"بسمه تعالی"

درس: اصول طراحی کامپایلر

تمرین شماره ۲

اتوسا طغیانی (۹۶۱۱۴۱۵۰۲۴)

تاریخ: ۹۹/۰۳/۱۹

۱- بخش‌های مختلف تهیه فرایند کد جاوا را به برنامه‌های نهایی apk. اندروید توضیح دهید.

همان طور که می‌دانید اندروید یک سیستم عامل موبایل است که توسط گوگل توسعه می‌یابد؛ این سیستم عامل بر پایه هسته لینوکس و دیگر نرم افزارهای متن باز طراحی شده است.

واژه نامه کمبریج، اندروید (Android) را به این صورت معنا کرده است:

" یک ربات که به گونه‌ای ساخته شده تا شکل ظاهری شبیه به انسان داشته باشد."

برنامه‌های اندروید را می‌توان با استفاده از زبان‌های Kotlin ، Java و C++ نوشت اما زبان اصلی برنامه نویسی اندروید Java است.

محیط برنامه نویسی اندروید، Android SDK است، که کد برنامه را به همراه هر سند و داده در منابع را در یک بسته Android ، که یک پرونده بایگانی با پسوند apk. است، گردآوری می‌کند.

APK مخفف Android application package است؛ یک فایل APK شامل تمام محتویات یک برنامه اندروید است و پرونده‌ای است که دستگاه‌های دارای سیستم عامل اندروید برای نصب برنامه‌ها از آن استفاده می‌کنند.

سیستم عامل اندروید، یک سیستم چند کاربره لینوکس است که در آن هر برنامه کاربر متفاوتی دارد؛ به طور پیش فرض، سیستم به هر برنامه یک شناسه کاربر (user ID) منحصر به فرد لینوکس اختصاص می‌دهد (شناسه فقط توسط سیستم استفاده می‌شود و برای برنامه ناشناخته است). این سیستم برای همه پرونده‌های یک برنامه مجوز تعیین می‌کند تا فقط شناسه کاربر اختصاص داده شده به آن برنامه بتواند به آن‌ها دسترسی پیدا کند.

هر فرآیند دارای یک ماشین مجازی (VM) خاص خود است، بنابراین کد برنامه به صورت جدا از سایر برنامه‌ها اجرا می‌شود.

به طور پیش فرض، هر برنامه در فرآیند لینوکس خود اجرا می‌شود. سیستم Android هنگامی که نیاز به اجرای هر یک از اجزای برنامه است، این فرآیند را شروع می‌کند و هنگامی که دیگر لازم نیست یا وقتی سیستم باید حافظه را برای سایر برنامه‌ها بازیابی کند، این روند را خاموش(غیر فعال) می‌کند.

سیستم اندرویدی اصل حداقل امتیاز را رعایت می‌کند؛ یعنی هر برنامه به طور پیش فرض فقط به مؤلفه‌هایی که برای انجام کار خود نیاز دارد دسترسی دارد و نه بیشتر. این قابلیت یک محیط بسیار امن ایجاد می‌کند که در آن برنامه نمی‌تواند به قسمت‌هایی از برنامه که سیستم به آن اجازه دستیابی نداده، دسترسی پیدا کند.

با این حال، روش‌هایی برای به اشتراک گذاشتن داده‌ها با سایر برنامه‌ها و دسترسی یک برنامه به سرویس‌های سیستم وجود دارد.

مولفه‌های برنامه

اجزای برنامه، بلوک‌های ساختاری اساسی یک برنامه اندرویدی هستند. هر مؤلفه یک نقطه ورود است که از طریق آن سیستم یا کاربر می‌تواند وارد برنامه شود. بعضی از مؤلفه‌ها به سایر مولفه‌ها بستگی دارند.

برای ساخت یک اپلیکیشن اندروید چهار Component اصلی وجود دارد، که عبارتند از:

- فعالیت‌ها (Activities)
- خدمات (Services)
- گیرنده‌های پخش (Broadcast receivers)
- ارائه‌دهندگان محتوا (Content providers)

هر کدام از مولفه‌ها یک هدف و چرخه عمر مشخص دارند که چگونگی ایجاد و از بین بردن مؤلفه را تعریف می‌کنند.

فعالیت ها:

یک Activities، نقطه ورود برای تعامل با کاربر است. می‌توان آن را به منزله یک صفحه از اپلیکیشن تصور کرد.به عنوان مثال در یک برنامه چند گزینه متفاوت وجود دارد که هرکدام یک کار مستقل انجام می‌دهند، با انتخاب یک گزینه به صفحه راهنمای برنامه هدایت می‌شوید و با انتخاب گزینه دیگر وارد صفحه اصلی برنامه می‌شوید؛ در واقع با انتخاب یک گزینه و رفتن به صفحه دیگر وارد یک Activities دیگر شده‌اید. به طور خلاصه میتوان گفت Activities ، UI را تنظیم می‌کند و تعامل کاربر با برنامه را کنترل می‌کند.

خدمات:

این مولفه از اهمیت زیادی برخوردار است و خارج از دید کاربر رخ می‌دهد، گاهی به مدت طولانی و گاهی مدت زمان کوتاهی.

این دسته از Component ها دارای هیچ گونه GUI نمی‌باشند و در Background (پس زمینه) برنامه اجرا می‌شوند. به عنوان مثال برنامه پخش موزیک، ممکن است شما در حال پیام دادن یا جست و جو در اینترنت باشید و موسیقی نیز پخش شود و پخش موسیقی عملکرد سایر برنامه‌ها را مختل نمی‌کند و در پس زمینه اجرا می‌شود. به طور خلاصه می‌توان گفت که این بخش، برنامه‌هایی که در پس زمینه جریان دارند را با دیگر برنامه‌ها مدیریت می‌کنند.

گیرنده های پخش:

این وظیفه را دارا است تا به هشدارهایی که در سطح کل سیستم بوجود می‌آیند پاسخ دهد. Broadcast Receiver ها به سادگی به پیام‌های منتشر شده از سوی سایر اپلیکیشن‌ها یا خود سیستم پاسخ می‌دهند. این پیام ها گاهی رویداد (event) یا مفهوم (intent) نامیده می‌شوند. برای مثال، اپلیکیشن‌ها می‌توانند برای آگاه کردن سایر اپلیکیشن‌ها از اینکه داده ی خاصی در دستگاه دانلود شده و آماده‌ی استفاده است، پیام‌هایی را به آن‌ها ارسال کنند یا هنگامی که باتری موبایل شما به ۳۰ درصد رسید به شما هشدار دهد.

ارائه دهندگان محتوا:

وظیفه این بخش، ذخیره سازی اطلاعات و قراردادن اطلاعات ذخیره شده در اختیار دیگر اپلیکیشن‌ها می‌باشد. به طور خلاصه تنها راه به اشتراک گذاری داده‌ها میان اپلیکیشن‌های مختلف به کارگیری Content Providerها می‌باشد. داده‌ها ممکن است به طور کامل در فایل‌های سیستمی خود اندروید ، پایگاه داده و یا جایی دیگر ذخیره شوند.

اگر بخواهیم داده‌ها را در اختیار سایر برنامه‌ها یا بخش‌های مختلف برنامه قرار دهیم، به سادگی می توانیم یک Content Provider ایجاد کرده و مابین بخش‌هایی که می‌خواهند از آن استفاده کنند ارتباطی برقرار سازیم.

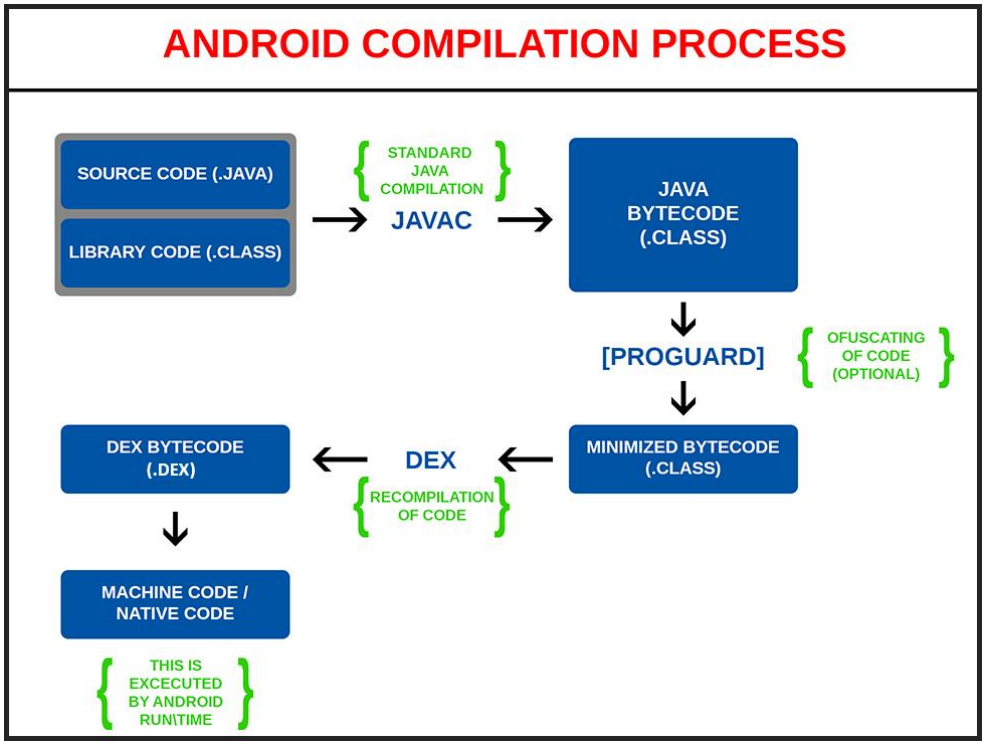
به عنوان مثال اگر یک برنامه نیاز به دفترچه تلفن موبایل شما داشته باشد، صرفاً نیاز است تا ارتباطی ابین اپلیکیشن خود و Content Provider مرتبط با Contact سیستم عامل اندروید برقرار سازیم.

به طور خلاصه میتوان گفت این بخش مدیریت اطلاعات و محتویات پایگاه داده را انجام می‌دهد.

کامپایل برنامه‌های اندروید

همان طور که در ابتدا گفته شد زبان اصلی برنامه نویسی اندروید جاوا است؛ **مراحل کامپایل برنامه‌های Android با سایر برنامه‌های جاوا بسیار متفاوت است.**

شکل زیر مراحل کامپایل کد اندروید را نشان می‌دهد:



در مرحله اول، سورس کدها و کتاب‌خانه‌ها توسط کامپایلر جاوا که JACAC نام دارد به بایت کد تبدیل شده و در پرونده‌های Class. وارد می‌شوند.

فایل‌های Class. حاوی بایت کد استاندارد Oracle JVM Java هستند اما سیستم عامل اندروید این قابلیت را ندارد در عوض دارای Dalvik است. بایت کدهای Jvm. مانند بایت کدهای Oracle JVM ، دستورالعمل‌های کد دستگاه برای پردازشگر نظری هستند.

تفاوت JVM و Dalvik (DVM) :

برنامه‌های جاوا برای اجرا نیاز به سیستم عامل خاصی ندارند، بلکه هرجایی که JVM نصب شده باشد قابل اجرا هستند. با نصب JVM روی هر سیستم عاملی می‌توان برنامه‌های جاوا را اجرا کرد. روی گوشی‌های اندروید هم یک JVM بهینه شده به نام Dalvik قرار دارد که برنامه نهایی را به زبان ماشین تبدیل و اجرا می‌کند.

JVM برای دسک‌تاپ طراحی شده است. تفاوت عمده دسک‌تاپ و موبایل در این است که موبایل‌ها نسبت به دسک‌تاپ و لپ‌تاپ از حافظه بسیار کمی (RAM) و سرعت پردازنده کمتری برخوردار هستند. JVM بسیار سنگین است و برای اجرای یک برنامه به رم و سرعت CPU زیادی احتیاج دارد. از آنجا که موبایل‌ها توانایی پردازش، حافظه رم و پردازنده قوی را ندارند، اندروید به نسخه بهینه شده و سبک تری از JVM نیاز دارد.
Dalvik برای جایگزینی JVM ایجاد شده است؛ Dalvik برای دستگاه‌های تعبیه شده که رم و سرعت CPU کمتری دارند، منظور شده است.

در مرحله دوم، بایت کد تولید شده توسط Proguard که از ابزارهای gradle است بهینه و کدگذاری می‌شود که فایل نتیجه این مرحله DEX نام دارد و پسوند آن هم dex. است.

در مرحله سوم، فرایند گرداوری نیاز به تبدیل پرونده‌های Class. و کتابخانه jar. به یک کلاس Class.dex دارد که حاوی بایت کدهای Dalvik است؛ این کار با دستور DEX انجام می‌شود.

دستور DEX تمام پرونده‌های class. و jar. را به هم پیوند می‌دهد تا در یک پرونده کلاس dex. با فرمت بایت کد Dalvik نوشته شود.

فایل‌های dex. ، کلاس‌ها و منابع برنامه مانند تصاویر در پرورنده‌ایی مشابه zip به نام یک بسته Android با پرونده apk. فشرده می‌شود.این کار با ابزار بسته بندی Android یا aapt انجام می‌شود.

یک توضیح کوتاه: پرونده dex یک نسخه فشرده از کلیه پرونده‌های class. است به گونه‌ای که dex خیلی سریع بارگیری می‌شود. حافظه کمتری می‌گیرد و در مقایسه با پرونده‌های class. به سرعت اجرا می‌شود.

پرونده apk. بسته برنامه‌ای است که توزیع می‌شود. با این حال، یک قدم دیگر وجود دارد که ممکن است لازم باشد ...

اگر می‌خواهید برنامه خود را از طریق فروشگاه Google Play توزیع کنید، باید آن را امضا کنید. امضای یک بسته برنامه به این معنی است که شما یک پرونده اضافی را در apk. ذخیره می‌کنید که براساس آن محتویات محتوای apk. و یک کلید خصوصی جداگانه تولید شده است.

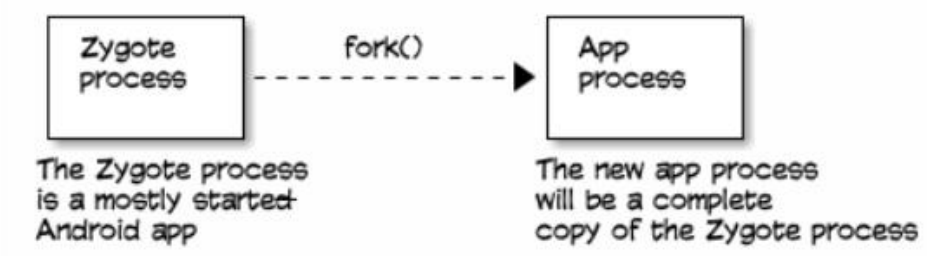
پرونده apk. از یک ابزار استاندارد jarsigner استفاده می‌کند که به عنوان بخشی از مجموعه توسعه جاوا در Oracle قرار دارد. ابزار jarsigner برای امضای پرونده‌های jar. ایجاد شده است ، اما همچنین با پرونده-های .apk کار خواهد کرد.

اگر پرونده apk. را امضا کنید، سپس باید آن را از طریق ابزاری به نام zipalign اجرا کنید، که اطمینان حاصل خواهد کرد که قطعات فشرده شده پرونده در مرزهای بایت تنظیم شده‌اند.

خب تا به اینجا برنامه ساخته شد، **حال نوبت شرح اجرای ان است:**

نحوه اجرای برنامه های Android اخیراً نسبتاً تغییر کرده است. از زمان API سطح ۲۱ ، ماشین مجازی قدیمی Dalvik با Runtime جدید Android جایگزین شده است.

هنگام اجرای یک برنامه کاربر درخواست راه اندازی یک برنامه را دارد.



روندی به نام Zygote برای راه اندازی برنامه استفاده می‌شود؛ Zygote یک نسخه ناقص از یک فرآیند Android است، فضای حافظه آن شامل کلیه کتابخانه‌های اصلی مورد نیاز هر برنامه است ، اما هنوز هیچ کد خاصی را برای یک برنامه خاص شامل نمی‌شود.

Zygote با استفاده از تماس سیستم fork (فورک نوعی کپی خیلی سریع) یک نسخه از خود را ایجاد می کند. Android یک سیستم لینوکس است و تماس fork می‌تواند خیلی سریع فرایندی مانند Zygote را کپی کند. به همین دلیل است که از فرآیند Zygote استفاده می‌شود؛ کپی کردن یک فرآیند نیمه شروع مانند Zygote خیلی سریعتر از آن است که بارگذاری یک فرآیند جدید از پرونده‌های اصلی سیستم انجام شود. Zygote یعنی برنامه شما سریعتر راه اندازی می‌شود.

Android کد dex. را به فرمت OAT بومی تبدیل می‌کند.

اکنون برنامه جدید باید کدهایی را که مخصوص برنامه شما است بارگیری کند. همان طور که گفته شد کد برنامه در پرونده کلاس dex. ، در بسته apk ذخیره شده است؛ بنابراین فایل کلاس dex. از apk. استخراج شده و در یک فهرست جداگانه قرار می‌گیرد. اما به جای قرار دادن کپی از فایل کلاس dex. ، اندروید کدهای بایت Dalvik را در کلاس dex. به کد دستگاه بومی تبدیل می‌کند.

اکنون تمام کدهای جاوا به کدهای کامپایل شده بومی تبدیل شده‌اند. از نظر فنی، کلاس dex. به یک شیء مشترک ELF تبدیل می‌شود. اندروید این فرمت کتابخانه را OAT می نامد و ابزاری که فایل کلاس dex. را تبدیل می‌کند dex2oat نامیده می‌شود.

کد تبدیل شده به کد دستگاه، اختصاصی برای CPU دستگاه Android خواهد بود.

سپس یک Dalvik Heap (حافظه) برای برنامه در نظر گرفته می‌شود و اپ آنقدر می‌تواند فضا اضافه کند تا به حداکثر برسد. اگر اپ از حافظه تخصیص داده شده فراتر رود پیام خطای " out of memory " نمایش داده می‌شود.

۲- بخش‌های مختلف مهندسی معکوس برای دستیابی به کد جاوا از برنامه اندرویدی apk. را توضیح دهید. از کدام ابزارها می‌توان در این فرآیند استفاده کرد؟

ابتدا به این می‌پردازیم که منظور از مهندسی معکوس چیست؟

مهندسی معکوس، "مهندسی سیاه" نیز نامیده می‌شود. فرایندی که از روی خروجی نهایی، نقشه یا طرح اولیه را می‌توان استخراج کرد؛ این فرآیند اغلب شامل جداسازی قطعات (وسیله مکانیکی ، مؤلفه الکترونیکی، برنامه رایانه‌ای، یا مواد بیولوژیکی، شیمیایی یا ارگانیک) و تجزیه و تحلیل دقیق مؤلفه‌ها و عملکردهای آن است. در اینجا به توضیح مهندسی معکوس برنامه‌های اندروید می‌پردازیم.

دو فرآیند در مهندسی معکوس وجود دارد: Disassembly و Decompilation

- Disassembly، فرایند ترجمه زبان ماشین به زبان مونتاژ است.
- Decompilation، معکوس کامپایل کردن است. ابزاری است که محتویات APK را به خود اختصاص داده و تلاش می‌کند تا کد اصلی را که برای ساختن کارکردهای مختلف برنامه استفاده شده است، نشان دهد. هیچ decompiler نمی‌تواند کد منبع دقیقی را که توسعه دهنده نوشته، دریافت کند.

همان طور که در سوال قبل گفته شد، APK حاوی داده‌های برنامه در قالب پرونده‌های Dalvik Zipped (.dex) فشرده شده است. پرونده‌های DEX از اجزای زیر تشکیل شده است:

- File Header
- String Table
- Class List
- Field Table
- Method Table
- Class Definition Table
- Field List
- Method List
- Code Header
- Local Variable List

یک استفاده مفید مهندسی معکوس نرم‌افزار، برای ایده گرفتن از برنامه‌های مطرح جهانی و درک چگونگی کارکرد آن برنامه‌ها است؛ همچنین استفاده مفیدتر بررسی عملکرد یک نرم افزار هست.

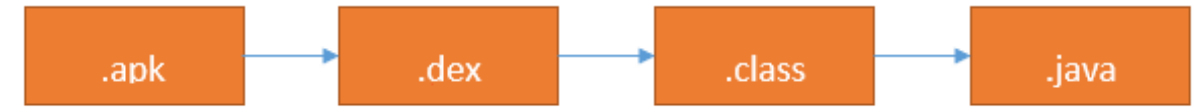
APK یک فایل فشرده مثل فایل‌های Zip است؛ با Rename کردن پسوند فایل به Zip ، به راحتی می‌توان محتویات برنامه را مشاهده کرد. پوشه‌هایی مثل assets و پوشه res وجود دارد که در آن‌ها عکس‌ها و

فایل‌های برنامه به صورت رمزنشده در این فایل زیپ قابل مشاهده هست. اما برای کدهای برنامه قضیه به این سادگی نیست؛ کدهای Java در یک فایل به اسمclasses.dex ذخیره شده‌اند.

بسیاری از ابزار برای مهندسی معکوس وجود دارد.

در اینجا به برخی از موارد که تقریباً همه استفاده‌کنندگان از آن استفاده می‌کنند، مانند: apktool ، Dex2jar، JD-Gui و ... می‌پردازیم.

مهندسی معکوس برنامه اندرویدی با نمودار زیر قابل درک‌تر است:



قبل از استفاده از ابزارها، اولین قدم دستیابی به پرونده APK است. بهترین راه این است که با استفاده از Android Debug Bridge (adb) آن را از تلفن همراه دریافت کنید.

ابتدا نام بسته برنامه را تعیین کنید.

```
adb shell pm list packages
```

نام کامل مسیر APK را برای بسته بندی مورد نظر دریافت کنید.

```
adb shell pm path com.example.someapp
```

فایل APK را از دستگاه Android به محیط آزمایش ببرید.

```
adb pull /data/app/com.example.someapp-2.apk
path/to/desired/destination
```

APKTOOL

یک ابزار برای مهندسی معکوس برنامه‌های باینری اندروید است.

- قادر به جدا کردن برنامه‌های کاربردی در شکل تقریباً اصلی و بسته بندی مجدد آنها پس از اصلاحات خاص است.
- می‌تواند برای اشکال زدایی از کد smali[1] استفاده کند.
- می‌تواند برای تغییر/ اضافه‌کردن برخی پشتیبانی‌ها یا ویژگی‌های سیستم‌عامل‌های مشتری و همچنین برای بومی سازی استفاده شود.
- این اتوماسیون برخی از کارهایی که در حال تکرار هستند مشخص و فراهم می‌کند که با ایجاد تعامل راحت تر کاربر برای کار با برنامه ها، به کاربر کمک می‌کند.

تصویر زیر تمام استفاده ممکن از apktool را نشان می‌دهد.

```
$ apktool
Apktool v2.0.3 - a tool for reengineering Android apk files
with smali v2.1.0 and baksmali v2.1.0
Copyright 2014 Ryszard Wi?niewski <brut.alll@gmail.com>
Updated by Connor Tumbleson <connor.tumbleson@gmail.com>

usage: apktool
 -advance,--advanced   prints advance information.
 -version,--version    prints the version then exits
usage: apktool if|install-framework [options] <framework.apk>
 -p,--frame-path <dir>   Stores framework files into <dir>.
 -t,--tag <tag>          Tag frameworks using <tag>.
usage: apktool d[ecode] [options] <file_apk>
 -f,--force              Force delete destination directory.
 -o,--output <dir>       The name of folder that gets written. Default is apk.out
 -p,--frame-path <dir>   Uses framework files located in <dir>.
 -r,--no-res             Do not decode resources.
 -s,--no-src             Do not decode sources.
 -t,--frame-tag <tag>    Uses framework files tagged by <tag>.
usage: apktool b[uild] [options] <app_path>
 -f,--force-all          Skip changes detection and build all files.
 -o,--output <dir>       The name of apk that gets written. Default is dist/name.apk
 -p,--frame-path <dir>   Uses framework files located in <dir>.
```

[1] Smali نسخه قابل خواندن انسان از bytecode Dalvik است.

Dex2Jar

همان طور که گفته شد، مؤلفه اصلی برنامه android ومنطق برنامه اصلی برنامه‌ی اندرویدی در پرونده کلاس dex. است. این پرونده‌ها قابل خواندن کاربر نیست. Dex2jar برای تبدیل پرونده به قالب کلاس‌های قابل خواندن برای مشاهده کاربر استفاده می‌شود. با استفاده از این ابزار می‌توان کد منبع برنامه را به عنوان کد جاوا مشاهده کرد.

طریقه استفاده:

تبدیل فایل dex. به فایل class. (zipped as jar)

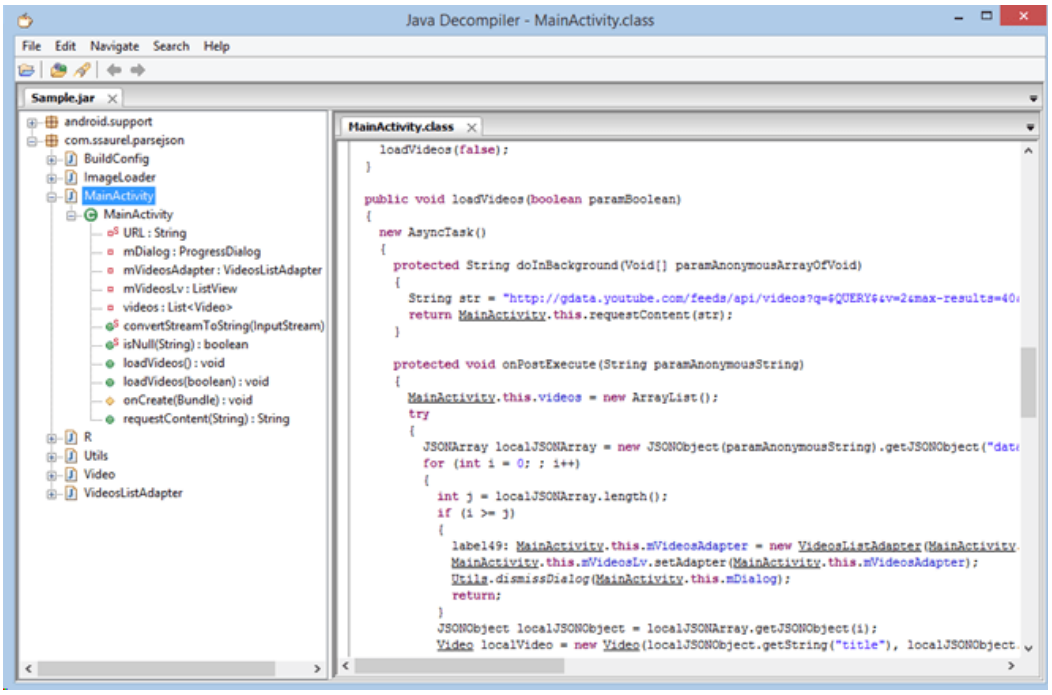
```
sh d2j-dex2jar.sh -f ~/path/to/apk_to_decompile.apk
```

و فایل خروجی به این صورت خواهد بود:

```
apk_to_decompile-dex2jar.jar.
```

JD-GUI

JD-GUI یک ابزار گرافیکی مستقل است که کدهای منبع جاوا پرونده های ".class" را نمایش می‌دهد. برای دسترسی سریع به روش‌ها و فیلدها می‌توان کد منبع بازسازی شده را با JD-GUI مرور کرد. بنابراین اساس آن decompiler GUI Java است.



روش مهندسی معکوس با استفاده از JD-GUI:

۱) تغییر نام test.apk به test.zip.
۲) استخراج test.zip و بازکردن فایل.
۳) فایل کلاس dex. را از پوشه آزمون کپی کنید.
۴) پوشه استخراج شده dex2jar گذشته است.
۵) دستور d2j-dex2jar.bat را در کلاس dex. اجرا کنید.
۶) پرونده کلاس -dex2jar.jar را به JD-GUI منتقل کنید.
۷) اکنون باید داده‌های مهندسی معکوس را تجزیه و تحلیل کنید.

Apk Analyzer

این یک ابزار تجزیه و تحلیل کاربردی استاتیک و مجازی است. می‌توان از آن برای تجزیه و تحلیل منابع API، مشاهده معماری برنامه و وابستگی‌ها و جدا کردن کدهای بایت در برنامه‌های Android استفاده کرد.

برای راه اندازی ان، دستور زیر را اجرا کنید:

```
java -jar ApkAnalyser.jar
```

استفاده از ان:

- برنامه آنالایزر Apk را راه اندازی کنید.
- مسیرهای مورد نیاز مانند Android sdk و محل فایل apk را تنظیم کنید.
- شروع به تجزیه و تحلیل کنید با انتخاب فایل انالیز (File -> Analyse).

شعار اصلی دریافت و خواندن کد ، اطلاع از فعال سازی و تعامل برنامه است؛ بنابراین با استفاده از تمام روش‌های فوق می‌توان یک کد قابل خواندن برای برنامه دریافت کنیم. اکنون یا به سادگی می‌توانیم آن را برای عملکردها یا بدافزارها تجزیه و تحلیل کنیم یا همه چیز را طبق هدف خود اصلاح کنیم.