问题：项目方法超 65535 编译报错。

产生原因：

我们都知道 Android apk 构建中会把所有 class 文件都转化为 Dex 文件，而单个 Dex 中方法数不能超过 2 的 16 次方，即 65535。既然是编译构建 dex 过程中发生的错误，肯定是由转化 dex 的工具 dx 抛出来的异常。

dx 工具描述为：把一系列 class 文件 或者 jar 包 转化为 dex 文件的工具。

顺便介绍一下 dx 的比较常见 options 参数，后面还是说到，这里简单介绍下：

- - multi-dex ：能够生成多个 dex files

- -main-dex-list= : 参数 file 是列出 main dex （即 classes.dex） 中所包含的类信息。通俗点就是你想要主 Dex 中有哪些类不被分出去就写在 file 里，把 file 传进参数命令里； 通常用来解决分包造成的 classNotDefException 问题。

- - minimal-main-dex : 只有- -main-dex-list= 指定的 file 中的类被保留在主Dex，用来减小 main dex 压力。

dx 脚本只是对 dx.jar 的使用做了封装，实际上 dx 工具只做了两件事：一是找到同级目录下 libs 下 dx.jar 的 path，二是读取你设置的 Java 虚拟机内存配置参数（可选）。

转换 dex 的时候可以通过--set-max-idx-number= 来改 dex 默认的引用 id 上限（默认 65535），当然只能往小了改。

multidex 分包转化 class 为 dex 的逻辑：runMultiDex —> processAllFiles —> 判断 inputFile 类型，如果是压缩类型，就解压后处理每一个 class 字节 —>processFileBytes—> 过滤掉resource 等字节，留下 class byte —> processClass —> 判断是否超过 DexFile id limit，超过就 new 新的 DexFile 继续处理字节 —> 最后把转化好格式的字节 写入多个 DexFile 中。

如果不启用 multidex 功能，项目很容易产生 65535 ，但是开启了 multidex 有时候仍然会造成 65535 （这个以后分析），但是开启不开启 multidex 功能，程序丢 65535 异常的原因肯定是一样的，就是向 DexFile 写入字节的时候，方法数超过了 DexFile 的默认最大上限 65535。

跟踪 DexWriter 方法最终到 DexFile 的 toDex0 方法，里面会对 Dex 文件格式每一个部分进行检查；比如：this.methodIds.prepare(); 猜测对 method count 判断在这里进行。跟踪进去后发现，MethodIdsSection 和 FiledIdsSection 的父类都有一个 getTooManyMembersMessage 方法：错误信息提示：如果 Dex id 引用太多，你可以开启 multidex 模式，如果已经开启，main dex 中的引用太多了。

解决方法：Multidex

Multidex 加载原理 ：

判断 Davlik 版本，是否支持 内置 multidex 功能；和 SDK 版本是否支持 support-multidex；

实例化 ClassLoader ，确保 secondary dexes 目录存在并清理；

MultidexExtractor 解压 apk 提取 classesN.dex 的字节到 /data/user/0//code\_cache/secondary-dexes/base.apk/classesN.dex.zip 中；每个 zip 中都只有一个 名为 classes.dex 的 entry；

把当前 apk crc 校验值和 sharepreference 中之前保存的做比较，

如果相等，就直接用上次的提取过的 files；

如果不等，就重新执行提取操作，并保存本次 lastModified time 和 crc；

安装 secondary dexes ：通过发射获取 BaseClassLoader 中 DexpathList 中的 Element[] 用包含 secondary dexes 的 新 Element[] 替换旧的。

Multidex分包Class依赖处理：

在 processAllFiles 处理所有 class 的时候会对 - -main-dex-list=<file> 设置的对应参数 this.args.mainDexListFile 进行判断。

它的作用就是在 dx 将 class 拆分成多个 dex 文件的时候，保证 classes.dex 中必要的类 不 被分出去，避免应用在启动时会造成 classNotFoundException 。

当我们使用 mulitdex 处理 apk 时候，在工程 app —> build —> intermidiates —> muliti-dex —> 下会有这么两个文件：

maindexlist.txt : 以 class 全路径形式列出了 keep 在 main dex 中的所有 class；

manifest\_keep.txt : 读取 appManifest 中的 application class，keep 住 application class 实例化过程中的调用类。

理下生成 maindexlist 流程：

mainDexClasses 脚本进行混淆 class file，调用 dx.jar 中 MainDexListBuilder Main 方法，将 mainDexClasses.rules 和 mainDexClassesNoAapt.rules 的 keep 规则作为参数传入；

main 方法处理参数，实例 ClassReferenceListBuilder 调用 addRoots 添加依赖；

判断 class 是否带有 运行时可见 注解

mainDexListBuilder.getMainDexList() 返回 main dex 中需要的 class list；