## 小米便签开源代码的质量分析报告

小米便签开源软件是一款较为流行的备忘录软件，其由Java语言编写，基于Android操作系统进行开发运行和维护。小米便签开源软件规模适中，通过初步阅读分析其代码质量较高，故对其进行进一步的质量分析和审查。

### 代码质量分析方法

代码质量分析主要由人工分析和利用工具分析两种方式，二者各有优缺点，在实际过程中需要结合使用以保证正确性、发挥效能。

1. **人工**

通过人工发现代码中存在的缺陷和质量问题是分析软件质量最直接的手段。软件开发相关人员在软件实现、测试、维护过程中，主动发现编写代码中存在的问题并给予修改，但是分析代码的效率低，寻找缺陷不够全面，比如其中软件的深层次问题可能难以发现。

1. **自动**

利用SonarQube，FindBugs等工具进行代码质量分析，从程序的语法、结构、接口等方面进行代码审查，并能够对代码风格进行分析。其优点是代码分析效率高，且能够发现软件开发人员忽略的错误，在实际应用中要结合人工审查才能完全明确代码缺陷。

本报告主要使用SonarLint工具进行针对小米便签开源软件进行质量分析，辅以少量的人工分析。

### Sonar及SonarQube简介

SonarQube软件是一种静态代码检查工具，采用B/S架构，帮助检查代码缺陷，改善代码质量，提高开发速度，通过插件形式，可以支持Java、C、C++、JavaScript等等二十几种编程语言的代码质量管理与检测。

Sonar客户端可以采用IDE插件、Sonar-Scanner插件、Ant插件和Maven插件方式，并通过各种不同的分析机制对项目源代码进行分析和扫描，并把分析扫描后的结果上传到Sonar的数据库，通过Sonar web界面对分析结果进行管理。

### 基于AndroidStudio配置SonarLint

1. 首先从SonarQube官网上下载SonarQube软件包（目前最新版本为7.3）。

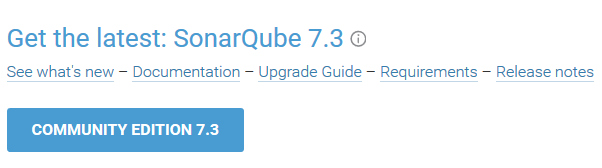


图1 SonarQube下载

1. 如图2，解压后运行bin\Window-x86-64目录下的StartSonar.bat文件启动SonarQube服务器（本机为Windows64位操作系统）。

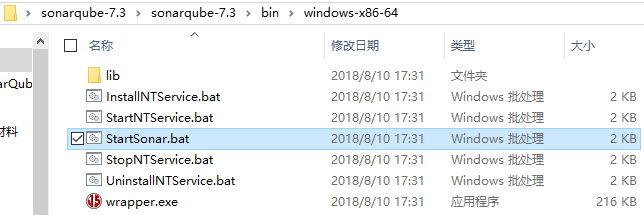


图2 SonarQube目录

运行至出现如图3输出时表示服务器已经建立:

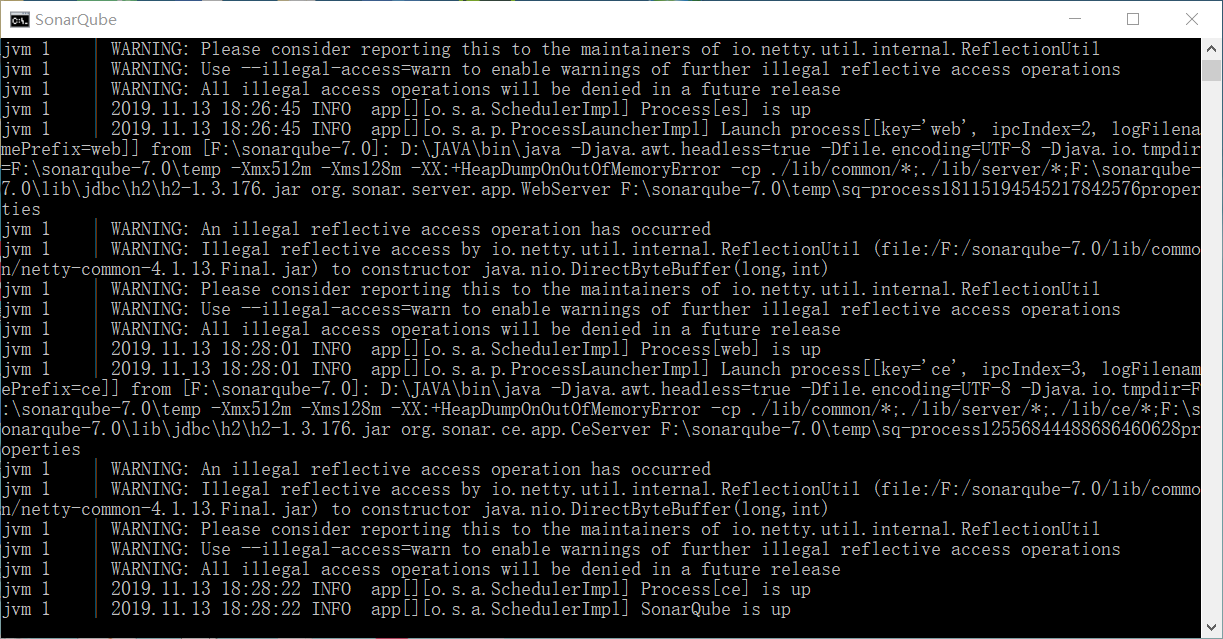


图3 建立SonarQube服务器

3、访问http：//localhost:9000并使用账户admin进行登陆，成功后即连接上SonarQube服务器。为了方便在本地的AndroidStudio中进行代码质量分析，下一步在AndroidStudio中配置SonarLint插件，该插件可以按照SonarQube中的规则对代码进行分析。

4、下载SonarLint插件：从Plugins.jetbrains.com下载SonarLint插件（目前版本为3.5.1），如图4。

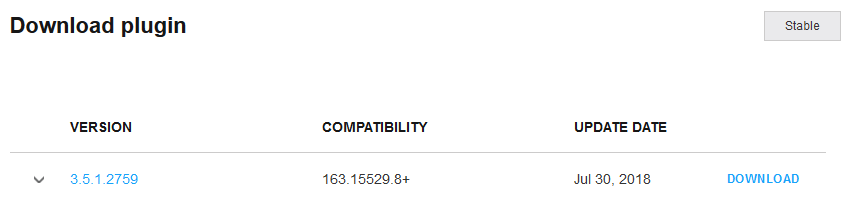


图4 下载SonarLint插件

5、在AndroidStudio中添加本地插件，如图5。

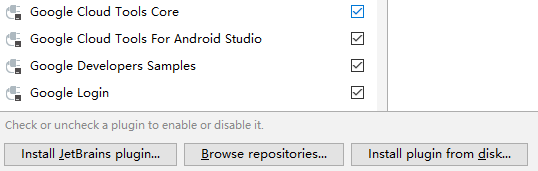


图5 添加本地插件

6、在Settting中的OtherSetting修改SonarLintGeneralSettings将本地项目和服务器绑定，使用SonarQube服务器的规则对本地代码进行质量分析：

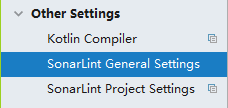


图6 选中SonarLintGeneralSettings

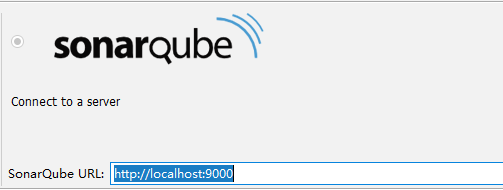


图7 连接到SonarQube服务器

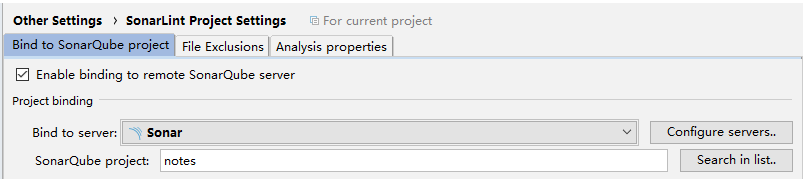


图8 与网上服务器进行绑定

### 使用SonarLint进行分析

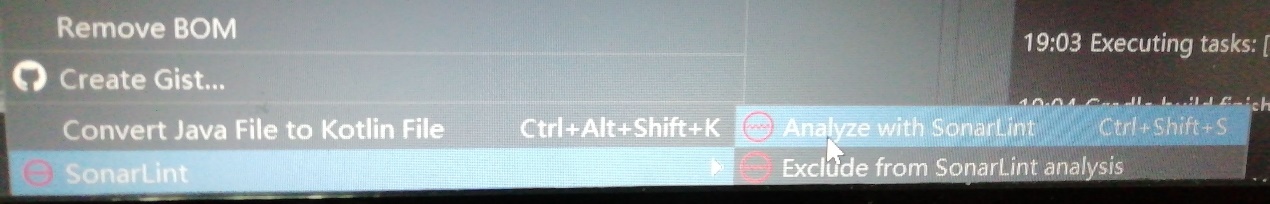


图9 使用SonarLint进行分析

在项目视图中选中待分析的包，在右键选项中使用SonarLint进行分析。对小米便签全部源码（39个文件）自动完成质量审查，得到如下代码质量分析报告：

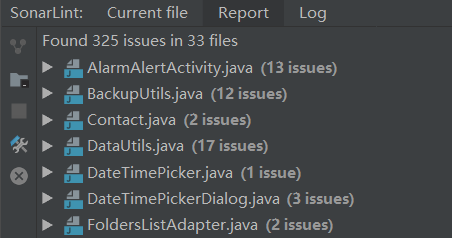


图10 SonarLint分析结果

在小米便签软件的源代码中，共审查出33个文件中的325个问题，经过人工审查，其中大量的问题在于规范和代码的简洁性要求，例如有308个Critical Issues为R.java类中的常量命名问题，所以总体来说，小米便签软件的质量较为不错。

代码中的缺陷分为如下几个等级：

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。

现对代码存在的质量问题进行汇总统计，对严重程度为Blocker、Critical和Major的问题进行逐一的分析：

严重程度：Blocker

1. Resources should be closed：资源未关闭

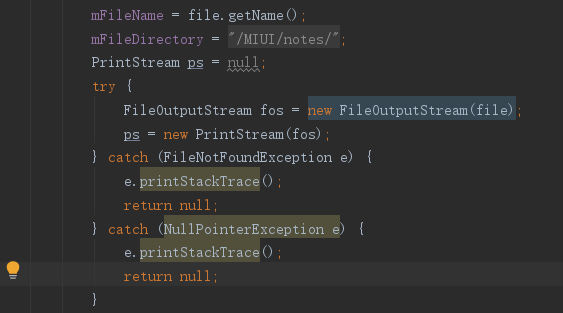


图11 BackUtils.java

资源的创建需要使用“try-with-resources”的模式，如果资源用尽则将自动关闭，防止发生内存泄漏的问题，应该将语句ps.close()写在finally块中,并把finally代码块放在第二个catch后面。

严重程度：Critical

1、Cognitive Complexity of methods should not be too high：方法的认知复杂度不应该过高

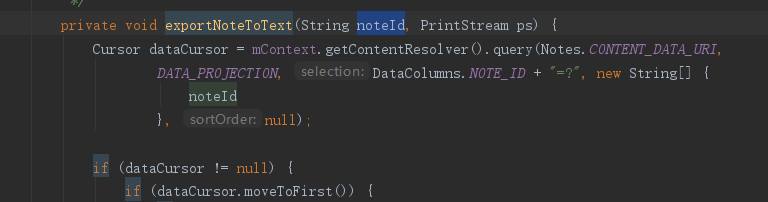


图12 BackupUtils.java

方法的认知复杂度是指衡量控制流理解难易程度的指标，可从以下方面进行评估：

代码块内部的嵌套控制流结构（对所有外层嵌套有影响）、使嵌套量增层数的控制流结构（for，while，do-while，if）和不使嵌套层数增加的控制流结构（else if，else，catch，递归调用，逻辑运算符序列）。如果多个语句在阅读时可以理解为一条，则忽略其对认知复杂度的影响（方法，是否为空的判断）。过高的认知复杂度可能会提高代码维护的难度。

1. String literals should not be duplicated：字符串不应该被复制

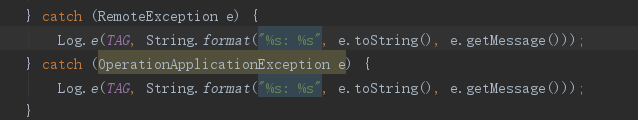


图13 DataUtils.java

在代码重构的过程中，重复的字符串如果需要进行修改，则容易发生因为仅有其中某个或某一部分值进行修改的数据一致性错误。软件开发者必须确保修改该字符串的所有出现。SonarLint工具给出了相应提示：使用常量代替字符串，避免出现多变量不一致的情况。

3、"static" base class members should not be accessed via derived types不应通过派生类型访问“静态”基类成员

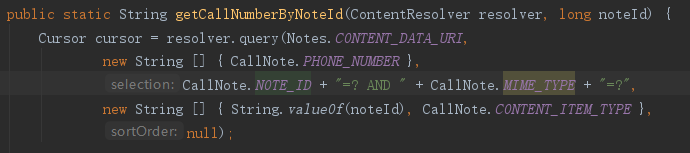


图14 DataUtils.java

为了代码清晰起见，永远不要使用派生类型的名称访问基类的静态成员。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。

4、Try-with-resources should be used 异常处理资源格式应该被使用

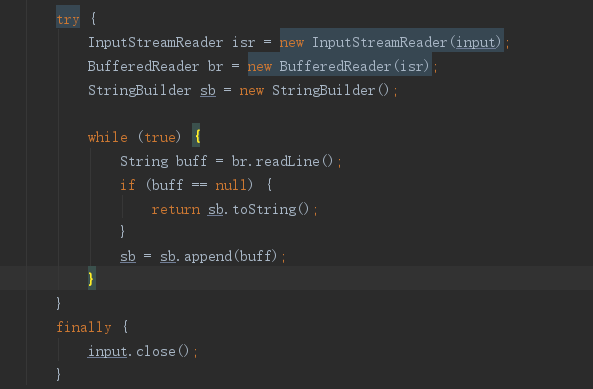


图15 GTaskClient.java

在java中，try,catch,finally应该搭配使用，缺一不可。

5、Cognitive Complexity of methods should not be too high 方法复杂度太高

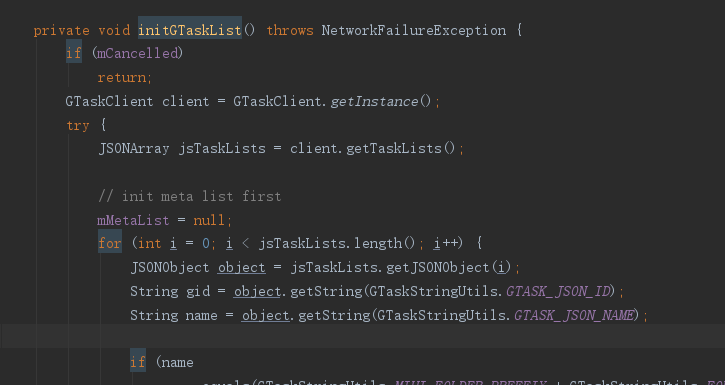


图16 GTaskManager.java

应该降低方法内if条件判断复杂度以及for的复杂度。

6、Constants should not be defined in interfaces：不应该在接口中定义常量

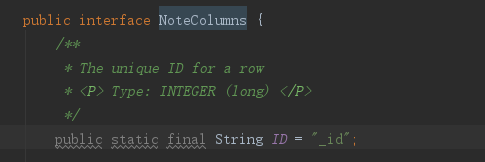


图17 Notes.java

在接口中定义常量会要求其子类必须含有这些常数，原则上应该将此接口中常数删去或将其接口变为final类型。

严重程度：Major

1. Utility classes should not have public constructors：工具类不应该含有公有构造器

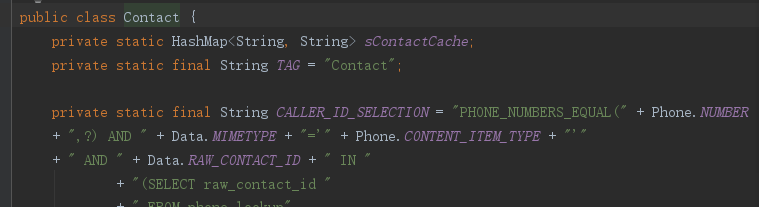


图18 Contact.java

工具类中，只含有静态的方法，可以使用类直接调用而不需要实例化对象，而使用私有构造器的目的是在类的外部无法调用该构造器，从而能够阻止对象的生成；Java在未显式声明私有构造器的时候默认添加一个公有构造器，可能会使该工具类出现未知的错误。

1. Collapsible "if" statements should be merged：可折叠的“if”语句需要被合并

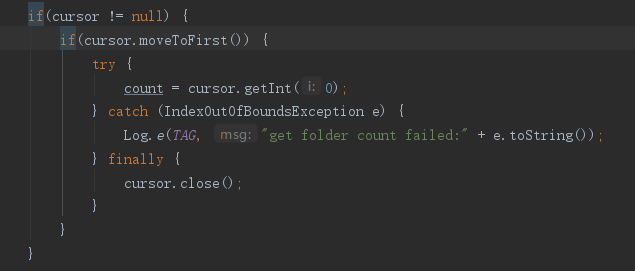


图19 DataUtils.java

从示例代码中可以看出，外层if语句中并没有分支，故可以将两个判断条件合并以增加程序的可读性。

1. All branches in a conditional structure should not have exactly the same implementation：分支结构的所有分支的实现不应该完全相同



图20 DateTimePickerDialog.java

1. Sections of code should not be commented out：不需要被使用的代码段不需要被注释掉，应该直接删除

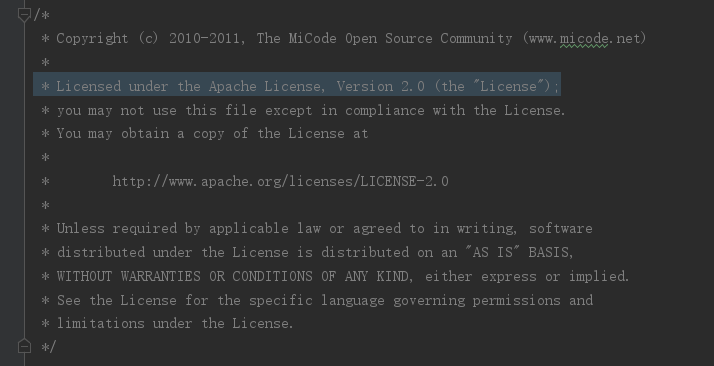


图21 GTaskASyncTask.java

解决方法就是直接删除掉。

5、Unused method parameters should be removed 方法中未使用的参数应该删除

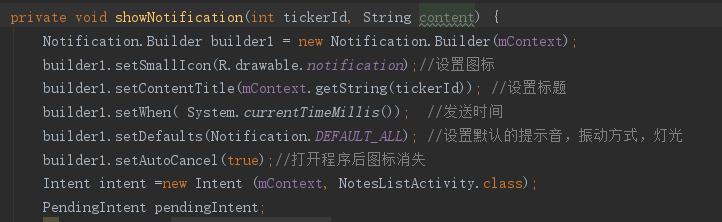


图22 GTaskASyncTask.java

6、 Boolean expressions should not be gratuitous：逻辑表达式不应无用，比如永真式作为分支判断条件：

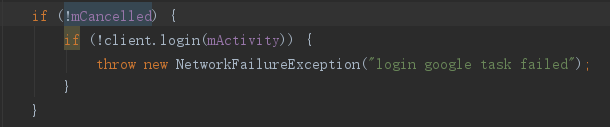


图23 GTaskManager.java

7、Constructors should not be used to instantiate "String", "BigInteger", "BigDecimal" and primitive-wrapper classes："String", "BigInteger", "BigDecimal"等类型不需要使用构造器进行实例化，直接使用ValueOf方法即可

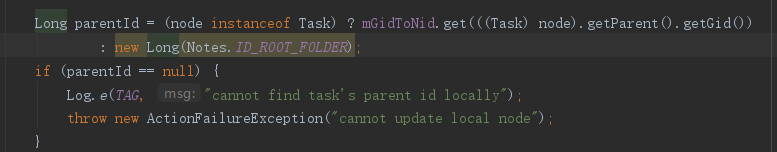


图24 GTaskManager.java

8、"entrySet()" should be iterated when both the key and value are needed ：若在循环体内map的关键字和值都需要，那么需要把键值对一起作为循环变量进行迭代

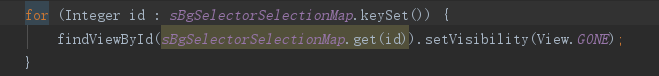


图25 NoteEditActivity.java

9、Methods should not have identical implementations：不同的方法不应该使用相同的实现

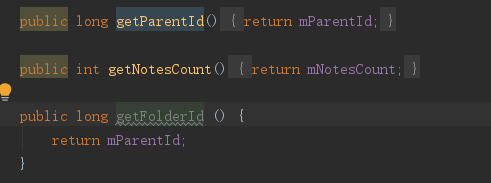


图26 NoteItemData.java

10、Nested blocks of code should not be left empty：嵌套结构内部不应该为空

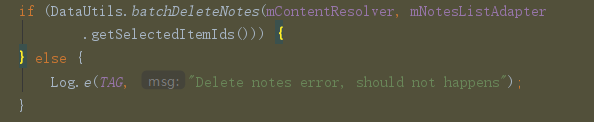


图27 NotesListActivity.java

11、Empty arrays and collections should be returned instead of null：数组返回值不应该为空

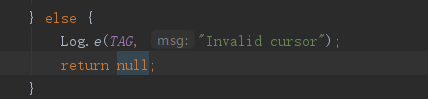


图28 NotesListAdapter.java

12、Null pointers should not be dereferenced： 空指针不应该被解除引用



图29 Task.java