Services, Libraries, Gradle plugin

DATE 2023

Lukas Prokop, Simona Kurnavova

Services

Services

- Long running operation in background
- Not bound with UI
- Can expose API for other applications
- By default runs on UI thread

Services

- **Types:**
  - Started
  - Bound
- **Visibility:**
  - Background
    - Limited since Oreo (API >= 26)
  - Foreground

Started service

- Independent from caller
- Do not return result to caller

Started service - starting

- Started by calling

  `Context#startService()`

- Override

  `Service#onStartCommand()`

Started service - ending

- Stop by self

  `Service#stopSelf()`

- From outside

  `Context#stopService()`

Bound service

- Client server interface for communication
- Lightweight RPC communication

Bound service - binding

- Component bind to it by calling

```
Context#bindService(service: Intent
                    conn: ServiceConnection,
                    flags: Int): Boolean
```

- Override

```
Service#onBind(intent: Intent): IBinder?
```

- Service returns IBinder object for interaction

Bound services - unbind

- Clients call

```
Context#unbindService(conn: ServiceConnection)
```

- System destroys service, when all clients unbond from it

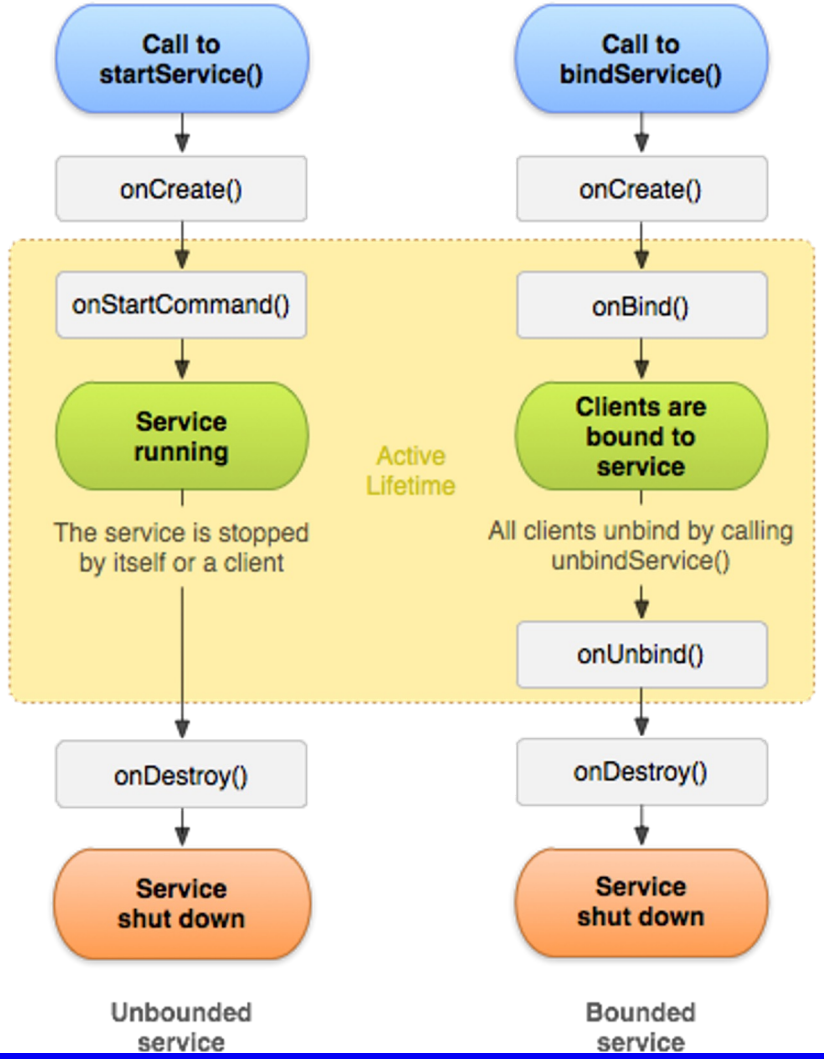
Service connection

- **Define callbacks for service binding**
- `fun onBindingDied(name: ComponentName)`
  - Binding is dead
  - Can happen during app update
  - Unbind and rebind
- `fun onNullBinding(name: ComponentName)`
  - Service#onBind returns null
  - Unbinding is still required
- `fun onServiceConnected(name: ComponentName, service: IBinder)`
  - Connection with the service has been established
- `fun onServiceDisconnected(name: ComponentName)`
  - Connection has been lost
  - Process hosting service crashed or been killed
  - Service connection remain active (onServiceConnected can be called again)

IBinder/Binder

- Remotable object for communication with bounded service
- Can be defined by AIDL

Service lifecycle



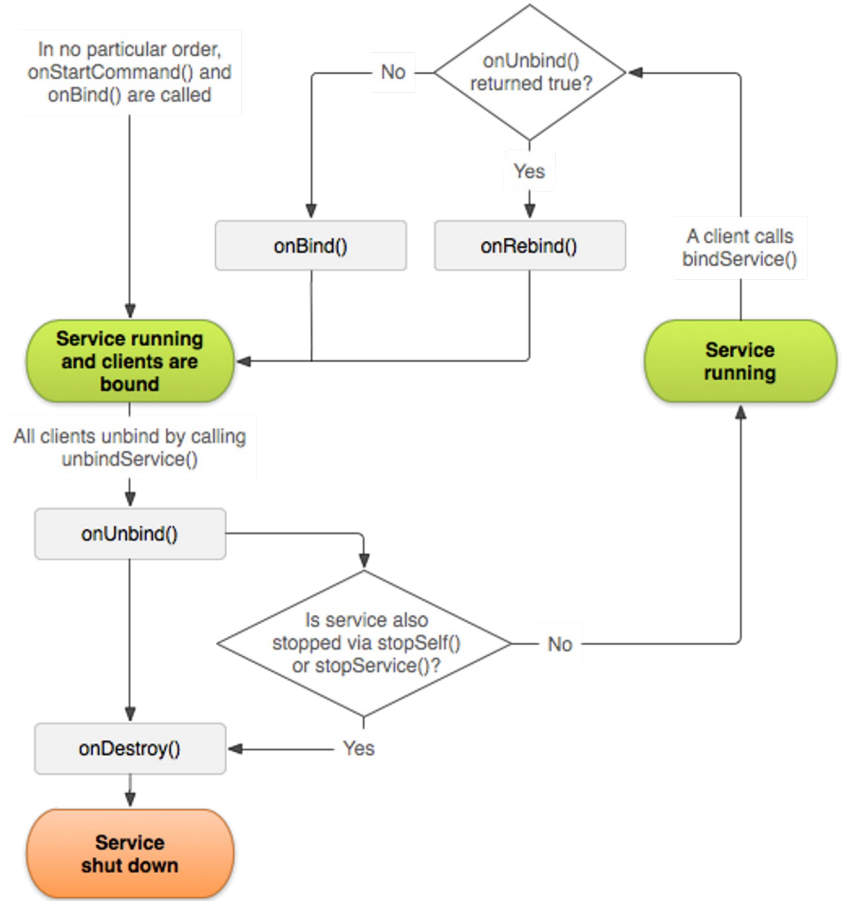
Service lifecycle

- **onCreate()**
  - Called when the service is being created (after first call of startService() or bindService())
- **onStartCommand()**
  - Called when startService() is called, delivers starting intent
  - Returned value specify behaviour when it’s killed by system
    - START_STICKY - don’t retain intent, later when system recreate service null intent is delivered (explicitly started/stopped services)
    - START_NOT_STICKY - if there is no start intent, take service out of the started state. Service is not recreated.
    - START_REDELIVER_INTENT - last delivered intent will be redelivered, pending intent delivered at the point of restart

Service - lifecycle

- **onBind()**
  - When another component binds to service
  - Returns Binder object for communication
- **onUnbind()**
  - When all clients disconnected from interface published by service
  - Returns true when onRebind should be called when new clients bind to service, otherwise onBind will be called
- **onRebind()**
  - Called when new clients are connected, after notification about disconnecting all client in its onUnbind
- **onDestroy()**
  - Called by system to notify a Service that it is no longer used and is being removed.
  - Cleanup receivers, threads..

Bound Service lifecycle



Background service

- **On background by default**
- **Strongly limited since Android Oreo (API 26)**
  - Not possible to start background service when app is not on the foreground

Foreground service

- **Service process has higher priority**
- **User is actively aware of it**
- **System not likely to kill foreground services**
- **Requires permanent notification (cannot be dismissed), it is under Ongoing header**
- **Use `Context#startForegroundService(Intent)`**
  - 5s window to make the service foreground
- **By calling `Service#startForeground(int, Notification)`**
- **Remove from foreground `stopForeground()`**
- **Apps targeting Android 9 (API 28) or higher must define**
  - FOREGROUND_SERVICE permission (normal permission)

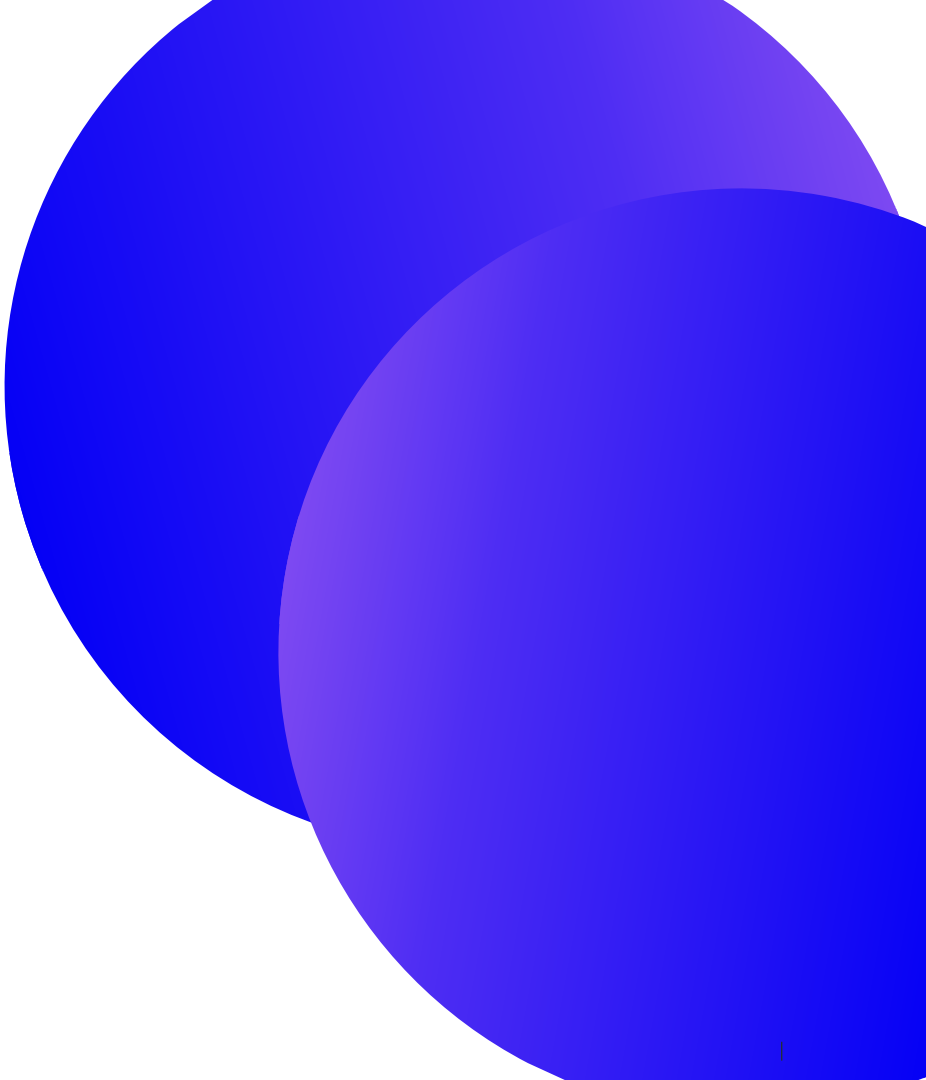
IntentService – Deprecated in API 30

- Subclass of Service
- Uses worker thread to handle requests
- Handle only one request at one time
- Creates work queue
- Stops when it run out of work
- Override on`HandleIntent(Intent)` for processing requests, runs on worker thread
- Replacement: Workers and WorkManager

JobIntentService – Deprecated

- Replacement of IntentService
- Part of support library
- Uses JobScheduler
- Requires WAKE_LOCK permission

Libraries, gradle plugins, etc...



Android jetpack

- **Set of libraries from google**
- **https://developer.android.com/jetpack**
- **Groups**
  - Foundation
    - AppCompat, Android KTX, Multidex, Test
  - Architecture
    - Data binding, Lifecycles, LiveData, Navigation, Paging, Room, ViewModel, WorkManager
  - Behavior
    - Download manager, Media & playback, Notifications, Permissions, Preferences, Sharing, Slices
  - UI
    - Animations & transitions, Auto, Emoji, Fragment, Layout, Palette, TV, Wear OS

DexCount plugin

- Computes methods count in APK
- Visualize count in nice chart
- https://github.com/KeepSafe/dexcount-gradle-plugin

Retrofit

- **A type-safe HTTP client for Android and Java**
- **Simplify communication with some API service**
- **Configurable**
  - OkHTTP 3 client - compression, timeouts
  - Supports multiple convertors
    - Gson
    - Jackson
    - Moshi
    - Protobuf
    - Wire
- **https://square.github.io/retrofit/**

OkHttp

- An HTTP & HTTP/2 client for Android and Java applications
- Supports sync/async calls
- Supports multiple addresses per URL (Load balancing, failover)
- http://square.github.io/okhttp/

Dagger

- Dependency injection framework
- Decouple code
- Better testing
- https://dagger.dev/android.html

Flipper

- **Debug platform by Facebook**
- **Inspect**
  - ViewHiearchy
  - Database
  - Shared preferences
  - Network traffic
- **https://fbflipper.com/**

LeakCannary

- **Helps with finding memory leaks**
- **https://github.com/square/leakcanary**

Ktor

- **Multiplatform http client/server**
- https://ktor.io

Kotlinx.serialization

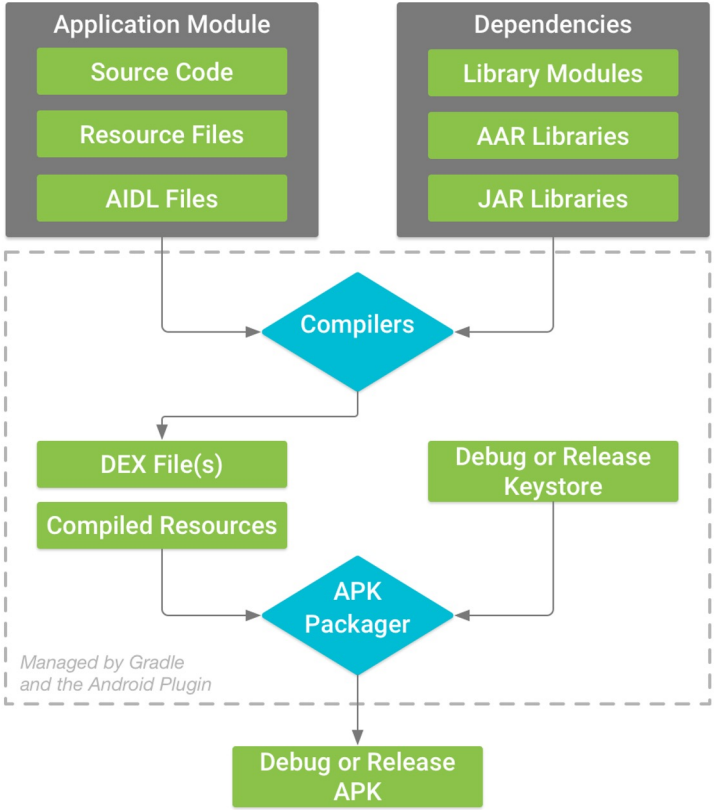
- **Data serialization/deseralization**
  - Json
  - Protocol buffers
- https://kotlinlang.org/docs/serialization.html

Demo

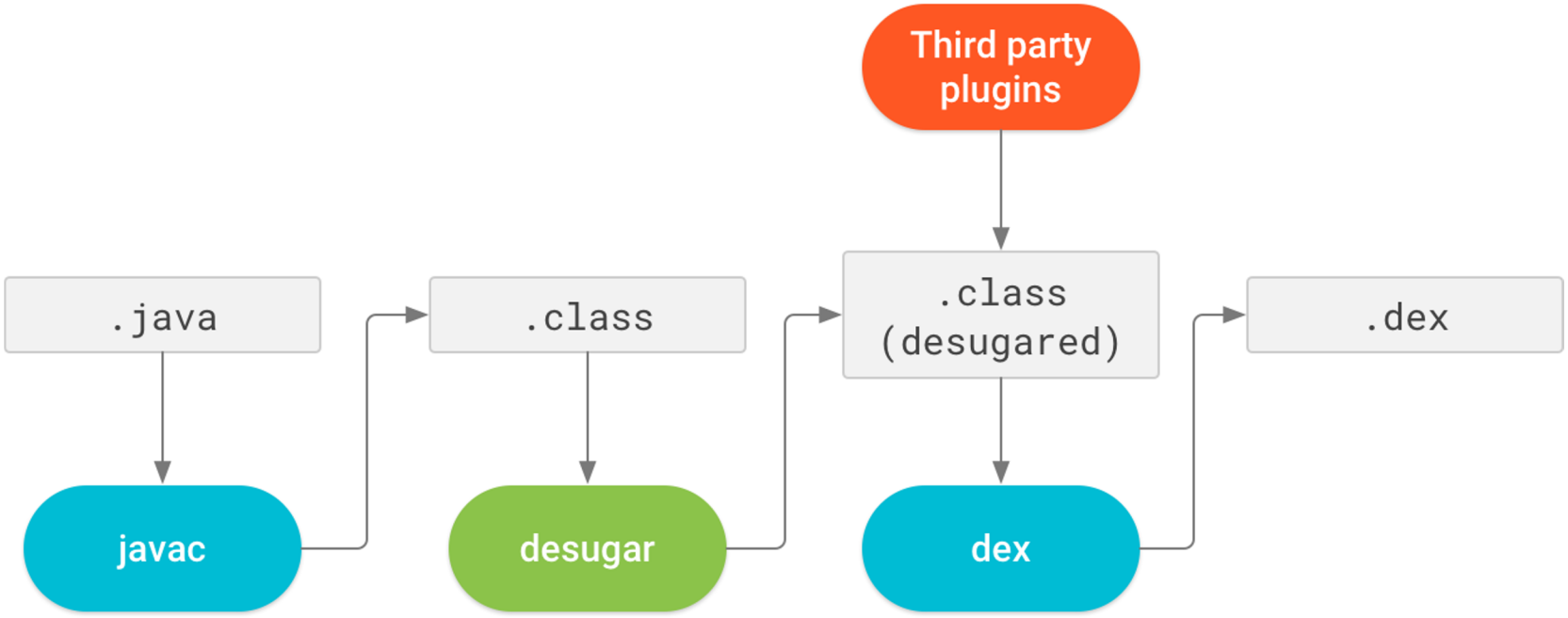
- **Implement network repository**

Build process and APK structure

Build process



Support java 8 features (d8)



Building android apps limitations

- **Dex limit 64k methods**
  - Shrink unused methods and classes (libraries)
    - You need to specify what is entry point, build dependency graph. Classes which are not part of graph are removed, unused methods as well
    - When you work on library, provide proguard rules together with library
  - Sometimes is better copy some classes from library into application
    - for example guava library
- **Google doesn’t allow code side load**

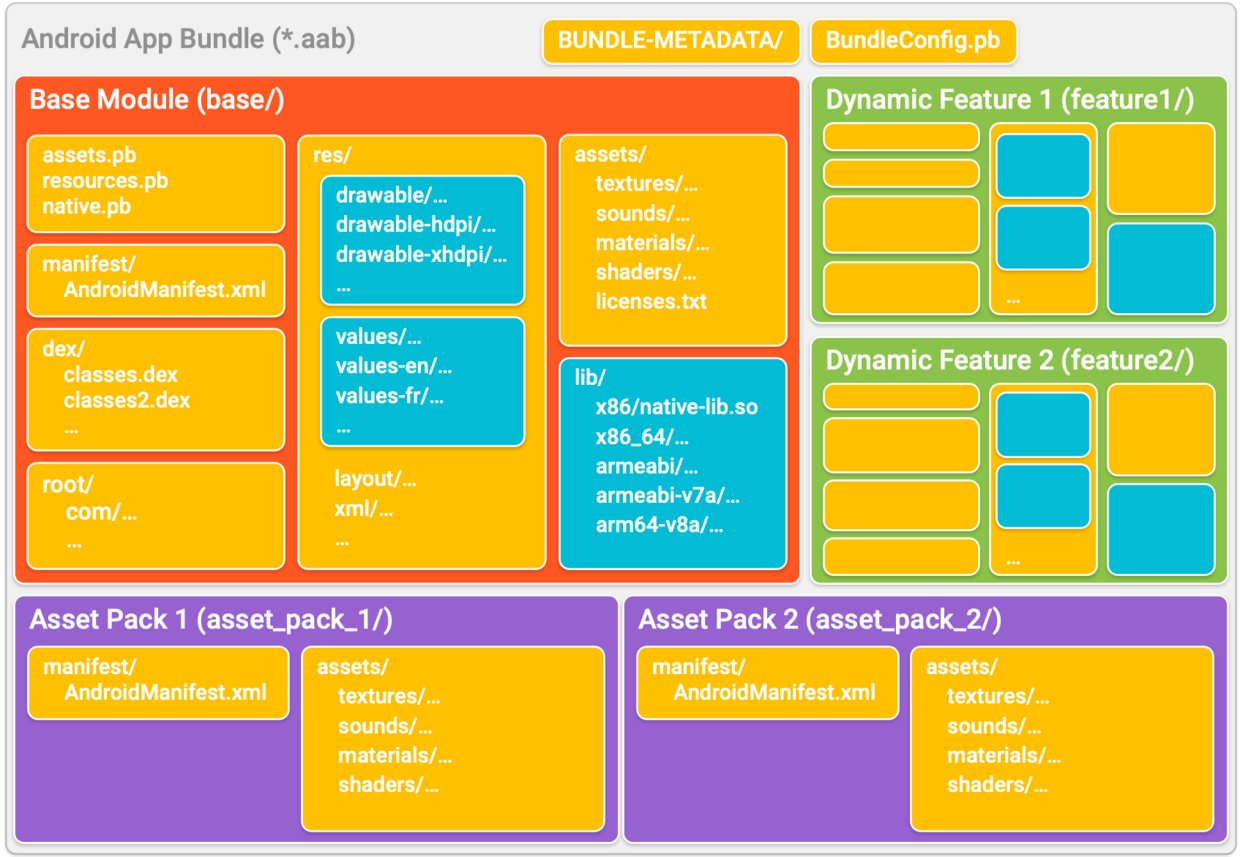
Multidex

- Native support since API-21 for older version support library
- Try to avoid using multidex, it slows down application start
- Splits classes to multiple dex files
- on API>=21 dex files are converted to single .oat file (ART runtime)
- Main dex file loaded when app is started
- Loading of additional dex files is performed during initialization
- Dex files are in app folder
- https://www.blackhat.com/docs/ldn-15/materials/london-15-Welton-Abusing-Android-Apps-And-Gaining-Remote-Code-Execution.pdf

Apk structure

```
.
├── AndroidManifest.xml- binary XML form of Android manifest
├── classes.dex      - classes compiled to dex
├── META-INF
│   ├── CERT.RSA       - Application certificate
│   ├── CERT.SF        - SHA-1 digest of corresponding lines in the MANIFEST.mf
│   └── MANIFEST.MF    - Manifest file list of files with their SHA-1 hash
├── assets             - assets files
├── res                  - resources
│   ├── drawable-hdpi-v4
│   ├── drawable-mdpi-v4
│   ├── drawable-xhdpi-v4
│   ├── drawable-xxhdpi-v4
│   ├── drawable-xxxhdpi-v4
│   ├── layout
│   ├── menu
│   └── xml
└── resources.arsc       - resources compiled to binary form
```

AppBundle



https://developer.android.com/guide/app-bundle

Proguard/R8

- Shrink - smaller code, faster build
- Optimize - faster code, removing static conditions
- Obfuscate - make it harder to read

Decompile apk

- **Unzip the apk**
- **Dex2Jar to convert classes.dex to jar archive**
  - https://github.com/pxb1988/dex2jar
- **jd-gui to view decompiled classes**
  - http://jd.benow.ca/
- **BytecodeViewer**
  - https://bytecodeviewer.com/
- **Android studio apk analyzer**
  - Easy to check resources
  - Compare multiple apk