小米便签代码质量分析报告

报告人：吴子吉 方群天

1. 待分析代码概述

Mi Note 便签是小米便签的社区开源版，由 MIUI 团队(www.miui.com) 发起并贡献第一批代码，是较为流行的记事软件。该开源应用代码总行数 12.5K，有效代码行 9.1K，其中 JAVA 代码行数 7.4K、XML 代码行数 1.7K，规模适中，主要使用 JAVA 语言编写，在 Android 平台上进行开发维护，涉及到 Android 编程的各方面。代码质量较高，注释较少，适合对其代码进行标注、维护开发与质量分析。

2. 质量分析方法及工具

针对小米便签代码的质量分析主要关注其代码的编码风格、设计质量以及代码的缺陷、错误等，采用人工审查与软件工具自动审查相结合的方式。

人工审查代码中的缺陷和质量问题是分析软件质量最直接的手段，在软件的编写、测试、维护过程中，软件开发者能够发现代码中的质量问题并修改，但是随着软件规模的扩大，人工审查的方法效率低、不能完全覆盖、难以发现隐含的深层次问题等弊端逐渐显现。

利用 SonarQube、Find Bugs、Check Style 等质量分析工具进行代码质量分析，从程序代码的语法、结构、接口等方面进行代码审查，还能够检查代码的编程规范。利用工具审查的优点是代码分析效率高，且能够发现软件开发者忽略的错误，在实际应用中要结合人工审查才能完全明确代码缺陷。

利用两种方法之间的优势互补，提高代码分析的效率与准确率。对小米便签的代码审查采用工具审查为主，人工审查为辅的方式。自动审查使用质量分析工具 SonarQube 与 Sonar Scanner 进行。

2.1 SonarQube 工具概述

SonarQube 是一个开源的、支持私有化部署的代码质量管理系统。与其他代码质量管理系统相比较，具有以下优势：

- 支持超过 25 种编程语言，涵盖 Java、C/C++、C#、PHP、Flex、Groovy、Python、PL/SQL、JavaScript、COBOL 等
- 提供重复代码、编码标准、单元测试、代码覆盖率、代码复杂度、潜在 Bug、注释的检测和软件设计报告的生成
- 提供了指标历史记录
- 支持与 Maven、Ant、Gradle 、Atlassian Bamboo、Jenkins、Hudson 等集成
- 支持与绝大多数主流 IDE 如 Android Studio、Eclipse、IntelliJ IDEA 等集成
- 支持 JIRA、Mantis、LDAP、Fortify 等外部工具集
- 支持插件扩展，可自行到应用市场下载安装所需插件

SonarQube 平台主要由 4 部分组成，一是 SonarQube 服务器，提供 Web Server 帮助开发者或者管理者浏览质量快照和配置 SonarQube 实例、提供 Search Server based on Elasticsearch，执行来自页面的查询请求以及提供 Compute Engine Server，处理代码分析报告并保存在 SonarQube 数据库；二是 SonarQube 数据库，存储 SonarQube 实例的配置（安全、插件设置等）以及项目、视图等的质量快照；三是用户安装的各种插件；四是一个或者多个 Sonar Scanner。

2.2 提供的功能

Sonar 可以从以下七个维度检测代码质量，作为软件开发人员至少需要处理前 5 种代码质量问题。

1. 不遵循代码标准：Sonar 可以通过 PMD，Check Style，Find bugs 等代码规则检测工具规范代码编写

2. 潜在的缺陷：sonar 可以通过 PMD，Check Style，Find bugs 等代码规则检测工具检测出潜在的缺陷

3. 糟糕的复杂度分布：文件、类、方法等，如果复杂度过高将难以改变，这会使得开发人员难以理解它们，且如果没有自动化的单元测试，对于程序中的任何组件的改变都将可能导致需要全面的回归测试。

4. 重复：显然程序中包含大量复制粘贴的代码是质量低下的，Sonar 可以展示源码中重复严重的地方

5. 注释不足或者过多：没有注释将使代码可读性变差，特别是当不可避免地出现人员变动时，程序的可读性将大幅下降，而过多的注释又会使得开发人员将精力过多地花费在阅读注释上，亦违背初衷

6. 缺乏单元测试：sonar 可以很方便地统计并展示单元测试覆盖率

7. 糟糕的设计：通过 sonar 可以找出循环，展示包与包、类与类之间的相互依赖关系，可以检测自定义的架构规则，还可以管理第三方的 jar 包，可以利用 LCOM4 检测单个任务规则的应用情况， 检测耦合。

Sonar 将代码中的缺陷分为如下五个等级，严重程度依次递减：

（1）阻断（Blocker）：极有可能影响应用程序表现的错误；

（2）严重（Critical）：可能影响应用程序表现的错误和表示安全缺陷的问题；

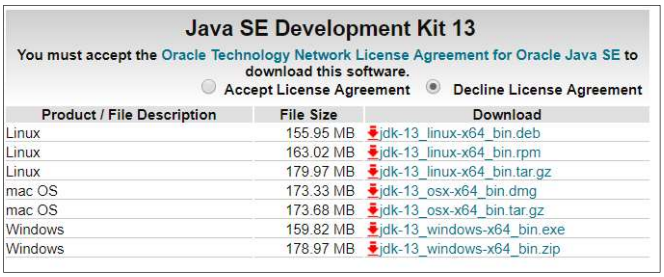
（3）主要（Major）：严重影响开发者效率的质量缺陷；

（4）次要（Minor）：轻微影响开发者效率的质量缺陷；

（5）提示（Info）：不是错误或者质量缺陷。

2.3 Windows 环境下 SonarQube 与 Sonar Scanner 安装使用

1. SonarQube 的运行依赖于 JDK，并且要求版本不低于 1.12，到官网下载 JDK，安装并添加环境变量。



Java SE Development Kit 13

You must accept the Oracle Technology Network License Agreement for Oracle Java SE to download this software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux	155.95 MB	jdk-13_linux-x64_bin.deb
Linux	163.02 MB	jdk-13_linux-x64_bin.rpm
Linux	179.97 MB	jdk-13_linux-x64_bin.tar.gz
mac OS	173.33 MB	jdk-13_osx-x64_bin.dmg
mac OS	173.68 MB	jdk-13_osx-x64_bin.tar.gz
Windows	159.82 MB	jdk-13_windows-x64_bin.exe
Windows	178.97 MB	jdk-13_windows-x64_bin.zip

图 1 官网下载合适版本 JDK

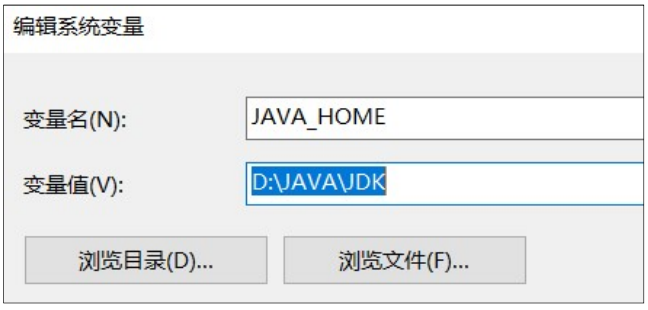


图 2 添加 JAVA_HOME 系统变量

图 3 添加 Path 环境变量

2. 到官网（https://www.sonarqube.org/downloads/）下载 SonarQube 软件包，目前最新版本为 7.9.1 LTS。

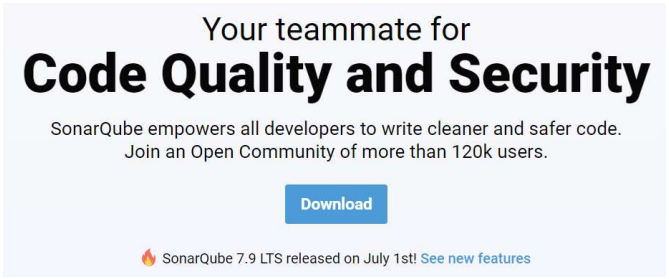


图 4 官网下载 SonarQube 软件包

3. 如图所示，解压下载的安装包，进入 bin 文件夹，根据电脑配置不同打开不同文件夹，此处我电脑为 64 位，运行 StartSonar.bat 文件，直至出现如下结果。

图 5 点击运行 StartSonar.bat 文件

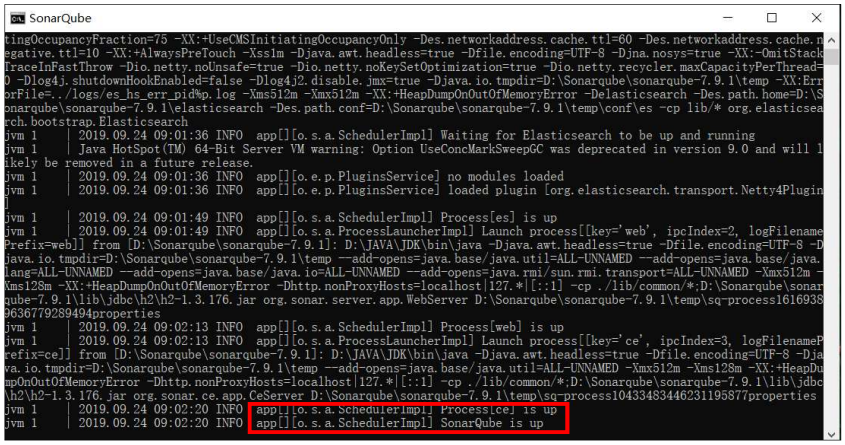


图 6 运行成功结果

4. 启动浏览器，访问 http://localhost:9000/，如果访问成功，说明安装成功。

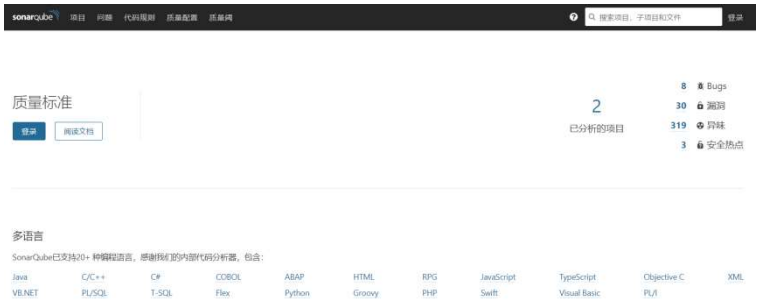


图 7 能打开网址，安装成功

5. 安装依赖软件 PostgreSQL。到官网（https://www.enterprisedb.com/downloads/postgres-postgresql-downloads）下载合适版本的 PostgreSQL 软件，并安装。安装完成后创建一个名为 SonarQube 的数据库。

PostgreSQL Database Download

PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
11.5	N/A	N/A	Download	Download	N/A
10.10	Download	Download	Download	Download	Download
9.6.15	Download	Download	Download	Download	Download
9.5.19	Download	Download	Download	Download	Download
9.4.24	Download	Download	Download	Download	Download
9.3.25 (Not Supported)	Download	Download	Download	Download	Download

图 8 官网下载依赖软件 PostgreSQL

6. 打开 SonarQube 文件夹下的 conf 文件夹，修改里面的 sonar.properties 文件。其中 username，password 是数据库的用户名和密码，根据自己的创建情况修改。重新运行 StartSonar.bat 程序，启动浏览器，如能访问 localhost:9000 表示配置成功。
7. 到官网（https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/）下载合适的 Sonar Scanner 安装包；下载完成后，解压即可。

SonarScanner

Download SonarScanner 4.0 - Compatible with SonarQube 6.7+ (LTS) By **SonarSource** – GNU LGPL 3 – **Issue Tracker** – **Source**

Linux 64-bit | **Windowx 64-bit** | **Mac OS X 64-bit** | **Any*** *Requires a pre-installed JVM - with the same requirements as the SonarQube server.

图 9 官网下载 Sonar Scanner 安装包

名称	修改日
bin	2019/
conf	2019/
jre	2019/
lib	2019/

图 10 解压后的文件夹

8. 在待分析的项目文件夹下新建一个名为 sonar-project.properties 的文件，按如下进行配置：

```
#Sonar Scanner 配置文件：
sonar-project.properties
```

```
# must be unique in a given SonarQube instance
sonar.projectKey=test1

# --- optional properties ---

# defaults to project key
#sonar.projectName=My project
# defaults to 'not provided'
#sonar.projectVersion=1.0

# Path is relative to the sonar-project.properties file. Defaults
to .
#sonar.sources=.
# Encoding of the source code. Default is default system encoding
sonar.sourceEncoding=UTF-8
sonar.java.binaries=res
```

9. 运行 cmd，进入项目所在文件夹，输入命令 sonar-scanner，待其完成后刷新 http://localhost:9000/，如果出现图 11 所示项目分析结果，说明分析成功。

图 11 小米便签全部代码分析结果概况

3. 人工分析代码的质量情况

3.1 代码风格

Java 程序语言遵循一系列的编码风格，包括**命名通则**，命名时应该采用有意义、一目了然的命名方式，应使用标准英文单词或缩写、不要使用拼音，应遵循望文知义原则、即名称应含义清晰、明确，不要过长、控制在规定的最大长度以内，所有命名都应尽量使用全称，少用缩写；**缩进**的使用，要用好 Tab 键；**程序注释：**加强程序理解，尤其是能帮助他人理解程序，要说明程序:(1) 做什么；(2)为什么这么做；(3) 注意事项，无需解释程序如何做，注解放在代码头、函数头、关键语句尾，太少和过多的注释不可取，随代码的修改而修改。

下面举例来分析小米便签代码中代码风格较好的地方：

1. **注释规范、命名合理、格式正确：**如图 12 代码片段为例，**一是**注释的格式与内容较为规范，对方法的注释说明置于方法首行，并且注释内容简洁、意义明确，该条注释说明下面代码功能是新便签的初始构造；**二是**命名合理，下例代码变量、函数命名遵循驼峰命名法，具体意义一目了然，大多使用单词全称，如 mNoteId 变量，一看便知其意义为便签 ID；**三是**格式正确，表现在大括号位置正确，左侧大括号位于函数行末端，右侧大括号位于函数体结束的下一行首端。

```
// New note construct
private WorkingNote(Context context, long folderId) {
    mContext = context;
    mAlertDate = 0;
    mModifiedDate = System.currentTimeMillis();
    mFolderId = folderId;
    mNote = new Note();
    mNoteId = 0;
    mIsDeleted = false;
    mMode = 0;
    mWidgetType = Notes.TYPE_WIDGET_INVALIDE;
}
```

图 12 WorkingNote.java 代码片段

2. **变量注释明了：**在小米便签中关于变量的注释比较明确，变量类型、用途都会在注释中说明。如图 13 的代码片段，对每一个变量的类型、用途都做了详细明确的解释，达到了使读者一目了然的目的。

```
public static final String SYNC_ID = "sync_id";

/**
 * Sign to indicate local modified or not
 * <P> Type: INTEGER </P>
 */
public static final String LOCAL_MODIFIED = "local_modified";

/**
 * Original parent id before moving into temporary folder
 * <P> Type : INTEGER </P>
 */
public static final String ORIGIN_PARENT_ID = "origin_parent_id";

/**
 * The gtask id
 * <P> Type : TEXT </P>
 */
public static final String GTASK_ID = "gtask_id";
```

图 13 Notes.java 代码片段

3. **函数命名符合规范：**小米便签中，函数的命名较好的符合函数命名规范，通过函数名称，能大概获取函数实现的功能，并且使用英文、无缩略语等。如图 14 代码片段所示，对很多函数的命名意义明确，直接明确了函数所要实现的功能，如建立新便签，批量删除等功能，十分有利于对代码的阅读理解和开发维护。

```
private void startAsyncNotesListQuery() {...}

private final class BackgroundQueryHandler extends AsyncQueryHandler {...}

private void showFolderListMenu(Cursor cursor) {...}

private void createNewNote() {...}

private void batchDelete() {...}
```

图 14 代码规范，函数命名正确

4. **处理 if-else 语句格式正确：**if 结束的右侧大括号后紧跟 else 和左括号。同时其缩进和空格使用合理，程序语句排列顺畅。

```
if (cursor != null) {
    if (cursor.moveToFirst()) {
        mFolderId = cursor.getLong(NOTE_PARENT_ID_COLUMN);
        mBgColorId = cursor.getInt(NOTE_BG_COLOR_ID_COLUMN);
        mWidgetId = cursor.getInt(NOTE_WIDGET_ID_COLUMN);
        mWidgetType = cursor.getInt(NOTE_WIDGET_TYPE_COLUMN);
        mAlertDate = cursor.getLong(NOTE_ALERTED_DATE_COLUMN);
        mModifiedDate = cursor.getLong(NOTE_MODIFIED_DATE_COLUMN);
    }
    cursor.close();
} else {
    Log.e(TAG, "No note with id:" + mNoteId);
    throw new IllegalArgumentException("Unable to find note with id " + mNoteId);
}
loadNoteData();
```

图 15 代码片段，格式正确

根据以上分析小米便签在绝大程度上都遵循了 Java 的编程风格，其命名和注释方式合理，换行符合要求，if-else 语句中{}使用规范，整体符合编程要求。

3.2 代码设计思路

Java 程序设计基本原则有**模块化、高内聚度**以及**低耦合度。**对小米便签整体架构分析如下：

小米便签由**界面层**、**业务层**、**模型层**和**数据层**构成。

界面层：包含 ui 包和 res 包，位于体系结构的最上层，为用户提供交互式的操作界面，其提供的主要功能包括应用图标的显示、各个操作界面元素的显示等。该层主要通过主包下的 AndroidManifest.xml 文件、res 库文件以及 ui\res 程序包中的类加以实现。

业务层：该层是小米便签系统体系结构的核心，用于连接界面层和模型曾、数据层，起到数据交换的作用。其提供的主要功能和服务包括定义软件的业务流程，接受界面层的用户请求，与模型层、数据层进行交互，将处理结果返回界面层。该层主要通过 tool 包、widget 包和 gtask 包中的 exception 和 remote 子包以及其他相关部件加以实现。

模型层：该层主要对单个便签项进行建模，小米便签的新建，以及实现了小米便签的基础工作功能。在 Note.java 中，分为用户进入，数据更新，以及安全检测这三个部分，在 WorkingNote.java 中，实现了许多功能项，如新建，模式切换，设置时钟等，主要是用于小米便签的正常工作。

数据层：该层负责组织和存储小米便签的相关数据，提供数据访问、数据和发行检验、数据访问却是异常处理等功能和服务。该层主要通过 data 和 gtask.data 程序包实现。

在整体上体现了模块化、高内聚度和低耦合度的设计理念，小米便签包结构如图 16 所示。

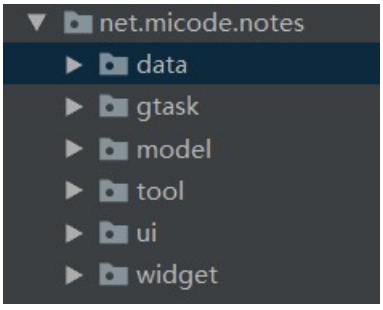


图 16 小米便签包结构

以上为小米便签代码的包的情况，从命名体系上看，各个包之间相互支撑，作用突出，

重叠部分少，程序模块的可重用性、移植性高，也是模块化思想的体现。

在具体的函数和类的实现中可以发现小米便签遵循良好的高内聚度与低耦合度的思想，例如在 gtask 包中的 data 子包中：

```
//利用cursor设置属性参数
private void loadFromCursor(Cursor c) {
    mId = c.getLong(ID_COLUMN);
    mAlertDate = c.getLong(ALERTED_DATE_COLUMN);
    mBgColorId = c.getInt(BG_COLOR_ID_COLUMN);
    mCreatedDate = c.getLong(CREATED_DATE_COLUMN);
    mHasAttachment = c.getInt(HAS_ATTACHMENT_COLUMN);
    mModifiedDate = c.getLong(MODIFIED_DATE_COLUMN);
    mParentId = c.getLong(PARENT_ID_COLUMN);
    mSnippet = c.getString(SNIPPET_COLUMN);
    mType = c.getInt(TYPE_COLUMN);
    mWidgetId = c.getInt(WIDGET_ID_COLUMN);
    mWidgetType = c.getInt(WIDGET_TYPE_COLUMN);
    mVersion = c.getLong(VERSION_COLUMN);
}
```

图 17 data 包中部分代码片段

如图 17，模块只对外暴露最小限度的接口，形成最低的依赖关系，只要对外接口不变，模块内部的修改，就不得影响其他模块。这样增强了内聚度。

```
//根据数据库查询id
private void loadDataContent() {
    Cursor c = null;
    mDataList.clear();
    try {
        c = mContentResolver.query(Notes.CONTENT_DATA_URI, SqlData.PROJECTION_DATA,
                selection: "(note_id=?)", new String[] {
                    String.valueOf(mId)
                }, sortOrder: null);
        if (c != null) {
            if (c.getCount() == 0) {
                Log.w(TAG, msg: "it seems that the note has not data");
                return;
            }
            while (c.moveToNext()) {
                SqlData data = new SqlData(mContext, c);
                mDataList.add(data);
            }
        } else {
            Log.w(TAG, msg: "loadDataContent: cursor = null");
        }
    } finally {
        if (c != null)
            c.close();
    }
}
```

图 18 代码片段

如图 18，在函数的实现过程中尽可能少地使用了全局变量，并且实现方法使用的是 private，模块实现功能化单一，为其他模块提供的调用机会少，这样有利于降低耦合度，提高内聚度。

3.3 编程技能

小米便签代码用的是 JAVA 编程语言结合 Android 中自带的库进行的面向对象的程序设计，其中没有用到特殊的算法知识等。在编程技能方面主要体现了对小米便签的整体模块掌握，层次清晰，功能实现具体，能够把小米便签的架构清晰展现出来；当然在一些细节的把握上也可以看出很多值得学习的编程技巧。

首先同样是几个包的构建是对小米便签整体框架的构建，如图 19 所示：

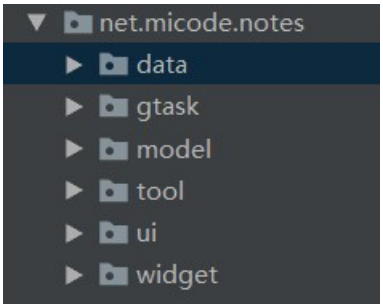


图 19 小米便签包框架

其次在一些具体的类以及函数实现中，如图 20，用 final 声明变量，可以防止自己在后面修改变量的值，有助于防错。

```
// Currently, the sdcard is not mounted
public static final int STATE_SD_CARD_UNMOUONTED            = 0;
// The backup file not exist
public static final int STATE_BACKUP_FILE_NOT_EXIST         = 1;
// The data is not well formated, may be changed by other programs
public static final int STATE_DATA_DESTROIED                = 2;
// Some run-time exception which causes restore or backup fails
public static final int STATE_SYSTEM_ERROR                  = 3;
// Backup or restore success
public static final int STATE_SUCCESS                       = 4;
```

图 20 代码片段，final 声明变量

如图 21 与图 22，下列代码对 null 空的情况进行了检查，这是一个好习惯，避免了很多考虑不全的报错。在代码中还尽量地使用了局部变量：

```
if (cursor != null && cursor.moveToFirst()) {
    try {
        return cursor.getString( i: 0);
    } catch (IndexOutOfBoundsException e) {
        Log.e(TAG,  msg: "Get call number fails " + e.toString());
    } finally {
        cursor.close();
    }
}
```

图 21 代码片段，null 情况检查

```
if (file == null) {
    Log.e(TAG,  msg: "create file to exported failed");
    return null;
}
```

图 22 代码片段，null 情况检查

如图 23，下列代码中很多地方都是使用局部变量来实现函数功能，使得临时变量保存在栈里，这样实现起来可以更快。

```
public static synchronized long getNewNoteId(Context context, long folderId) {
    // Create a new note in the database
    ContentValues values = new ContentValues();
    long createdTime = System.currentTimeMillis();
    values.put(NoteColumns.CREATED_DATE, createdTime);
    values.put(NoteColumns.MODIFIED_DATE, createdTime);
    values.put(NoteColumns.TYPE, Notes.TYPE_NOTE);
    values.put(NoteColumns.LOCAL_MODIFIED, 1);
    values.put(NoteColumns.PARENT_ID, folderId);
    Uri uri = context.getContentResolver().insert(Notes.CONTENT_NOTE_URI, values)

    long noteId = 0;
    try {
        noteId = Long.valueOf(uri.getPathSegments().get(1));
    } catch (NumberFormatException e) {
        Log.e(TAG,  msg: "Get note id error :" + e.toString());
        noteId = 0;
    }
    if (noteId == -1) {
        throw new IllegalStateException("Wrong note id:" + noteId);
    }
    return noteId;
}
```

图 23 代码片段，局部变量使用

3.4 代码质量问题

当然小米便签的代码也有做得不足的质量问题，通过人工阅读代码的形式，发现了一系列小米便签中的质量问题，下面举例说明：

1. 对象使用完之后未被关闭，易发生**内存泄露**问题，如图 24：

```
try {
    FileOutputStream fos = new FileOutputStream(file);
    ps = new PrintStream(fos);
} catch (FileNotFoundException e) {
    e.printStackTrace();
    return null;
} catch (NullPointerException e) {
    e.printStackTrace();
    return null;
}
return ps;
```

图 24 质量问题，内存泄漏

如图 24，这里应该采用“try-with-resources”的模式，防止出现内存泄漏的问题，需在 finally 块中加入 fos.close()。

2. 代码复杂，**不易维护**，如图 25：

```
if (noteCursor != null) {
    if (noteCursor.moveToFirst()) {
        do {
            ps.println(String.format(getFormat(FORMAT_NOTE_DATE), DateFormat.format(
                    "MMMd kk:mm",
                    noteCursor.getLong(NOTE_COLUMN_MODIFIED_DATE))));
            // Query data belong to this note
            String noteId = noteCursor.getString(NOTE_COLUMN_ID);
            exportNoteToText(noteId, ps);
        } while (noteCursor.moveToNext());
    }
    noteCursor.close();
}
```

图 25 质量问题，代码复杂

如图 25，有多个嵌套的 if 语句使得整个程序的控制流嵌套层数较多，导致程序控制结构复杂，不利于对代码的理解和维护。

3. **参数含义不明**

```
public static final int TYPE_NOTE     = 0;
public static final int TYPE_FOLDER   = 1;
public static final int TYPE_SYSTEM   = 2;

/**
```

图 26 质量问题，参数含义不明

如图 26，定义了三个全局变量参数，但是根据其名称并不能很好地判断出来它们的用途，所以这里不利于代码的理解与维护。

4. 异常处理缺失

```
private void showSoftInput() {
    //获取输入法服务
    InputMethodManager inputMethodManager = (InputMetho
    if (inputMethodManager != null) {
        inputMethodManager.toggleSoftInput(InputMethodM
    }
}
```

图 27 质量问题，无异常处理

如图 27，这里要获取输入法的服务，但是没有考虑到获取失败的情况，也没有将异常添加到日志中，所以同样不利于找出错误和完善程序。

4. 工具分析代码的质量情况

使用 Sonar Scanner 工具对小米便签开源代码进行分析，发现了 8 个 Bug、30 个安全漏洞、253 个异味，整体质量较高，但是仍有一些缺陷，主要表现在变量命名不够规范、重复使用字符串文字、方法定义、公有私有变量与函数处理等方面。

图 28 工具分析，出现的问题

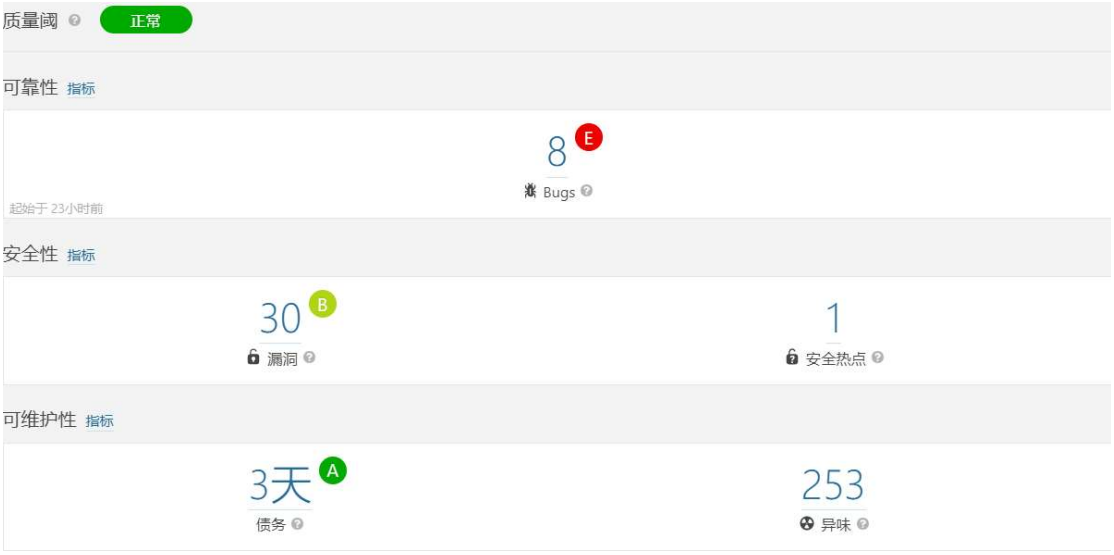


图 29 工具分析，整体概况

下面选取有代表性的错误进行分析。

1) 严重

此处字符串重复使用了 11 次，但是并没有把字符串定义为一个常量，因此报错。在字符串多次被使用时，应当使用一个常量来代替，能减少犯错的可能性，提高维护性能。类似这样的错误在小米便签代码中有很多。

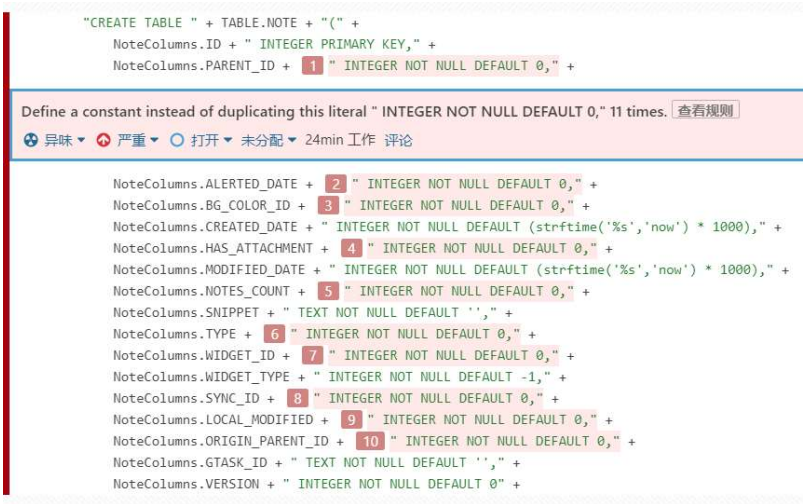


图 30 工具分析漏洞，严重问题

2) 主要

Js 变量在此处可能为 Null，如果 js 为 Null，此时第二点处会产生异常，但是代码并没有考虑将这个异常抛出，导致程序会出现问题。

图 31 工具分析漏洞，主要问题

3) 次要

此处由于变量命名不符合规范导致报错。变量命名建议使用驼峰命名法，而不是全部采用大写形式。

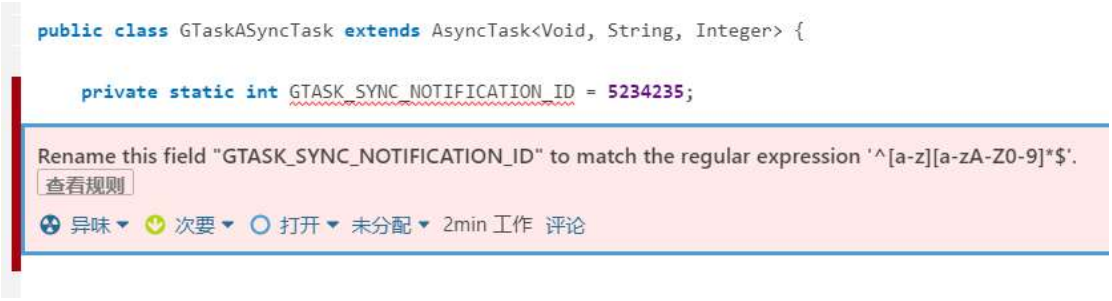


图 32 工具分析漏洞，次要问题

4) 提示

TODO 标记用于提醒开发人员需要修改代码的位置，但是开发人员有时会忘记回到这个标记，这个用于跟踪这些标记确保这些标记不被忽视。


public String getType(Uri uri) {
// TODO Auto-generated method stub
Complete the task associated to this TODO comment.
查看规则
异味
提示
打开
未分配
评论

图 33 工具分析，提示