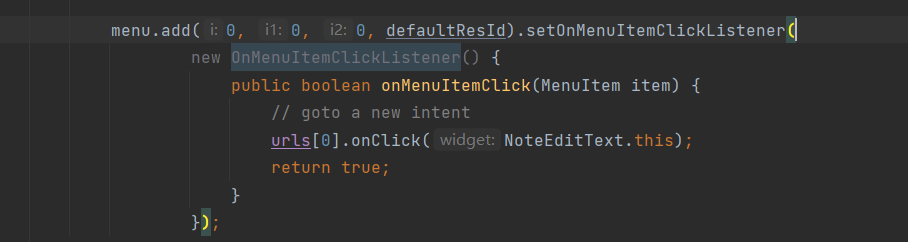
**问题：**

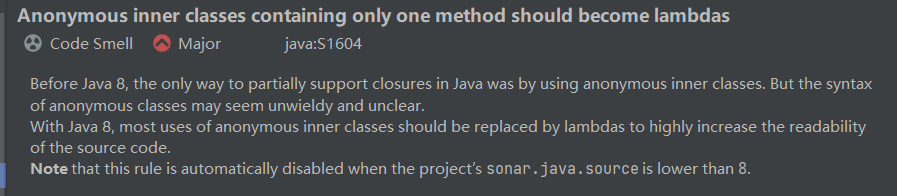
在这个函数里，调用了switch case选择分支，但是没有设置默认case（default情况），如果出现给定情况之外的情况，则容易出现问题。

**解决办法：**

应该设置一个default case的情况来抛出异常，如果出现对应问题，则可以知道问题来源。

#### (2)*ui/NoteEditText.java* Critical型问题





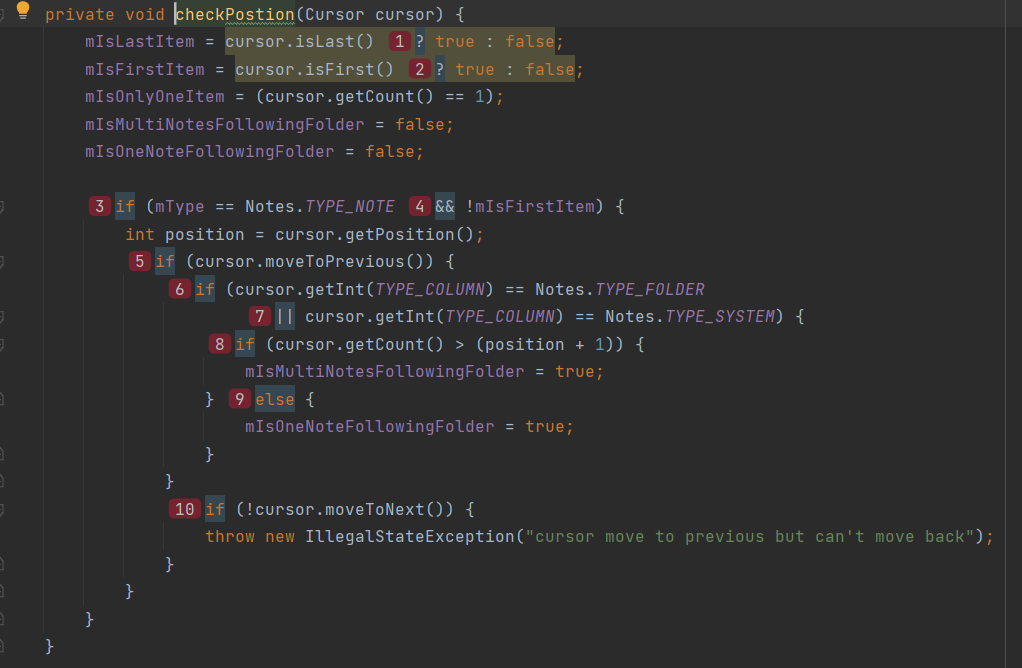
**问题：**

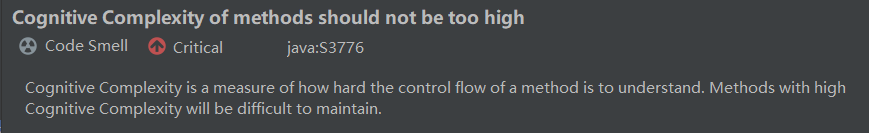
使用了匿名内部类（anonymous inner classes），但是在JAVA 8的后续版本中，具有更高可读性的匿名函数（lambdas）显然是更好之选。

**解决办法：**

使用匿名函数来代替此处的匿名类

#### (3)*ui/NoteEditText.java* Critical型问题





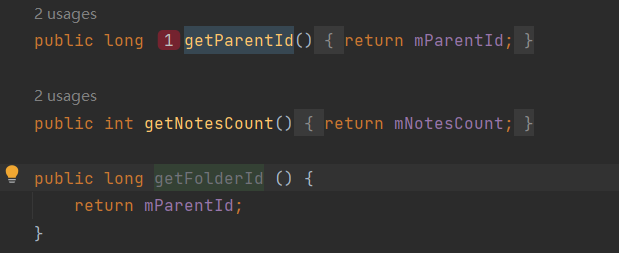
**问题：**

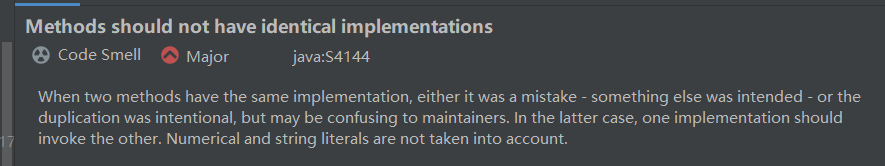
设置的if判断过多，函数的认知复杂度太高，可读性较差。

**解决办法：**

通过数学分析，来降低if判断的个数，提高判断质量，以降低认知复杂度。

#### (4)*ui/NoteEditText.java* Critical型问题





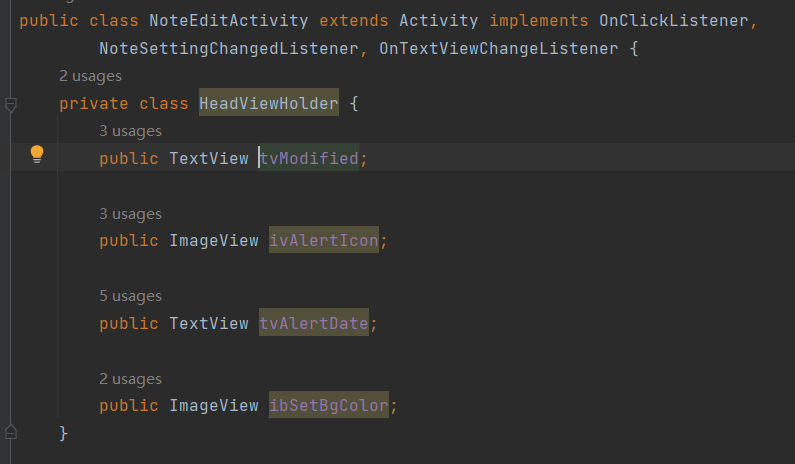
**问题：**

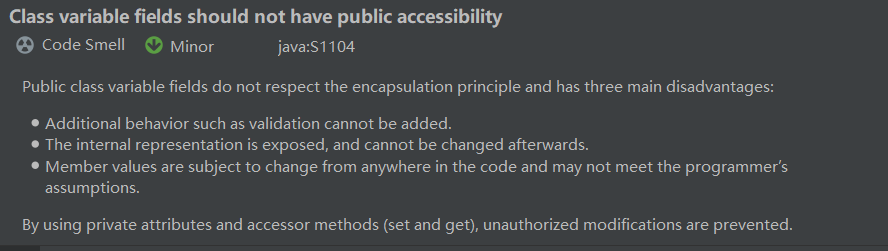
两个函数只有函数名是不同的，但是起到的作用是一致的，出现了冗余的现象。

**解决办法：**

在调用getFolderId的地方调用getParentId即可，利用注释在调用处说明调用的目的。

#### (5)*ui/NoteEditActivity.java* Minor型问题





**问题：**

使用了public变量字段，而没有使用私有声明。封装不是很好。

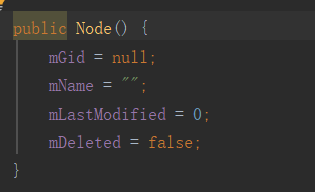
**解决办法：**

声明时声明为private字段，将变量字段封装起来。

## 手动分析

### 维护前问题

#### (1)*Node.java*



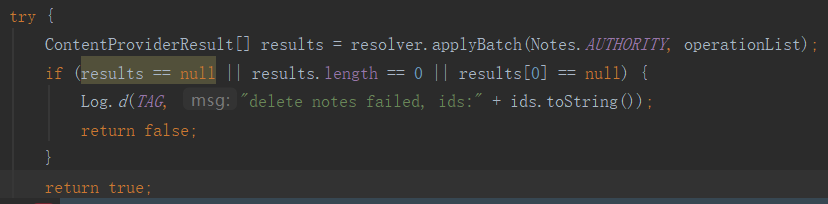
**问题：**

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。

#### (3)*ui/NoteEditActivity.java*



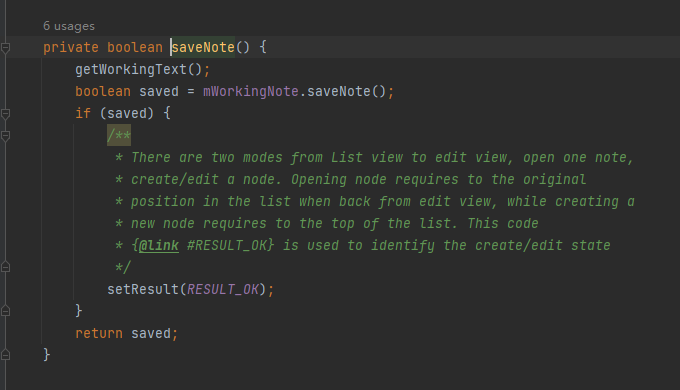
**问题：**

便签编辑时，当用户按下返回键自动保存，而没有手动保存功能，如果程序中途崩溃，很可能会使内容丢失。

**解决方案：**

在便签编辑ui添加手动保存按键，并实现手动保存功能，保存函数可以与此onBackPressed一致而去掉最后一步。

#### (4)*ui/NoteEditActivity.java*



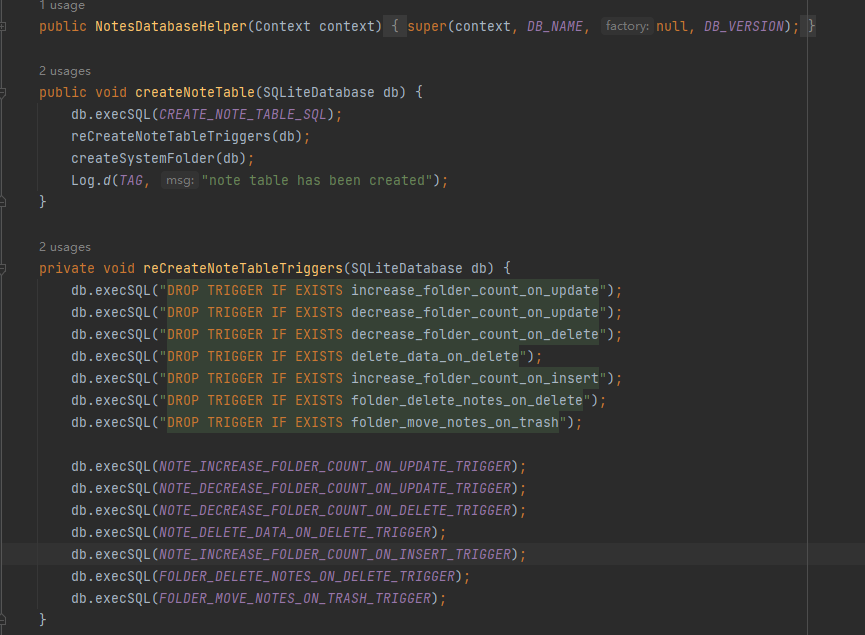
**问题：**

saveNote函数通过获取便签界面文本框的内容进行写入保存，这样会加大实现完全加密便签的难度。即加密的情况下仍有短暂的时间会在文本框显示便签内容。

**解决方案：**

可以修改WorkingNote，使其保存一个是否允许访问的变量，如果不允许访问则不对数据库进行修改，但经过评估该改进方法实现工作量太大，需要修改许多文件有关于WorkingNote的调用，故不予考虑。

#### (5)*data/NotesDatabaseHelper.java*



**问题：**

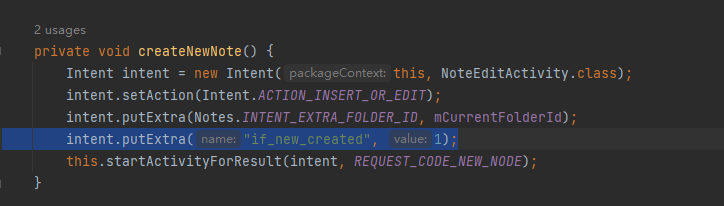
SQLite数据库并未加密（也可能是我没找到），数据库安全性不足。而且数据库丢失会造成很大程度的损失。

**解决方案：**

1.使用SQLCipher对SQLite进行加密操作，

2.在云端创建数据库，并支持用户将便签内容上传至云端。

#### (6)*ui/NotesListActivity.java*



**问题：**

Activity传值不够完备，在NoteEditActivity中不能有效区分该便签是新建的还是已经有的，导致在添加功能的时候出现bug。

**解决方案：**

添加“if\_new\_created字段进行传值”。

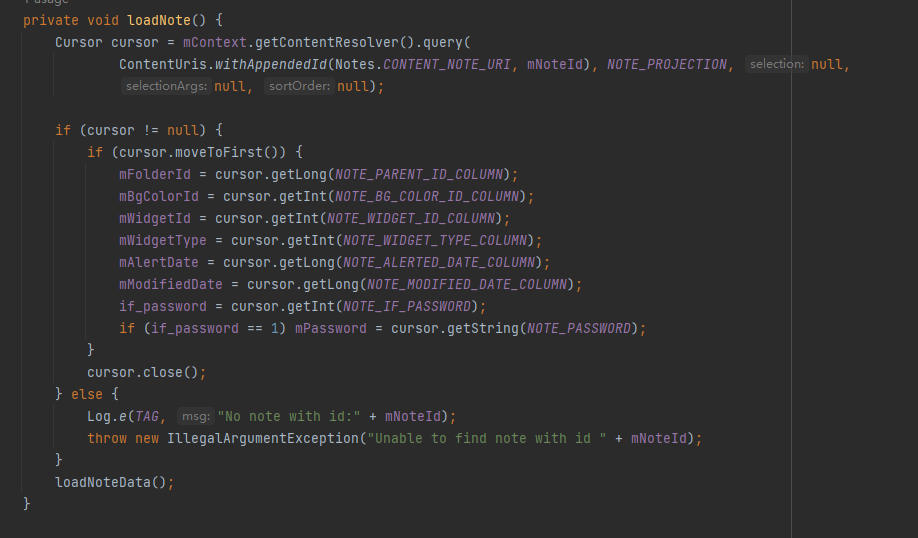
**问题：**

startActivityForResult方法已经过时

**解决方案：**

使用registerForActivityResult。

#### (7)*model/WorkingNote.java*



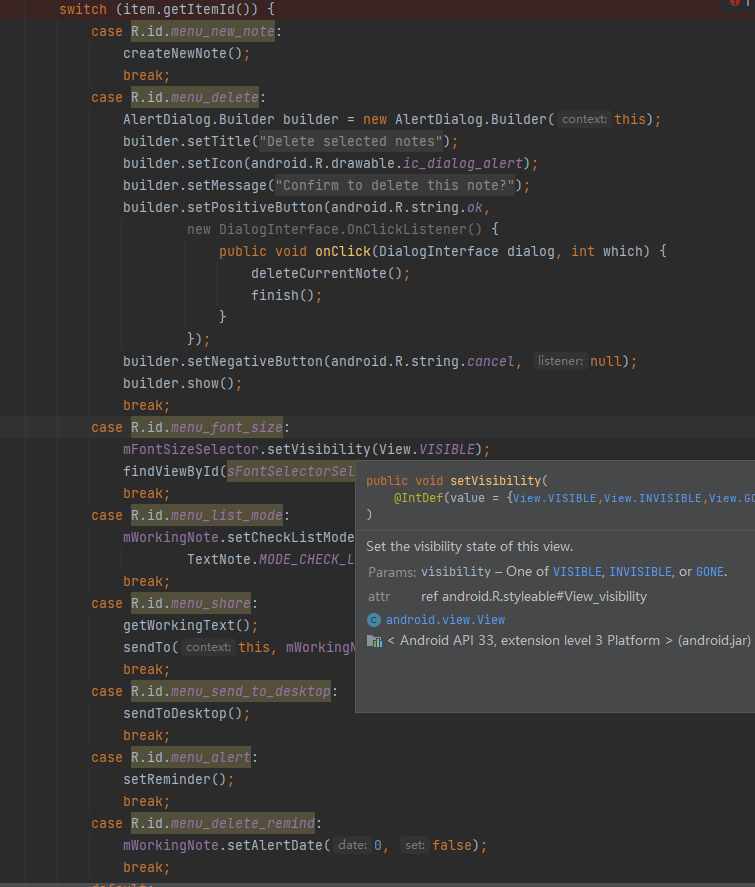
**问题：**

使用函数链进行主操作，可读性差

**解决方案：**

将所有函数在onCreate函数中调用，这样可读性好，便于观察Activity整体流程

#### (8)*ui/NoteEditActivity.java*



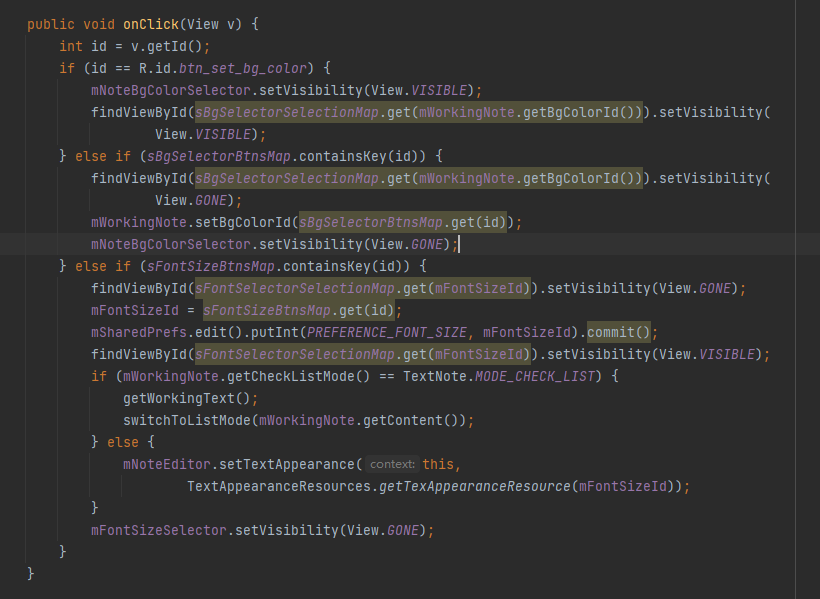
**问题：**

并未找到下拉菜单功能（推测是新版本的Android删除了某些固有组件）

**解决方案：**

原计划将设置密码与解除密码操作置于下拉菜单中，由于未找到下拉菜单，所以在便签界面增加按钮作为替代。

#### (9)*ui/NoteEditActivity.java*



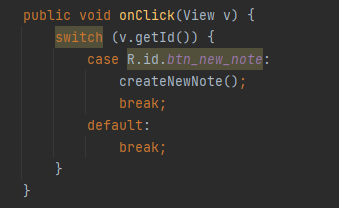
**问题：**

Onclick函数将所有组件集中在一起进行处理

**解决方案：**

对每个组件分别进行注册点击响应函数代码可能会更清晰简洁。

#### (10)*ui/NotesListActivity.java*



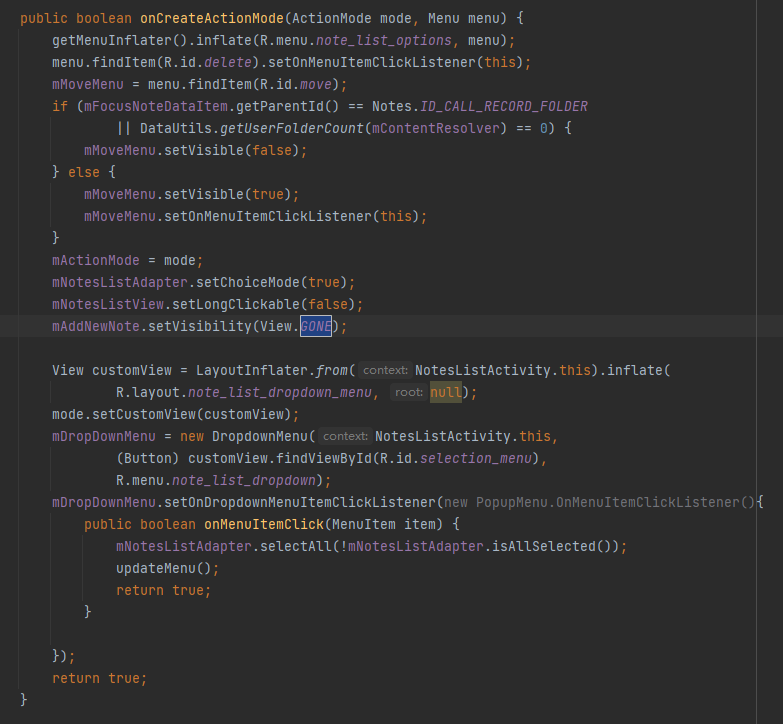
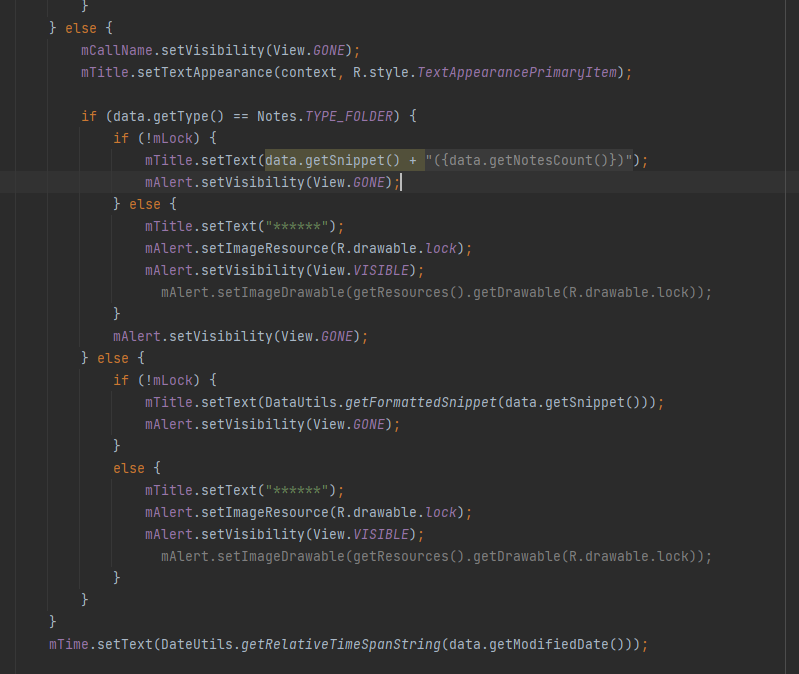
**问题：**

这里的switch只有一种可能

**解决方案：**

去掉switch。

#### (11)*ui/NotesListItem.java*



问题：

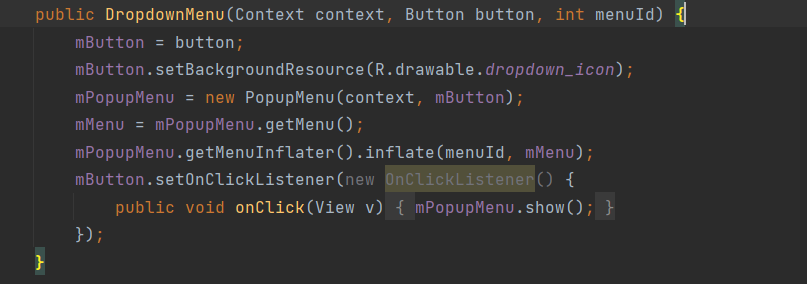
对同一个视图反复的判断并设置可视化，逻辑混乱，代码维护较复杂。

解决方案：

可以将不同的图标设为不同的视图，分别进行判断。

### 维护后问题

#### (1)*ui/DropdownMenu.java*



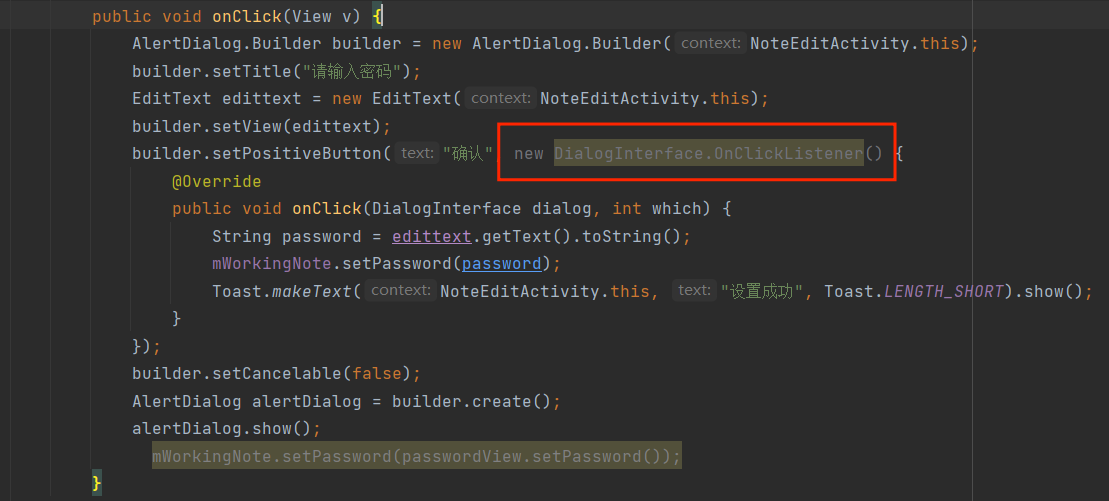
**问题：**

使用了匿名内部类，而没有使用可读性更好的匿名函数。

**解决办法：**

使用匿名函数代替匿名内部类。

#### (2)*ui/NoteEditActivity.java*



**问题：**

没有注释，可读性较差。且使用了匿名内部类，没有使用匿名函数。

**解决办法：**

对于红框内的部分，使用匿名函数类加以代替。

同时使用注释，向读者解释，这是在进行密码的设置。