目录

安卓反编译利器：jadx

- 最新版本：`v1.0`
- 更新时间：`20230912`

简介

介绍整理安卓逆向中静态分析之反编译器中比较好用的反编译器：jadx。包括如何下载，以及jadx的基本用法，包括从apk转java，从dex转java；然后是jadx的高级用法，包括显示坏代码、开启反混淆；然后是jadx的gui图形界面版本，包括如何打开，参数设置页面，以及参数映射关系，和其他一些页面；然后是jadx使用心得，以及help语法帮助。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

HonKit源码

- crifan/android_re_decompile_jadx: 安卓反编译利器：jadx

如何使用此HonKit源码去生成发布为电子书

详见：crifan/honkit_template: demo how to use crifan honkit template and demo

在线浏览

- 安卓反编译利器：jadx book.crifan.org
- 安卓反编译利器：jadx crifan.github.io

离线下载阅读

- 安卓反编译利器：jadx PDF
- 安卓反编译利器：jadx ePub
- 安卓反编译利器：jadx Mobi

版权和用途说明

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

其他

作者的其他电子书

本人 `crifan` 还写了其他 `150+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

关于作者

关于作者更多介绍，详见：

关于CrifanLi李茂 – 在路上

jadx概览

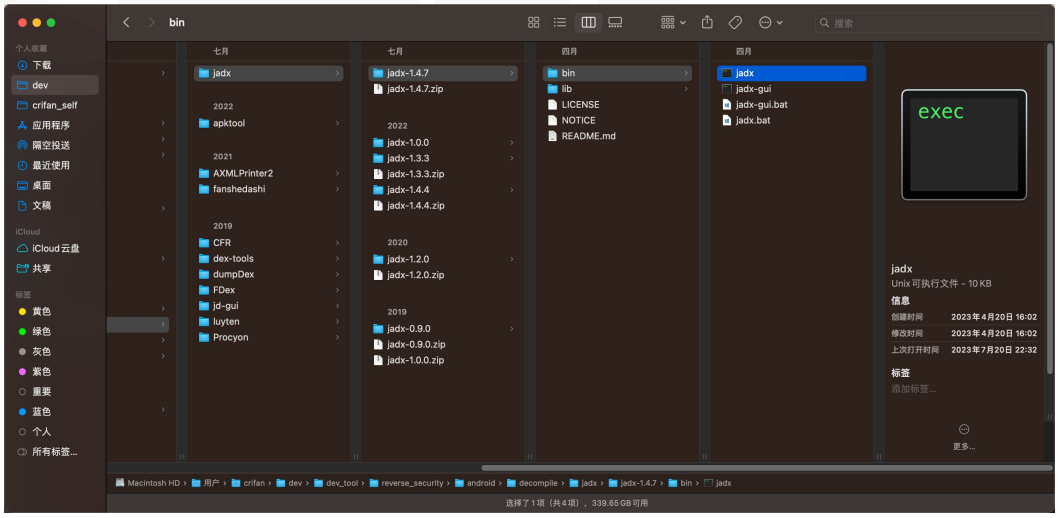
安卓逆向期间用到的，静态分析时常涉及到的，从 `apk` / `dex` 转 `java` 的反编译器，有多种：

- `jadx`
- `JEB`
- `GDA`

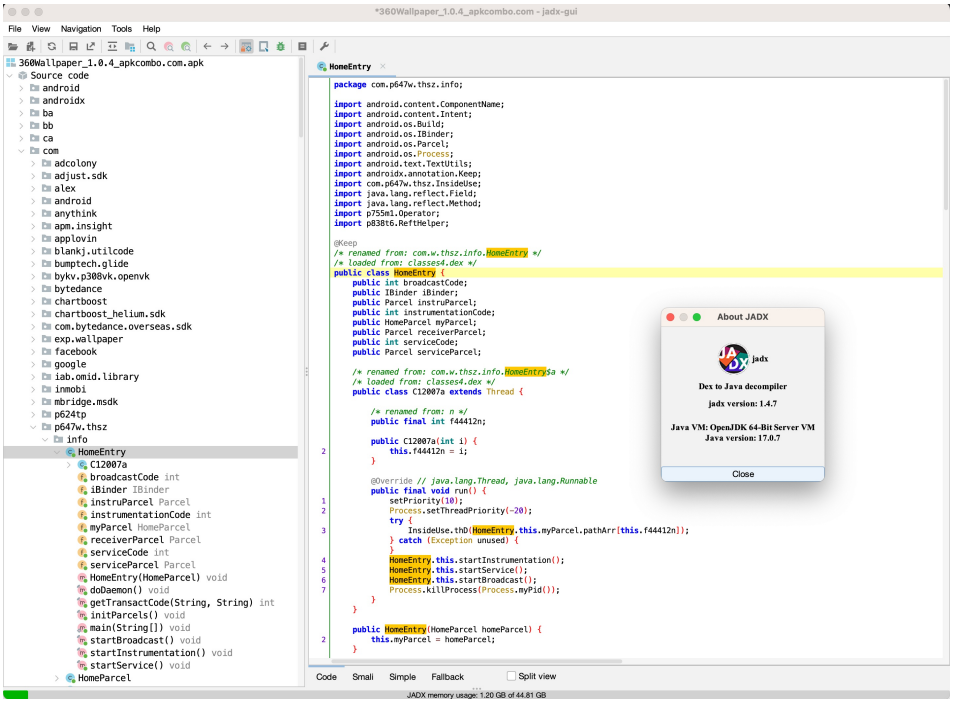
等。

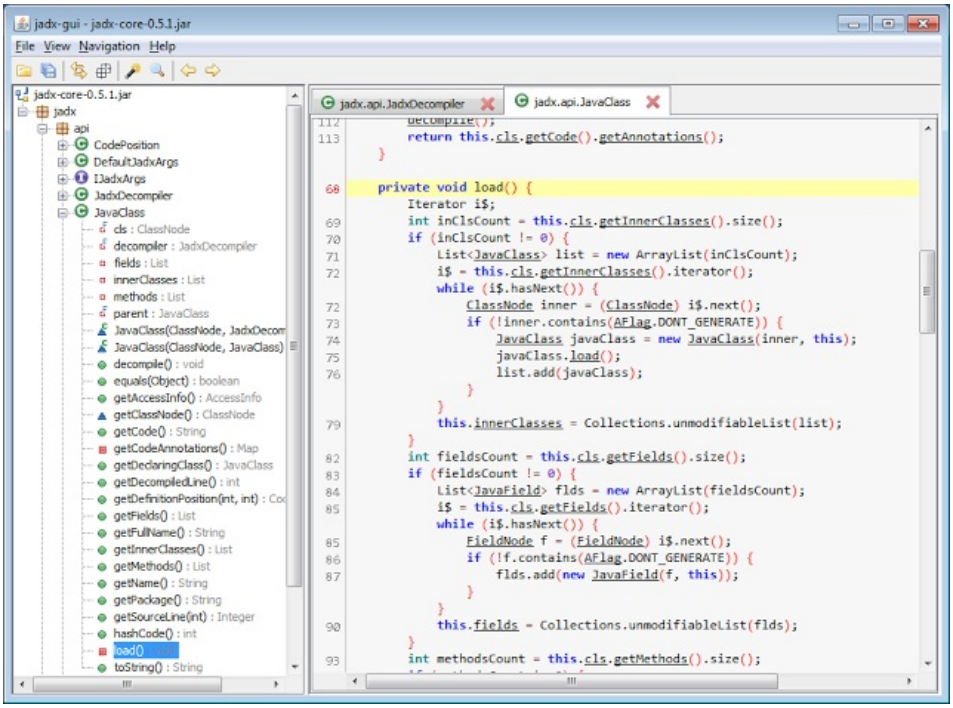
其中比较好用（之一的）是：`jadx`

jadx

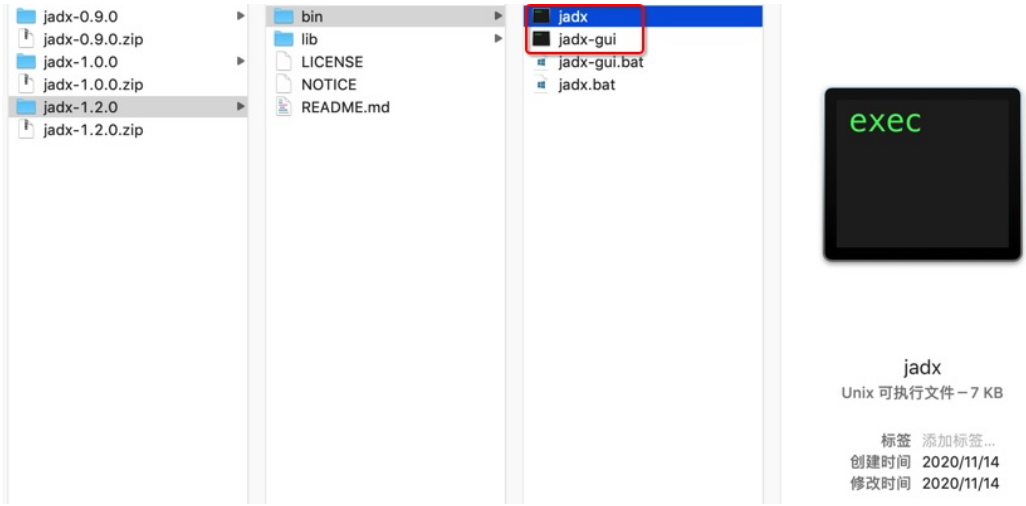
- `jadx`
  - 主页
    - skylot/jadx: Dex to Java decompiler
  - 功能
    - 从 `dex` 或 `apk` 文件中转换出 `java` 源代码的反编译器
  - 两种模式/版本
    - 命令行 版本= `command line version` = `jadx-cli` ： `jadx`
      - 

    - 图形界面 版本= `GUI` = `graphical version` ： `jadx-gui` = `JadxGUI`
      - 注：也有很多人常把GUI版本 `jadx-gui` 简称为 `jadx` ，注意不要搞混淆了
      - jadx的gui版
        - 新版

- 

- 旧版
- 

jadx下载

- 从jadx的release页面，可下载到最新版的 `jadx`
  - 比如：
    - jadx-1.2.0.zip
  - 得到 `jadx-1.2.0.zip` ，解压后的 `bin` 目录中有：
    - 图
      - 

    - 文字说明
      - 命令行版本=也称为： `jadx-cli`
        - `Mac` / `Linux` 的 `jadx`
        - `Win` 的 `jadx.bat`
      - 带图形界面的GUI版本
        - 文件
          - `Mac` / `Linux` 的 `jadx-gui`
          - `Win` 的 `jadx-gui.bat`
        - 说明
          - 双击即可运行
          - 可以用来查看反编译后的代码，也支持导出反编译后的代码

所以想要使用jadx反编译出代码，可以用

- 命令行： `jadx`
  - 直接导出代码
- 图形界面： `jadx-gui`
  - 查看反编译的结果，再导出代码

jadx基本用法

命令行：`jadx`

- 命令行：`jadx`
  - 处理`apk`
    - 语法

      ```
      jadx -d output_folder your_apk_file.apk
      ```

    - 举例

      ```
      jadx/jadx-0.9.0/bin/jadx -d from_jadx_command xiaohuasheng-v1.5.apk
      jadx/jadx-0.9.0/bin/jadx -d exported_java_src mafengwo_ziyouxing.apk
      ```

  - 处理`dex`
    - 语法

      ```
        jadx -d output_folder your_dex_file.dex
      ```

    - 举例

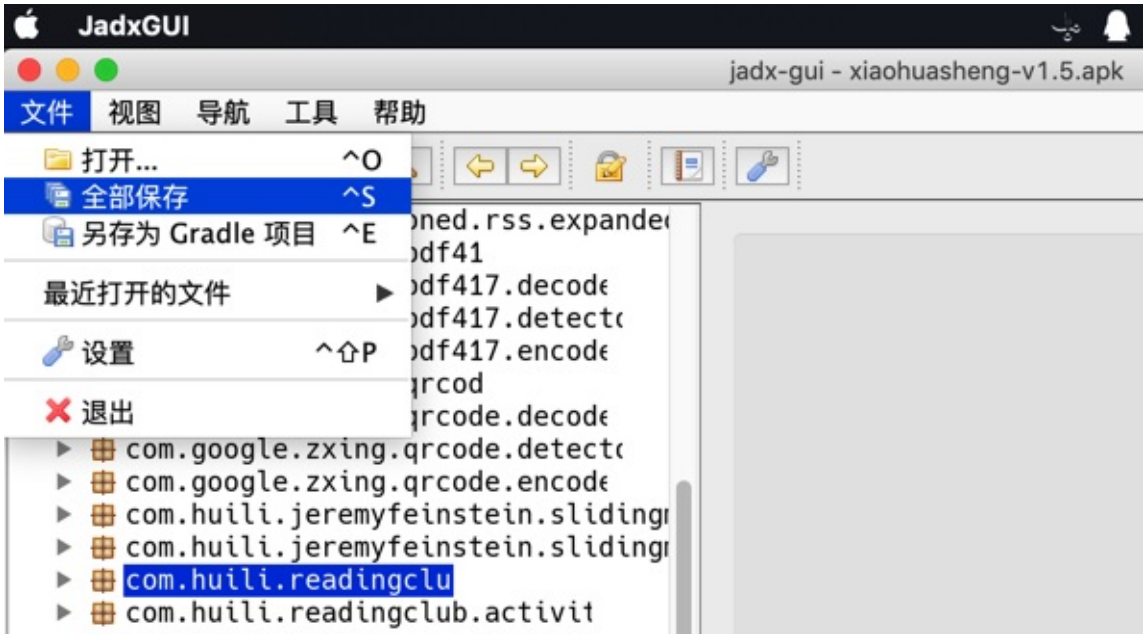
      ```
      jadx-0.9.0/bin/jadx some_dex_file.dex -d
      jadx-1.0.0/bin/jadx com.ishowedu.child.peiyin8392664.dex -d com.ishowedu.chi
      ld.peiyin8392664_java
      ```

GUI：`jadx-gui`

使用方式：双击`bin/jadx-gui`，即可打开界面

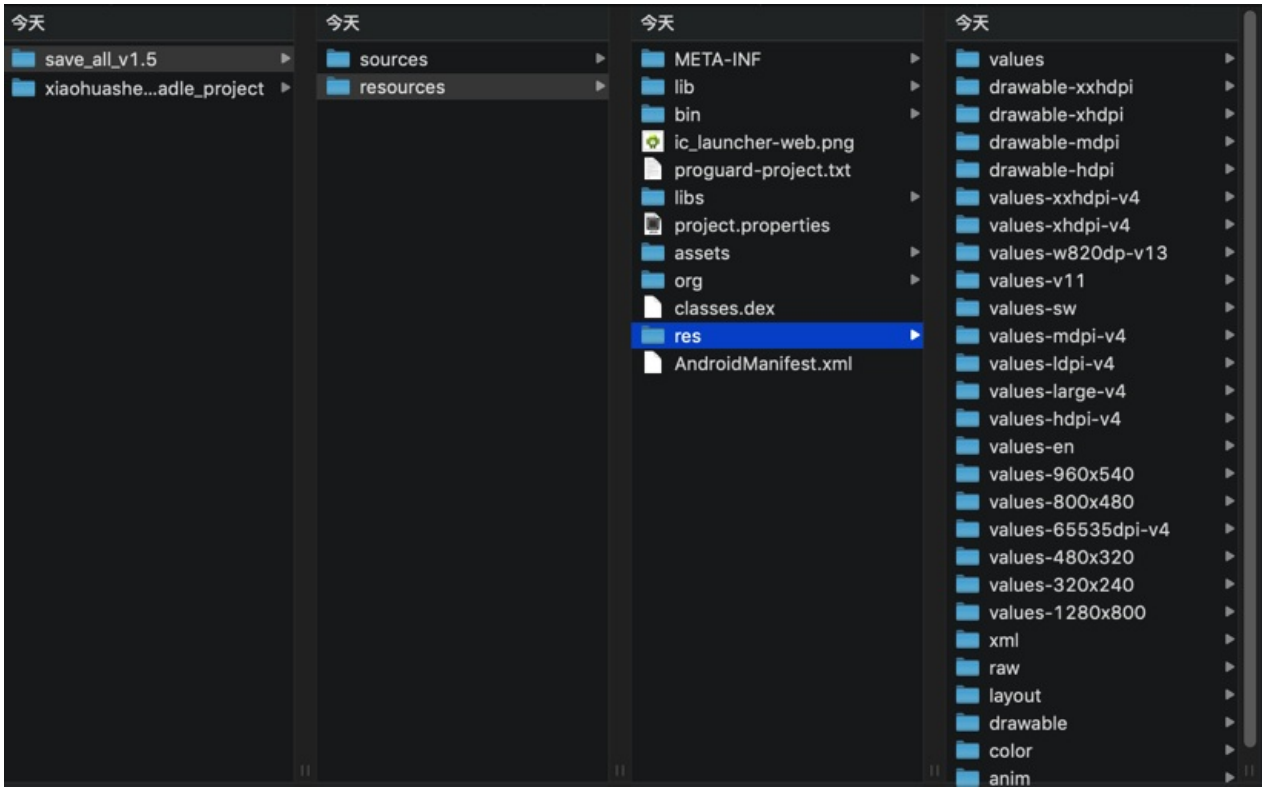
用`jadx-gui`导出全部代码

`文件 -> 全部保存`

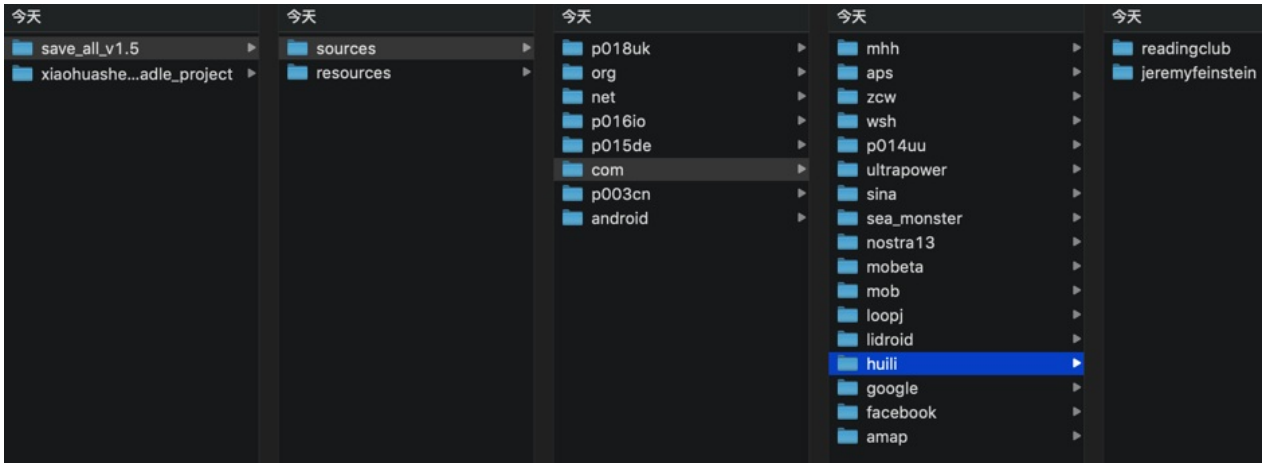


即可下载到：

- 各种资源： `resources`
- 源码： `sources`
  - 其中的 `sources` ，和 `文件 -> 另存为Gradle项目` 所导出的代码是一样的



其中就有我们希望的app的业务逻辑的代码：



jadx从apk导出java

此处用 `jadx` 工具，直接从 `apk` 转换出 `java` 源代码

准备

下载 `jadx` ：

从skylot/jadx: Dex to Java decompiler的releases页面下载最新版本，比如jadx-0.9.0.zip

解压得到：

- `bin/jadx` ： 命令行版本
  - `bin/jadx.bat` ： Windows版
- `bin/jadx-gui` ： 带图形界面的版本
  - `bin/jadx-gui.bat` ： Windows版

jadx命令行版

直接通过命令行去反编译，效率最高。

语法：

```
jadx -d output_folder your_apk_file.apk
```

举例：

```
jadx/jadx-0.9.0/bin/jadx -d from_jadx_command xiaohuasheng-v1.5.apk
jadx/jadx-0.9.0/bin/jadx -d exported_java_src mafengwo_ziyouxing.apk
jadx -d . LiftFileManager-FileClean_1.3.1_Apkpure.apk
jadx -d . 360Wallpaper_1.0.4_apkcombo.com.apk
```

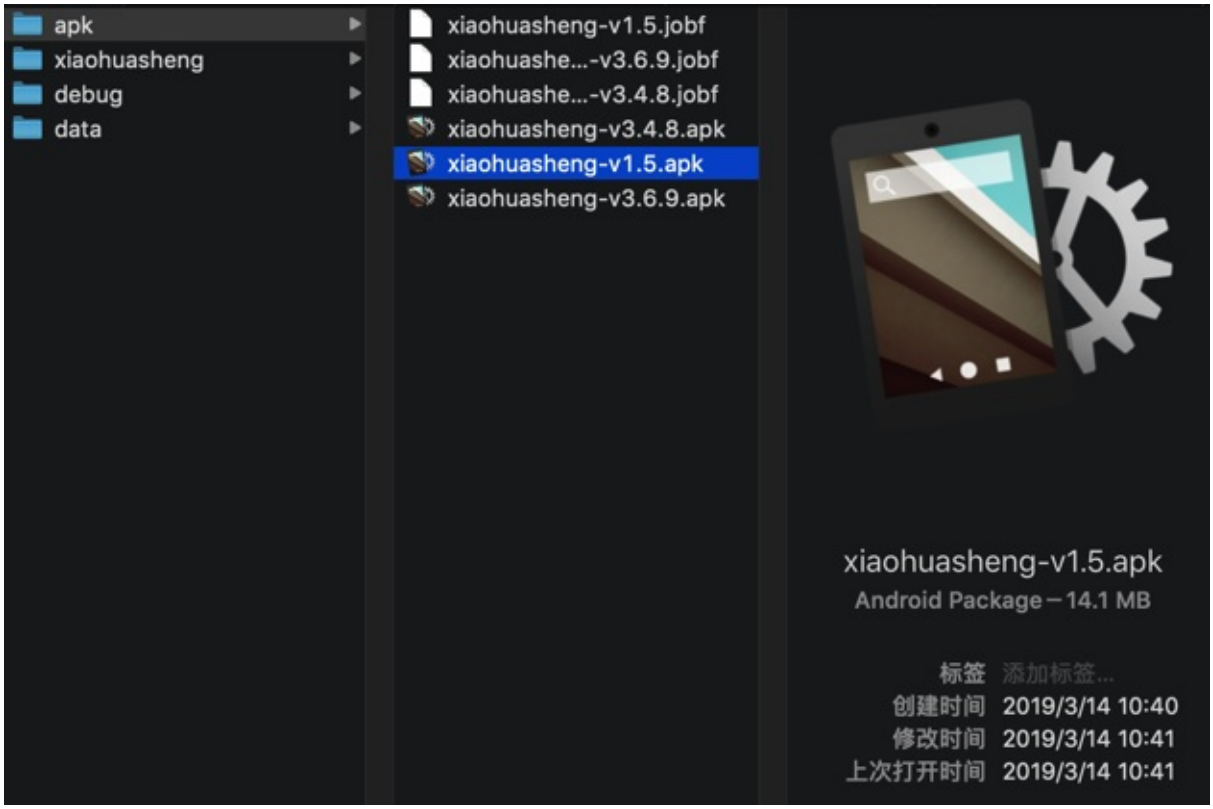
输出举例：

```
➜  input jadx -d . 360Wallpaper_1.0.4_apkcombo.com.apk
INFO  - loading ...
INFO  - processing ...
ERROR - finished with errors, count: 74
```

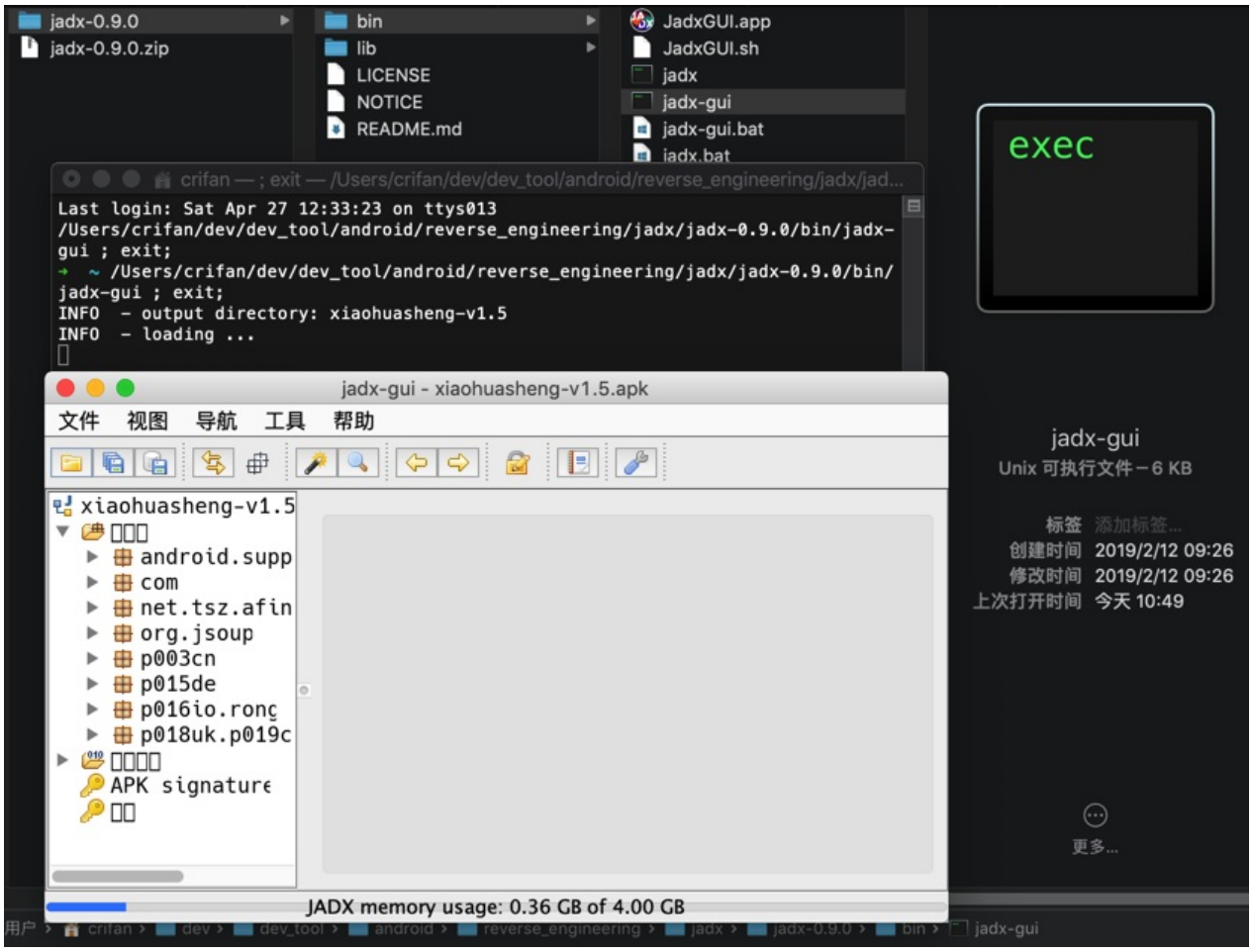
jadx gui图形界面版

下面以 `jadx-gui` (已被我改名为 `JadxGUI`)为例去解释。

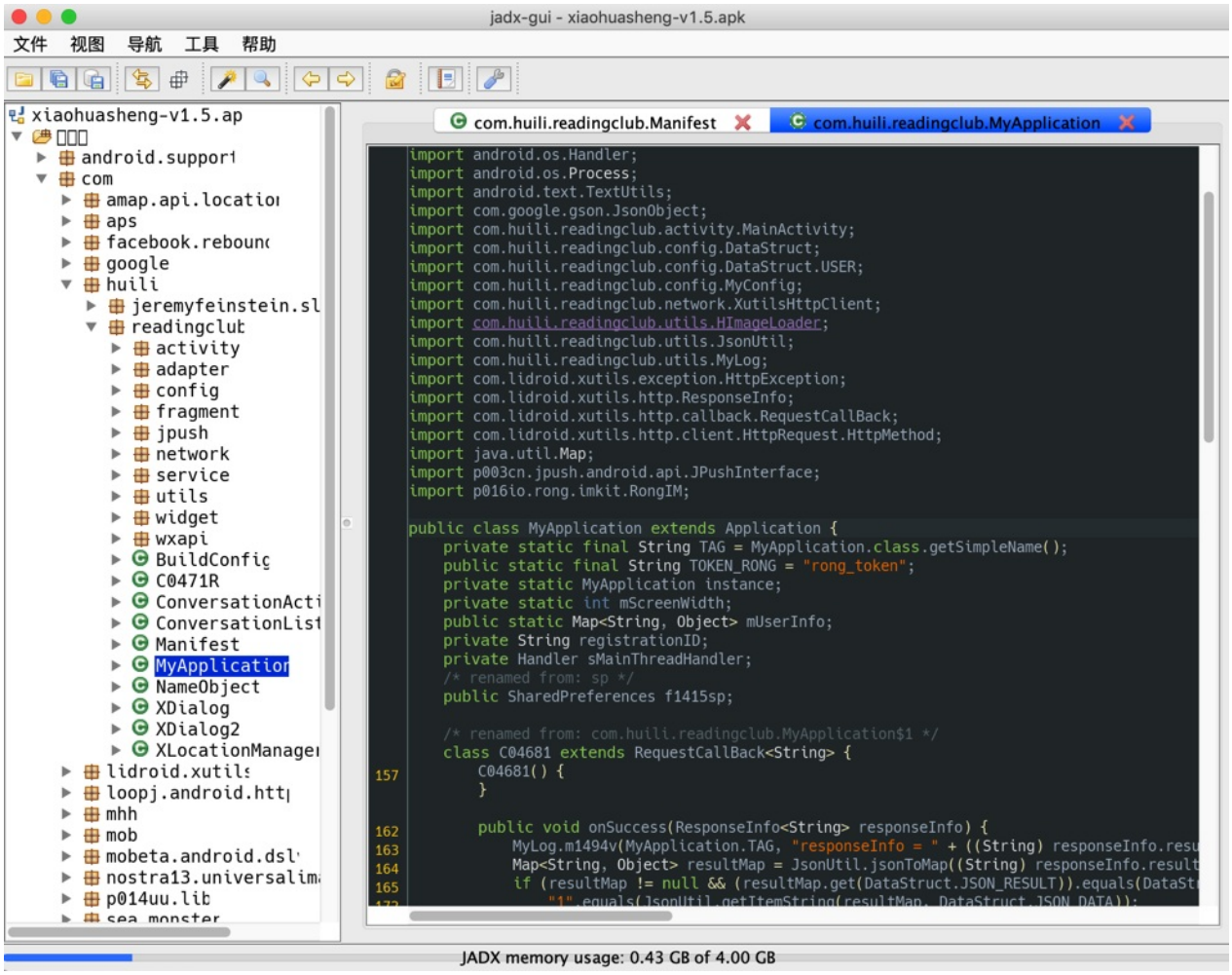
对于此处v1.5这种没有加固的apk：



`jadx` （此处指的是 `jadx-gui` ）打开后：



可以看到源码：

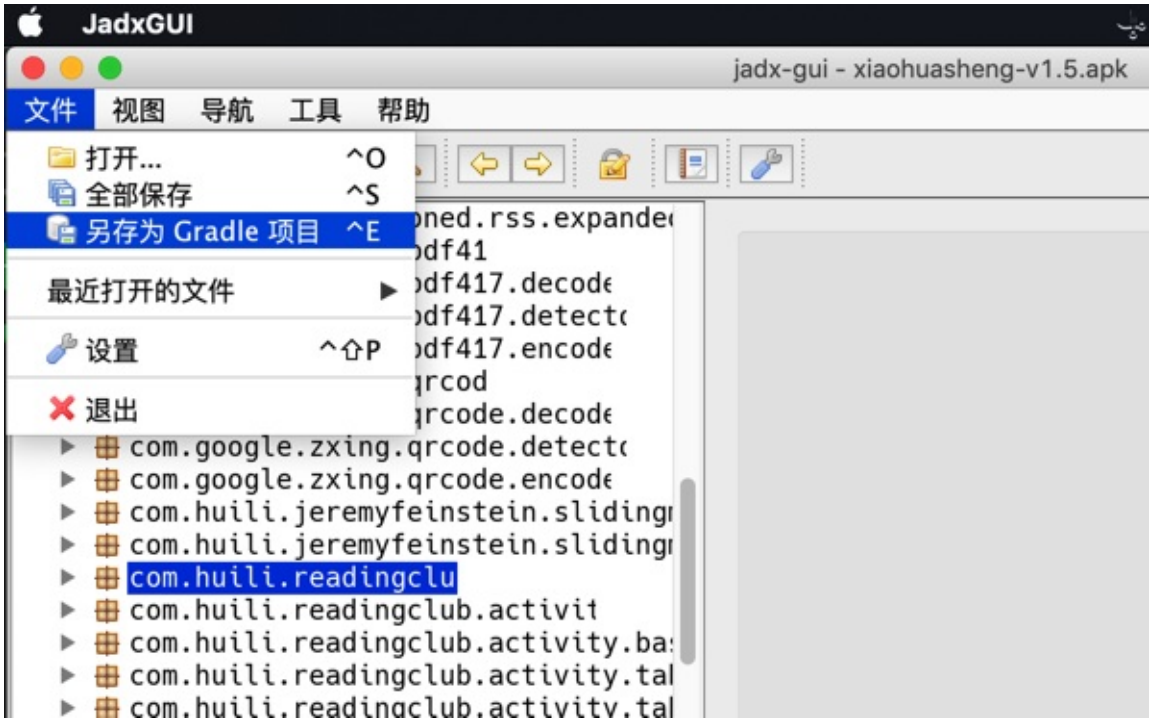


想要导出全部源码，则可以去

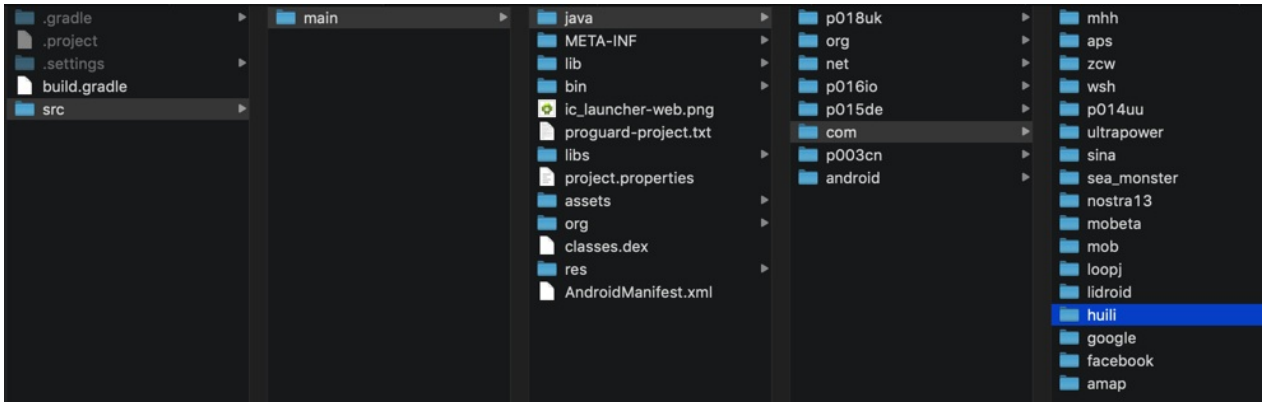
`文件 -> 全部保存`

或

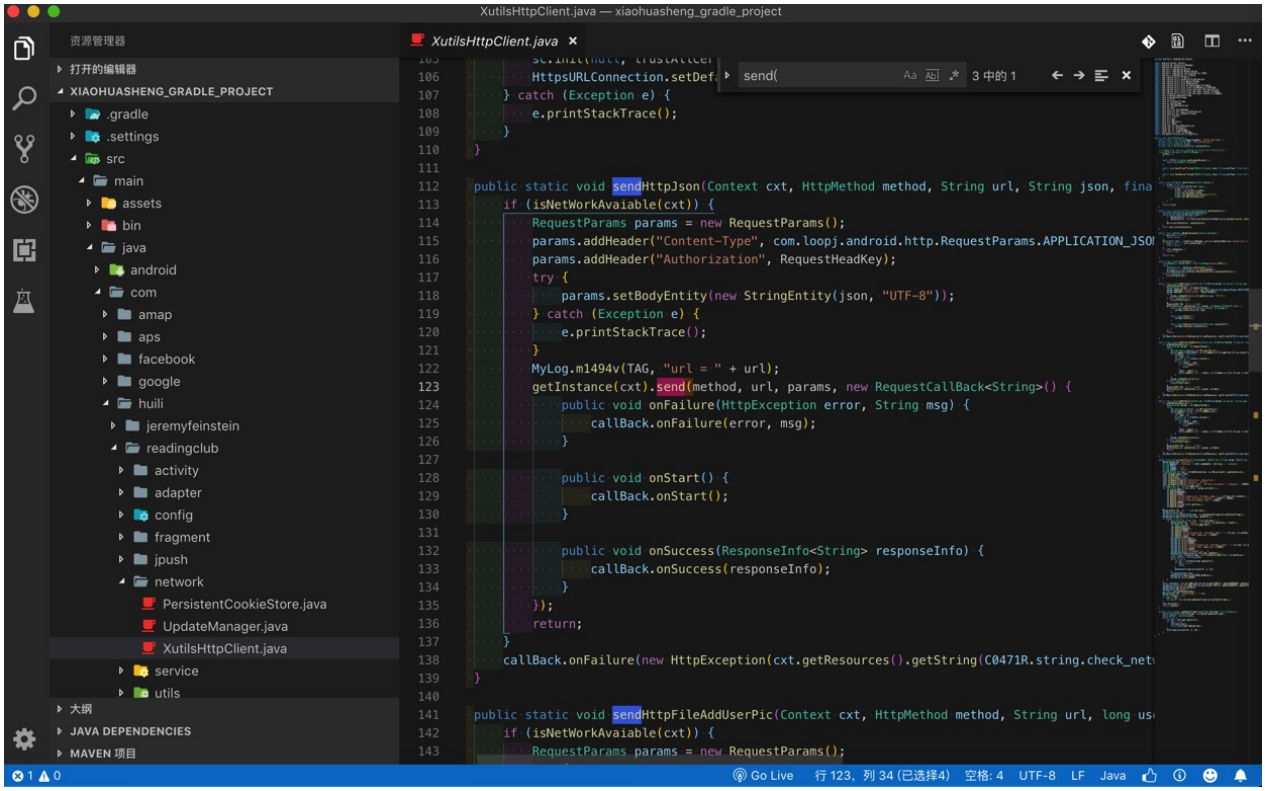
`文件 -> 另存为Gradle项目`



即可导出全部的代码：



用VSCode打开后即可找到（希望研究的app对应的业务逻辑的）代码：



jadx从dex导出java

`jadx` 可以直接从 `dex` 导出 `java` 源码：

切换到要导出代码的目录，已有dex文件要导出，则可以直接运行：

- 语法=命令

```
jadx dex_file.dex -d output_folder
```

  - 举例

```
jadx-0.9.0/bin/jadx dex_file.dex -d .

jadx/jadx-1.0.0/bin/jadx com.ishowedu.child.peiyin8392664.dex -d com.ishowedu.child.peiyin8392664_java
```

- 输出
  - 即可转换出源代码到当前目录下，输出有：
    - `resources`
    - `sources`
      - 有你要的源码
  - 转换速度还是不错的

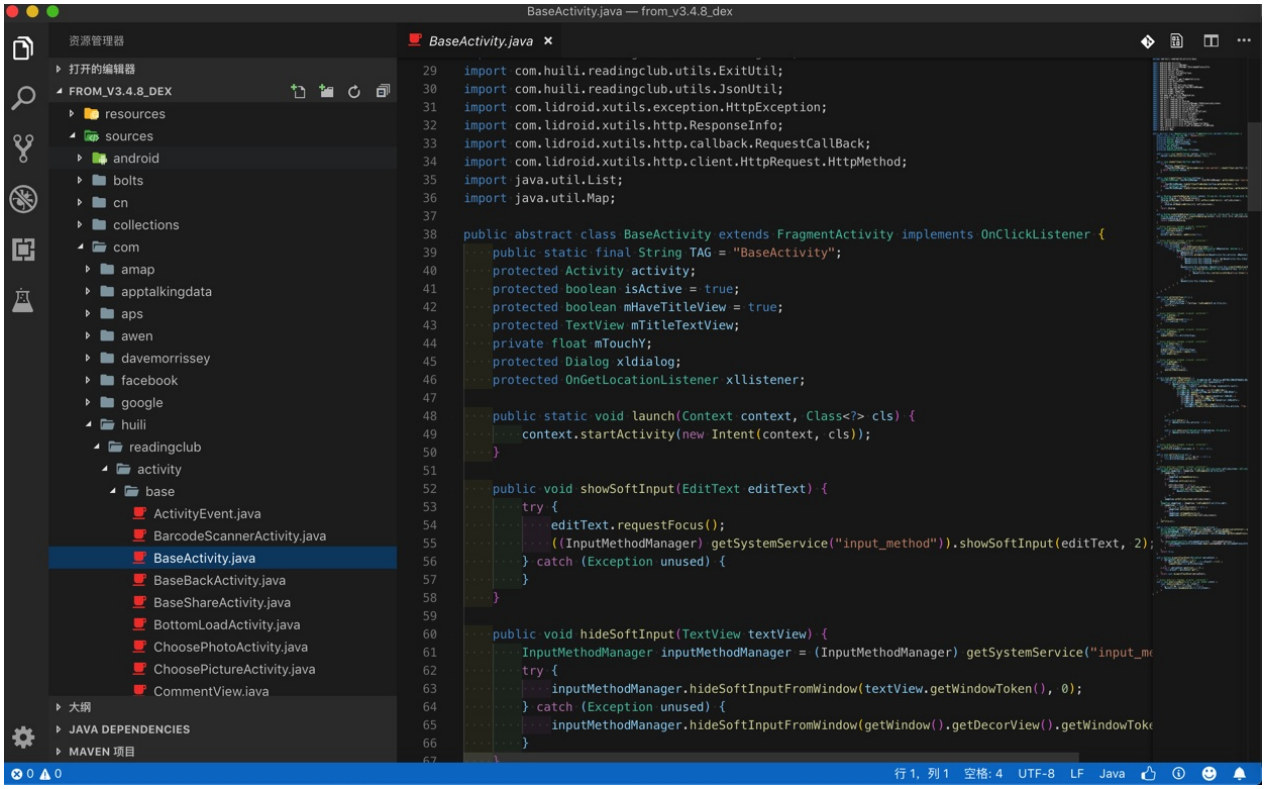
举例

```
from_v3.4.8_dex /Users/crifan/dev/dev_tool/android/reverse_engineering/jadx/jadx-0.9.0/bin/jadx ../../../../xiaohuasheng/app_hook_dump_dex/FDex2/v3.4.8/com.huili.readingclub8825612.dex -d .
...
中间很多错误
...
WARN  - Found 75 references to unknown classes
ERROR - 6 errors occurred in following nodes:
ERROR -   Method: android.support.v4.provider.FontsContractCompat.getFontFromProvider(android.content.Context, android.support.v4.provider.FontRequest, java.lang.String, android.os.CancellationSignal):android.support.v4.provider.FontsContractCompat$FontInfo[]
ERROR -   Method: cn.addapp.pickers.util.LogUtils.getTraceElement():java.lang.String
ERROR -   Method: cn.jiguang.a.a.b.c.a(android.os.Message):void
ERROR -   Method: cn.jiguang.d.b.f.a(int):boolean
ERROR -   Method: cn.jiguang.d.d.m.a(android.content.Context, boolean):java.util.List java.io.File>
ERROR -   Method: cn.jiguang.g.e.a(java.lang.String, java.util.Map):cn.jiguang.g.e
WARN  - 2299 warnings in 454 nodes
ERROR - finished with errors
```

转换后：

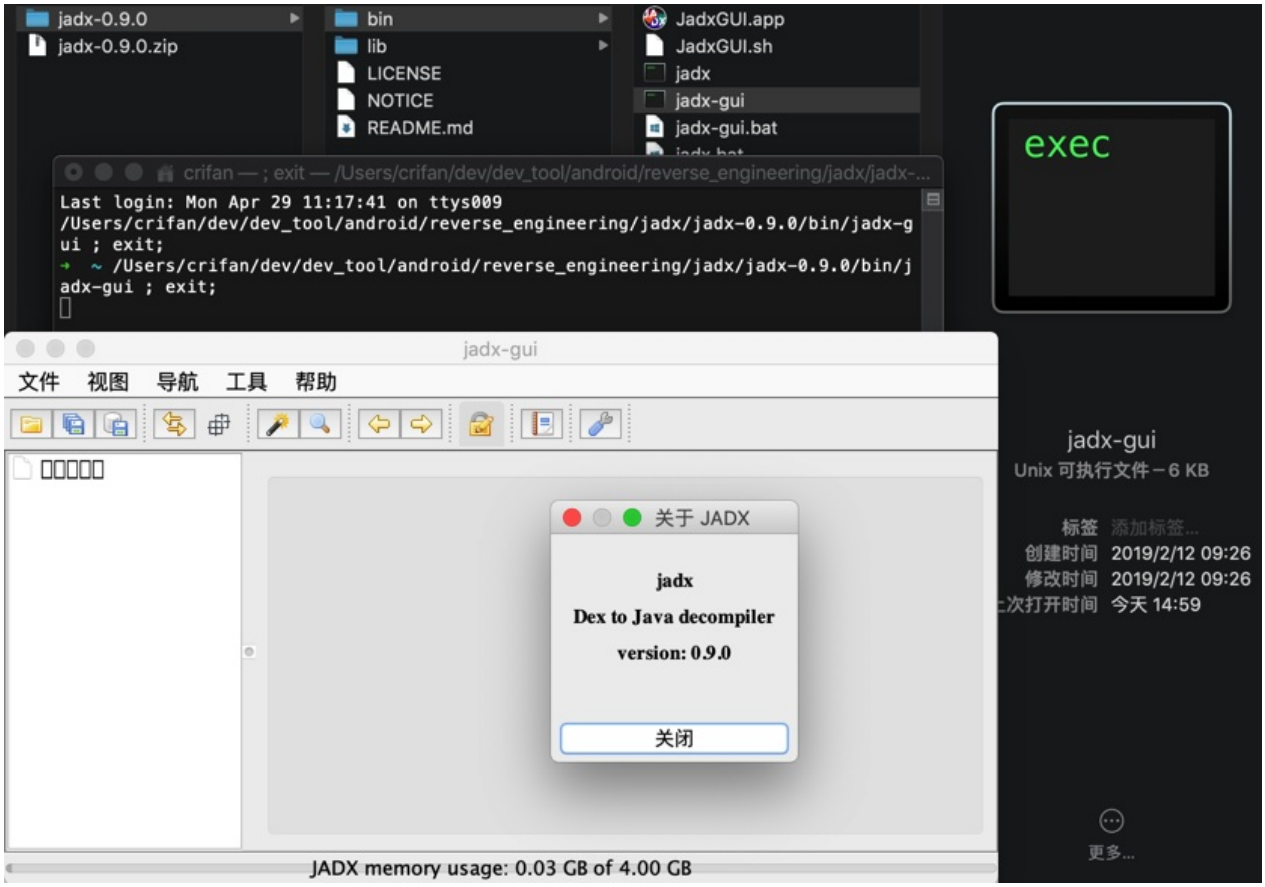
```
➜  from_v3.4.8_dex ll
total 0
drwxr-xr-x   3 crifan  staff    96B  4 29 15:29 resources
drwxr-xr-x  13 crifan  staff   416B  4 29 15:30 sources
```

转换后的代码用VSCode去打开的效果：

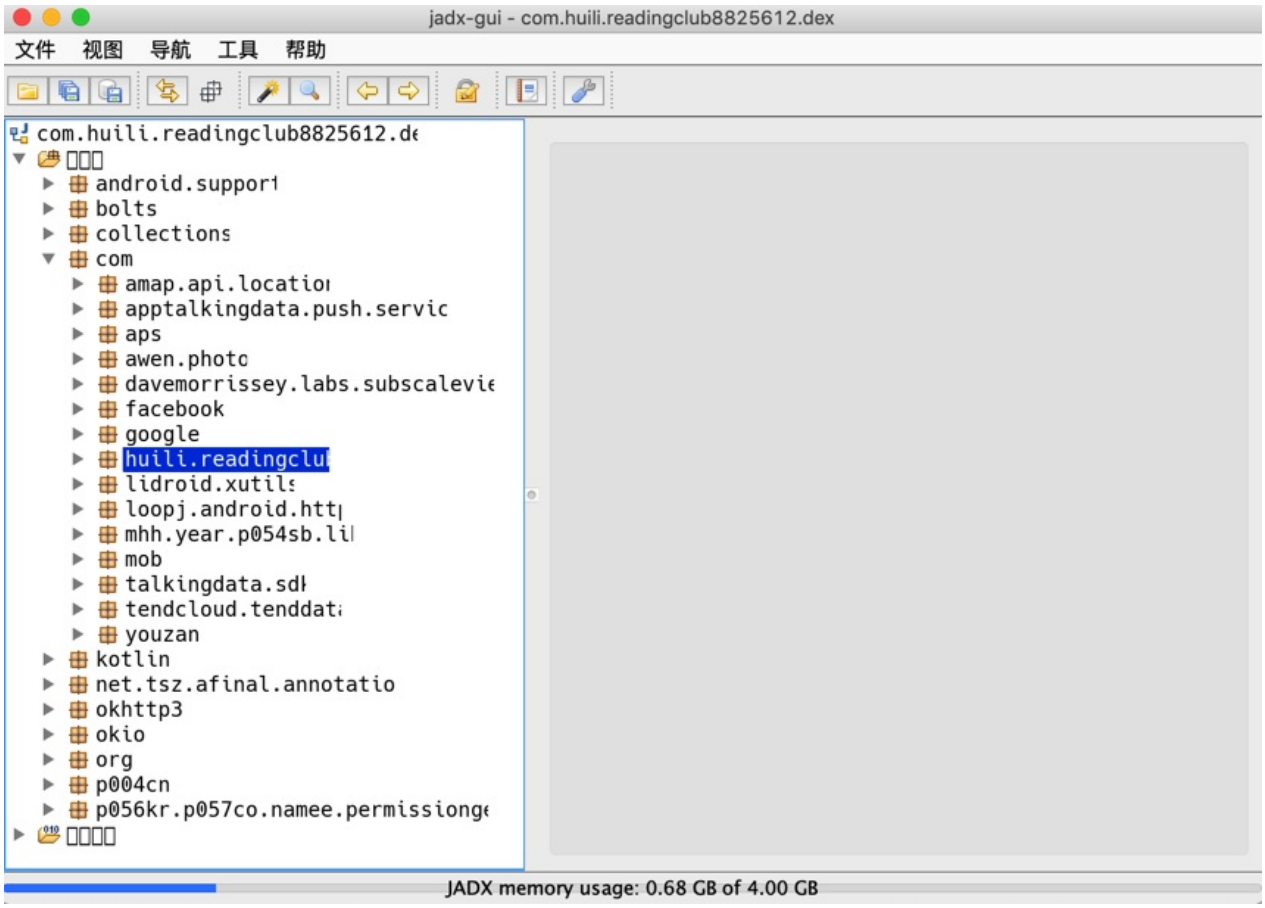


`jadx-gui` 查看和导出代码

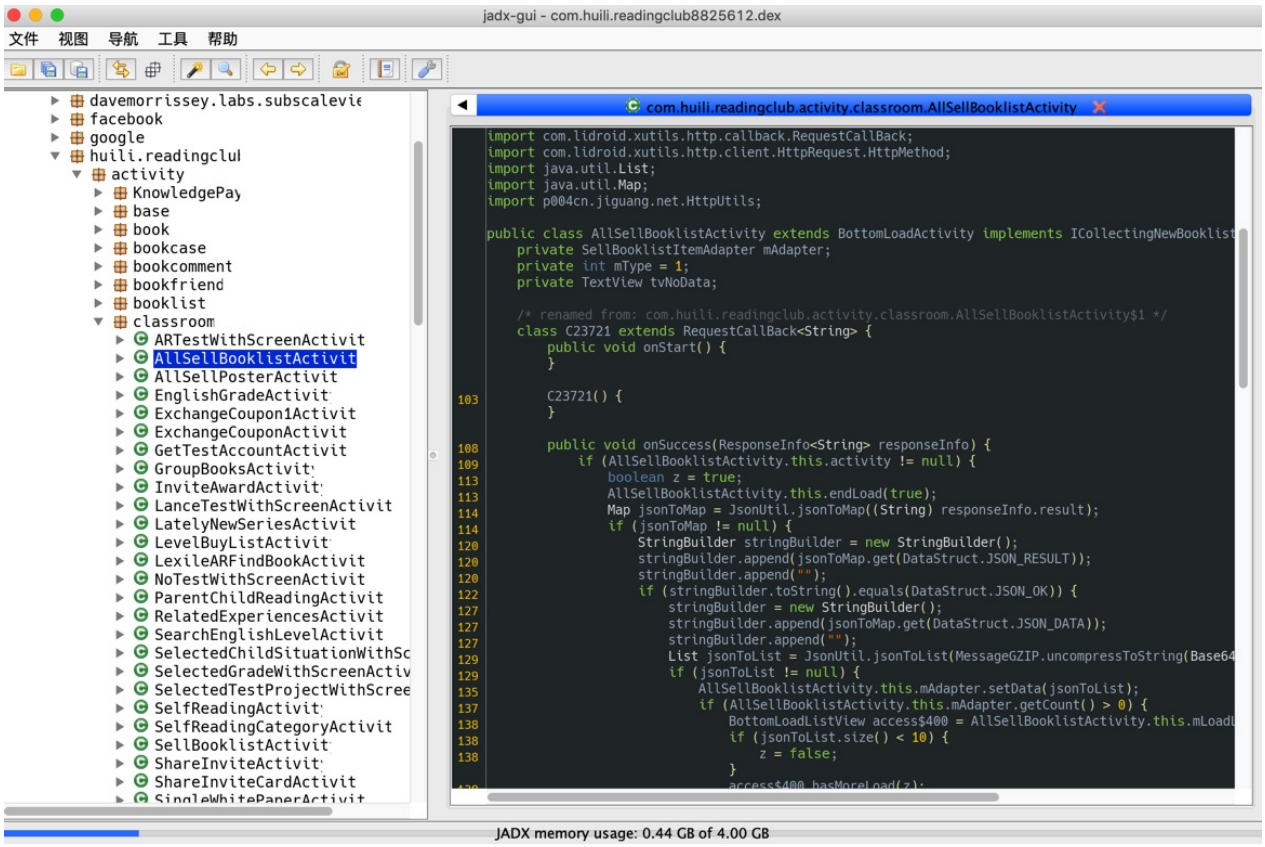
双击 `jadx-gui` 即可运行：



然后去打开对应的jar文件：`com.huili.readingclub8825612-dex2jar.jar`，即可看到包含了app业务逻辑的代码结构和包名：

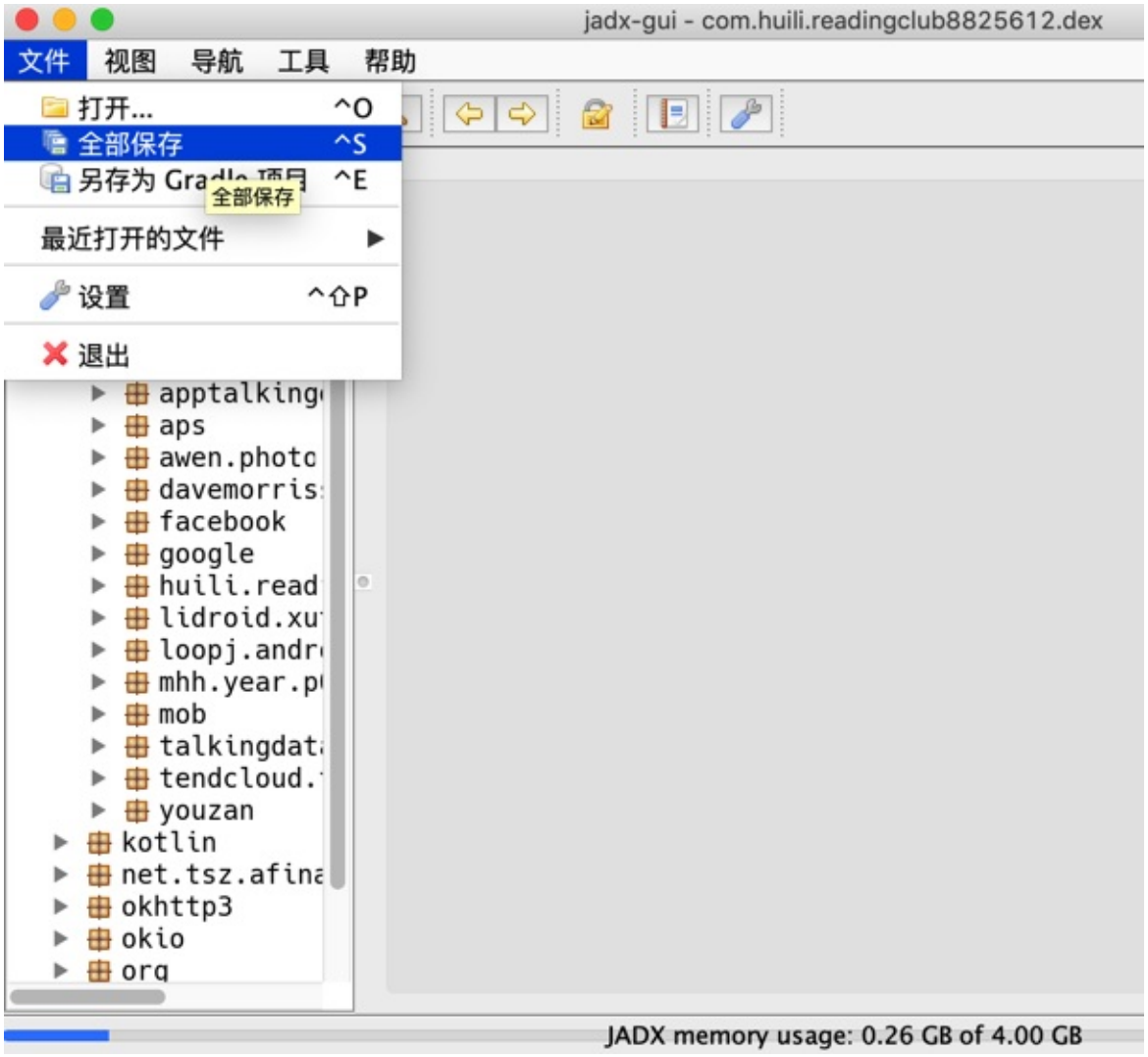


然后展开后可以看到详细的代码：

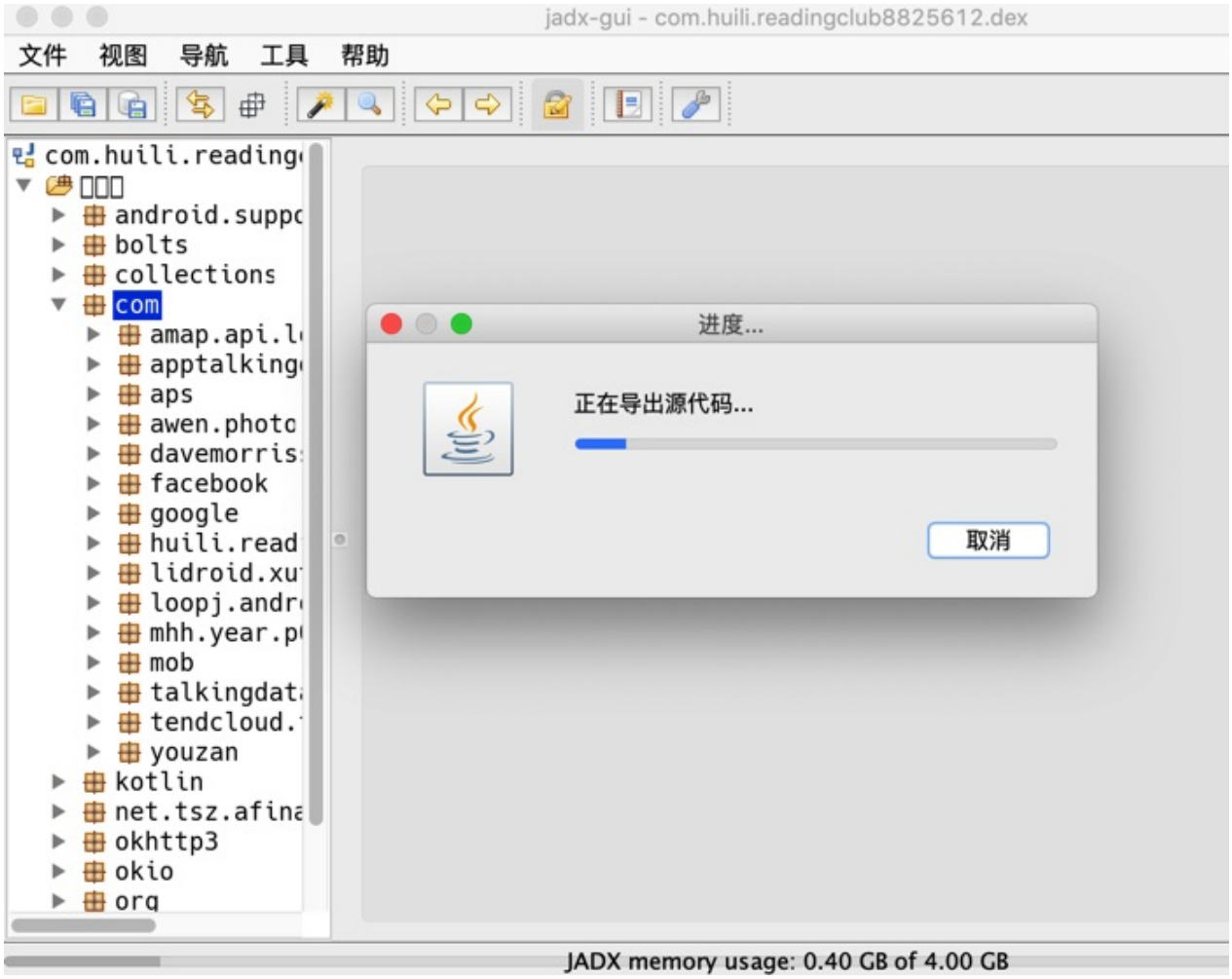


然后如果想要导出全部代码，则可以去：

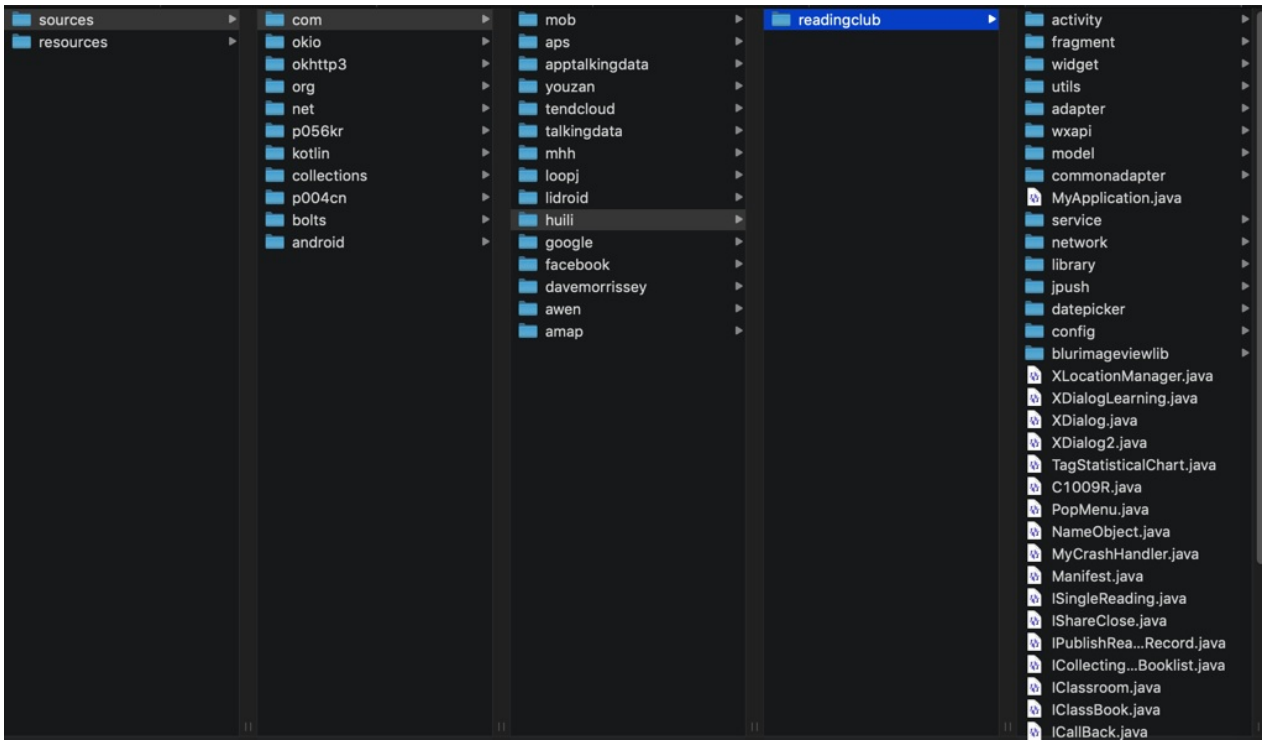
```
File -> Save All
```


jadx-gui - com.huili.readingclub8825612.dex
文件
视图
导航
工具
帮助
打开...
^O
全部保存
^S
另存为 Gradle 项目
^E
全部保存
最近打开的文件
设置
^⇧P
退出
apptalking
aps
awen.photo
davemorris
facebook
google
huili.read
lidroid.xu
loopj.andr
mhh.year.p
mob
talkingdat
tendcloud.
youzan
kotlin
net.tsz.afina
okhttp3
okio
org
JADX memory usage: 0.26 GB of 4.00 GB

然后稍等片刻：



即可在导出的 `sources` 文件夹中找到你要的源码：



jadx高级用法

下面介绍一些，相对来说算是jadx高级的用法。

显示坏代码

用jadx反编译出的java代码中，有时候会看到类似的输出：

```
    /*
        Code decompiled incorrectly, please refer to instructions dump.
        To view partially-correct add '--show-bad-code' argument
    */
```

举例：

- `360Wallpaper_jadx/sources/androidx/core/content/pm/ShortcutManagerCompat.java`

```
    /* JADX WARN: Removed duplicated region for block: B:7:0x002b  */
    /*
        Code decompiled incorrectly, please refer to instructions dump.
        To view partially-correct add '--show-bad-code' argument
    */
    private static List<ShortcutInfoChangeListener> getShortcutInfoListeners(Context r8) {
        /*
            java.util.List<androidx.core.content.pm.ShortcutInfoChangeListener> r0 = androidx.core.content.pm.ShortcutManagerCompat.sShortcutInfoChangeListeners
            if (r0 != 0) goto L71
            java.util.ArrayList r0 = new java.util.ArrayList
            r0.<init>()
            android.content.pm.PackageManager r1 = r8.getPackageManager()
            android.content.Intent r2 = new android.content.Intent
            java.lang.String r3 = "androidx.core.content.pm.SHORTCUT_LISTENER"
            r2.<init>(r3)
            java.lang.String r3 = r8.getPackageName()
            r2.setPackage(r3)
            r3 = 128(0x80, float:1.794E-43)
            java.util.List r1 = r1.queryIntentActivities(r2, r3)
            java.util.Iterator r1 = r1.iterator()
...
...
...
        L6b:
            java.util.List<androidx.core.content.pm.ShortcutInfoChangeListener> r8 = androidx.core.content.pm.ShortcutManagerCompat.sShortcutInfoChangeListeners
            if (r8 != 0) goto L71
            androidx.core.content.pm.ShortcutManagerCompat.sShortcutInfoChangeListeners = r0
        L71:
            java.util.List<androidx.core.content.pm.ShortcutInfoChangeListener> r8 = androidx.core.content.pm.ShortcutManagerCompat.sShortcutInfoChangeListeners
            return r8
        */
        throw new UnsupportedOperationException("Method not decompiled: androidx.core.content.pm.ShortcutManagerCompat.getShortcutInfoListeners(android.content.Context):java.util.List");
```

```
    }
```

其含义是：

当jadx反编译某个类=输出的单个java文件期间，部分内容，比如某个类的某个函数、某个函数中部分的代码等，无法完整的反编译=反编译期间遇到一些无法解析的错误，此时，就会显示出上述提示

而我们，其实对于部分解析出错的代码，并不是特别关心

反编译结果虽然有错误，但是对于伪代码用来查看大致的代码逻辑，则总体影响不大

所以希望：忽略这些解释的错误，尽可能多的显示出反编译的结果（有些个别的错误，可以忽略）

为了实现此目的，则可以去：

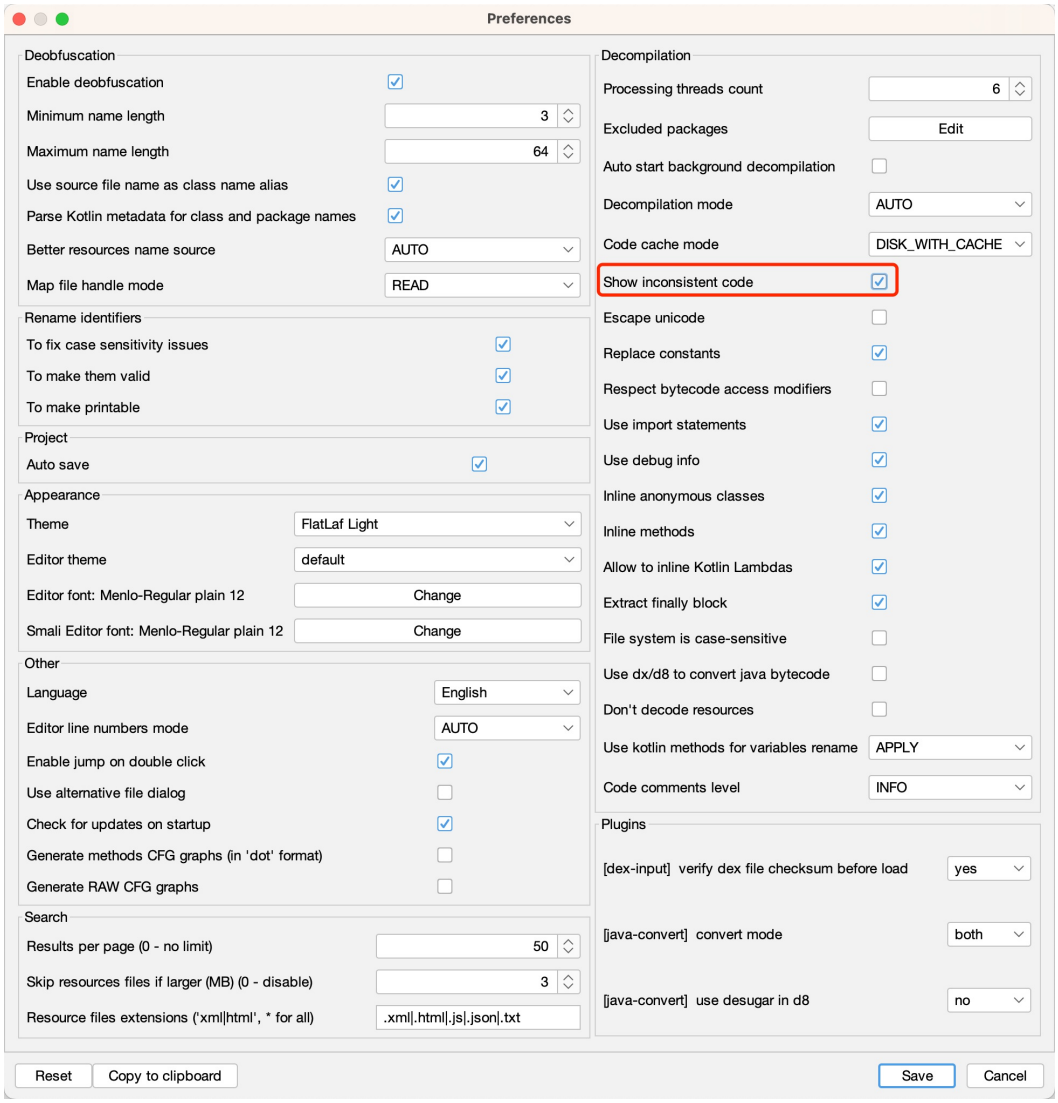
- 开启坏代码

如何开启坏代码功能

- 如何开启坏代码功能 = 开启坏代码的具体操作方式
  - cli=命令行：加上 `--show-bad-code`
    - 举例

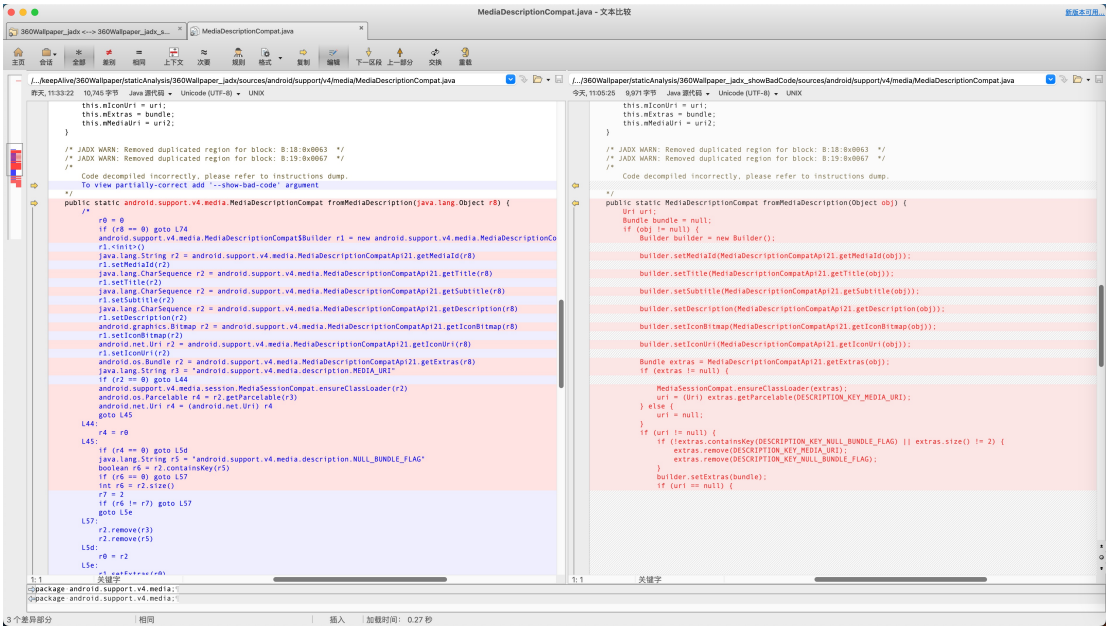
      ```
      jadx --show-bad-code -d .\ 360Wallpaper_1.0.4_apkcombo.com.apk
      ```

  - GUI=图形界面中：勾选： `Show inconsistent code`
    - 举例



开启坏点前后的效果对比

- 文件：`360Wallpaper_jadx/sources/android/support/v4/media/MediaDescriptionCompat.java`
  - 截图对比



- 代码对比
  - 开启坏代码之前

```
/* JADX WARN: Removed duplicated region for block: B:18:0x0063  */
/* JADX WARN: Removed duplicated region for block: B:19:0x0067  */
/*
  Code decompiled incorrectly, please refer to instructions dump.
  To view partially-correct add '--show-bad-code' argument
*/
public static                          MediaDescriptionCompat fromMediaDescr
iption(          Object r8) {
  /*
      r0 = 0
      if (r8 == 0) goto L74
      android.support.v4.media.MediaDescriptionCompat$Builder r1 = new andro
id.support.v4.media.MediaDescriptionCompat$Builder
      r1.<init>()
      java.lang.String r2 = android.support.v4.media.MediaDescriptionCompatA
pi21.getMediaId(r8)
      r1.setMediaId(r2)
      java.lang.CharSequence r2 = android.support.v4.media.MediaDescriptionC
ompatApi21.getTitle(r8)
      r1.setTitle(r2)
      java.lang.CharSequence r2 = android.support.v4.media.MediaDescriptionC
ompatApi21.getSubtitle(r8)
      r1.setSubtitle(r2)
      java.lang.CharSequence r2 = android.support.v4.media.MediaDescriptionC
ompatApi21.getDescription(r8)
      r1.setDescription(r2)
      android.graphics.Bitmap r2 = android.support.v4.media.MediaDescription
CompatApi21.getIconBitmap(r8)
      r1.setIconBitmap(r2)
      android.net.Uri r2 = android.support.v4.media.MediaDescriptionCompatAp
i21.getIconUri(r8)
      r1.setIconUri(r2)
      android.os.Bundle r2 = android.support.v4.media.MediaDescriptionCompat
Api21.getExtras(r8)
      java.lang.String r3 = "android.support.v4.media.description.MEDIA_URI"
```

```
      if (r2 == 0) goto L44
      android.support.v4.media.session.MediaSessionCompat.ensureClassLoader(
r2)
      android.os.Parcelable r4 = r2.getParcelable(r3)
      android.net.Uri r4 = (android.net.Uri) r4
      goto L45
  L44:
      r4 = r0
  L45:
      if (r4 == 0) goto L5d
      java.lang.String r5 = "android.support.v4.media.description.NULL_BUNDL
E_FLAG"
      boolean r6 = r2.containsKey(r5)
      if (r6 == 0) goto L57
      int r6 = r2.size()
      r7 = 2
      if (r6 != r7) goto L57
      goto L5e
  L57:
      r2.remove(r3)
      r2.remove(r5)
  L5d:
      r0 = r2
  L5e:
      r1.setExtras(r0)
      if (r4 == 0) goto L67
      r1.setMediaUri(r4)
      goto L6e
  L67:
      android.net.Uri r0 = android.support.v4.media.MediaDescriptionCompatAp
i23.getMediaUri(r8)
      r1.setMediaUri(r0)
  L6e:
      android.support.v4.media.MediaDescriptionCompat r0 = r1.build()
      r0.mDescriptionObj = r8
  L74:
      return r0
  */
  throw new UnsupportedOperationException("Method not decompiled: android.su
pport.v4.media.MediaDescriptionCompat.fromMediaDescription(java.lang.Object)
:android.support.v4.media.MediaDescriptionCompat");
}
```

- 开启坏代码之后

```
/* JADX WARN: Removed duplicated region for block: B:18:0x0063  */
/* JADX WARN: Removed duplicated region for block: B:19:0x0067  */
/*
  Code decompiled incorrectly, please refer to instructions dump.
*/
public static MediaDescriptionCompat fromMediaDescription(Object obj) {
  Uri uri;
  Bundle bundle = null;
  if (obj != null) {
      Builder builder = new Builder();
```

```
        builder.setMediaId(MediaDescriptionCompatApi21.getMediaId(obj));
        builder.setTitle(MediaDescriptionCompatApi21.getTitle(obj));
        builder.setSubtitle(MediaDescriptionCompatApi21.getSubtitle(obj));
        builder.setDescription(MediaDescriptionCompatApi21.getDescription(obj))
;
        builder.setIconBitmap(MediaDescriptionCompatApi21.getIconBitmap(obj));
        builder.setIconUri(MediaDescriptionCompatApi21.getIconUri(obj));
        Bundle extras = MediaDescriptionCompatApi21.getExtras(obj);
        if (extras != null) {
            MediaSessionCompat.ensureClassLoader(extras);
            uri = (Uri) extras.getParcelable(DESCRIPTION_KEY_MEDIA_URI);
        } else {
            uri = null;
        }
        if (uri != null) {
            if ( extras.containsKey(DESCRIPTION_KEY_NULL_BUNDLE_FLAG) || extras
.size() != 2) {
                extras.remove(DESCRIPTION_KEY_MEDIA_URI);
                extras.remove(DESCRIPTION_KEY_NULL_BUNDLE_FLAG);
            }
            builder.setExtras(bundle);
            if (uri == null) {
                builder.setMediaUri(uri);
            } else {
                builder.setMediaUri(MediaDescriptionCompatApi23.getMediaUri(obj
));
            }
            MediaDescriptionCompat build = builder.build();
            build.mDescriptionObj = obj;
            return build;
        }
        bundle = extras;
        builder.setExtras(bundle);
        if (uri == null) {
        }
        MediaDescriptionCompat build2 = builder.build();
        build2.mDescriptionObj = obj;
        return build2;
    }
    return null;
}
```

反混淆

- 反混淆概述
  - 混淆
    - 很多app都做了代码混淆处理，导致解析出来的（类、属性、函数等）变量名，都是混淆后的值
      - 典型的是 `a` 、 `b` 、 `c` 、 `d` 等值
      - 让（逆向的）你看难直接看出代码逻辑
  - 反混淆
    - 部分反编译器，比如此处的jadx，支持（基本的）反混淆
      - 所谓的基本的反混淆，主要指的是：变量的重名名
        - 比如，至少可以把 `a` 变成 `C12657a` ， `w` 变成 `p647w` 等，至少看起来能稍微容易区分出不同变量值，让（逆向的）你，看反混淆后的代码的逻辑，稍微降低点难度

Jadx中如何开启反混淆 `deobfuscation`

- jadx中开启反混淆
  - cli=命令行：加参数 `--deobf`
    - 举例

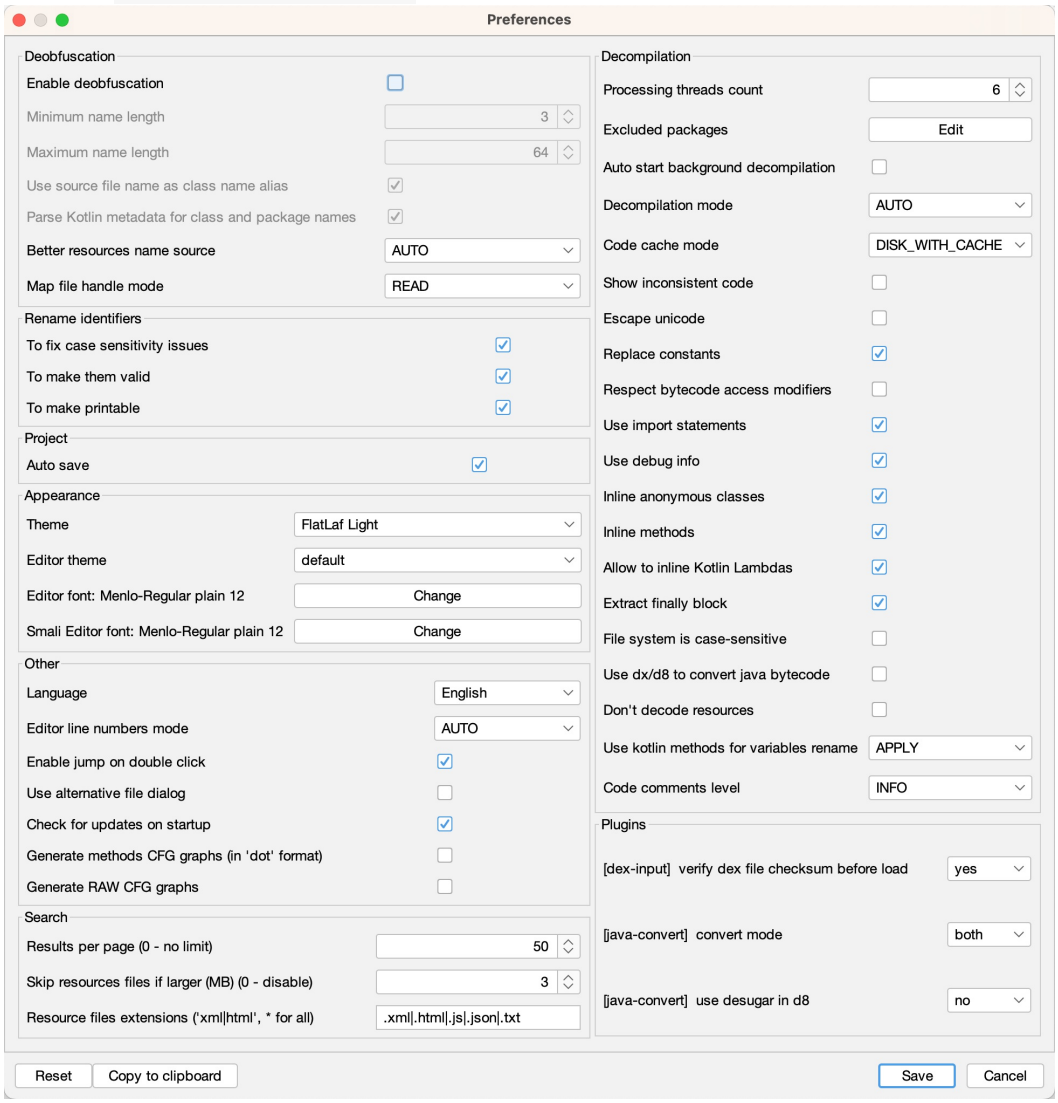
```
jadx --deobf -d . 360Wallpaper_1.0.4_apkcombo.com.apk
```

    - 更多相关子参数，可以根据需求去设置

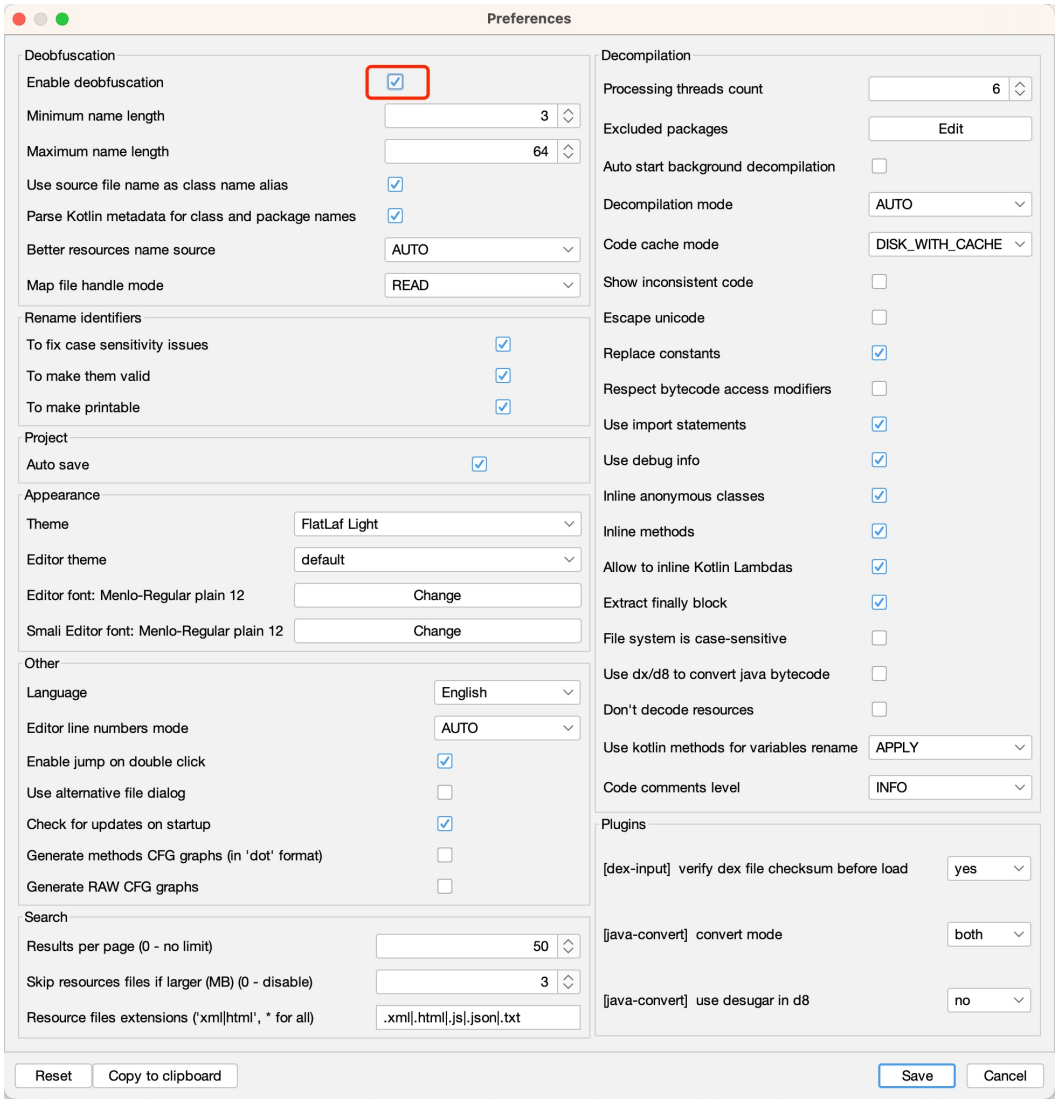
```
  --deobf                               - activate deobfuscation
  --deobf-min                           - min length of name, renamed if short
er, default: 3
  --deobf-max                           - max length of name, renamed if longe
r, default: 64
  --deobf-cfg-file                      - deobfuscation map file, default: sam
e dir and name as input file with '.jobf' extension
  --deobf-cfg-file-mode                 - set mode for handle deobfuscation ma
p file:
                                          'read' - read if found, don't save (
default)
                                          'read-or-save' - read if found, save
 otherwise (don't overwrite)
                                          'overwrite' - don't read, always sav
e
                                          'ignore' - don't read and don't save
  --deobf-use-sourcename                - use source file name as class name a
lias
  --deobf-parse-kotlin-metadata         - parse kotlin metadata to class and p
ackage names
  --deobf-res-name-source               - better name source for resources:
                                          'auto' - automatically select best n
ame (default)
                                          'resources' - use resources names
                                          'code' - use R class fields names
```

- 详见：jadx的help语法

- gui=图形界面：勾选 `Enable deobfuscation`
  - 默认没有勾选 `Enable deobfuscation` ，各子参数也是灰色不可设置



  - 勾选上 `Enable deobfuscation`
    - 注：相关子项参数，则可以（根据自己需要去）设置了



反混淆前后效果对比

cli命令行版本输出代码结果详细对比

- `AndroidMenifest.xml` 反混淆前后对比

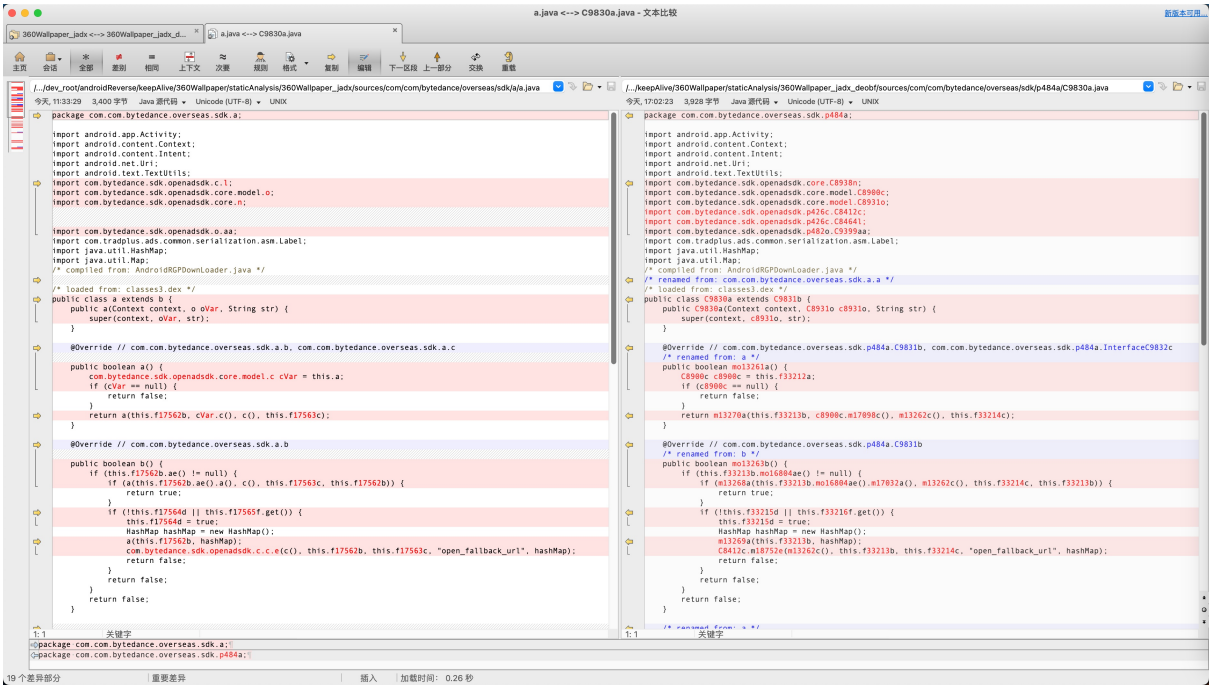
-
-
- 类名目录对比

- 某个类：ComponetDialog

-

某个类的代码详细对比

总体效果的截图对比：

- 

详细代码：

- 混淆前：`360Wallpaper_jadx/sources/com/com/bytedance/overseas/sdk/a/a.java`

```
package                                 ;


import              Activity;
import                  Context;
import                  Intent;
import              Uri;
```

```
import              TextUtils;
import                                ;
import                                    ;
import                                  ;
import                                   ;
import                                          Label;
import           HashMap;
import           Map;
/* compiled from: AndroidRGPDownLoader.java */
/* loaded from: classes3.dex */
public class a extends b {
    public a(Context context, o oVar, String str) {
        super(context, oVar, str);
    }


    @Override // com.com.bytedance.overseas.sdk.a.b, com.com.bytedance.overseas.sdk.a.c
    public boolean a() {
        com.bytedance.sdk.openadsdk.core.model.c cVar = this.a;
        if (cVar == null) {
            return false;
        }
        return a(this.f17562b, cVar.c(), c(), this.f17563c);
    }


    @Override // com.com.bytedance.overseas.sdk.a.b
    public boolean b() {
        if (this.f17562b.ae() != null) {
            if (a(this.f17562b.ae().a(), c(), this.f17563c, this.f17562b)) {
                return true;
            }
            if (!this.f17564d || this.f17565f.get()) {
                this.f17564d = true;
                HashMap hashMap = new HashMap();
                a(this.f17562b, hashMap);
                com.bytedance.sdk.openadsdk.c.c.e(c(), this.f17562b, this.f17563c, "ope
n_fallback_url", hashMap);
                return false;
            }
            return false;
        }
        return false;
    }


    public static boolean a(o oVar, String str, Context context, String str2) {
        Intent a;
        if (oVar != null && oVar.av() == 0) {
            return false;
        }
        try {
            if (TextUtils.isEmpty(str) || (a = aa.a(context, str)) == null) {
                return false;
            }
            a.putExtra("START_ONLY_FOR_ANDROID", true);
```

```
            if (!(context instanceof Activity)) {
                a.addFlags(Label.FORWARD_REFERENCE_TYPE_SHORT);
            }
            context.startActivity(a);
            HashMap hashMap = new HashMap();
            a(oVar, hashMap);
            com.bytedance.sdk.openadsdk.c.c.e(context, oVar, str2, "click_open", hashMap
);
            return true;
        } catch (Throwable unused) {
        }
        return false;
    }


    public static boolean a(String str, Context context, String str2, o oVar) {
        try {
            if (TextUtils.isEmpty(str)) {
                return false;
            }
            Uri parse = Uri.parse(str);
            Intent intent = new Intent("android.intent.action.VIEW");
            intent.setData(parse);
            if (!(context instanceof Activity)) {
                intent.addFlags(Label.FORWARD_REFERENCE_TYPE_SHORT);
            }
            HashMap hashMap = new HashMap();
            a(oVar, hashMap);
            com.bytedance.sdk.openadsdk.c.c.e(n.a(), oVar, str2, "open_url_app", hashMap
);
            context.startActivity(intent);
            l.a().a(oVar, str2);
            return true;
        } catch (Throwable unused) {
            return false;
        }
    }


    private static void a(o oVar, Map<String, Object> map) {
        if (oVar != null) {
            map.put("auto_click", Boolean.valueOf(oVar.a()));
        }
    }
}
```

- 混淆后：（类相应的也改名了）`360Wallpaper_jadx_deobf/sources/com/com/bytedance/overseas/sdk/p484a/C9830a.java`

```
package                                    ;


import            Activity;
import               Context;
```

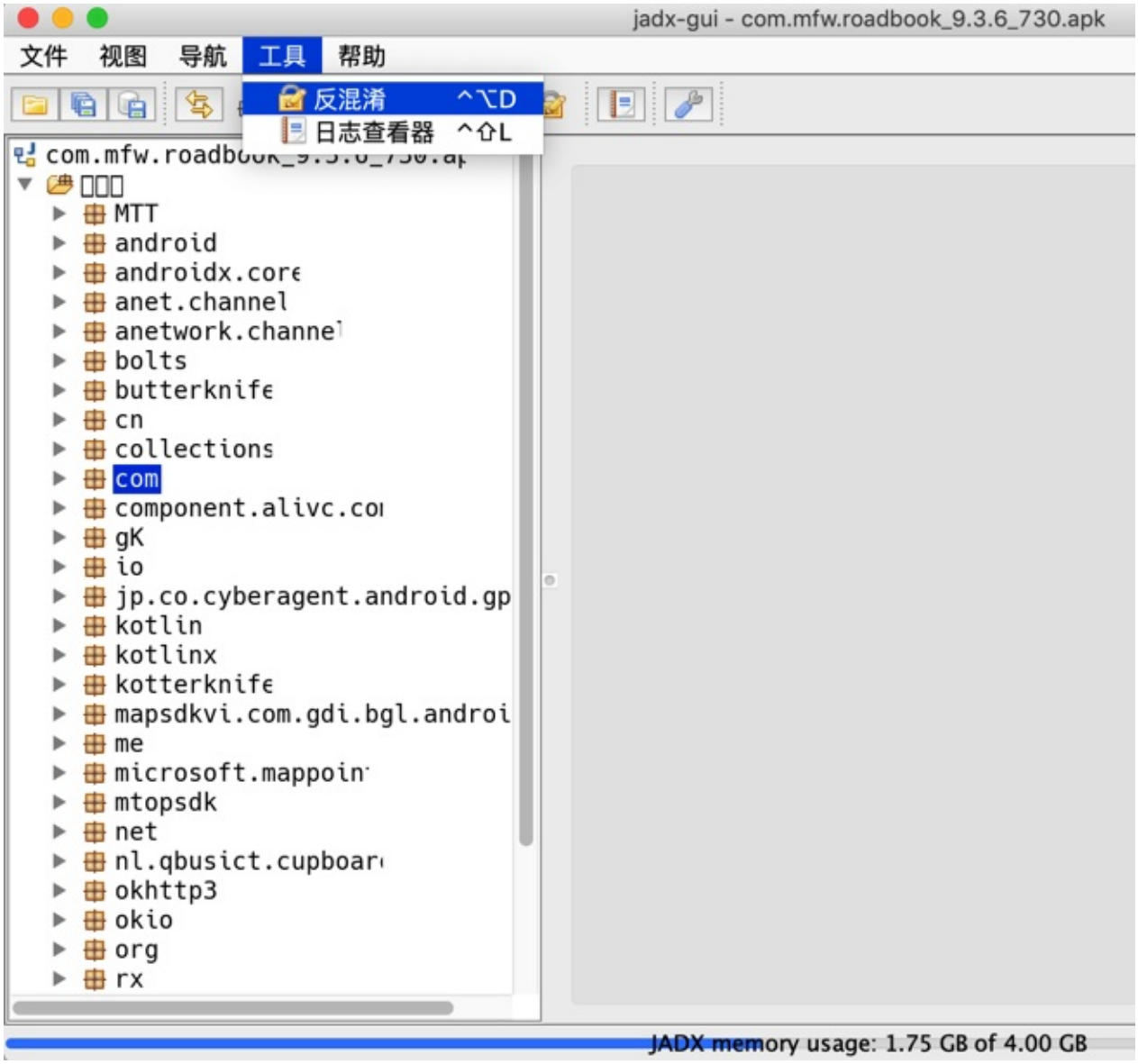
```
import                 Intent;
import             Uri;
import              TextUtils;
import                                  C8936n;
import                                       C8900c;
import                                       C8931o;
import                                   C8412c;
import                                   C8464l;
import                                   C9399aa;
import                                         Label;
import           HashMap;
import           Map;
/* compiled from: AndroidRGPDownLoader.java */
/* renamed from: com.com.bytedance.overseas.sdk.a.a */
/* loaded from: classes3.dex */
public class C9830a extends C9831b {
    public C9830a(Context context, C8931o c8931o, String str) {
        super(context, c8931o, str);
    }


    @Override // com.com.bytedance.overseas.sdk.p484a.C9831b, com.com.bytedance.oversea
s.sdk.p484a.InterfaceC9832c
    /* renamed from: a */
    public boolean mo13261a() {
        C8900c c8900c = this.f33212a;
        if (c8900c == null) {
            return false;
        }
        return m13270a(this.f33213b, c8900c.m17098c(), m13262c(), this.f33214c);
    }


    @Override // com.com.bytedance.overseas.sdk.p484a.C9831b
    /* renamed from: b */
    public boolean mo13263b() {
        if (this.f33213b.mo16804ae() != null) {
            if (m13268a(this.f33213b.mo16804ae().m17032a(), m13262c(), this.f33214c, th
is.f33213b)) {
                return true;
            }
            if (!this.f33215d || this.f33216f.get()) {
                this.f33215d = true;
                HashMap hashMap = new HashMap();
                m13269a(this.f33213b, hashMap);
                C8412c.m18752e(m13262c(), this.f33213b, this.f33214c, "open_fallback_ur
l", hashMap);
                return false;
            }
            return false;
        }
        return false;
    }


    /* renamed from: a */
```

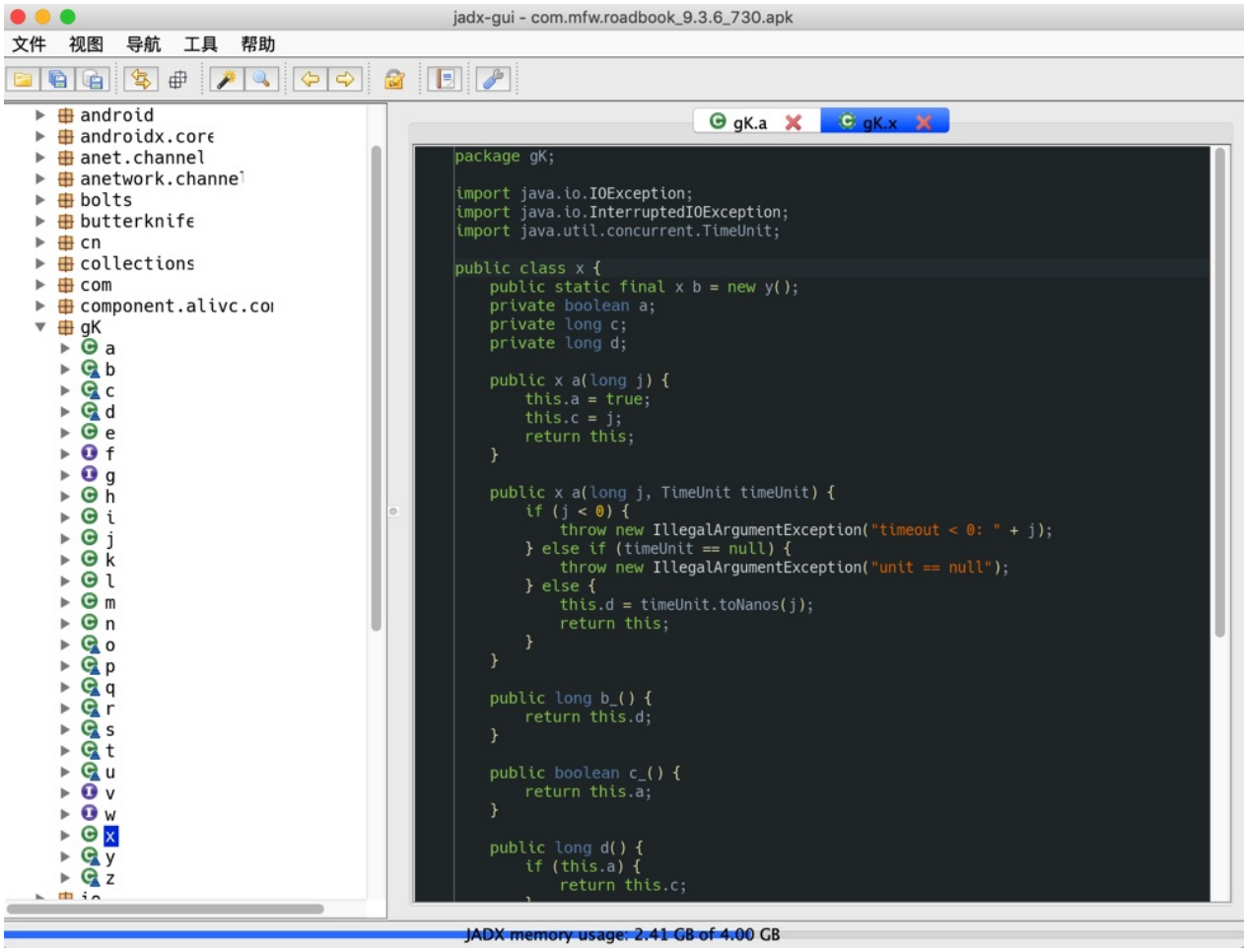
```
    public static boolean m13270a(C8931o c8931o, String str, Context context, String st
r2) {
        Intent m14850a;
        if (c8931o != null && c8931o.mo16787av() == 0) {
            return false;
        }
        try {
            if (TextUtils.isEmpty(str) || (m14850a = C9300aa.m14850a(context, str)) ==
null) {
                return false;
            }
            m14850a.putExtra("START_ONLY_FOR_ANDROID", true);
            if (!(context instanceof Activity)) {
                m14850a.addFlags(Label.FORWARD_REFERENCE_TYPE_SHORT);
            }
            context.startActivity(m14850a);
            HashMap hashMap = new HashMap();
            m13269a(c8931o, hashMap);
            C8412c.m18752e(context, c8931o, str2, "click_open", hashMap);
            return true;
        } catch (Throwable unused) {
        }
        return false;
    }


    /* renamed from: a */
    public static boolean m13268a(String str, Context context, String str2, C8931o c893
1o) {
        try {
            if (TextUtils.isEmpty(str)) {
                return false;
            }
            Uri parse = Uri.parse(str);
            Intent intent = new Intent("android.intent.action.VIEW");
            intent.setData(parse);
            if (!(context instanceof Activity)) {
                intent.addFlags(Label.FORWARD_REFERENCE_TYPE_SHORT);
            }
            HashMap hashMap = new HashMap();
            m13269a(c8931o, hashMap);
            C8412c.m18752e(C8938n.m16660a(), c8931o, str2, "open_url_app", hashMap);
            context.startActivity(intent);
            C8484j.m18615a().m18612a(c8931o, str2);
            return true;
        } catch (Throwable unused) {
            return false;
        }
    }


    /* renamed from: a */
    private static void m13269a(C8931o c8931o, Map<String, Object> map) {
        if (c8931o != null) {
            map.put("auto_click", Boolean.valueOf(c8931o.m16924a()));
        }
```

```
    }
}
```

GUI版本中的对比

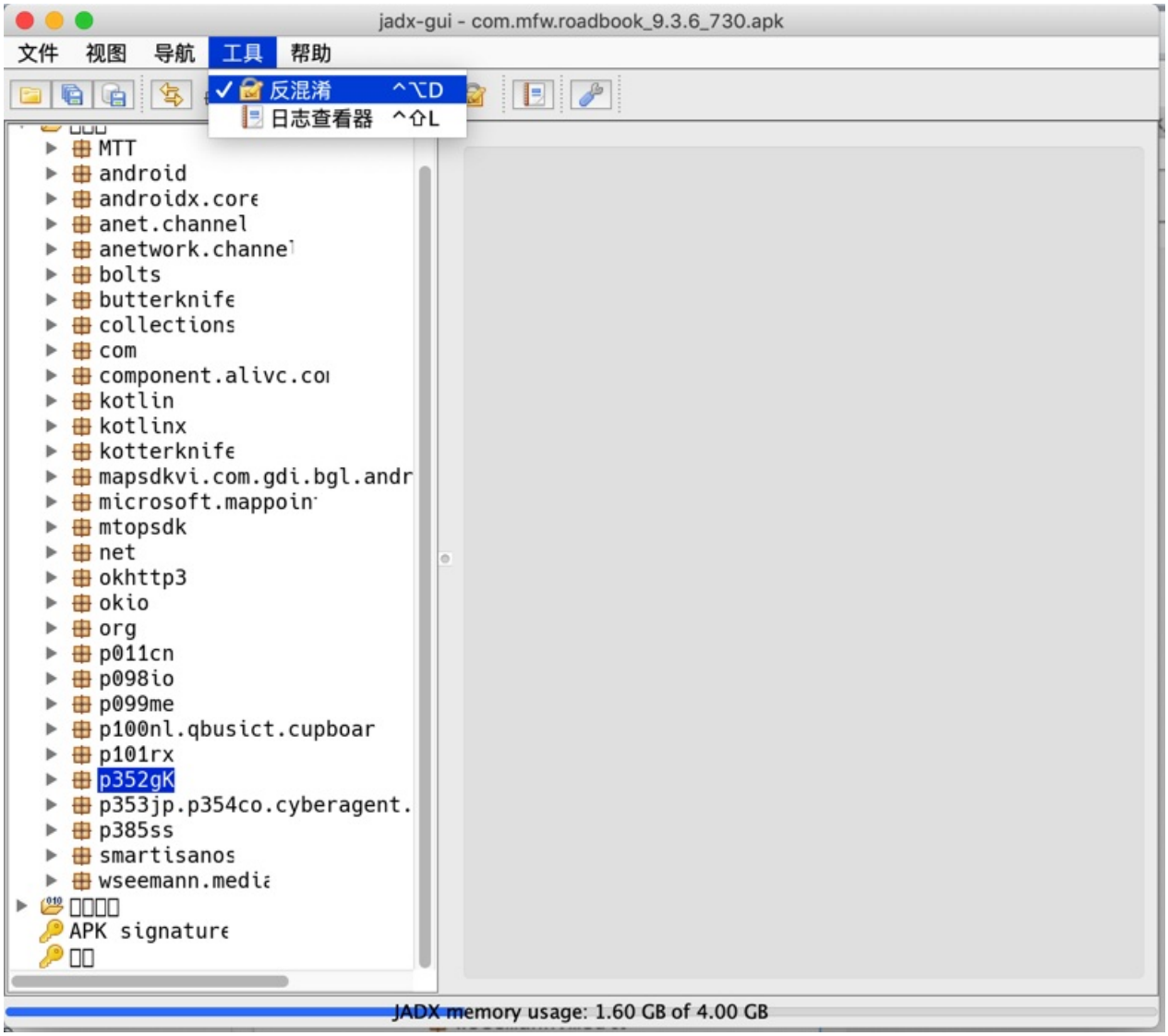
未开启反混淆



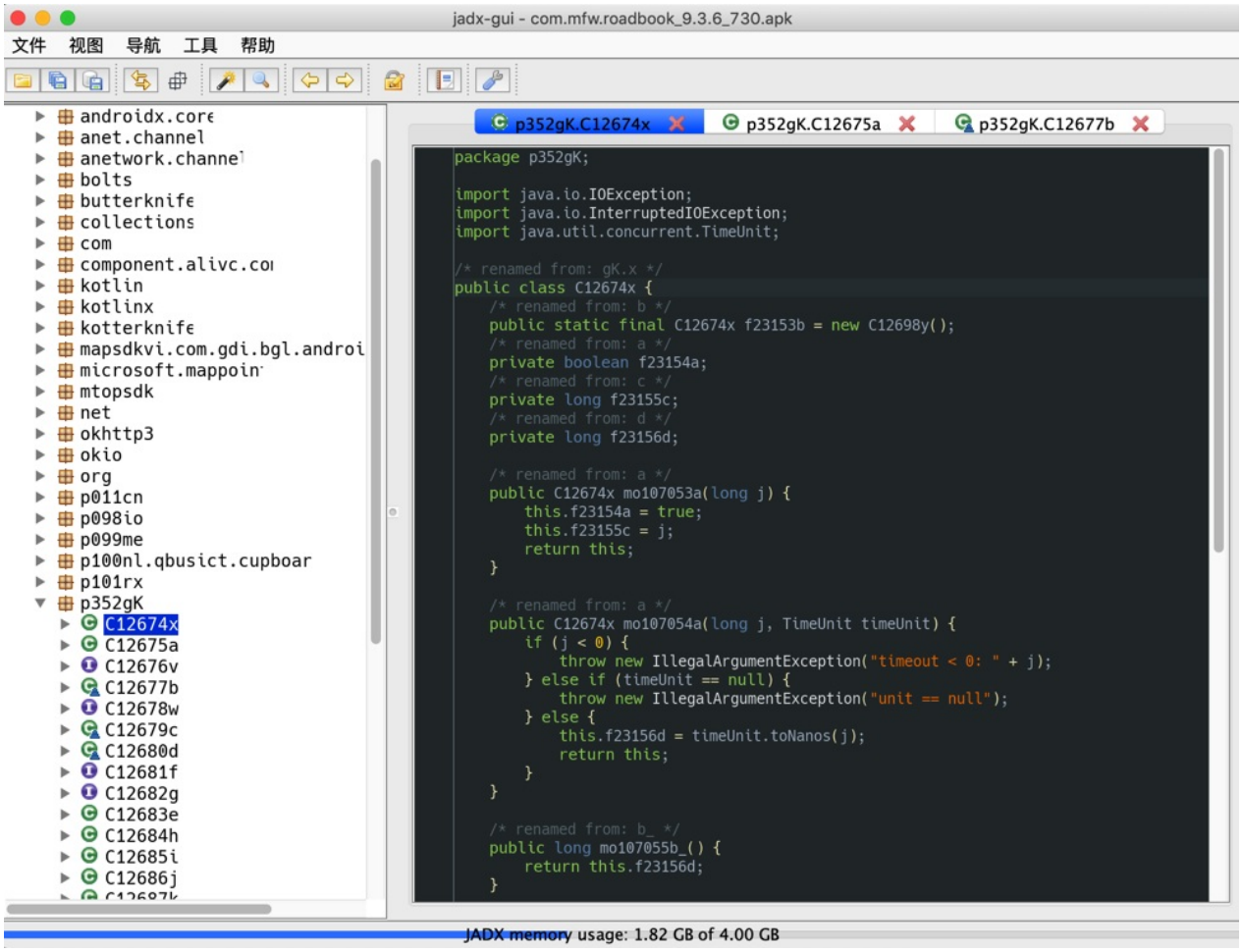


-》都是a,b,c,d,j等变量名

开启反混淆



之前的gK,io等，就反混淆了：



变量名改为了：`f23154a`，`f23155c`，虽然反混淆后的效果很一般，但是至少比a,b,c更容易看懂一些。

jadx-gui图形界面版

jadx除了cli的命令行版，还有个 `jadx-gui` =图形界面版。

启动运行jadx-gui

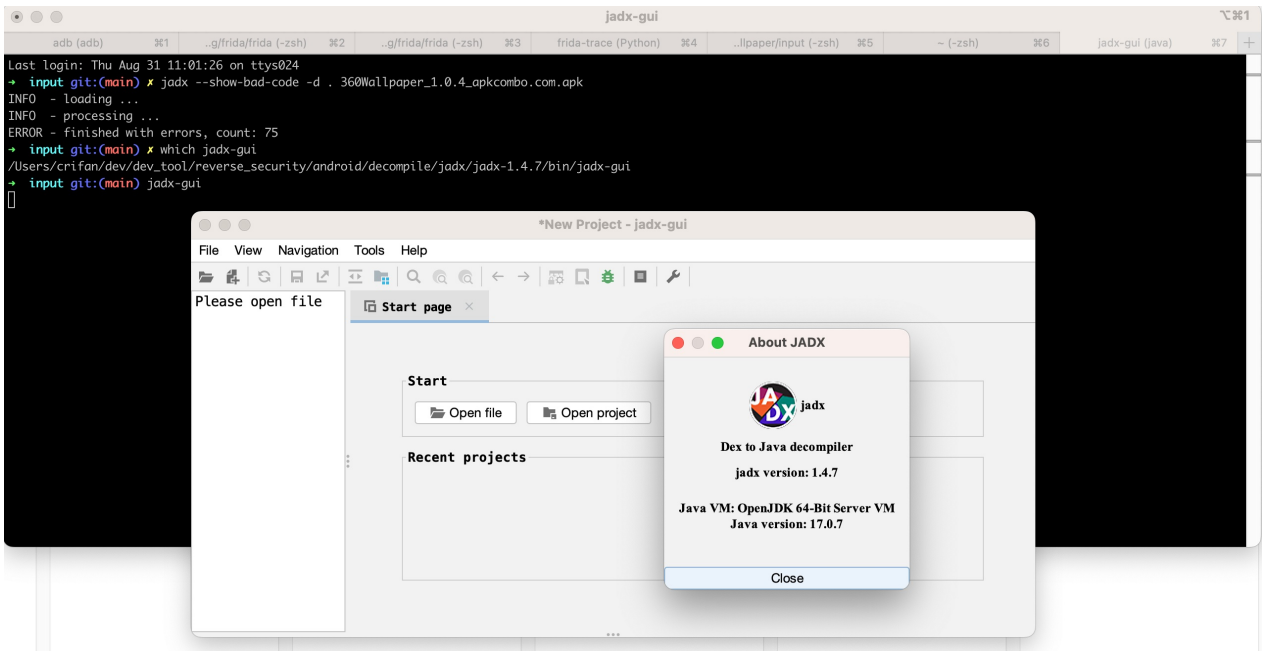
去运行 `jadx-gui` 脚本：

- Mac/Linux的 `jadx-gui`
  -
- Win的 `jadx-gui.bat`

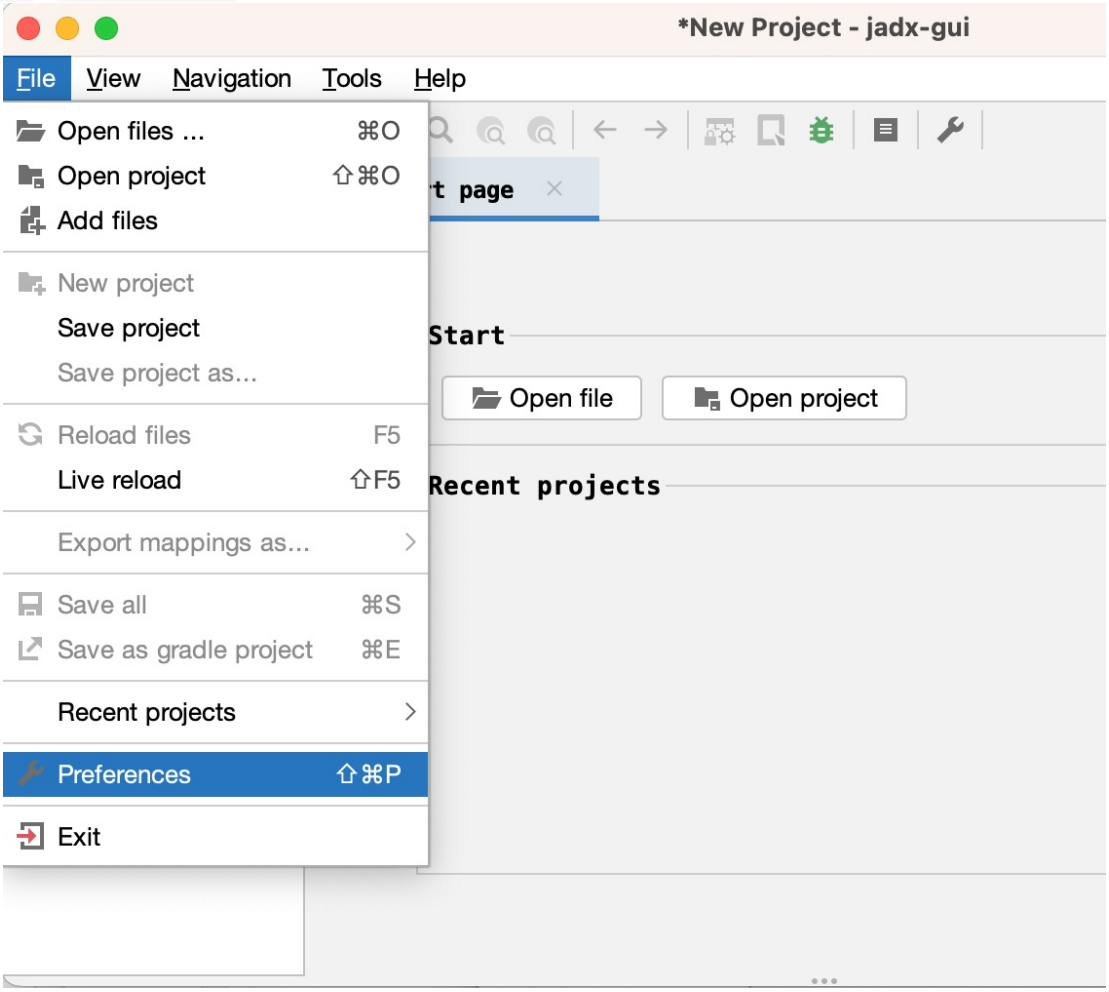
比如，Mac中是终端中运行：

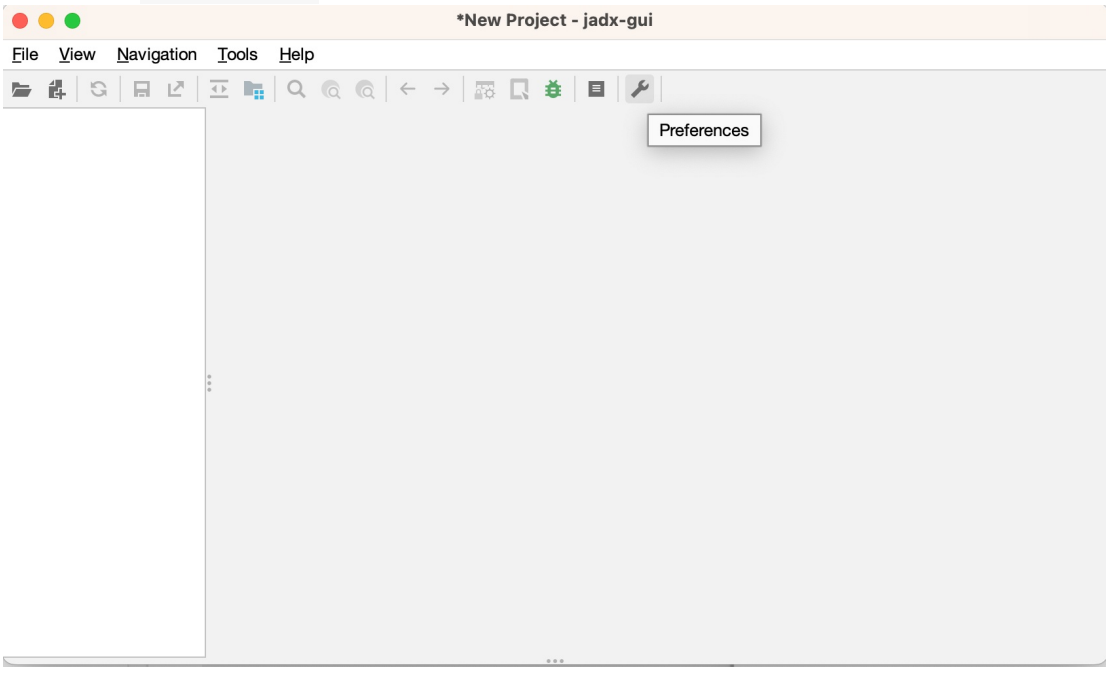
```
./jadx-gui
```

即可启动GUI图形界面版本：

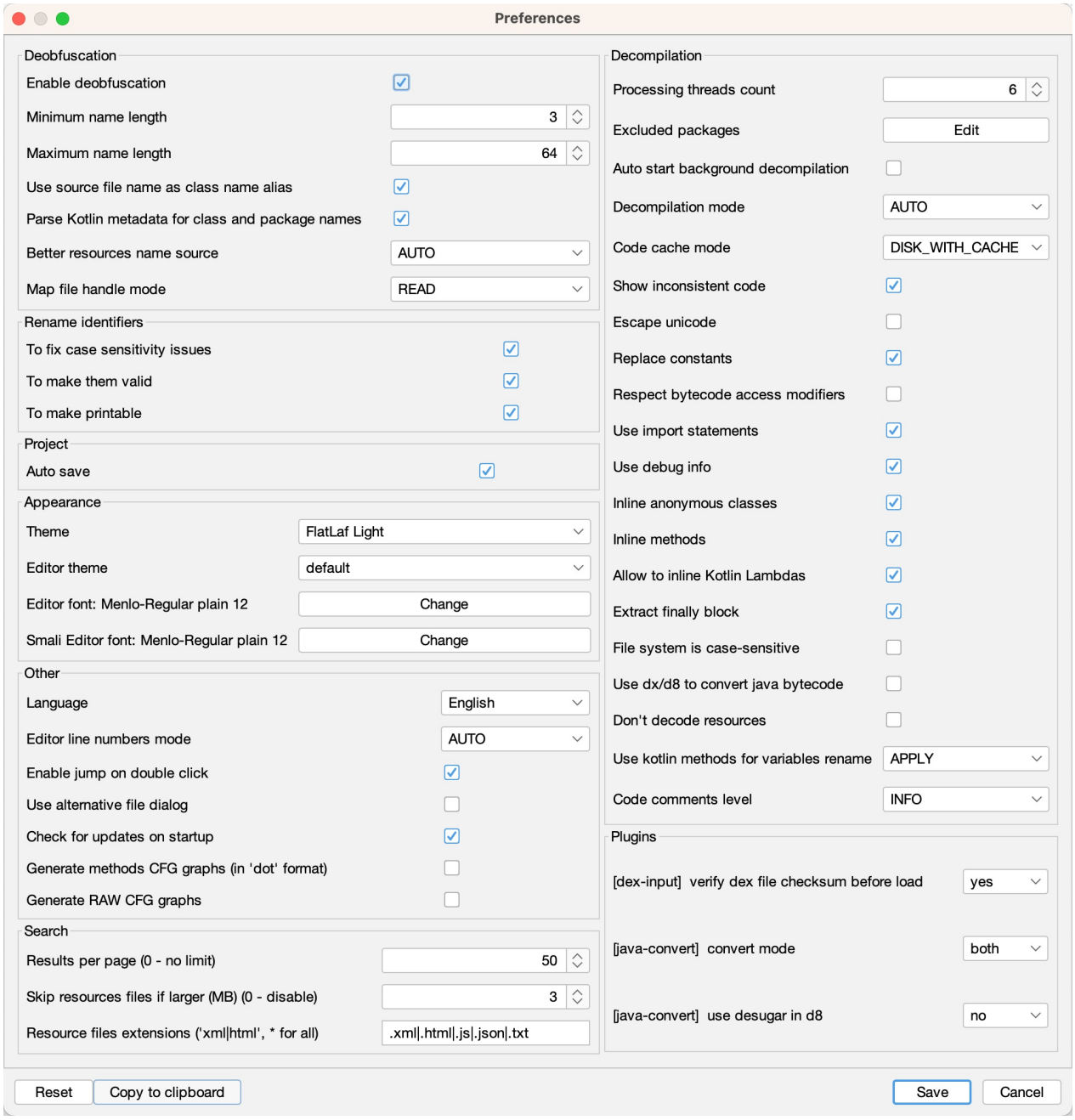


GUI版本中的参数设置页面

- 打开参数设置页面的2种方式
  - `File -> Preferences`
    - 

  - 点击工具栏中的 `Preferences` 图标
    - 

参数设置界面：



以及点击 `Copy to clipboard` 后得到的（JSON格式的）配置信息：

```
{
  "flattenPackage": false,
  "checkForUpdates": true,
  "fontStr": "",
  "smaliFontStr": "",
  "editorThemePath": "/org/fife/ui/rsyntaxtextarea/themes/default.xml",
  "lafTheme": "FlatLaf Light",
  "langLocale": {
    "locale": "en_US"
  },
  "autoStartJobs": false,
  "excludedPackages": "",
  "autoSaveProject": true,
```

```
  "showHeapUsageBar": false,
  "alwaysSelectOpened": false,
  "useAlternativeFileDialog": false,
  "codeAreaLineWrap": false,
  "srhResourceSkipSize": 3,
  "srhResourceFileExt": ".xml|.html|.js|.json|.txt",
  "searchResultsPerPage": 50,
  "useAutoSearch": true,
  "keepCommonDialogOpen": false,
  "smaliAreaShowBytecode": false,
  "lineNumbersMode": "AUTO",
  "mainWindowVerticalSplitterLoc": 300,
  "debuggerStackFrameSplitterLoc": 300,
  "debuggerVarTreeSplitterLoc": 700,
  "adbDialogPath": "",
  "adbDialogHost": "localhost",
  "adbDialogPort": "5037",
  "codeCacheMode": "DISK_WITH_CACHE",
  "jumpOnDoubleClick": true,
  "treeWidth": 130,
  "settingsVersion": 18,
  "skipResources": false,
  "skipSources": false,
  "exportAsGradleProject": false,
  "threadsCount": 6,
  "decompilationMode": "AUTO",
  "showInconsistentCode": true,
  "useImports": true,
  "debugInfo": true,
  "addDebugLines": false,
  "inlineAnonymousClasses": true,
  "inlineMethods": true,
  "allowInlineKotlinLambda": true,
  "extractFinally": true,
  "replaceConsts": true,
  "escapeUnicode": false,
  "respectBytecodeAccessModifiers": false,
  "deobfuscationOn": true,
  "deobfuscationMinLength": 3,
  "deobfuscationMaxLength": 64,
  "deobfuscationMapFileMode": "READ",
  "deobfuscationUseSourceNameAsAlias": true,
  "deobfuscationParseKotlinMetadata": true,
  "resourceNameSource": "AUTO",
  "useKotlinMethodsForVarNames": "APPLY",
  "renameFlags": [
    "CASE",
    "VALID",
    "PRINTABLE"
  ],
  "fsCaseSensitive": false,
  "cfgOutput": false,
  "rawCfgOutput": false,
  "fallbackMode": false,
  "useDx": false,
  "commentsLevel": "INFO",
```

```
  "pluginOptions": {}
}
```

cli和gui参数对应关系

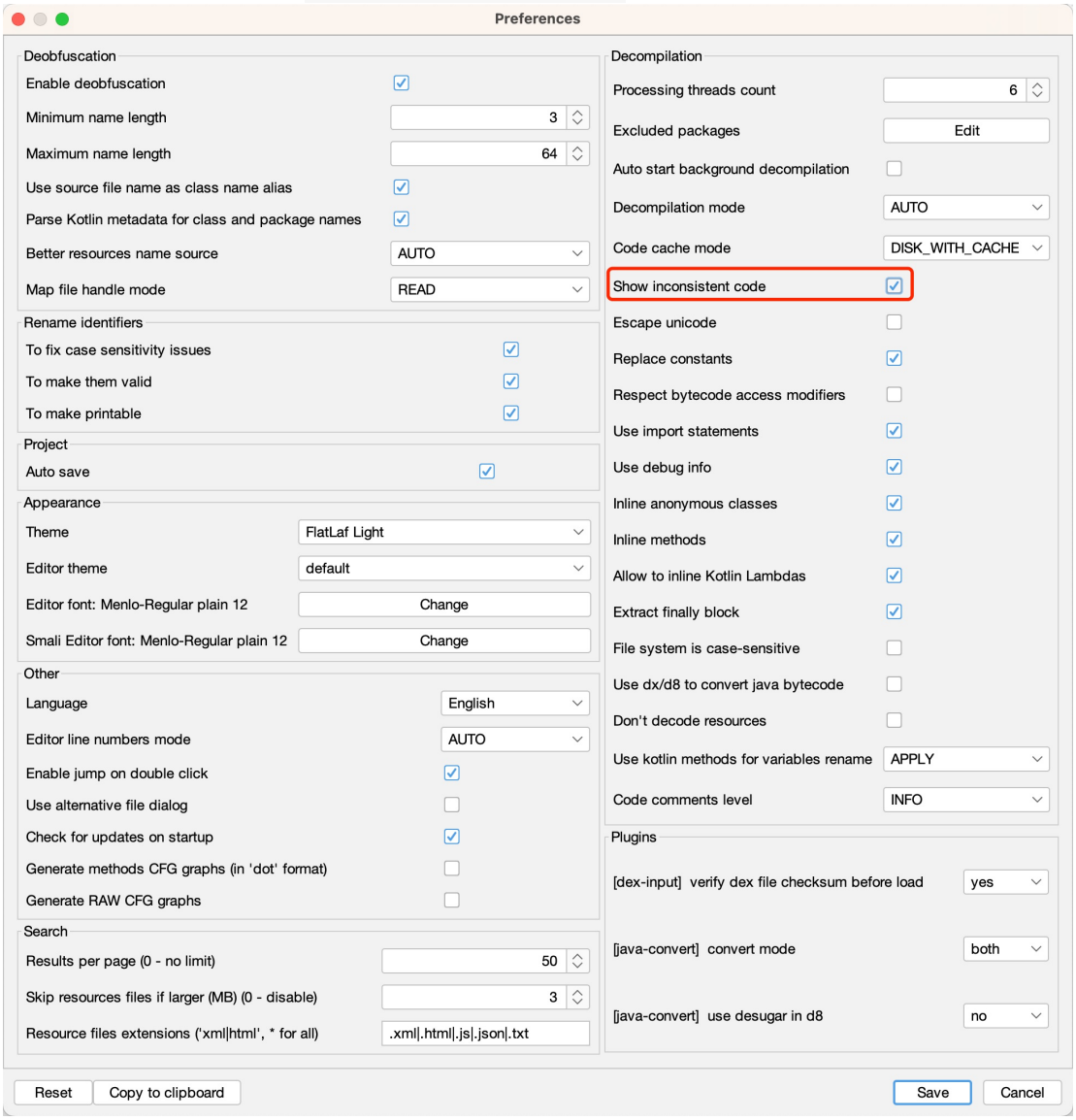
jadx分cli=命令行和gui=图形界面，共2个版本。

而其中很多功能，本质上是一样的

举例：

- 开启坏代码
  - jadx-cli中加参数 `--show-bad-code`
    - 举例

```
jadx --deobf --show-bad-code -d ./ 360Wallpaper_1.0.4_apkcombo.com.apk
```

  - jadx-gui中参数设置页中勾选：`Show inconsistent code`
    - 

因此就有了：同样的功能，不同版本中的开启（关闭）的方式，即：

- cli=命令行 和 gui=图形界面 的不同功能的映射关系

功能点	jadx-gui	jadx-cli
显示坏代码	勾选 Show inconsistent code	--show-bad-code
开启反混淆	勾选 Enable deobfuscation	--deobf
		--deobf-min
		--deobf-max
		--deobf-cfg-file
		--deobf-cfg-file-mode read/read-or-save/overwrite/ignore
		--deobf-use-sourcename
		--deobf-parse-kotlin-metadata
		--deobf-res-name-source
	取消勾选 all Rename identifiers flags	--rename-flags none
关闭内联	取消勾选 Inline anonymous classes	--no-inline-anonymous
	取消勾选 Inline methods	--no-inline-methods
	勾选 Generate methods CFG graphs	--cfg
	勾选 Generate RAW CFG grapths	--raw-cfg
	取消勾选 Move inner classes into parent	--no-move-inner-classes

相关页面

菜单栏

- View
    - Navigation
    - Tools
    - Help

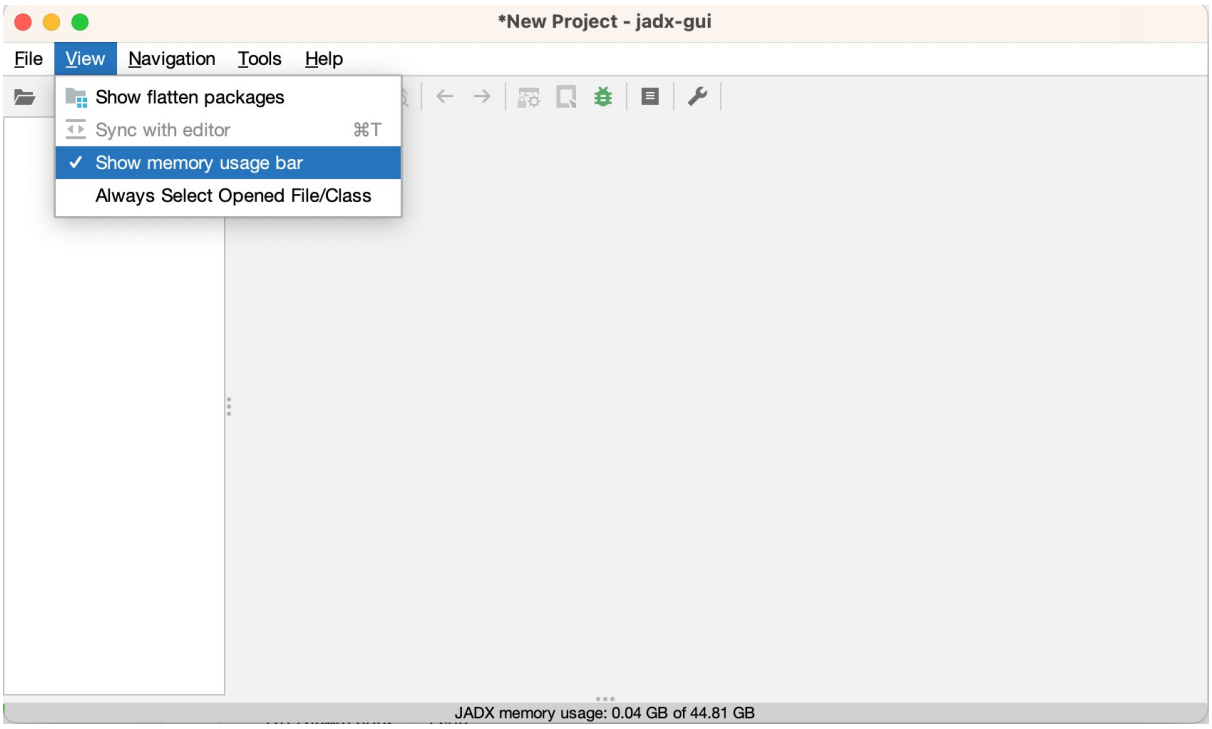
-

View视图

显示内存使用

勾选：`View -> Show memory usage bar`

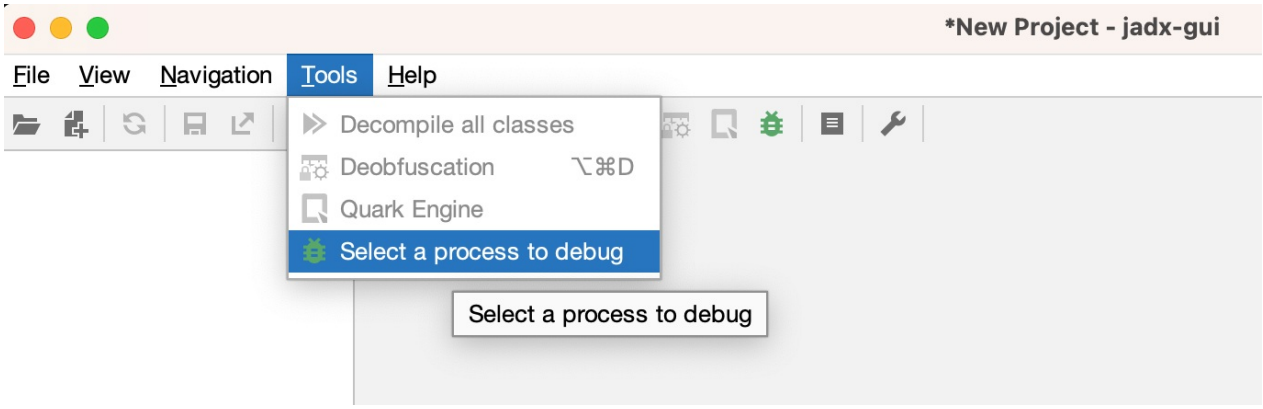
可以在底部显示JADX的内存使用量：

- 

-》估计是防止，反编译太大的apk，导致内存占用太多，而导致崩溃时，至少知道是内存方面的问题。

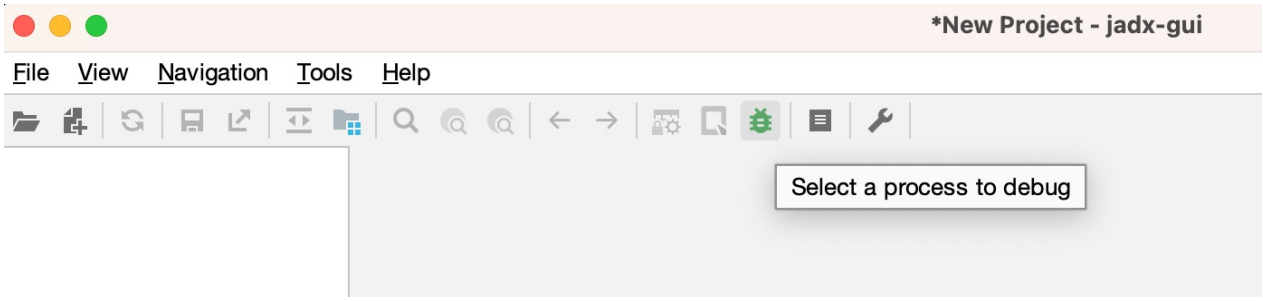
调试进程

`Tool -> Select a process to debug`



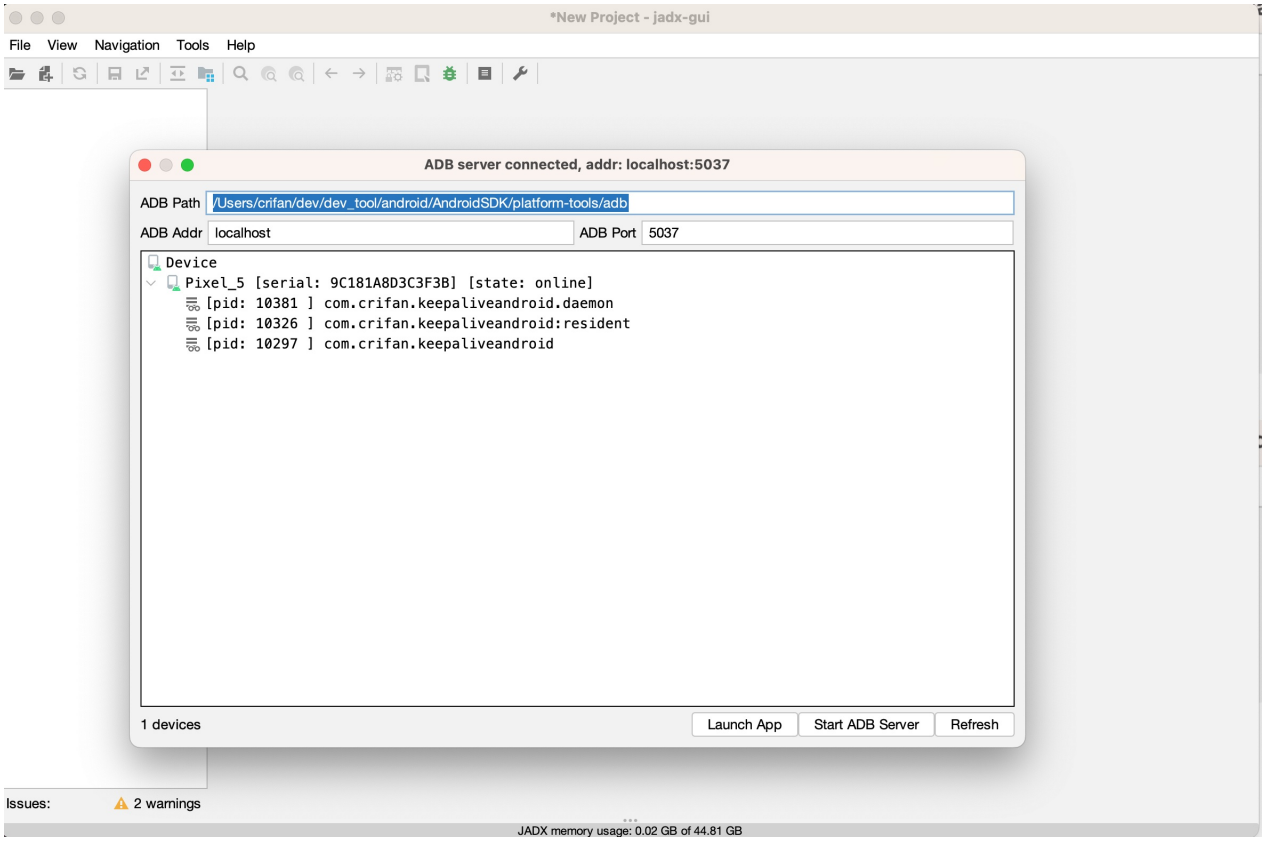
或点击：

工具栏中bug虫子的图标：



可以打开：

ADB调试窗口：



此处自动找到了：

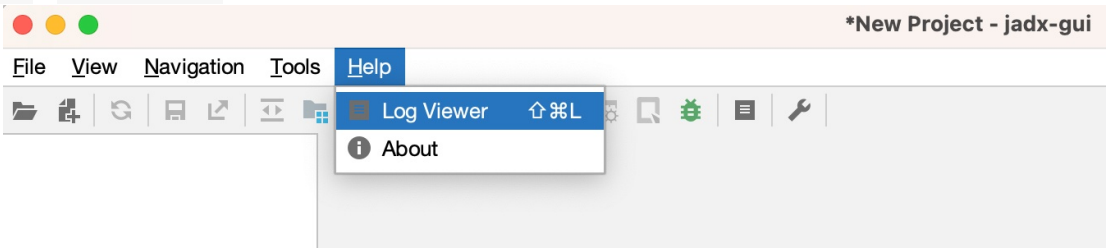
- ADB的path

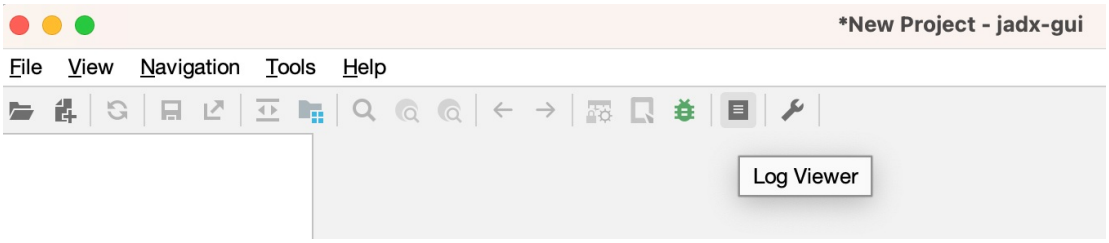
- ADB的Address 和port

以及：

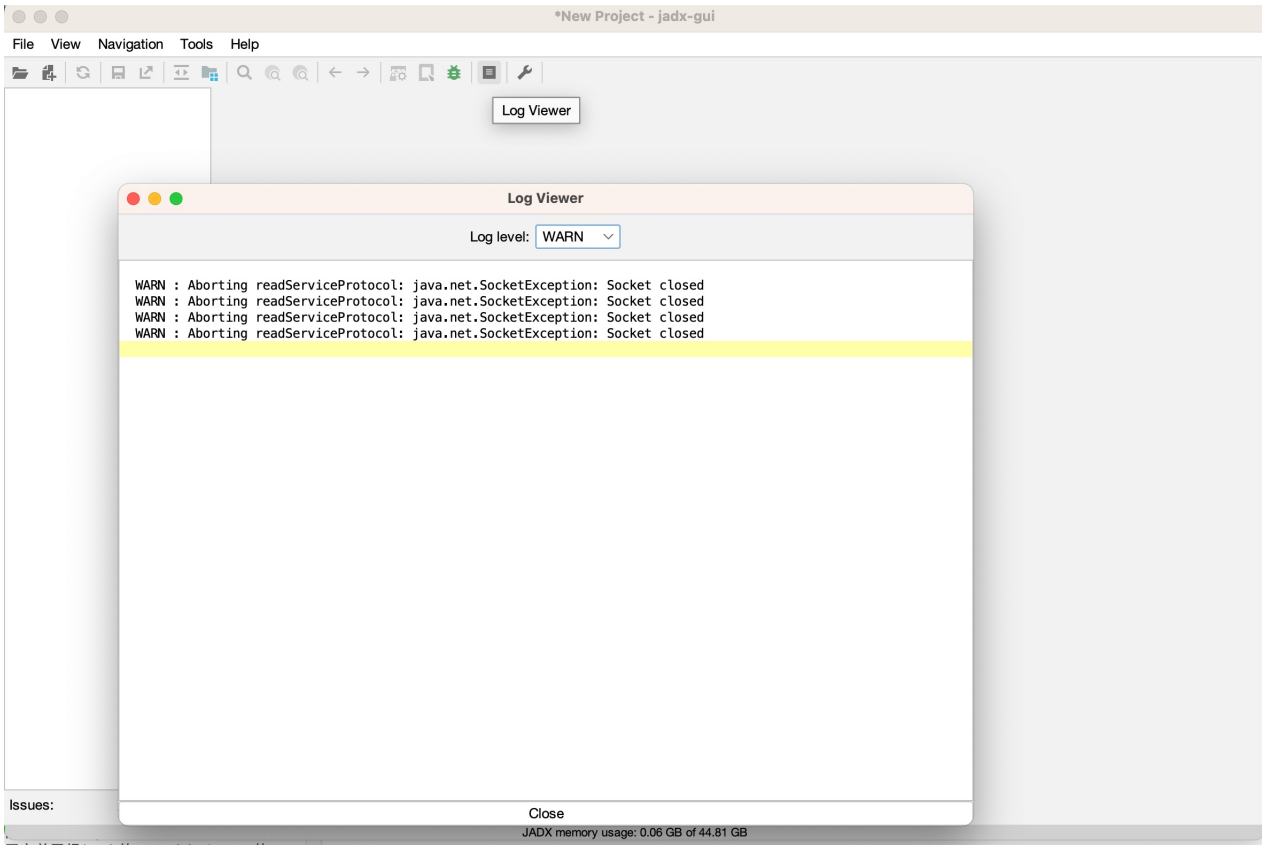
目标安卓设备中，（debuggable）可以调试的进程 = 此处是自己之前AS调试安装过的一个app：KeepAliveAndroid的相关进程

日志查看器

- 打开 `Log Viewer` = `日志查看器` 的方式
  - `Help -> Log Viewer`
    - 

  - 工具栏中点击：日志图标
    - 

可以打开：日志查看器



2023-09-12 22:20:11

jadx使用心得

经验心得

从 `jadx-gui` 打开的结构看出是否加固和是哪家的加固

对于某个安卓的apk，用 `jadx-gui` 打开不同版本的apk的效果是：

- v1.5
  -
- v3.4.8

- v3.6.9

-》可以看出apk是否被加固以及用了何种加固方案：

- `v1.5`：没有被加固
- `v3.4.8`：加固方案 qihoo奇虎360
- `v3.6.9`：加固方案 腾讯乐固legu

常见错误

`jadx` 不能从 `jar` 导出 `java`，否则会报错

举例：

```
../../../../reverse_engineering/jadx/jadx-1.0.0/bin/jadx ../dex_to_jar/com.ishowedu.
child.peiyin9201516-dex2jar.jar -d .
INFO  - loading ...
INFO  - converting to dex: com.ishowedu.child.peiyin9201516-dex2jar.jar ...
ERROR - jadx error: Error load file: ../dex_to_jar/com.ishowedu.child.peiyin9201516-dex
2jar.jar
jadx.core.utils.exceptions.JadxRuntimeException: Error load file: ../dex_to_jar/com.ish
owedu.child.peiyin9201516-dex2jar.jar
    at jadx.api.JadxDecompiler.loadFiles(JadxDecompiler.java:138)
    at jadx.api.JadxDecompiler.load(JadxDecompiler.java:102)
    at jadx.cli.JadxCLI.processAndSave(JadxCLI.java:32)
    at jadx.cli.JadxCLI.main(JadxCLI.java:18)
```

```
Caused by: jadx.core.utils.exceptions.DecodeException: java class to dex conversion err
or:
 dx exception: Translation has been interrupted
    at jadx.core.utils.files.InputFile.loadFromJar(InputFile.java:191)
    at jadx.core.utils.files.InputFile.searchDexFiles(InputFile.java:82)
    at jadx.core.utils.files.InputFile.addFilesFrom(InputFile.java:40)
    at jadx.api.JadxDecompiler.loadFiles(JadxDecompiler.java:136)
    ... 3 common frames omitted
Caused by: jadx.core.utils.exceptions.JadxException: dx exception: Translation has been
 interrupted
    at jadx.core.utils.files.JavaToDex.convert(JavaToDex.java:63)
    at jadx.core.utils.files.InputFile.loadFromJar(InputFile.java:182)
    ... 6 common frames omitted
Caused by: java.lang.RuntimeException: Translation has been interrupted
    at com.android.dx.command.dexer.Main.processAllFiles(Main.java:614)
    at com.android.dx.command.dexer.Main.runMultiDex(Main.java:365)
    at com.android.dx.command.dexer.Main.runDx(Main.java:286)
    at jadx.core.utils.files.JavaToDex.convert(JavaToDex.java:49)
    ... 7 common frames omitted
Caused by: java.lang.InterruptedException: Too many errors
    at com.android.dx.command.dexer.Main.processAllFiles(Main.java:606)
    ... 10 common frames omitted
```

jadx转换出错：java.lang.OutOfMemoryError

如果用 `jadx` 转换代码期间出错：

```
java.lang.OutOfMemoryError: GC overhead limit exceeded
    at jadx.core.dex.visitors.blocksmaker.BlockProcessor.computeDominators(BlockProcess
or.java:189)
    at jadx.core.dex.visitors.blocksmaker.BlockProcessor.processBlocksTree(BlockProcess
or.java:52)
    at jadx.core.dex.visitors.blocksmaker.BlockProcessor.visit(BlockProcessor.java:42)
    at jadx.core.dex.visitors.DepthTraversal.visit(DepthTraversal.java:27)
    at jadx.core.dex.visitors.DepthTraversal.lambda$visit$1(DepthTraversal.java:14)
    at jadx.core.dex.visitors.DepthTraversal$$Lambda$19/469590976.accept(Unknown Source)

    at java.util.ArrayList.forEach(ArrayList.java:1249)
    at jadx.core.dex.visitors.DepthTraversal.visit(DepthTraversal.java:14)
    at jadx.core.ProcessClass.process(ProcessClass.java:32)
    at jadx.api.JadxDecompiler.processClass(JadxDecompiler.java:292)
    at jadx.api.JavaClass.decompile(JavaClass.java:62)
    at jadx.api.JadxDecompiler.lambda$appendSourcesSave$0(JadxDecompiler.java:200)
    at jadx.api.JadxDecompiler$$Lambda$13/1425454633.run(Unknown Source)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
    at java.lang.Thread.run(Thread.java:745)
...
```

且同时伴有：

- CPU占用率很高

- 内存消耗也很大
  - 比如此处JadxCLI占用了4G的内存
    - 

就是典型的：`OOM` = `Out Of Memory` 的问题了。

解决办法，有两种：

- 增加JVM最大内存
  - 逻辑：修改 `jadx` 脚本，增大 `-Xmx` 的值
  - 步骤：
    - 编辑 `jadx-0.9.0/bin/jadx`，找到 `DEFAULT_JVM_OPTS` 的配置，修改其中 `-Xmx` 的值
    - 比如把此处的

```
DEFAULT_JVM_OPTS='"-Xms128M" "-Xmx4g"'
```

    - 改为：

```
DEFAULT_JVM_OPTS='"-Xms128M" "-Xmx6g"'
```

    - 即表示，把JVM最大内存，从之前的 `4G`，增大到 `6G`
    - 这样就运行 `jadx` 使用更多的内存，从而降低或消除 `OOM` 的问题了
- 减少线程数
  - 逻辑：通过 `-j N`，N=1/2之类，减少进程数，从而降低内存占用，减少OOM的概率
  - 步骤：
    - 在命令行运行jadx时，传递 `-j` 参数，指定线程数，比如

```
jadx -d output_folder -j 1 your_apk.apk
```

  - 缺点
    - 处理速度会有所降低

    - 因为默认 `4` 线程处理，反编译等速度会比较快
    - 线程数减少后，反编译等速度可能会有所影响

说明：

- 一般反编译小的不复杂的 `apk` 或 `dex` ，不会遇到 `OOM` 问题
- 反编译比较大型的，比较复杂的 `apk` 或 `dex` ，才可能会遇到 `OOM`
    - 比如之前用jadx反编译马蜂窝时遇到了 `OOM`

jadx的help=帮助信息=语法参数

```
➜ jadx --help

jadx - dex to java decompiler, version: 1.4.7

usage: jadx [options] input files (.apk, .dex, .jar, .class, .smali, .zip, .aar, .ars
c, .aab)
options:
  -d, --output-dir                    - output directory
  -ds, --output-dir-src               - output directory for sources
  -dr, --output-dir-res               - output directory for resources
  -r, --no-res                        - do not decode resources
  -s, --no-src                        - do not decompile source code
  --single-class                      - decompile a single class, full name, raw or ali
as
  --single-class-output               - file or dir for write if decompile a single cla
ss
  --output-format                     - can be 'java' or 'json', default: java
  -e, --export-gradle                 - save as android gradle project
  -j, --threads-count                 - processing threads count, default: 6
  -m, --decompilation-mode            - code output mode:
                                        'auto' - trying best options (default)
                                        'restructure' - restore code structure (normal
java code)
                                        'simple' - simplified instructions (linear, wi
th goto's)
                                        'fallback' - raw instructions without modifica
tions
  --show-bad-code                     - show inconsistent code (incorrectly decompiled)
  --no-imports                        - disable use of imports, always write entire pac
kage name
  --no-debug-info                     - disable debug info
  --add-debug-lines                   - add comments with debug line numbers if availab
le
  --no-inline-anonymous               - disable anonymous classes inline
  --no-inline-methods                 - disable methods inline
  --no-inline-kotlin-lambda           - disable inline for Kotlin lambdas
  --no-finally                        - don't extract finally block
  --no-replace-consts                 - don't replace constant value with matching cons
tant field
  --escape-unicode                    - escape non latin characters in strings (with \u
)
  --respect-bytecode-access-modifiers - don't change original access modifiers
  --deobf                             - activate deobfuscation
  --deobf-min                         - min length of name, renamed if shorter, default
: 3
  --deobf-max                         - max length of name, renamed if longer, default:
64
  --deobf-cfg-file                    - deobfuscation map file, default: same dir and n
ame as input file with '.jobf' extension
  --deobf-cfg-file-mode               - set mode for handle deobfuscation map file:
                                        'read' - read if found, don't save (default)
```

```
                                        'read-or-save' - read if found, save otherwise
 (don't overwrite)
                                        'overwrite' - don't read, always save
                                        'ignore' - don't read and don't save
  --deobf-use-sourcename              - use source file name as class name alias
  --deobf-parse-kotlin-metadata       - parse kotlin metadata to class and package name
s
  --deobf-res-name-source             - better name source for resources:
                                        'auto' - automatically select best name (defau
lt)
                                        'resources' - use resources names
                                        'code' - use R class fields names
  --use-kotlin-methods-for-var-names  - use kotlin intrinsic methods to rename variable
s, values: disable, apply, apply-and-hide, default: apply
  --rename-flags                      - fix options (comma-separated list of):
                                        'case' - fix case sensitivity issues (accordin
g to --fs-case-sensitive option),
                                        'valid' - rename java identifiers to make them
 valid,
                                        'printable' - remove non-printable chars from
identifiers,
                                      or single 'none' - to disable all renames
                                      or single 'all' - to enable all (default)
  --fs-case-sensitive                 - treat filesystem as case sensitive, false by de
fault
  --cfg                               - save methods control flow graph to dot file
  --raw-cfg                           - save methods control flow graph (use raw instru
ctions)
  -f, --fallback                      - set '--decompilation-mode' to 'fallback' (depre
cated)
  --use-dx                            - use dx/d8 to convert java bytecode
  --comments-level                    - set code comments level, values: error, warn, i
nfo, debug, user-only, none, default: info
  --log-level                         - set log level, values: quiet, progress, error,
warn, info, debug, default: progress
  -v, --verbose                       - verbose output (set --log-level to DEBUG)
  -q, --quiet                         - turn off output (set --log-level to QUIET)
  --version                           - print jadx version
  -h, --help                          - print this help

Plugin options (-P name=value ):
 1) dex-input: Load .dex and .apk files
    - dex-input.verify-checksum       - verify dex file checksum before load, values: [
yes, no], default: yes
 2) java-convert: Convert .class, .jar and .aar files to dex
    - java-convert.mode               - convert mode, values: [dx, d8, both], default:
both
    - java-convert.d8-desugar         - use desugar in d8, values: [yes, no], default:
no

Examples:
  jadx -d out classes.dex
  jadx --rename-flags "none" classes.dex
  jadx --rename-flags "valid, printable" classes.dex
  jadx --log-level ERROR app.apk
  jadx -Pdex-input.verify-checksum no app.apk
```

附录

下面列出相关参考资料。

参考资料

- 【已解决】mac中用jadx从apk中导出java源码
- 【记录】安卓逆向LiftFileManager的apk：jadx反编译
- 【记录】用jadx开启--show-bad-code去反编译360Wallpaper安卓apk
- 【记录】用jadx开启反混淆去反编译360Wallpaper安卓apk
- 【记录】jadx的GUI版本jadx-gui功能试用
- 【记录】用jadx反编译360Wallpaper的安卓apk
-
- 【整理】java反编译器对比：JD-GUI，CFR，Procyon，Jadx
- 【已解决】用jadx把安卓dex文件转换提取出jar包和java源代码
- 【已解决】mac中用jadx命令行CLI从apk中导出java源码
- 【已解决】用jadx命令行从dex文件转换出java源代码
-
- skylot/jadx: Dex to Java decompiler
- How to decompile an APK or DEX file using jadx in Windows | Our Code World
- Android 反编译利器，jadx 的高级技巧 - 简书
- 1.2.1 APK反编译工具之：Procyon，Jadx和AndroidDecompiler - Coder-Pig的猪栏 - CSDN博客
- Troubleshooting Q&A · skylot/jadx Wiki (github.com)
-