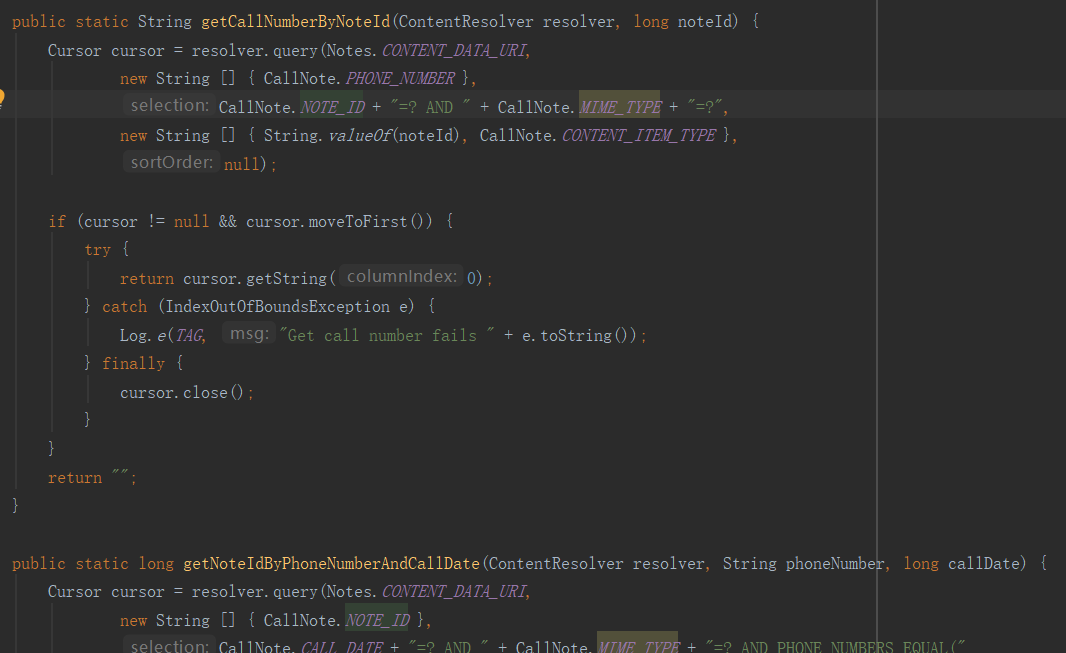
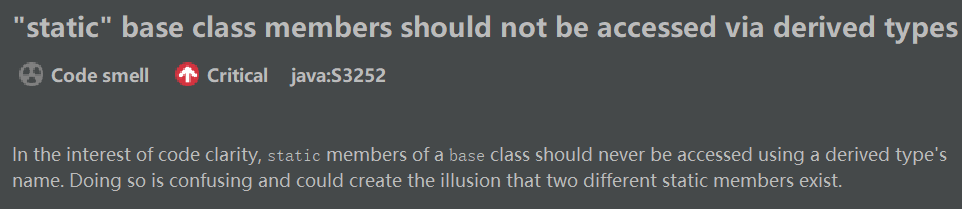
5处较严重的问题、分析及解决方案：

(1)DataUtils.java:

Critical型问题：



问题：

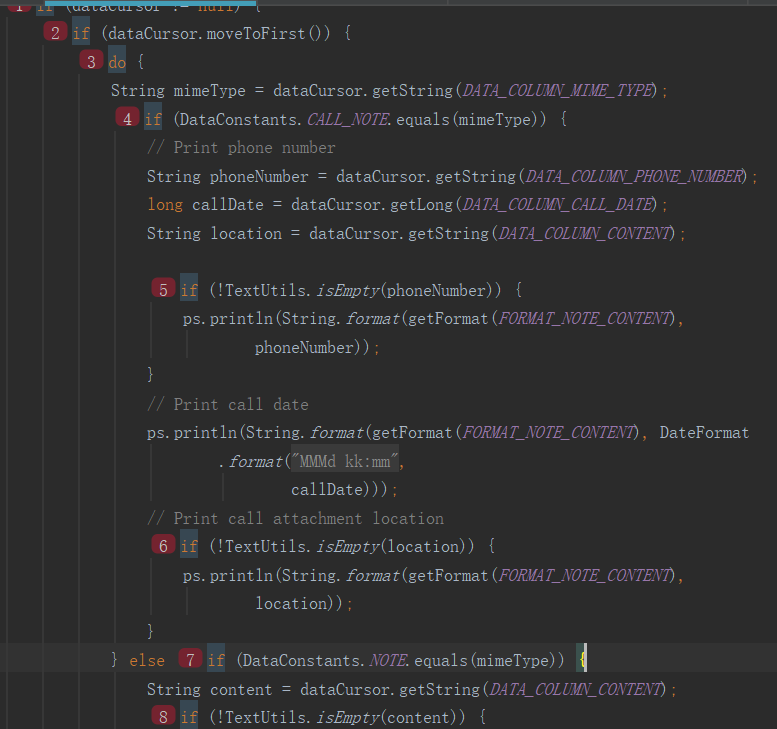
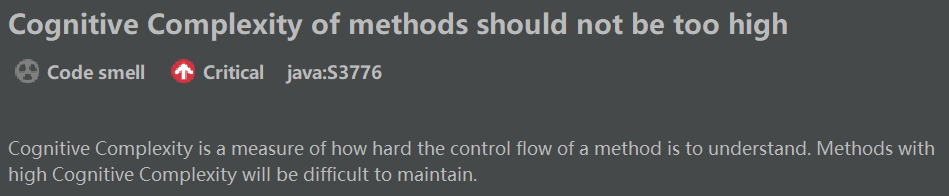
在这个函数里，使用子类的名称访问父类的静态成员。这样做会造成混淆，并且可能会导致存在两个不同的静态成员的错觉。

解决办法：

应该通过父类来调用

(2)BackupUtils.java:

Critical型问题：



问题：

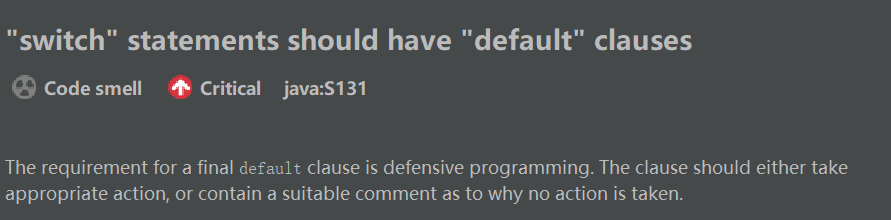
if判断分支过多，函数的认知复杂度太高，可读性较差。

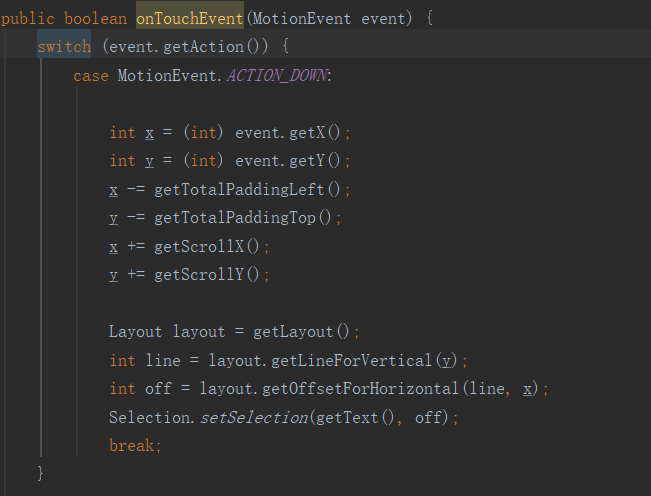
解决办法：

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)NoteEditText.java

Critical型问题：





问题：

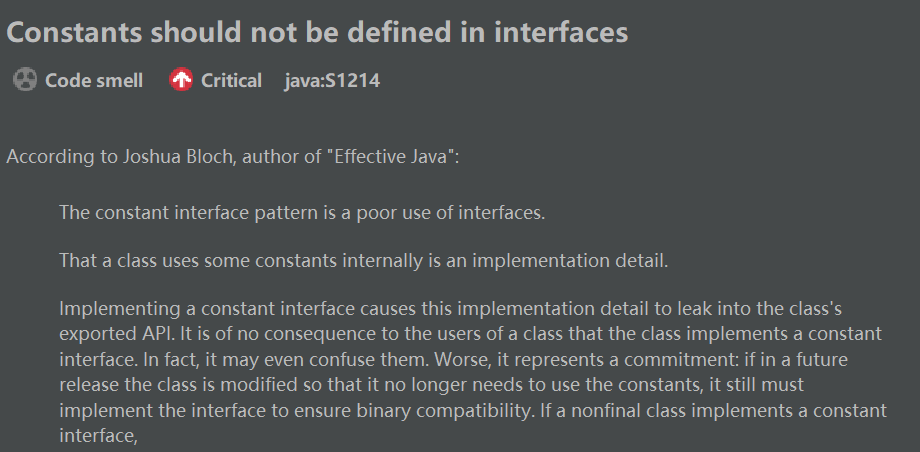
在这个函数里，使用了switch结构，但是没有设置default分支，如果出现给定情况之外的情况，则容易出现问题。

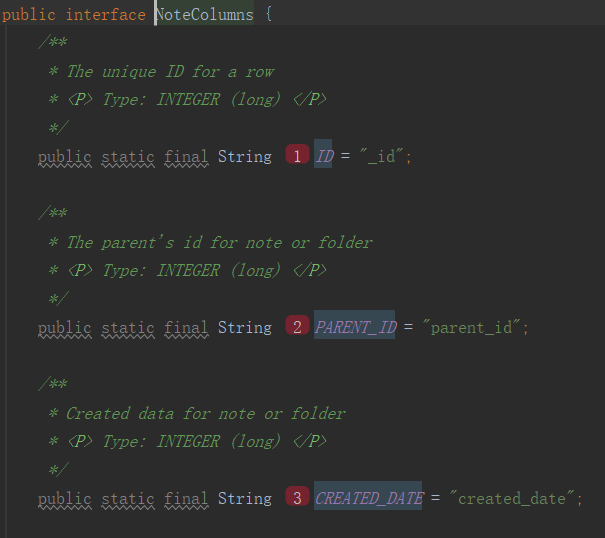
解决办法：

应该设置一个default case的情况来抛出异常，如果出现对应问题，则可以通过输出来解决问题。

(4)Notes.java

Critical型问题：





问题：

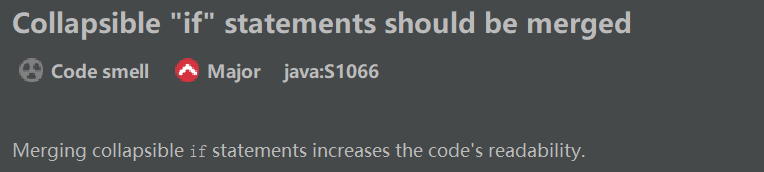
常量不应在接口中定义。

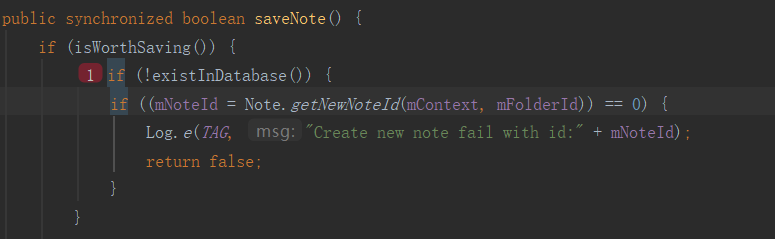
解决办法：

可将接口改为其他类，例如公共类

(5)WorkingNote.java：

Major型问题：





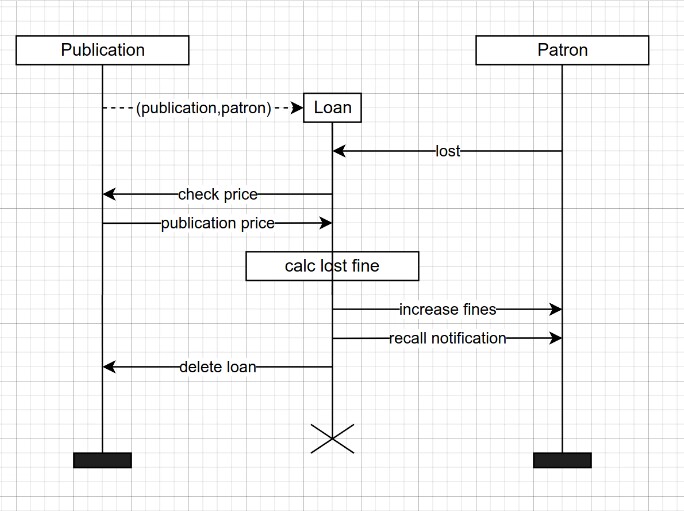
问题：

可折叠的if语句应该合并。

解决办法：通过进一步的逻辑分析，将可合并的多级if合成一个if，增加代码可读性。

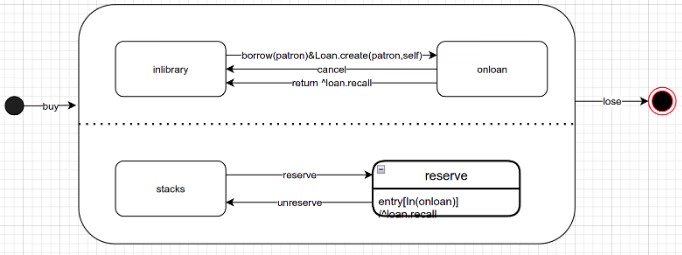
（1）用CASE工具画出课本P115，图4-7 图书馆借出事务的消息时序图MSC。

参考P118 图4-11，在图4-7中补充lost场景的消息时序



（2）用CASE工具画出课本P117，图4-9 Publication类的UML状态图。

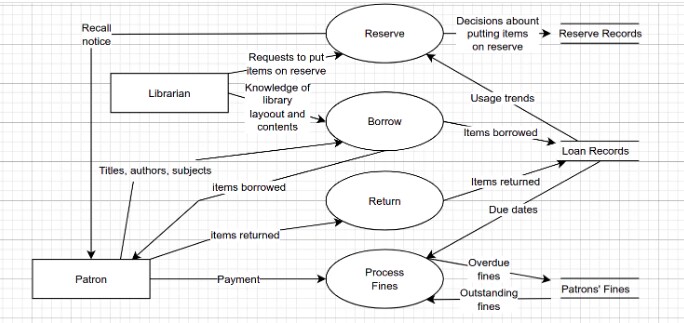
分析它为什么比图4-10简洁、清晰?



这张图更简洁的原因在于它将两个不相关的子状态机给分开了;如果不进行区分，就要对所有的状态进行组合，导致状态数变多，从而导致状态转换变得更加复杂。

（3）用CASE工具画出课本P122，图4-14 图书馆问题的数据流图

该DFD有几个数据存储？



该DFD有3个数据存储