Android 앱의 크기를 줄이는 10가지 방법

Jirawatee | 2019.09.06
I am a technology evangelist at LINE.

> *참고. 이번 블로그는 LINE DEVELOPER DAY 2018에서 Jirawat Karanwittayakarn 님이 발표한 'Build your Android app Faster and Smaller than ever' 세션을 기록한 내용을 각색하여 옮긴 글입니다(원문 기록 및 제공: logmi).*

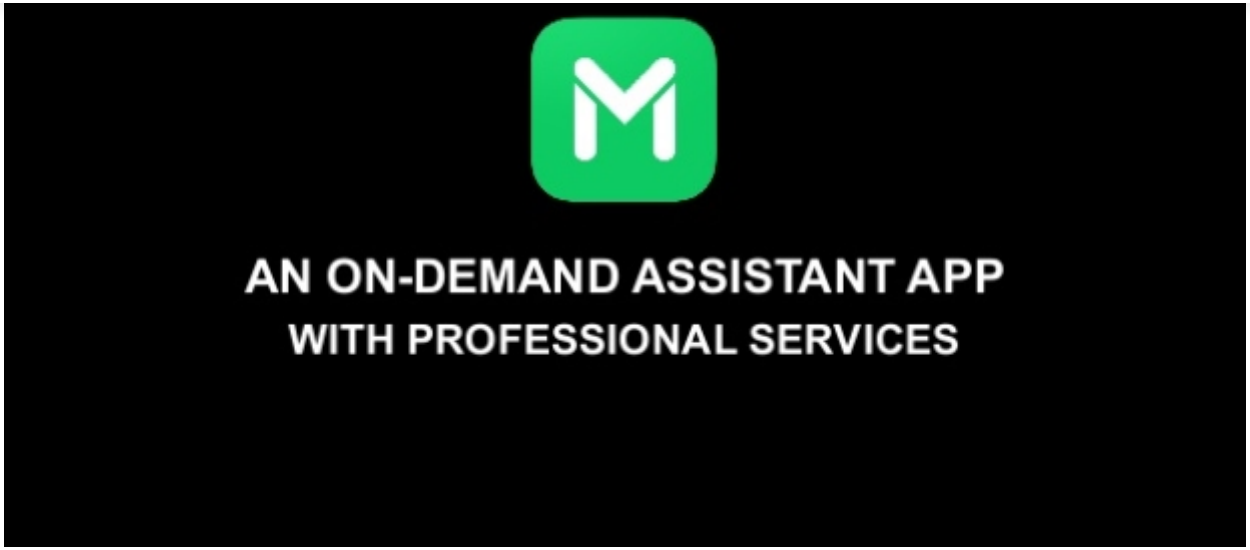
들어가며

안녕하세요. LINE Thailand에서 테크놀로지 에반젤리스트를 맡고 있고, Android 개발자로 5년 넘게 일해왔으며, Firebase의 Google Developer Expert이기도 한 Jirawat Karanwittayakarn입니다.

저는 이번 글에서 앱 크기를 줄이는 방법을 공유하려고 합니다. 먼저 이번 글에서 예시로 사용할 LINE MAN에 대해 간단하게 소개하겠습니다.

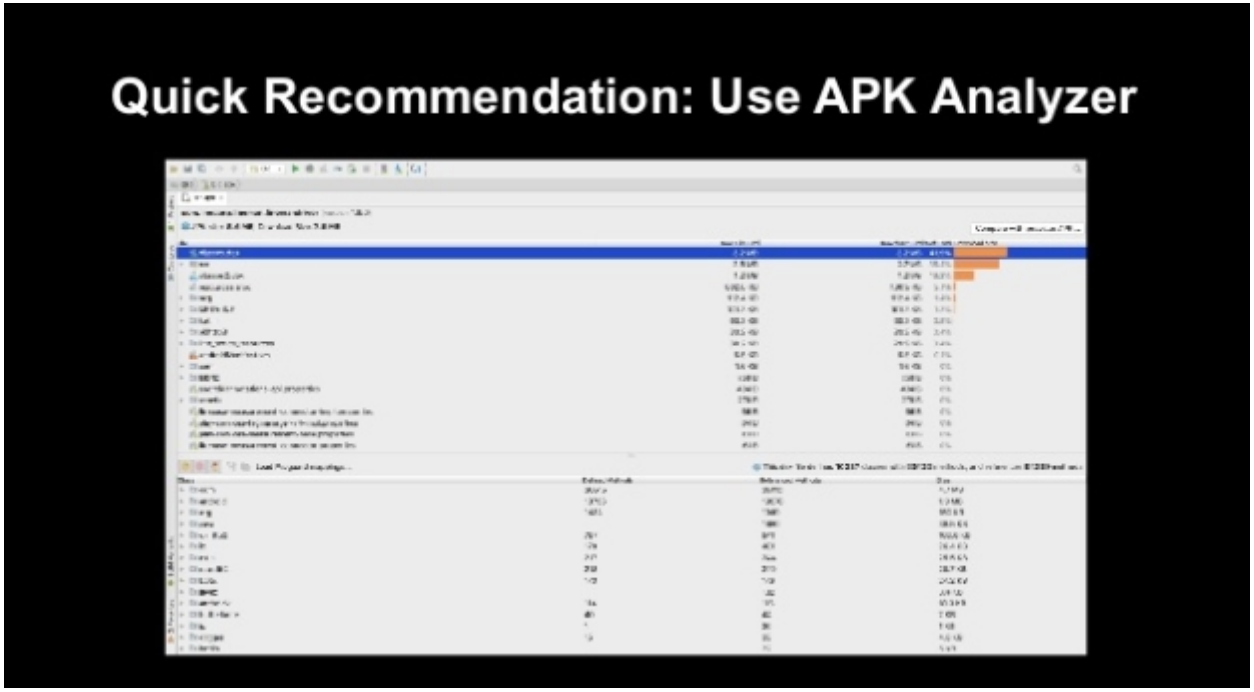
LINE MAN이란?

LINE MAN은 태국에서 서비스하고 있는 맞춤형(On-demand) 어시스턴트 앱으로, 방콕 내 사용자들에게 있어 '일상 생활의 No. 1 어시스턴트(No.1 Daily Life Assistant)'라 할 수 있습니다. 현재 음식 배달과 편의점 상품 배달, 메신저, 택배, 택시까지 총 5가지의 서비스를 제공하고 있는데요. 태국에서 월 평균 70만 명 이상이 LINE MAN을 사용하고 있으며, 음식 배달의 경우 현재 5만 곳 이상의 레스토랑과 제휴를 맺고 있습니다.



LINE MAN 앱은 드라이버용과 클라이언트용이 있는데요. 여기서는 드라이버용 앱을 사용해서 설명하겠습니다.

방법을 설명하기에 앞서 작업에 필요한 도구인 APK Analyzer를 소개하겠습니다. APK Analyzer는 Android Studio 버전 2.2부터 포함되었는데요. APK를 Android Studio에 업로드해서 레이아웃이나 스크린 등 작성된 함수 코드(function code)를 분석해 주는 훌륭한 툴입니다. 이 툴을 사용하면 '어떤 것이 큰 용량을 차지하고 있는지' 또는 '어느 부분의 용량이 작은지'를 파악할 수 있고, 어디서부터 먼저 수정해야 할지 판단하는데 도움이 됩니다.



이번 글에서 소개할 내용에는 저희의 경험에 Google I/O에서 들은 내용, DEVELOPER BUILD CLINIC[1]에서 얻은 내용이 포함되어 있습니다. 또한, 이번 글의 내용은 각 프로젝트의 성격이나 환경에 따라 적용 방식이나 결과에 차이가 발생할 수 있다는 점을 염두에 두시기 바랍니다.

Android 앱의 크기를 줄이는 10가지 방법

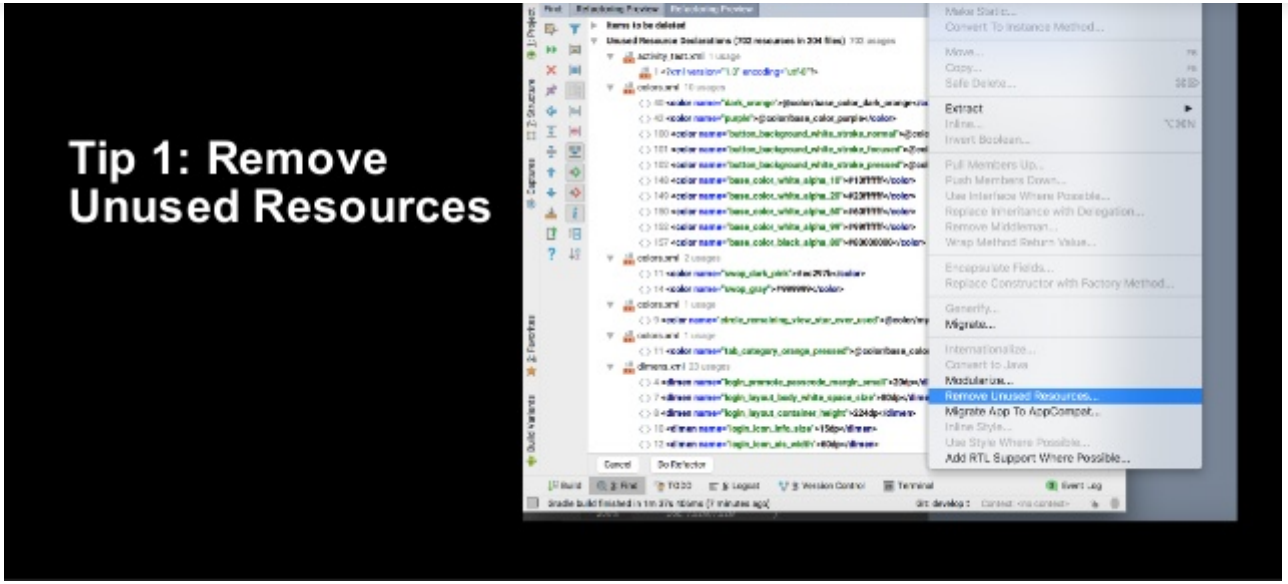
앱 크기는 사용자 참여(user engagement)에 큰 영향을 미치는 중요한 요인입니다. 그럼 앱 크기를 줄이는 10가지 방법에 대해 알아보겠습니다.

적용 전

각 팁을 적용하기 전에 확인한 LINE MAN 드라이버용 앱의 크기는 약 9MB입니다.

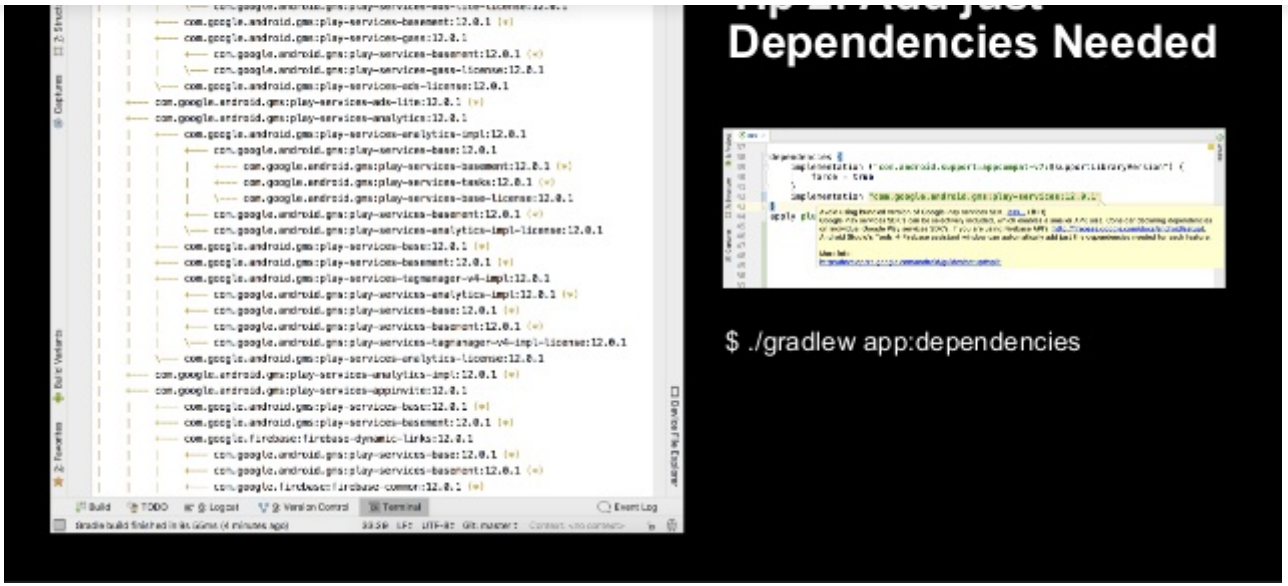
1. 사용하지 않는 리소스 삭제

많은 분들이 현재 레거시 프로젝트에도 참여하고 있을 텐데요. 사용하지 않는 리소스가 있다는 걸 알아도 그걸 직접 삭제하고 싶지는 않을 겁니다. 그럴 때 아래처럼 작업표시줄에서 클릭만 하면 Android Studio가 이미지나 레이아웃 등 사용하지 않는 리소스를 자동으로 탐지하고 미리보기로 목록을 만들어 줍니다. 탐지와 삭제가 모두 원 클릭으로 가능해 아주 편리합니다.



2. 필요한 종속성만 추가

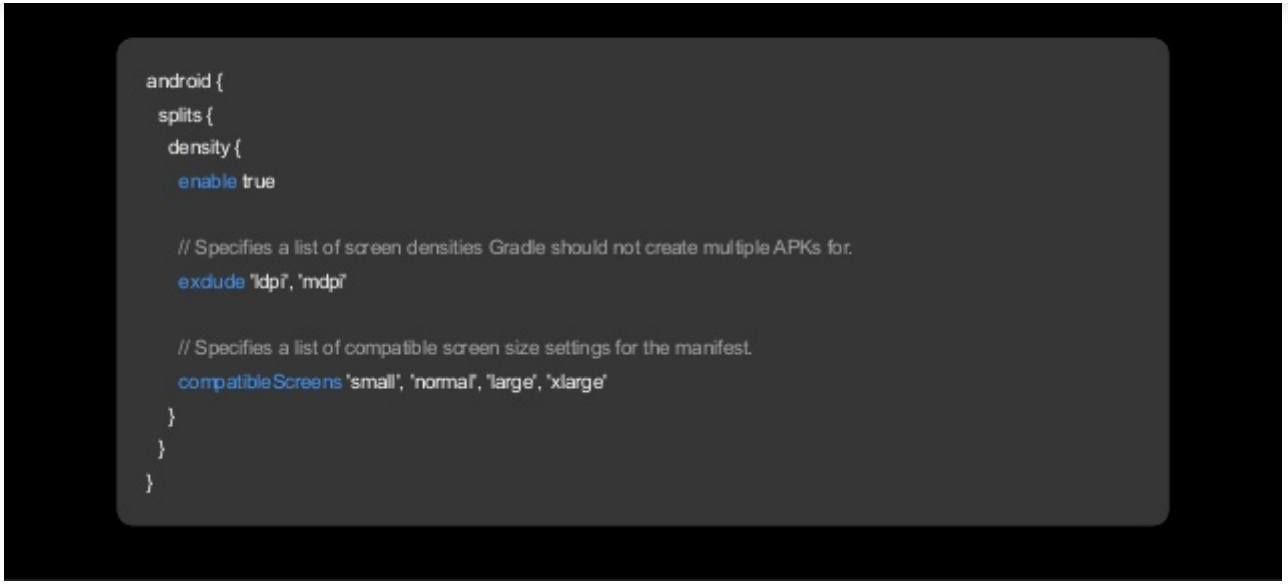
종속성(dependency) 중에는 Google Play Services나 Facebook SDK처럼 아주 많은 라이브러리를 포함하고 있는 것이 있는데요. 별도 지정없이 전체를 선택하면 앱의 크기가 너무 커져버리기 때문에 어느 라이브러리가 필요한지 지정해야 합니다. 예를 들어 인증이 필요할 땐 `googleservice-authentication` 만 선택하면 앱 크기를 줄일 수 있습니다.



라이브러리가 얼마나 많이 포함되어 있는지 모르겠다면 종속성을 분석해서 트리 형태로 보여주는 `./gradlew app:dependencies` 명령어를 사용하여 확인 후 필요 없는 것은 지우시기 바랍니다.

3. 여러 개의 APK를 화면 밀도에 맞춰 생성

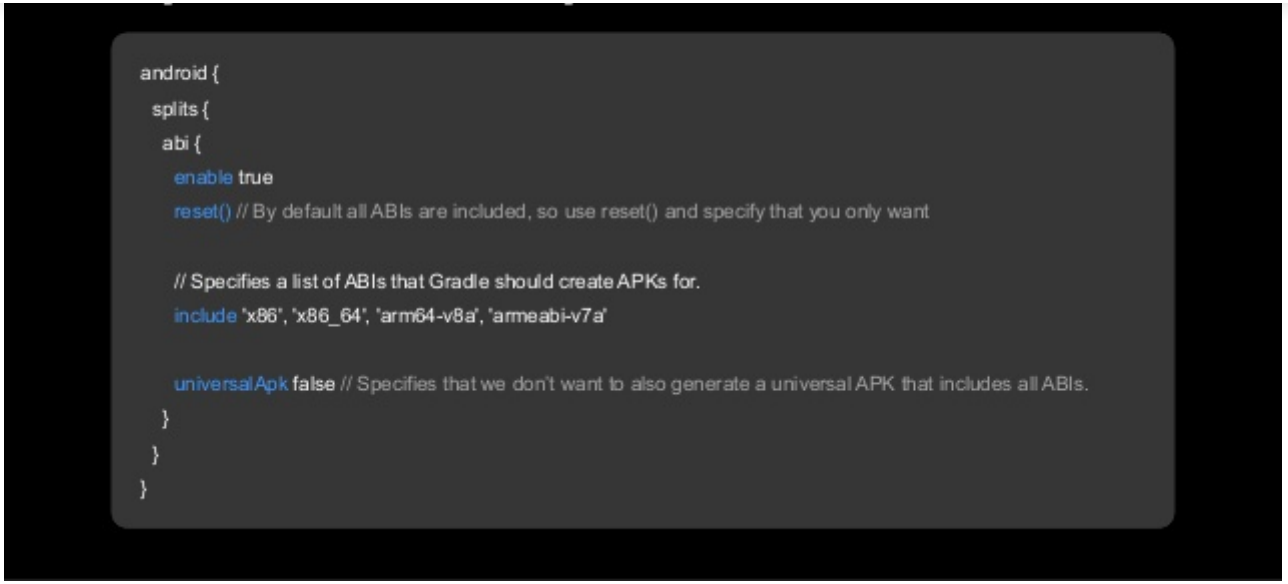
Android Studio에서는 모든 환경에서 실행할 수 있는 유니버설한 APK를 생성합니다. APK 하나로 모든 화면 밀도를 커버할 수 있게 되어 있는데요. 밀도가 낮은 화면에서 중간 정도인 화면, 매우 높은 화면까지 모두 다 지원합니다. 여기서 '어느 정도의 밀도를 지원할 것인지를 선택하는 것'이 앱 크기를 줄일 수 있는 팁입니다.



예를 들어 밀도가 낮은 화면은 고려 대상이 아니거나 지원하고 싶지 않을 수 있습니다. 특히 Android 시장에서는 화면 밀도가 낮은 경우가 별로 없기 때문에 이를 제외한 후에 APK를 빌드하고 싶은 경우가 있을 텐데요. Android Studio는 여러 개의 APK를 생성해 주기 때문에 그걸 Google Play에 올리기만 하면 사용자는 화면 밀도에 맞는 APK만 자동으로 다운로드하게 됩니다.

4. 여러 ABI(application binary interface)를 지원하는 여러 개의 APK 생성

현재 Android 시장에서 사용되는 CPU 아키텍처에는 7가지 종류가 있는데요. 그 중 아래 이미지에 보이는 4가지가 주로 사용되고 나머지는 많이 사용되고 있지 않습니다. 따라서 많이 쓰이지 않는 ARM API, MIPS, NVS64를 제외하고 생성한 여러 개의 APK를 모두 Google Play에 업로드하면 각 사용자에 맞는 APK가 다운로드됩니다.



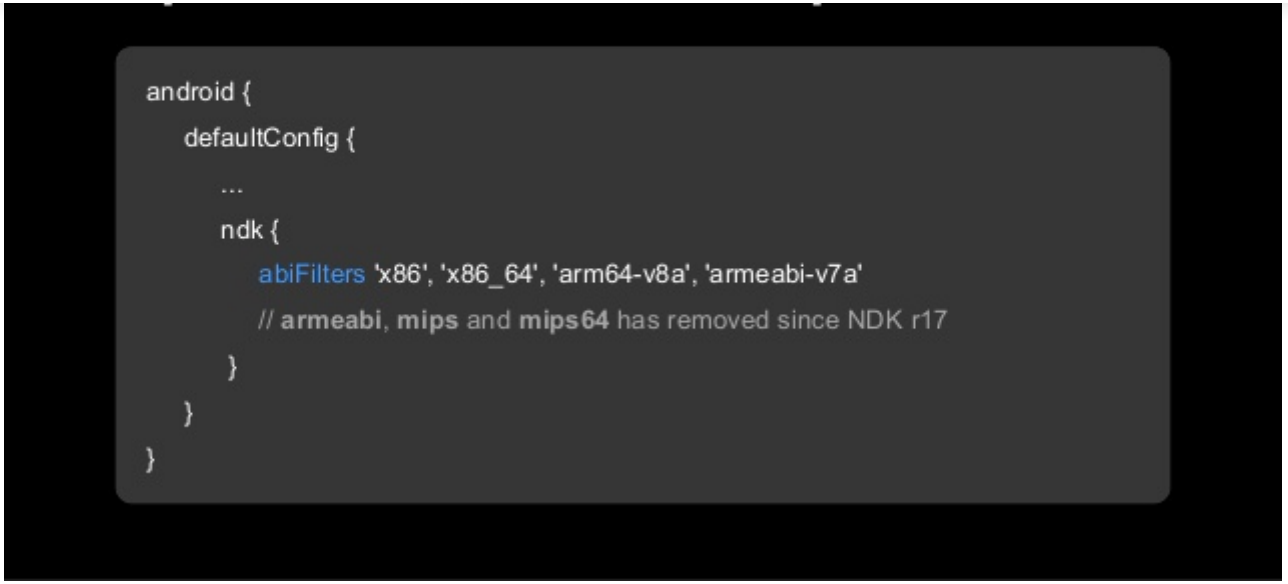
```
android {
  splits {
    abi {
      enable true
      reset() // By default all ABIs are included, so use reset() and specify that you only want

      // Specifies a list of ABIs that Gradle should create APKs for.
      include 'x86', 'x86_64', 'arm64-v8a', 'armeabi-v7a'

      universalApk false // Specifies that we don't want to also generate a universal APK that includes all ABIs.
    }
  }
}
```

5. 특정 ABI를 지원하는 하나의 APK를 생성

네 번째 팁과 약간 다른데요. 지원할 CPU의 아키텍처를 정한 뒤 Android Studio로 그 아키텍처에 맞춘 APK를 하나만 생성하는 방법입니다.



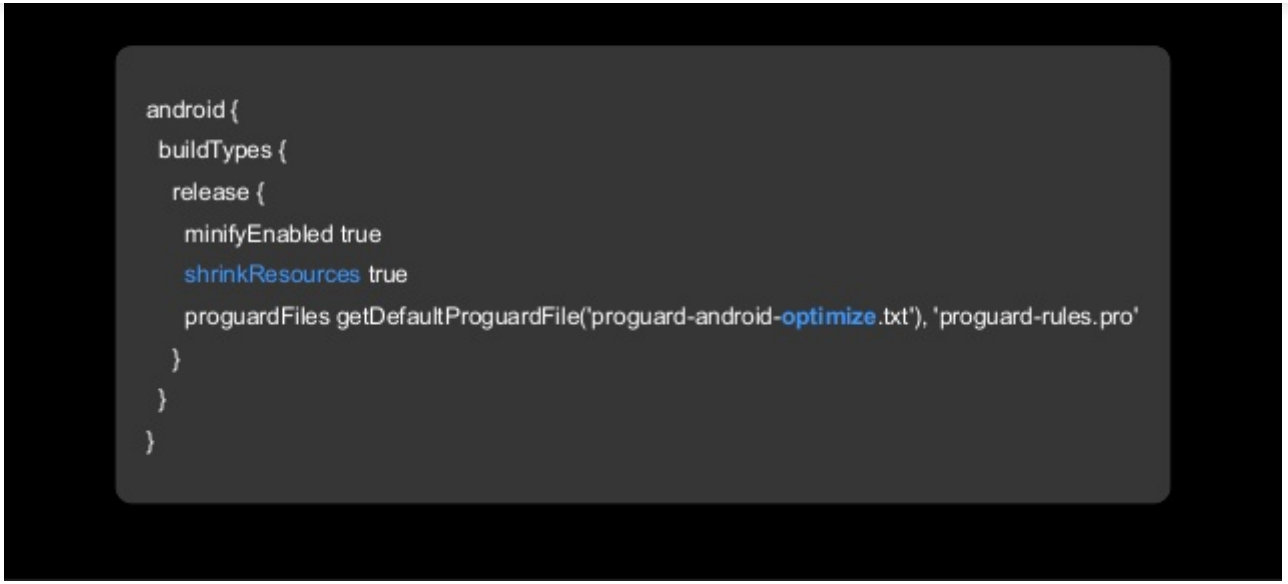
여러 개의 APK를 생성해서 Google Play에 배포하면 기기에 따라 가끔씩 앱이 크래쉬되는 일을 겪은 적이 있어서 저는 이 방법이 더 안전하다고 생각하여 추천합니다.

6. 사용하지 않는 선택적(alternative) 리소스 삭제

특정 종속성이나 라이브러리에는 전 세계의 언어가 포함되어 있는 경우가 있는데요. 앱에서 특정 언어만 지원 대상으로 삼는 경우에는 굳이 전부 앱에 포함시킬 필요가 없습니다. 예를 들어 LINE MAN 드라이버 앱은 현재 태국에서만 사용 가능하니 언어를 영어와 태국어로 한정할 수 있습니다. 아래 이미지처럼 `resConfigs` 설정을 사용해서 필요한 언어만 포함시키면 앱 크기를 줄일 수 있습니다.

7. 리소스 축소(shrinkResources) 사용

Android Studio로 작업할 때 `minifyEnabled` 라는 속성을 볼 수 있습니다. 보통 `false` 로 설정되어 있고 소스코드를 난독화해서 보안을 강화하고 싶거나 앱의 크기를 줄이고 싶을 때 `true` 로 변경하는데요. 이때 `shrinkResources` 라는 속성도 추가해서 `true` 로 설정하면 minify 작업 후 사용하지 않는 리소스를 삭제합니다.

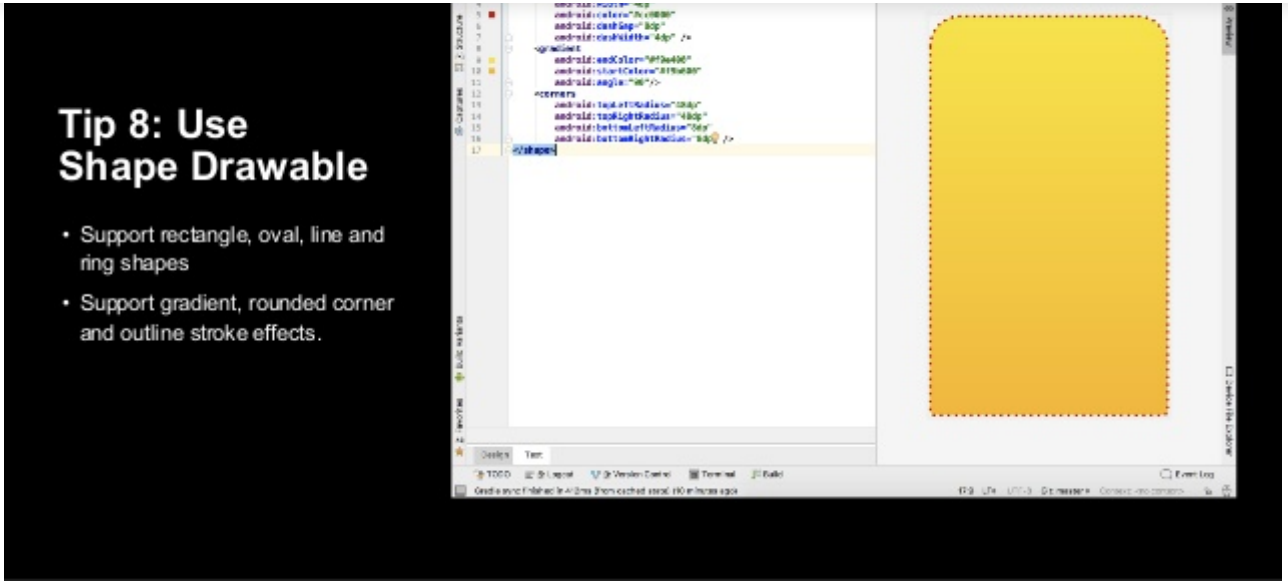


```
android {
 buildTypes {
  release {
   minifyEnabled true
   shrinkResources true
   proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
  }
 }
}
```

Gradle로 `-optimize` 를 `proguard-android` 뒤에 추가하면 앱 생성을 위한 최적화가 추가로 진행되어 앱 크기가 줄어듭니다. 단, 빌드 시간이 늘어나기 때문에 위 코드처럼 `release` 블록에서 이 방법을 사용하는 것은 괜찮지만, 개발 단계에서 사용하면 시간이 너무 오래 걸릴 수도 있으니 주의해야 합니다.

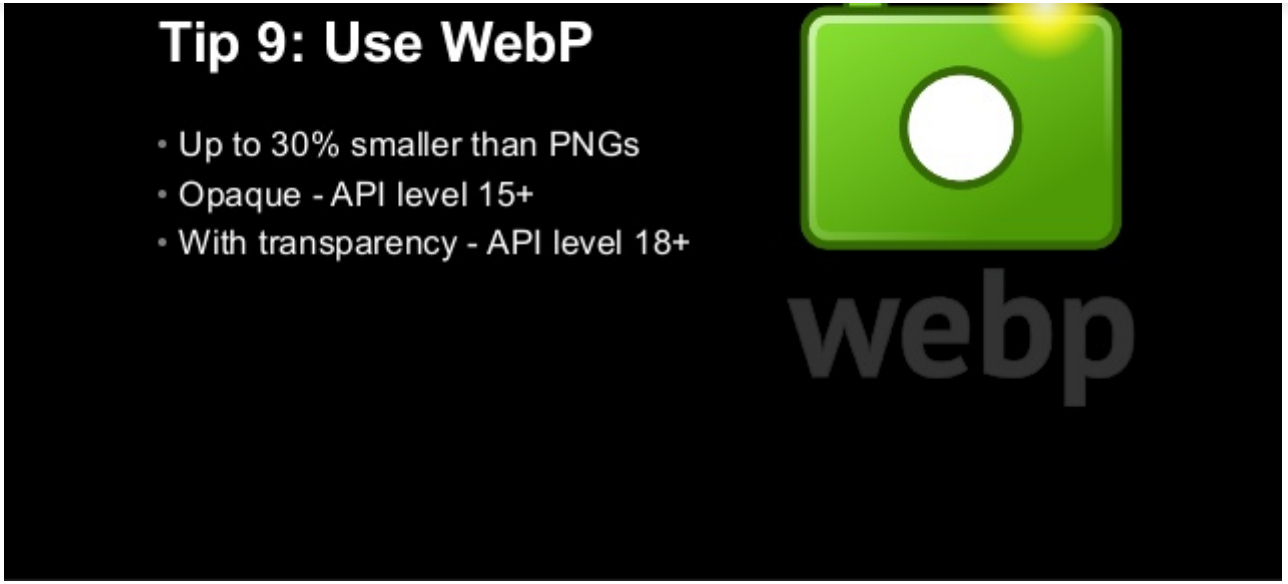
8. 셰이프 드로어블(shape drawable) 사용

Shape Drawable은 많은 분들이 사용하는 비트맵보다 용량이 작으면서 직사각형이나 타원형, 모서리 라운딩 등을 모두 XML로 작성할 수 있습니다. 따라서 앱 크기는 물론, 개발자가 직사각형이나 타원형, 그라데이션이 있는 배경을 제작하고 싶어 디자이너 팀에 의뢰한 후 기다리는 시간도 줄일 수 있습니다.

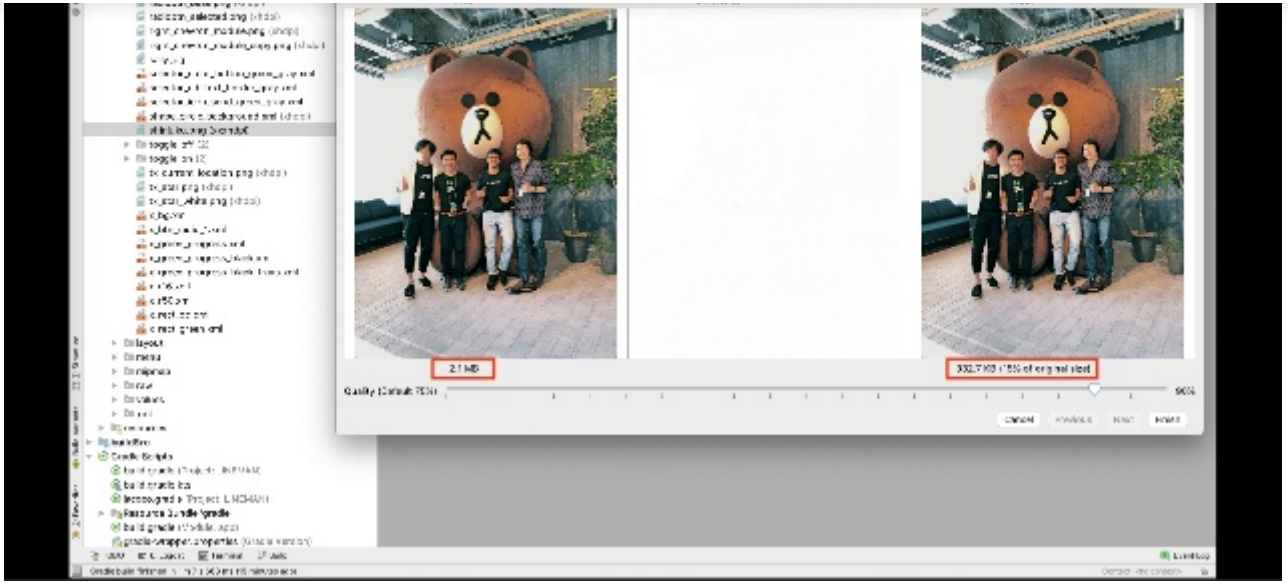


9. WebP 사용

WebP는 Google이 제공하는 이미지 포맷으로 PNG 포맷보다 크기가 30% 정도 작아서 앱 크기를 줄일 수 있습니다. 다만, 애플리케이션 작업할 때 Minimum SDK를 지원하고 싶다면 Android API 레벨 18 이상을 사용해야 배경 투명도를 사용할 수 있습니다.

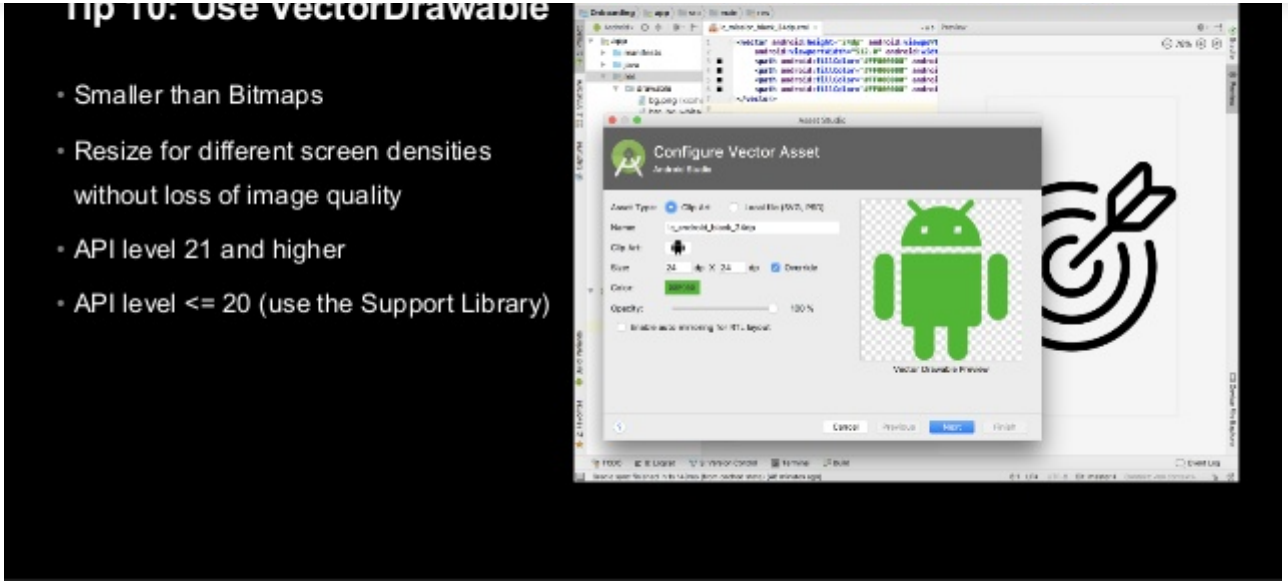


예를 보여드리겠습니다. LINE MAN 프로젝트에서 다운로드한 PNG 파일을 WebP로 변환해 보았습니다. Android Studio에서는 아주 간단합니다(참고). 원래 이미지의 사이즈는 2.1MB였는데요. 90% 정도의 퀄리티로 WebP로 변환하니 대략 300KB가 되어 원래 크기의 15% 정도로 작아졌습니다.



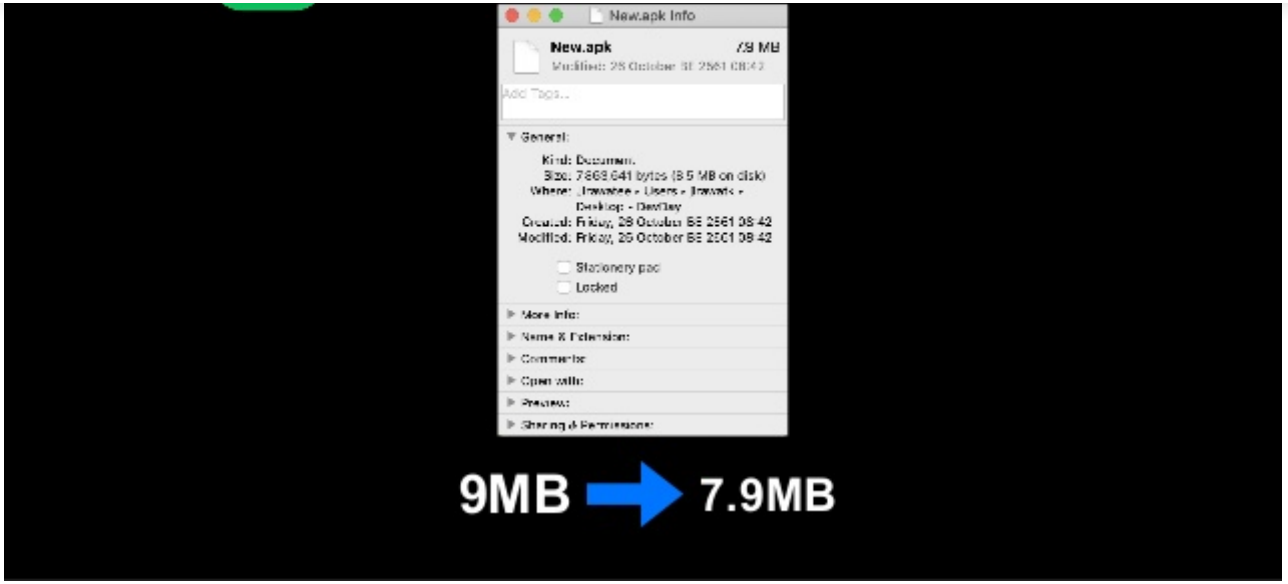
10. VectorDrawable 사용

벡터 이미지의 크기는 비트맵 이미지보다 작습니다. Android 개발에서 단편화를 지원하려면 밀도가 낮은 화면, 중간 정도인 화면, 매우 높은 화면 등 다양한 화면 밀도를 커버해야 하는데요. VectorDrawable은 한 번만 클릭하면 화질은 그대로 유지한 채 화면 밀도에 맞춰 크기를 자동으로 변환해 줍니다. Android API 레벨이 11 이상이면 지원되니 꼭 사용해 보세요. 아주 유용합니다. Minimum SDK의 버전이 11 미만이라도 지원 라이브러리의 버전이 23.2 이상이면 사용할 수 있습니다.



적용 결과

모든 방법을 적용하니 원래 9MB였던 LINE MAN 드라이버 앱의 크기가 7.9MB까지 줄어들었습니다! 어쩌면 이보다 더 줄어들 수도 있을 겁니다. 이런 팁을 활용하면 앱 크기 뿐 아니라 작업 시간도 상당히 절약할 수 있습니다.



한 가지 더 공유하고 싶은데요. Android 버전 3.2부터 Android App Bundle 기능이 새로 지원됩니다. Android App Bundle은 앱 크기를 쉽게 줄일 수 있는 새로운 앱 공식 게시 형식인데요. 자세한 것은 여기를 참고하시기 바랍니다.

마치며

여기까지 앱 크기를 줄일 수 있는 방법에 대해 알아보았는데요. 아직 끝난 게 아닙니다. 다음 편에선 안드로이드 앱 빌드 속도를 빠르게 하는 10가지 기술을 공유할 계획이니 많은 기대 바랍니다.

---

1. DEVELOPER BUILD CLINIC은 빌드 성능을 개선하는 방법을 Android Studio팀이 1대1 컨설팅 형식으로 답변해 주는 클리닉입니다. 저희는 2016년부터 이 프로그램에 참가하여 다양한 팁을 얻고 조언을 받고 있습니다.