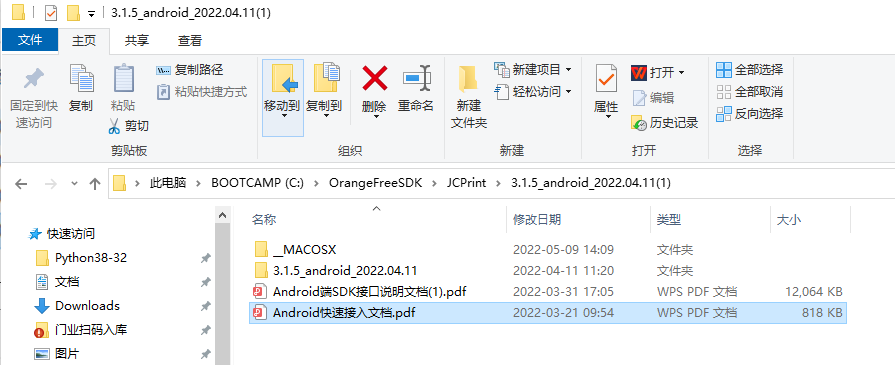
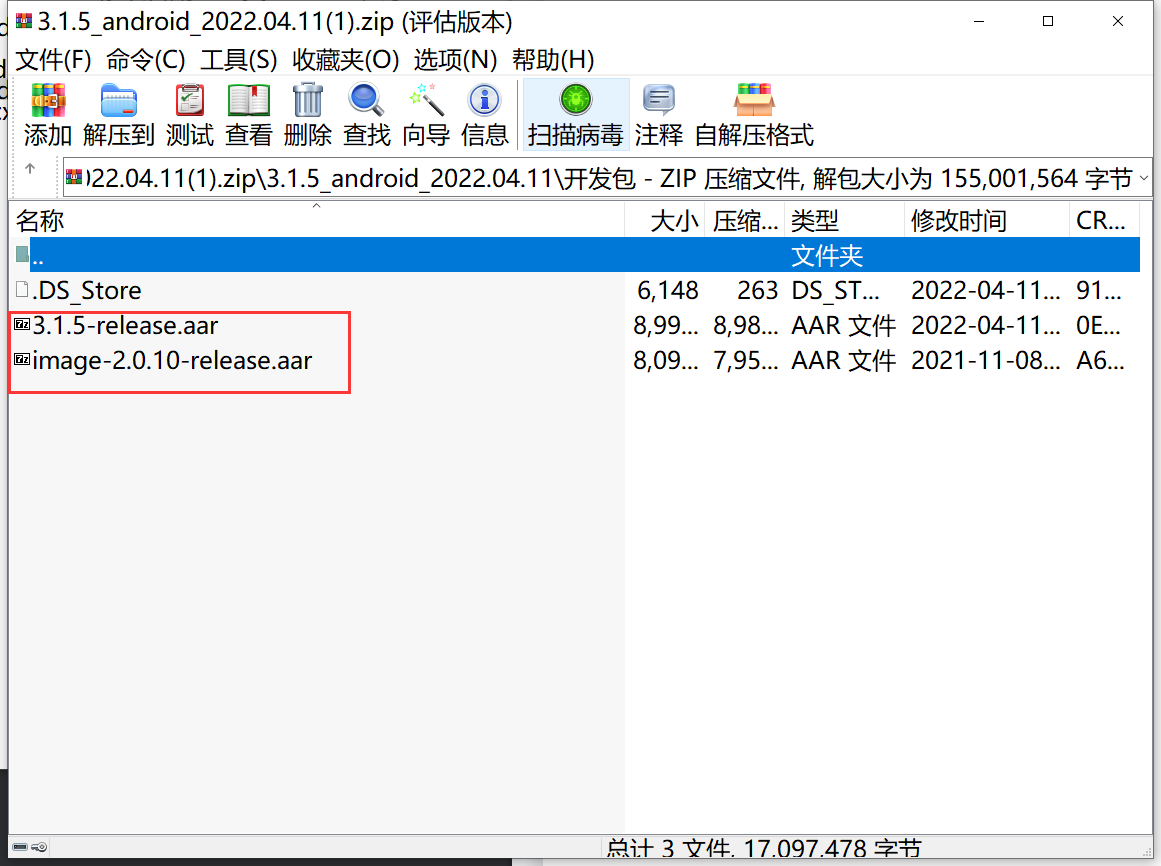
有必要升级到Delphi 11吗？我用10.4.2对接三方Android SDK时的翻车现场！

最近在帮群里朋友对接一个标签打印机的SDK，我自以为是这方面的老司机了，没想到还是翻车了！！！

**拿到了SDK压缩包，我先看了看里面有哪些文件，**

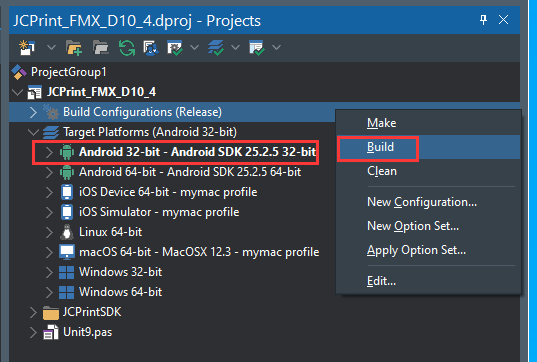




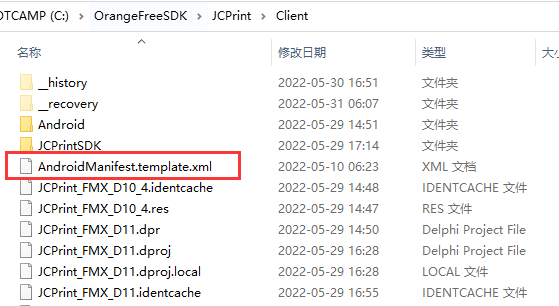
可以看到有两个aar，其中的3.1.5-release就是我们要对接的主要SDK，另一个image-2.0.10.aar应该只是一个辅助。

我一开始是用Delphi 10.4.2进行对接的，10.4.2是开发Android APP非常稳定的版本。

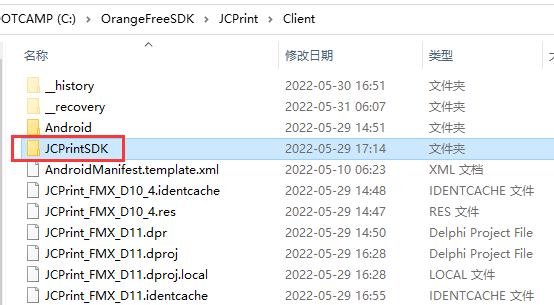
新建一个工程，专门用于测试这个SDK，将工程的当前目标平台设置为Android平台，并且编译一次，



主要是用于生成AndroidManifest.template.xml，因为在这个文件里面可以配置安卓的权限。



在工程目录，新建一个存放该SDK相关文件的目录，这次我对接的是精臣打印机，就取名为JCPrintSDK吧，把SDK相关的文件放进去，这样SDK对接测试好之后，移植到正式的项目里的时候只需要将这个文件夹复制过去即可。



**这里要用到我的三方SDK智能布署工具，**

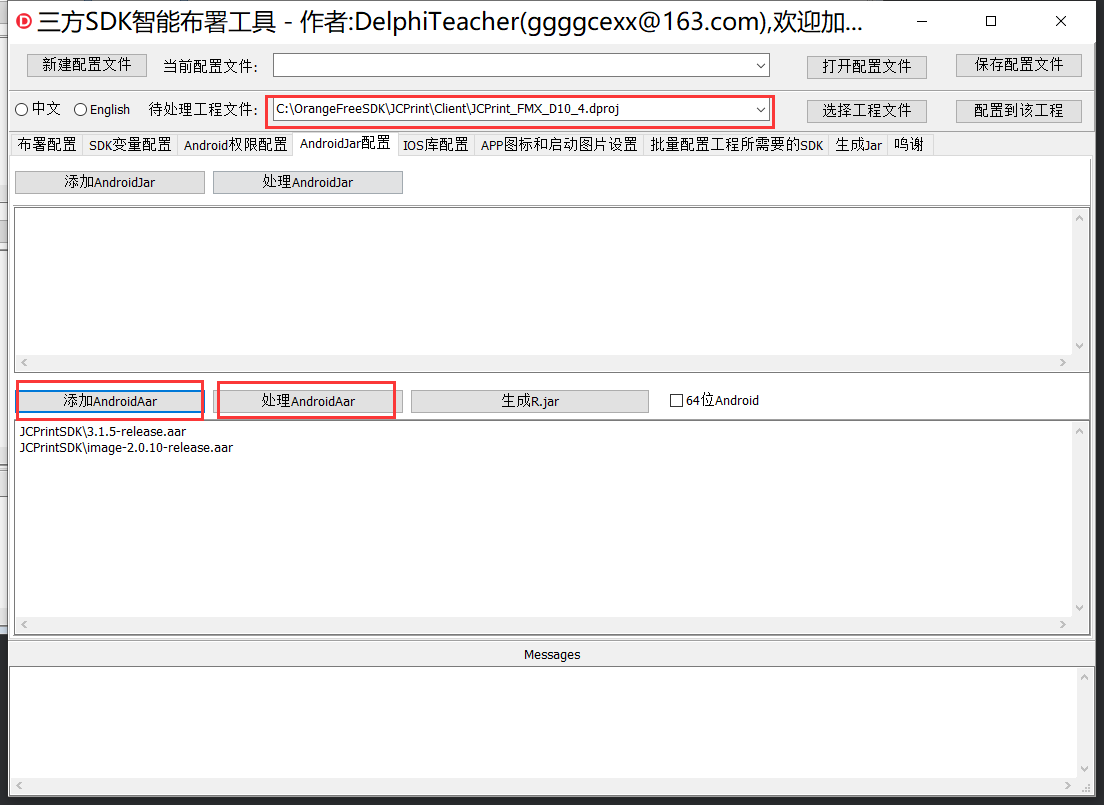
它可以批量布署文件、对接jar和aar、配置权限、批量设置工程各种尺寸的图标等，并可以将这些操作保存成配置文件，别的项目中可以直接加载这个配置。

并且这个工具是免费而且开源的：

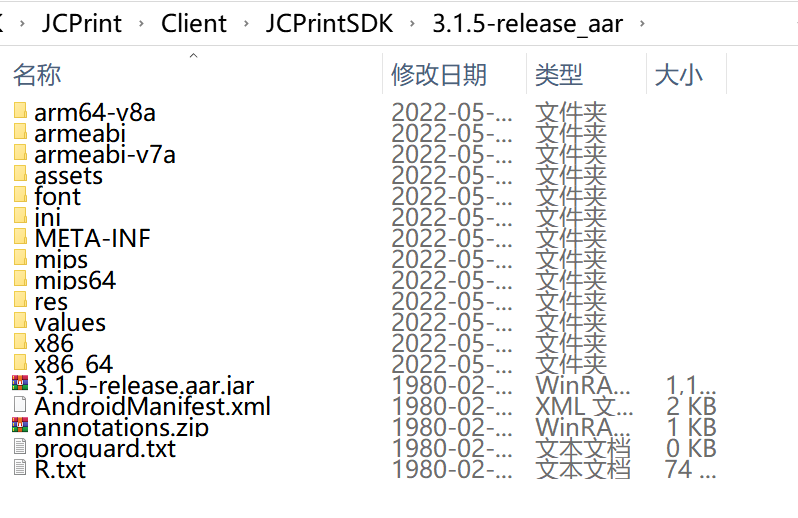
<https://github.com/DelphiTeacher/OrangeFreeSDK>

（github现在需要科学上网工具才能访问了）

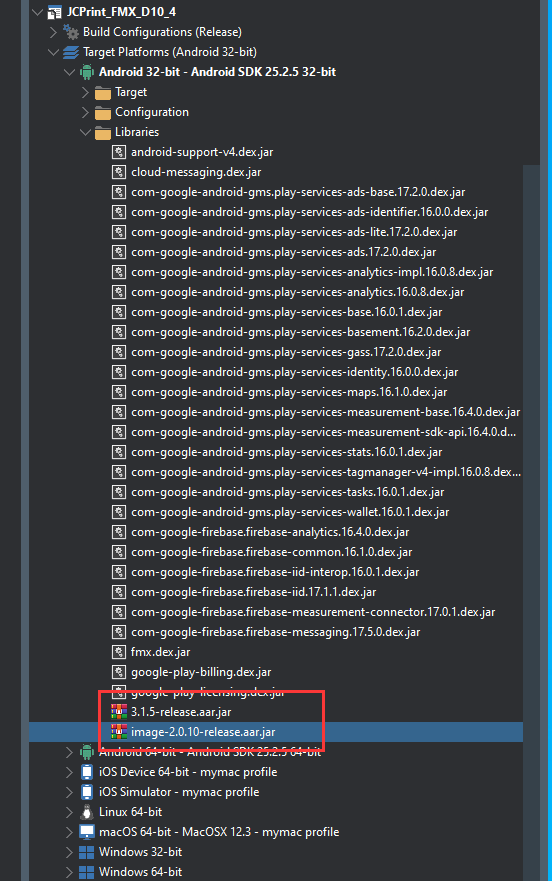
在三方SDK智能布署工具中选择上面新建的工程，添加aar，再点击处理AndroidAar，



这个工具会自动帮你解压aar，

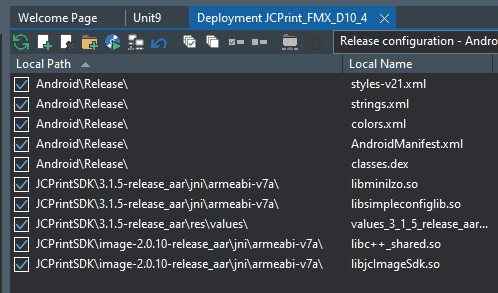


把里面的jar添加到你的工程中去，



工具也会自动帮你布署资源和SO，打开布署，你可以看到：

这里我们需要把其他CPU架构的so布署去掉，只留armeabi-v7a的

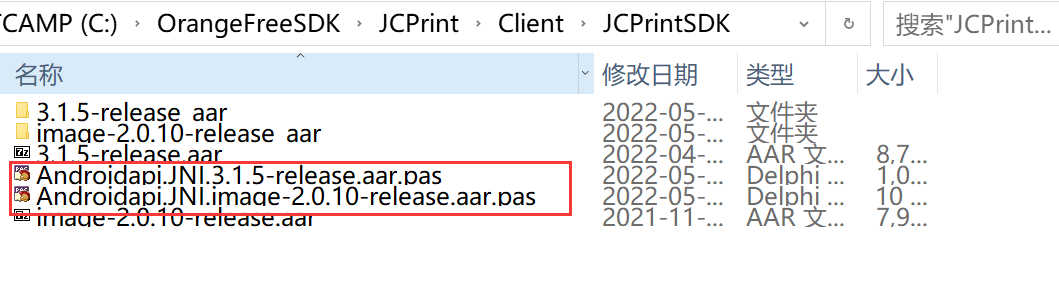


**接下来，用JavaClassToDelphiUnit工具将jar转为pas接口文档**

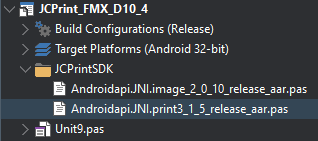
这个工具是老猫开发的，离开了它，对接不了任何Android SDK，在这里感谢老猫的辛苦付出，大家可以到老猫的QQ群②FireMonkey[移动开发]-群号165232328下载。

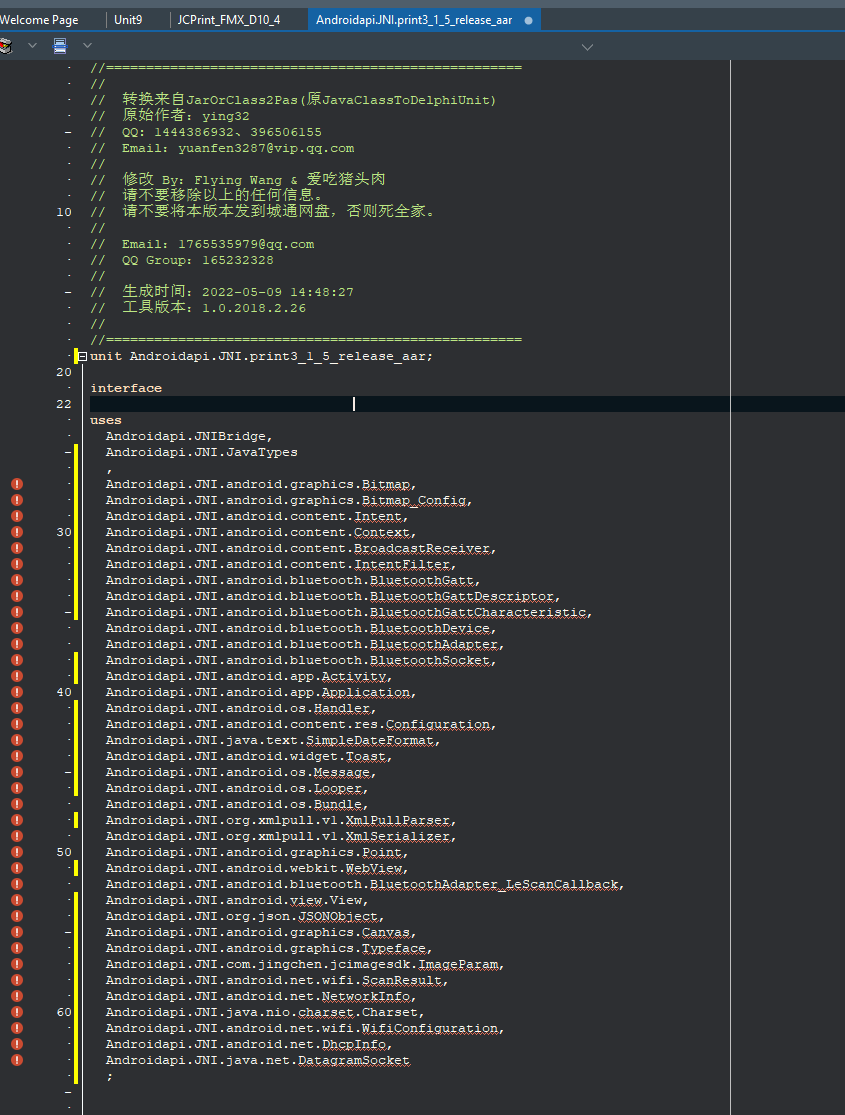
我们选择jar，再选择要将生成的pas保存到哪个目录，勾选”AutoCommentBadInterface”可以注释掉绝大部分不需要的接口，再点击开始转换：



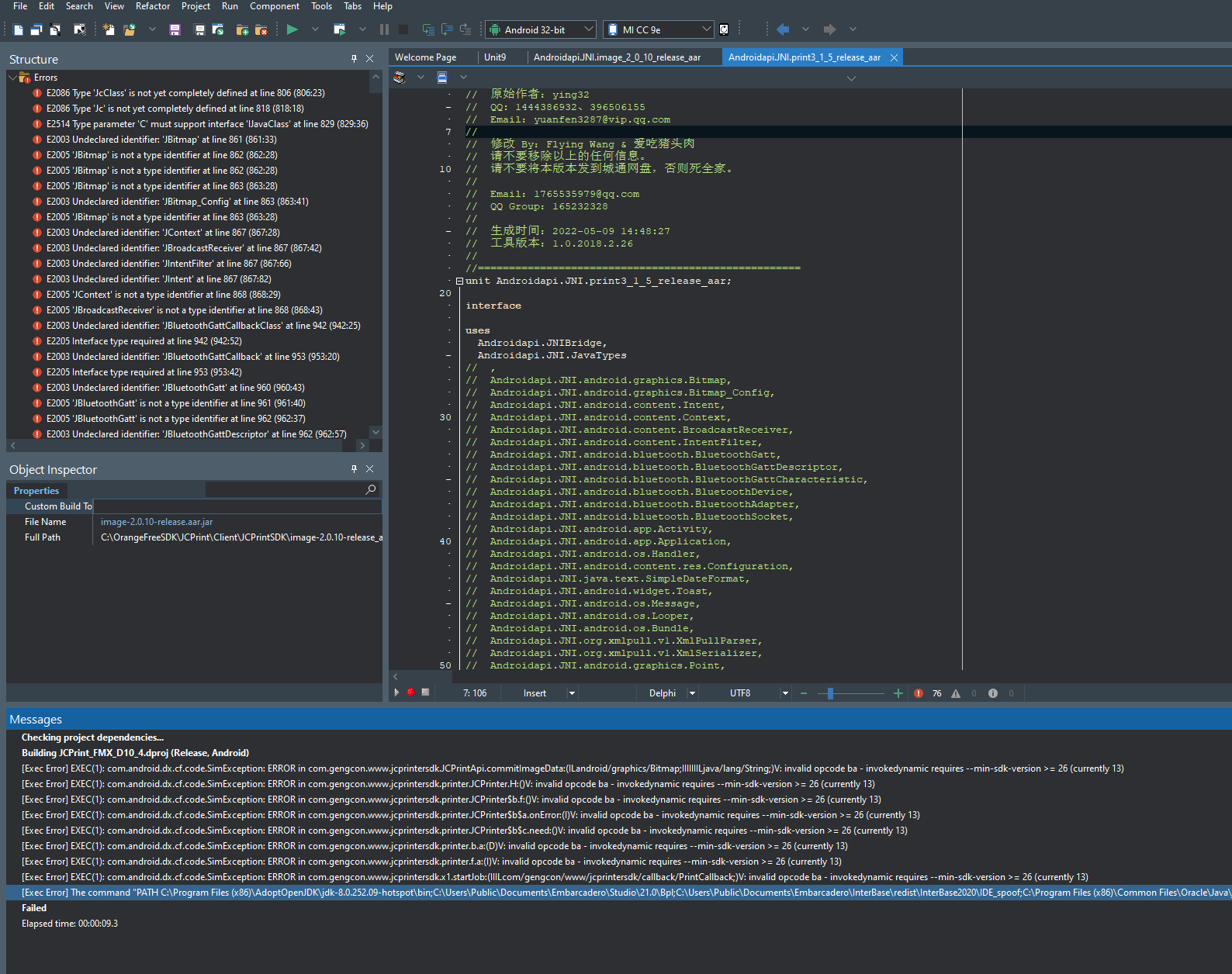


修复下文件名，把后面的-.可以替换成下划线\_，再添加到工程中去

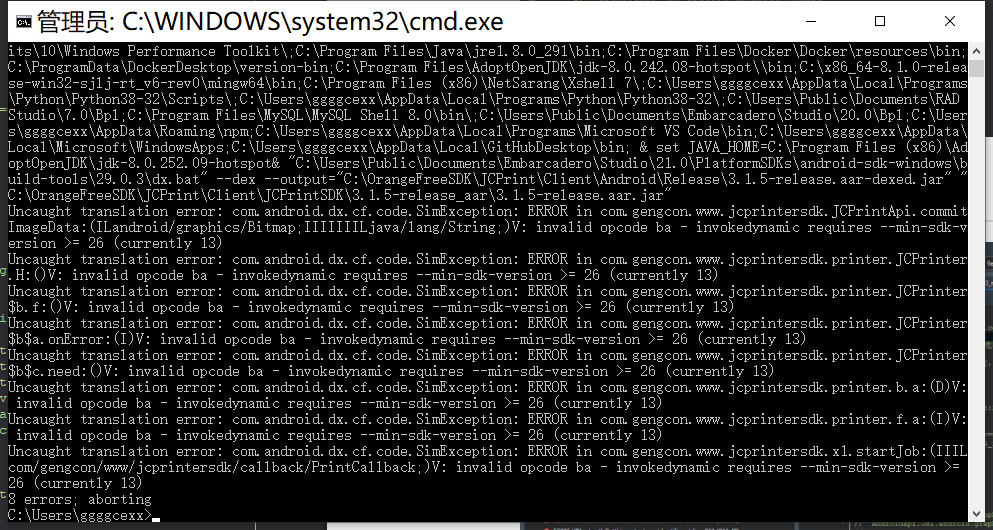


再需要手动将编译不过的引用单元和声明都先注释掉，确保要能先编译通过。

不过，又出问题了，还没到代码编译的时候报错了，



出现这种问题，Delphi会将出错的命令显示出来，我们只需要将这个出错的命令拷出来，单独在命令行里运行，或者保存成bat执行，这个命令是将3.1.5.jar转换为dexed.jar。

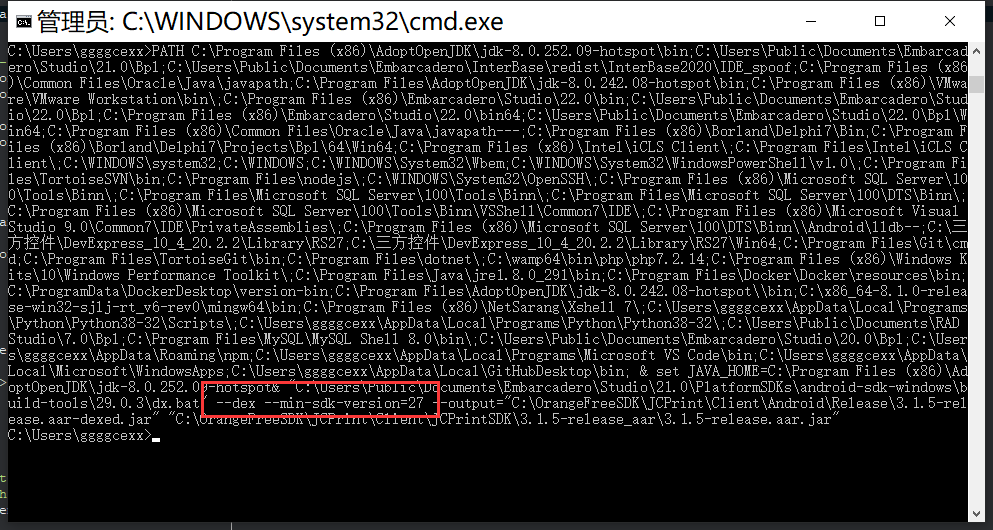


提示说我们指定的最低的Android SDK版本太低了，那就是厂家在生成这个jar的时候指定的Java版本有点高。那么我们该怎么解决呢？

在Android Studio中的话，只需要配置java编译器的版本就行了，



但我在Delphi中没有发现配置它的地方，只能先按照提示，给这个命令行加入--min-sdk-version=26这个参数运行一下，



这下没有提示错误，在去生成目录看，对应的dexed.jar已经生成了，这个文件只需要生成一次就行了。



再将编译不过的引用单元和声明都先注释掉，这一步工作量有点大的，非常考验大家的耐心。

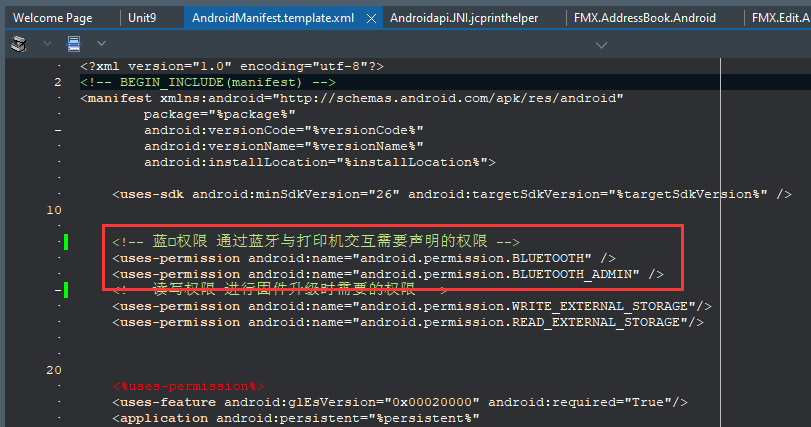
**接下来，按着厂家的SDK对接文档，**



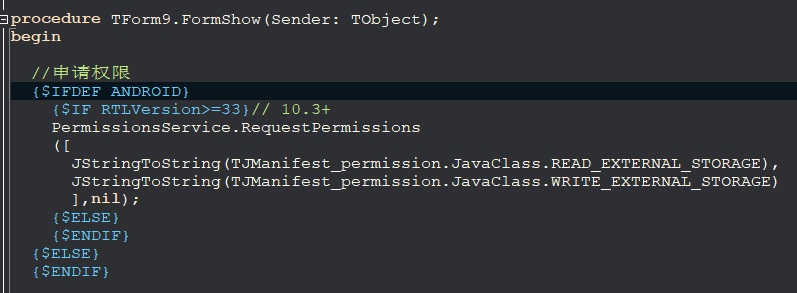
文档里第一点需要配置权限：



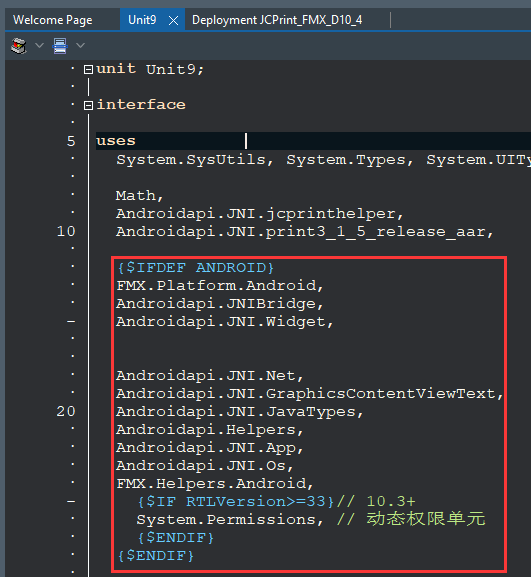
那我们也添加权限，蓝牙权限不需要动态申请，打开工程目录下面的AndroidManifest.template.xml直接添加：



但是读写外部存储的权限现在需要动态申请了，



需要引用 下面这些常用的单元：



大家是否注意到这些代码我都用编译指令**{$IFDEF ANDROID}{$ENDIF}**包起来了，这个非常有用，这样写的话，包起来的代码只会在Android平台才会编译进去，在其他平台下就不会被编译进去，当你的项目用到了很多安卓相关的代码，在windows上编译不过，那么用它，你就能在windows上编译和测试项目的其他功能了。

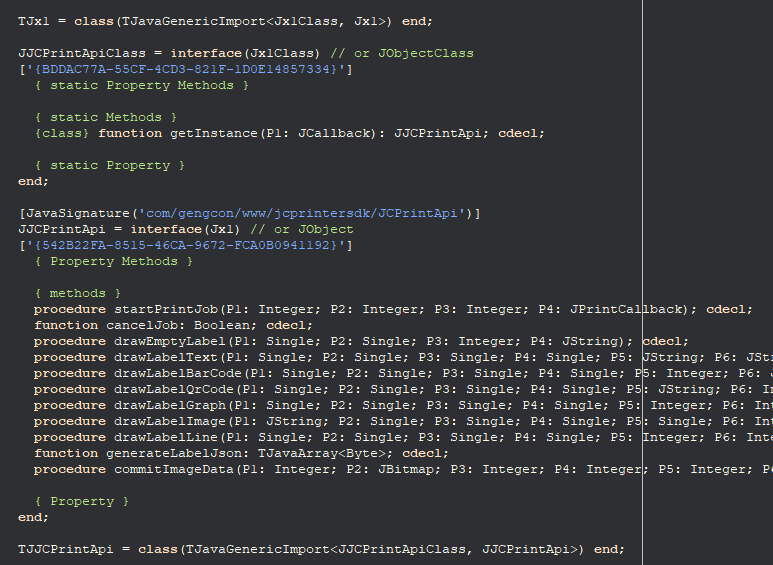
对着文档初始化SDK

先看文档中如何用java代码进行初始化的：



java代码首先声明了一个JCPrintApi的变量jcapi，那么相应的，我们在Delphi下也需要一个类型为JCPrintApi的jcapi的对象。

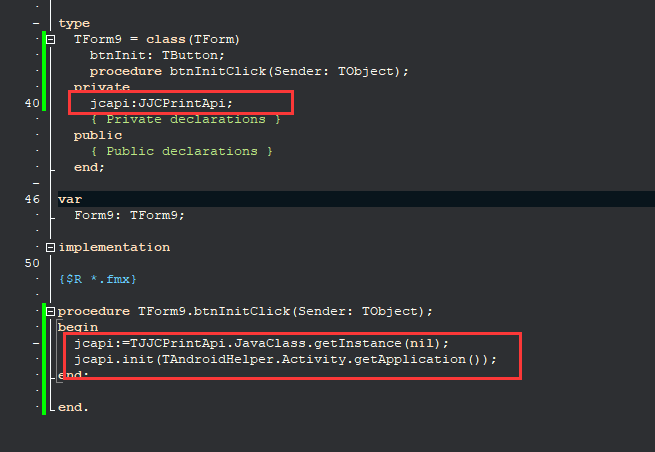
工具已经根据JCPrintApi生成了一个对应的Delphi接口JJCPrintApi，我们需要将jcapi声明成为JJCPrintApi，



翻译成Delphi的话，声明如下：

jcapi:JJCPrintApi;

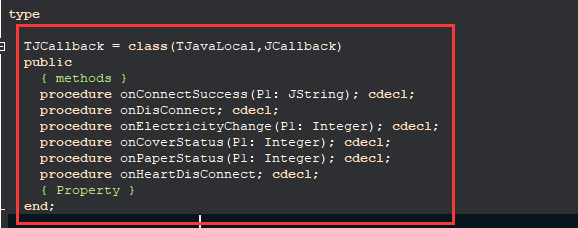
Delphi通过JNI的方式，即用生成的接口来调用对应java类的方法，在这里:Java类JCPrintApi在Delphi下被转换为JJCPrintApi接口，Java类JCPrintApi的类类型在Delphi下被转换为JJCPrintApiClass接口，在java代码中它是调用JCPrintApi的类方法getInstance来创建这个jcpai的对象的，类方法不用创建对象就可以直接调用，那么我们在Delphi中如何创建呢？TJJCPrintApi.JavaClass.类方法即可。对象创建了之后 ，调用初始方法init，



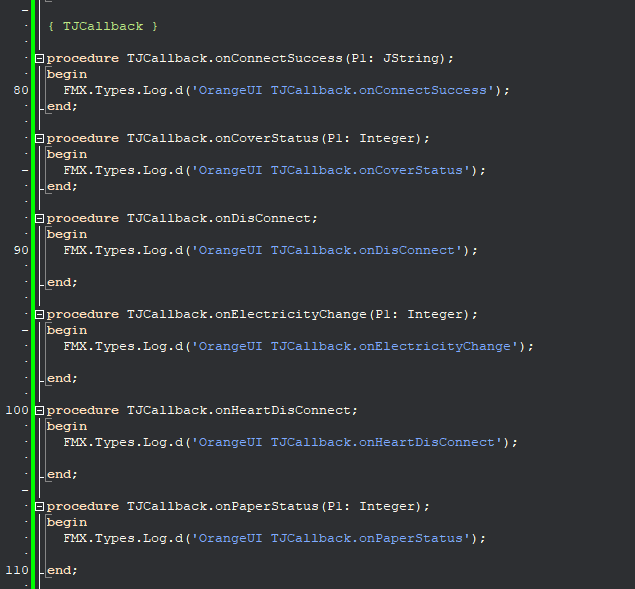
设置回调



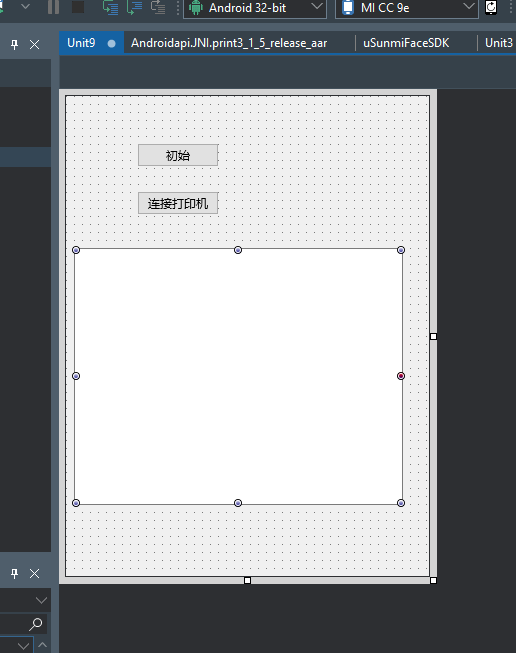
翻译成Delphi来实现这个回调接口，



再实现这个接口的方法，我们在每个方法中写上日志：



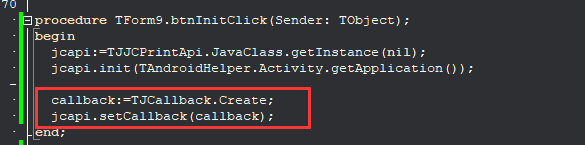
我们为了直观能看到效果，在窗体上放一个Memo，将日志输入到Memo中。



我们也声明一个callback变量，

callback:JCallback;

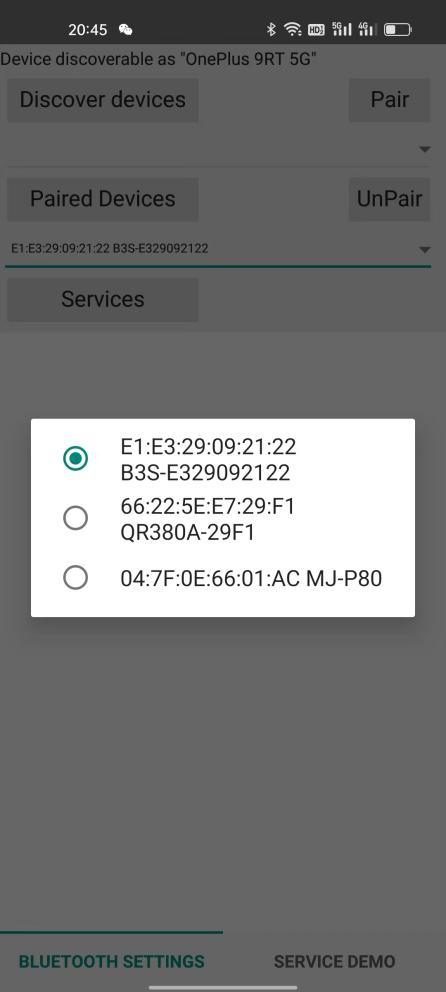
直接使用TJCallback.Create来创建对象，



**连接打印机：**

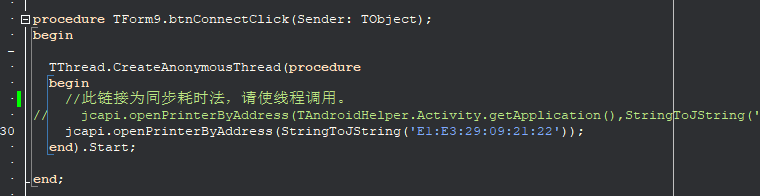


图中这个地址为蓝牙打印机的设备地址，我们通过Delphi自带的蓝牙示例来获取到

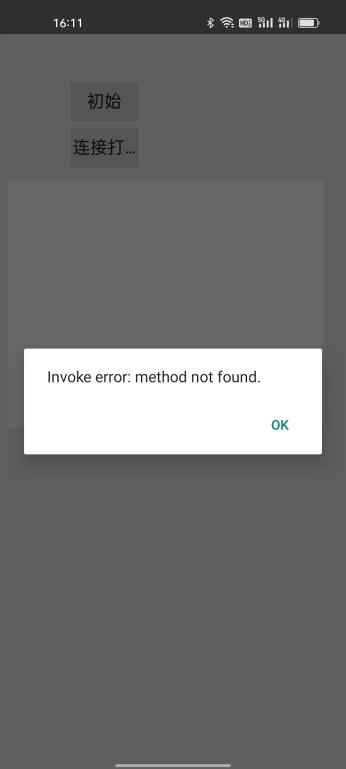


图中第一个设备就是这次我们要对接的，

文档中让我们用子线程调用，那么我们也满足它，把它放在一个匿名线程中。

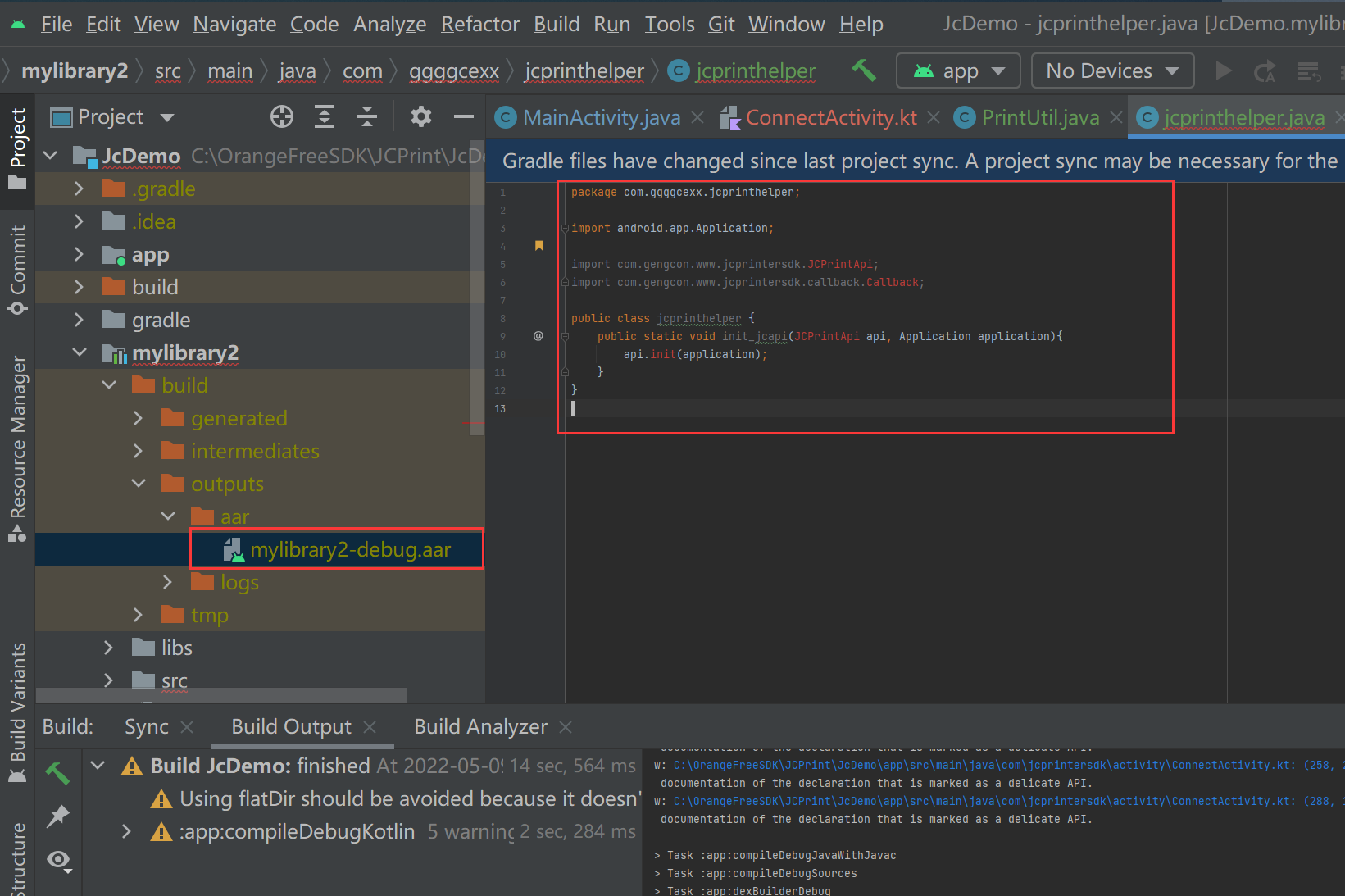


好了，先测试一下能否成功连接打印机



经过测试发现，在执行jcapi.init(TAndroidHelper.Activity.getApplication())这句代码的时候报init方法，可是这个方法明明存在。这是Delphi调用jar的一个不足的地方，它已经将init固定为一个构造方法，如果有些init在java类中用于初始，则delphi的init构造方法会覆盖它，以至于找不到这个方法。那么如何解决呢？那就是自己封装一个jar来调用init方法。

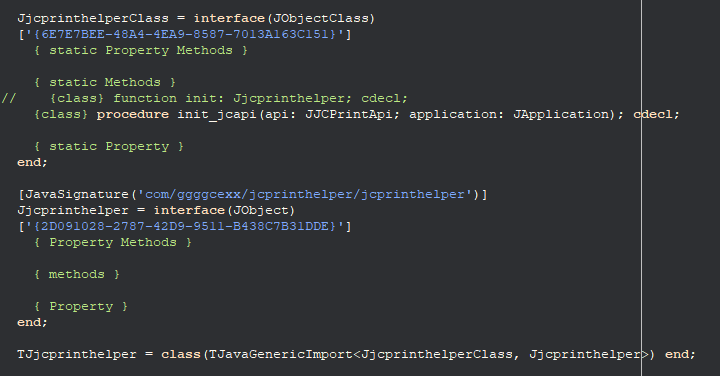
我们用AndroidStudio建个库，用它来调用init。具体细节就不讲了，大致步骤如下：用AndroidStudio来打开SDK自带的示例Demo，新建一个Android Library模块，它用来生成aar，在模块中新建一个jcprinthelper类，在类中添加一个静态的类方法，传入jcapi，调用jcapi.init方法即可。然后编译这个库，将生成的aar拷出来，aar里面的jar解压出来。



将aar解压出来的jar命名为jcprinthelper.jar，添加到工程中去，并生成Pas接口文件，



可以看到，生成的delphi接口如下：



修改jcapi.init(TAndroidHelper.Activity.getApplication())这句代码为：

TJjcprinthelper.JavaClass.init\_jcapi(jcapi,TAndroidHelper.Activity.getApplication());

再次测试连接



一番折腾之后，心想，这下没有别的拦路虎了吧，

**那就对接打印方法试下能不能打印**

这一步就是对着厂家给的android示例和文档将用java写的代码转换成我们delphi的，我贴几段给大家看看：

java的变量声明：



我转成Delphi的：



java的方法调用：



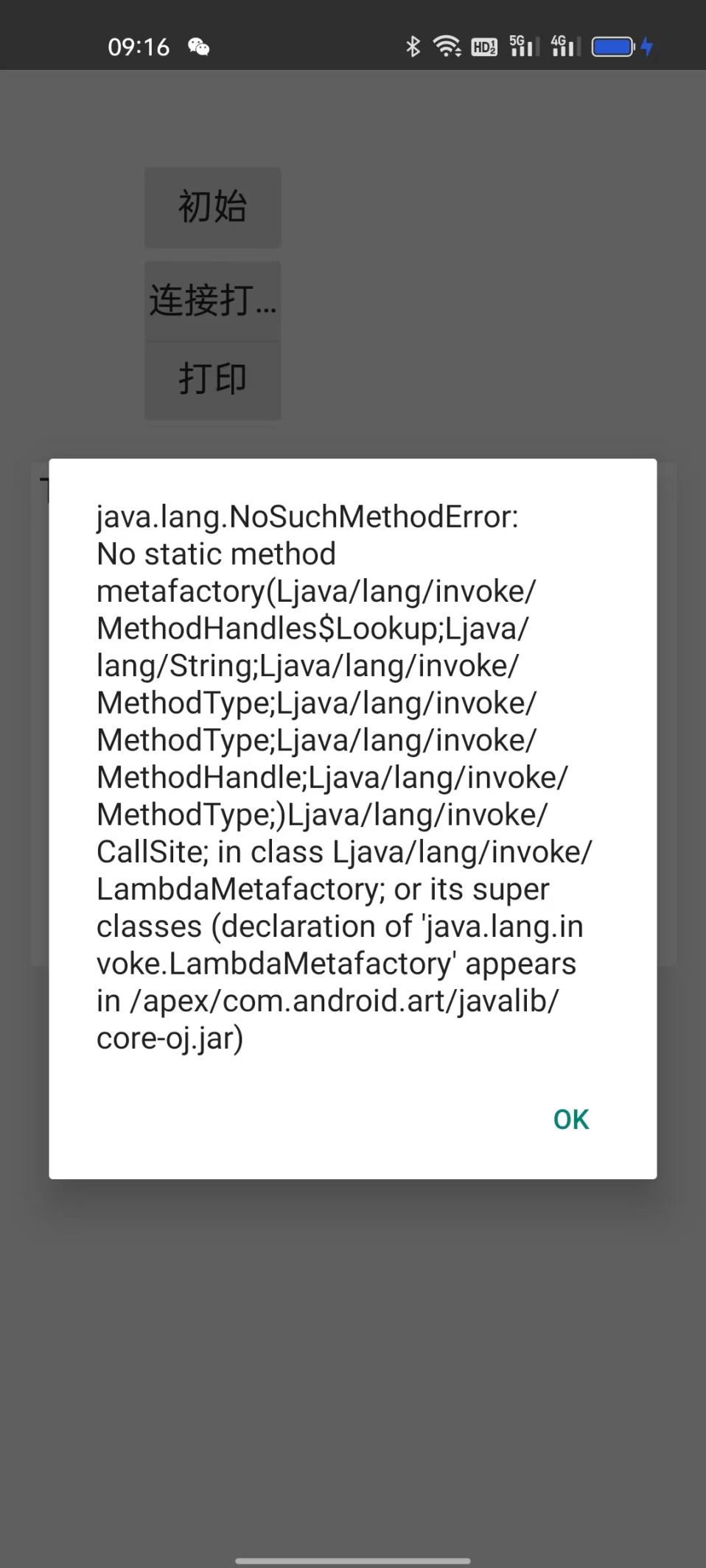
我转成Delphi的：



是不是很简单。

**翻车现场**

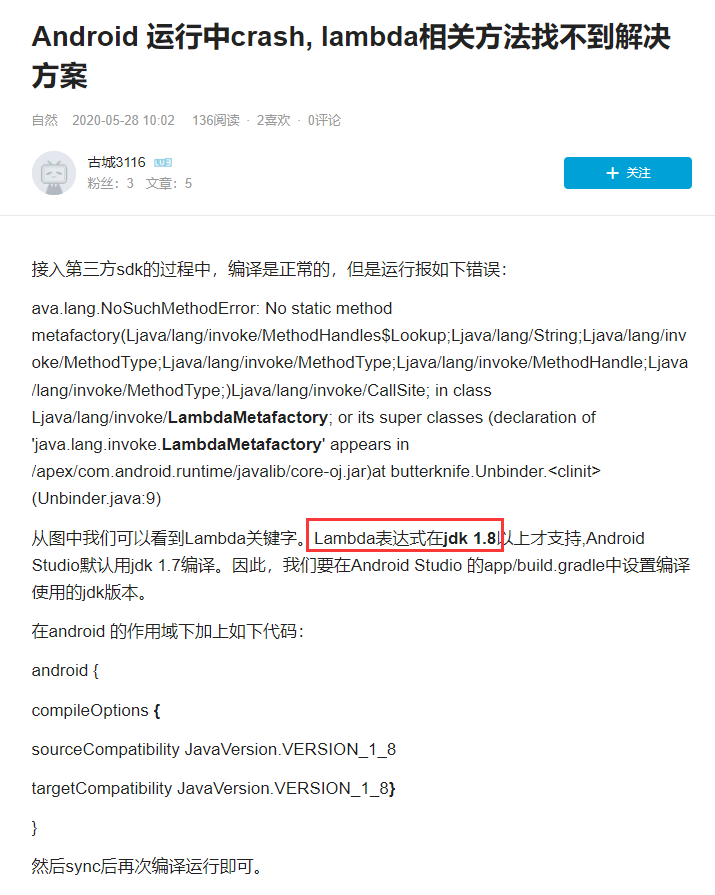
接下来测试一下：



java.lang.NoSuchMethodError:No static method

metafactory(Ljava/lang/invoke/MethodHandles$Lookup;Ljava/lang/String;Ljava/lang/invoke/MethodType;Ljava/lang/invoke/MethodType;Ljava/lang/invoke/MethodHandle;Ljava/lang/invoke/MethodType;)Ljava/lang/invoke/CallSite; in class Ljava/lang/invoke/LambdaMetafactory; or its superclasses (declaration of 'java.lang.invoke.LambdaMetafactory' appearsin /apex/com.android.art/javalib/core-oj.jar)

网上一搜，都是说旧版本的java编译出来的不支持lambda表达式，

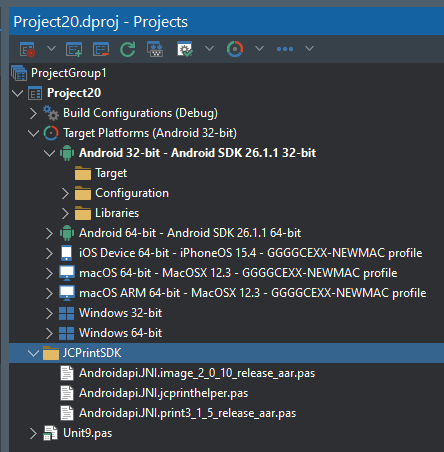


java编译的时候指定高版本就可以了，可是我用的是Delphi，到哪里去指定高版本。。。。。

至此，Delphi 10.4对接这个SDK失败，这相当于对我造成了一万点伤害，想放弃了。

**要不试下Delphi 11吧，它总对安卓方面的支持有所提升吧。**

打开最新的Delphi 11.1，新建了一个工程，重新配置一遍，



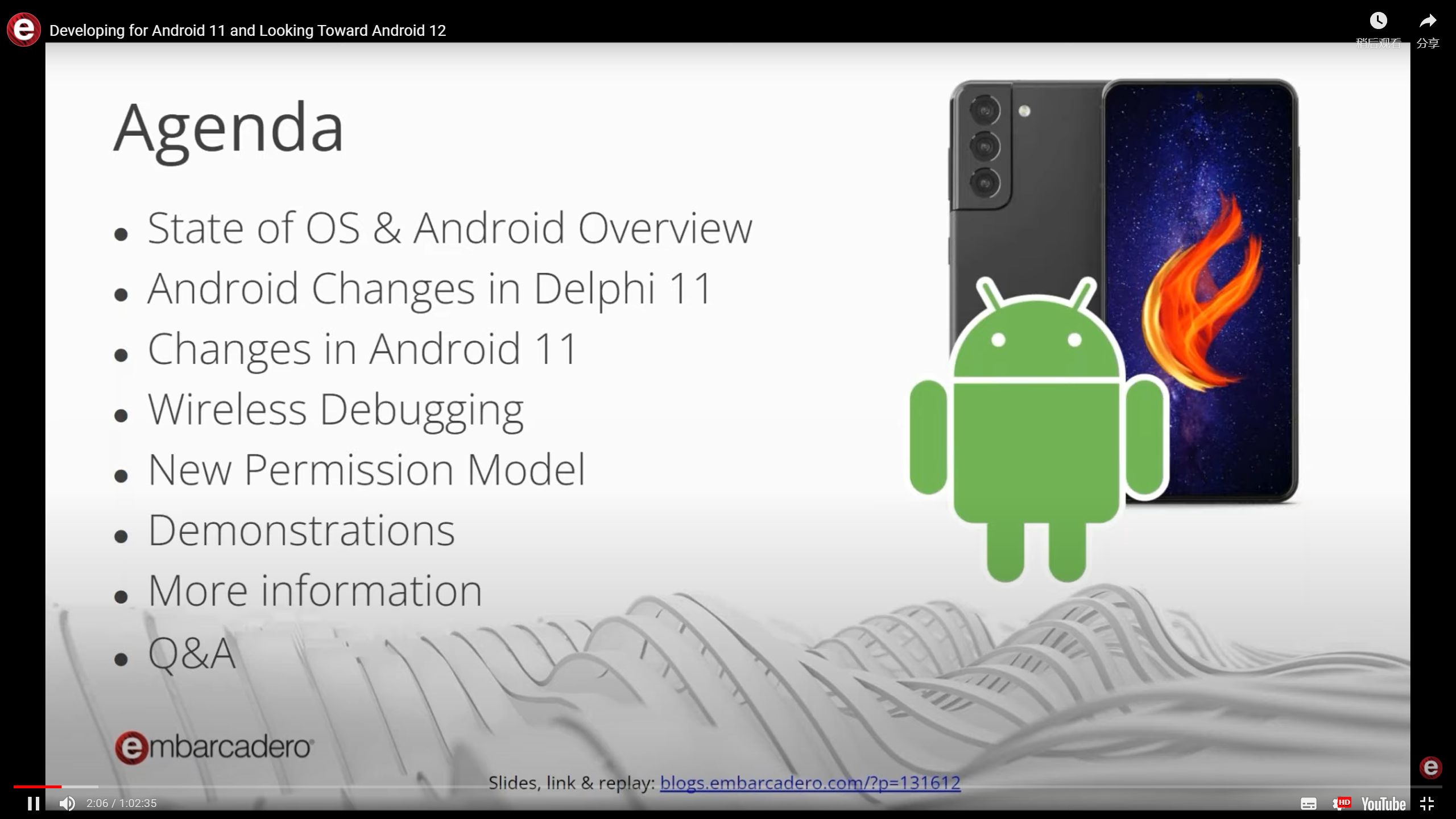
其他啥也没改，编译运行，竟然打印出来了，太神奇了。

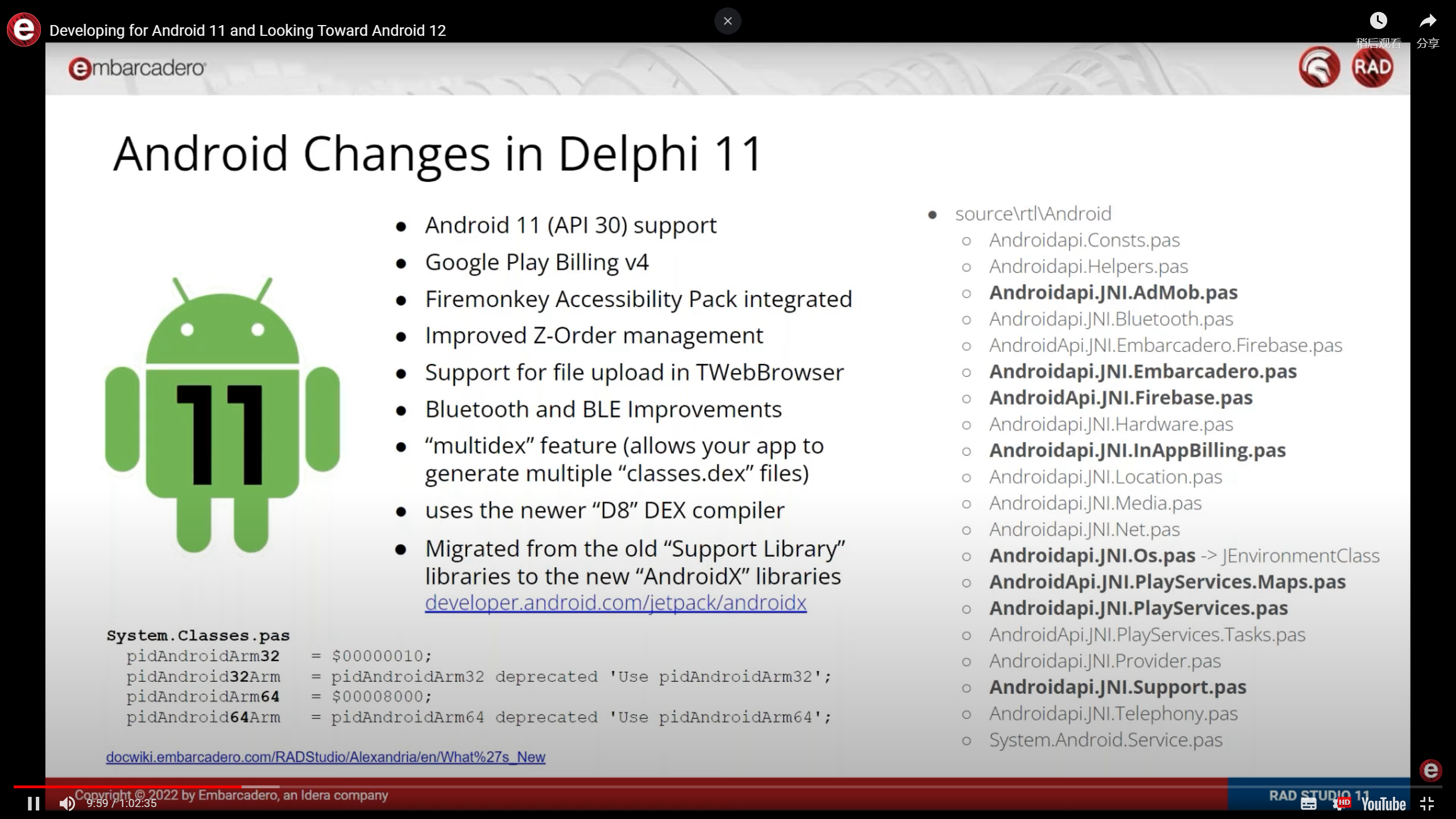


这就不禁令我产生了好奇之心，Delphi 11究竟对Android做了哪些方面的提升？

找了一下官方的资料：

<https://blogs.embarcadero.com/developing-for-android-11-12-with-delphi-11-alexandria/#Android_Changes_in_Delphi_11>





我注意到其中有一个”uses the newer D8 DEX compiler”，找到了这么一个链接有介绍 ：

<https://developer.android.com/studio/command-line/d8>



d8 是一种命令行工具，Android Studio 和 Android Gradle 插件使用该工具来将项目的 Java 字节码编译为在 Android 设备上运行的 DEX 字节码，该工具支持您在应用的代码中使用 Java 8 语言功能。

再看，又发现了D8支持lambda，



疑团解开了，就是因为Delphi 11中使用了这个D8编译器来编译Android，所以我们也不需要配置什么java的编译器版本了，直接就解决了我这个报错的问题。

所以，我一定要好好吸取这个教训，抽出时间来多看看Delphi升级了哪些方面！避免翻车。

最近发了几篇公众号文章，感谢大家的关注和几位朋友的打赏，大家的支持就是我的动力！