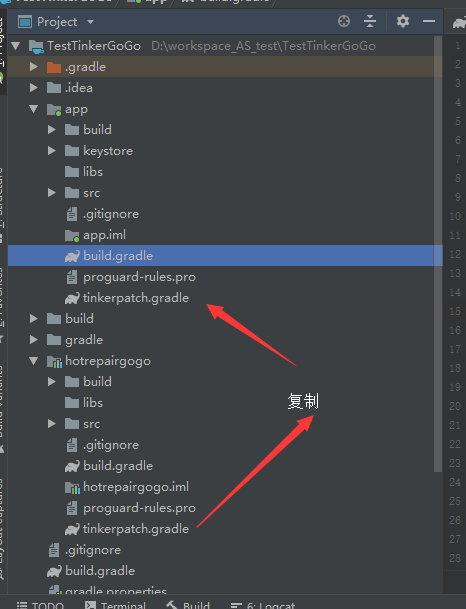
# 热修复库-hotrepairgogo

## *引入Module*

（略）

## 使用准备好的gradle文件

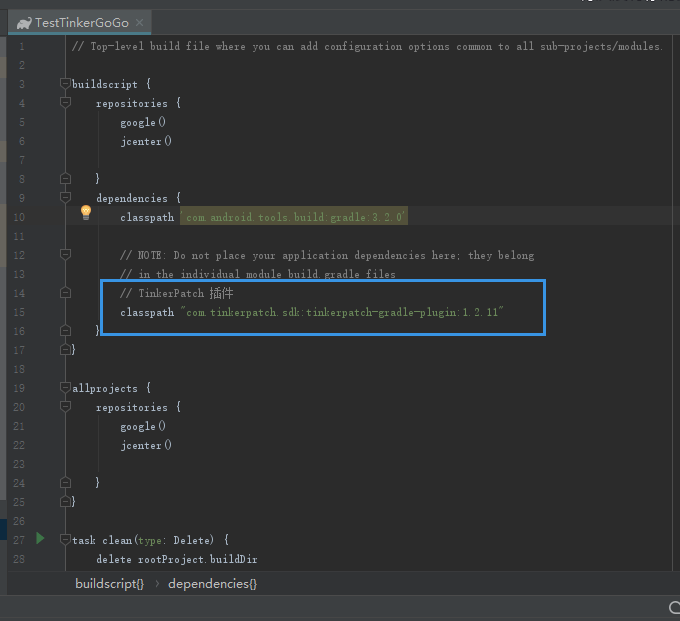
把库里的tinkerpatch.gradle 文件复制到 app文件夹下，即与app的build.gradle文件同级



## Android Studio 安装插件

工程build.gradle 加入插件地址

**classpath "com.tinkerpatch.sdk:tinkerpatch-gradle-plugin:1.2.11"**



app build.gradle 加入

引用第二步复制过来的 grade文件： **apply from: 'tinkerpatch.gradle'**

解除方法数限制：

**multiDexEnabled true**

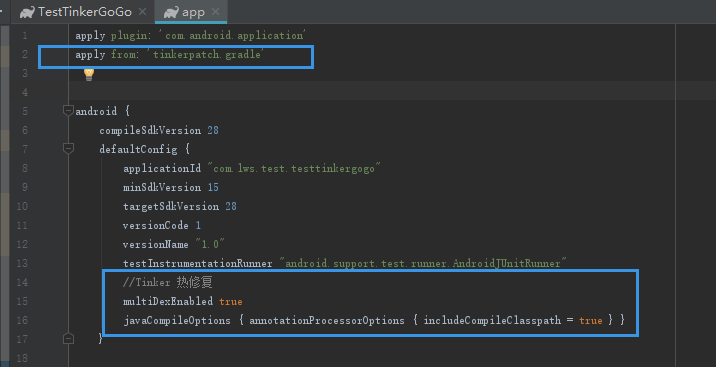
**javaCompileOptions {**

**annotationProcessorOptions {**

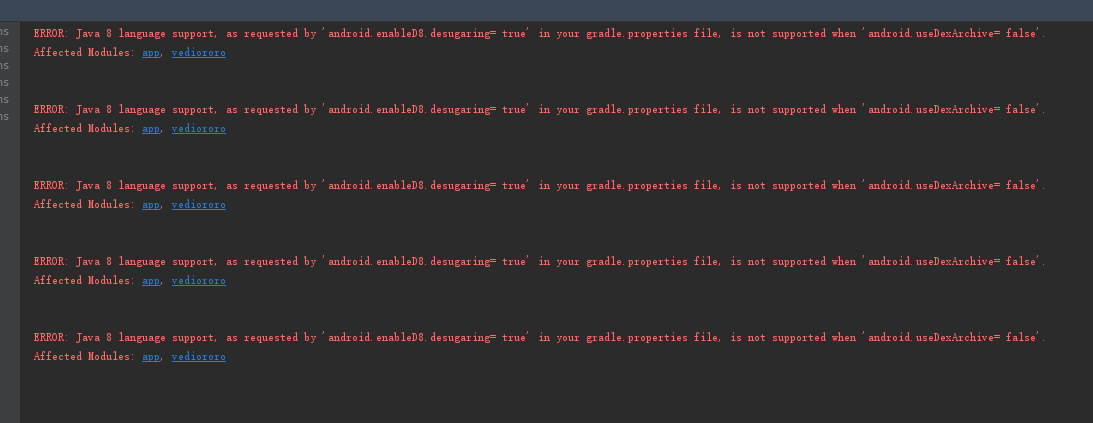
**includeCompileClasspath = true**

**}**

**}**



**如果引入同步出错：原因Tinker不支持jdk8，如果工程可以修改为jdk8以下最好，实在不方便修改可尝试以下修改方法**



可尝试：

方案1：gradle版本降低3.1.4

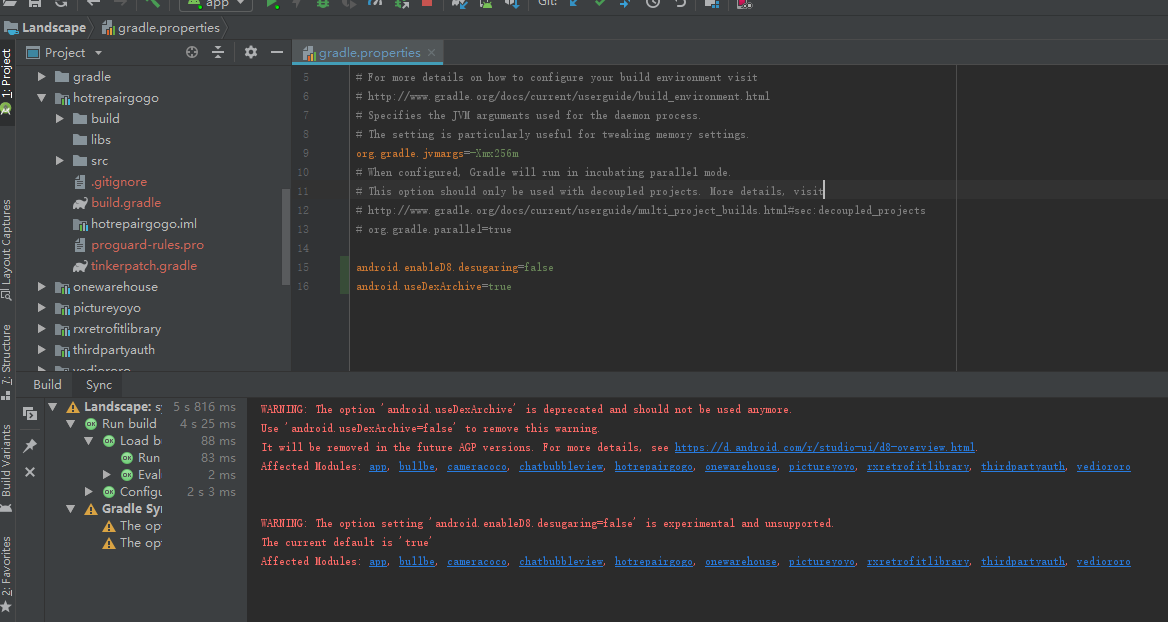
**classpath 'com.android.tools.build:gradle:3.1.4'**

方案2：gradle.properties文件添加

**android.enableD8.desugaring=false**

**android.useDexArchive=true**

同步后也许会出现以下警告，但可以忽略，网上说这是影响构建速度而已，以后tinker兼容java8才修改其他方案



## 注册平台

[Tinker Platform](http://tinkerpatch.com/) 官网，即<http://tinkerpatch.com/> ，上去把app登记到平台



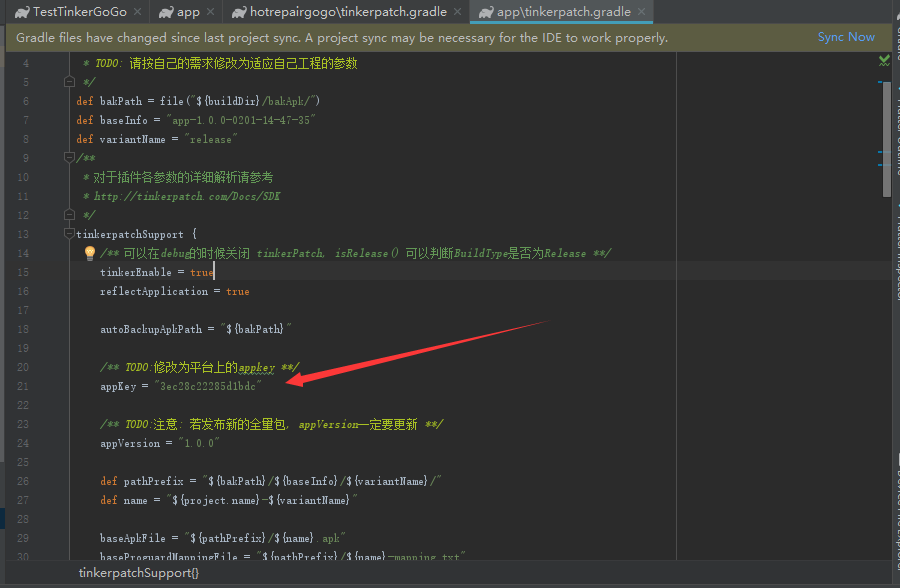
点击新增APP，新增一个app，名字可随时更改



新增后 可以看到appKey 复制下来



回到 tinkerpatch.gradle 文件，把appKey的值应用到图示位置，Sync Now



到此，配置完成，下面开始测试。

## 调试

### 权限

下载补丁 读写补丁需要的权限，动态获取的功能自行实现

<uses-permission android:name="android.permission.INTERNET"/>  
<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"/>  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

### 新建application

public class MyApplication extends Application {  
  
 @Override  
 protected void attachBaseContext(Context base) {  
 super.attachBaseContext(base);  
 MultiDex.*install*(base);  
 }  
  
 @Override  
 public void onCreate() {  
 super.onCreate();  
 if (BuildConfig.*TINKER\_ENABLE*) { //这个值判断是否开启补丁热修复功能 在tinkerpatch.gradle文件修改  
 RepairManager.*With*(new RepairCallBack() {  
 @Override  
 public void repairResult(PatchResult patchResult) {  
 */\*\*  
 \* 补丁安装成功的话会返回  
 \*/* Log.*e*("TAG","修复 成功~~~~~~~");  
 }  
  
 @Override  
 public void onSuccessConfig(HashMap<String, String> hashMap) {  
 */\*\*  
 \* Tinker 平台上有新建 在线参数 就会有返回，如果没有新建 会在onFailConfig返回异常  
 \* 是否有新建不影响 补丁的更新 可以不管  
 \*/* Log.*e*("TAG","成功了 HashMap:" + hashMap.toString());  
 }  
  
 @Override  
 public void onFailConfig(Exception e) {  
 Log.*e*("TAG","失败了啊:" + e.toString());  
 }  
 });  
 }  
 }  
}

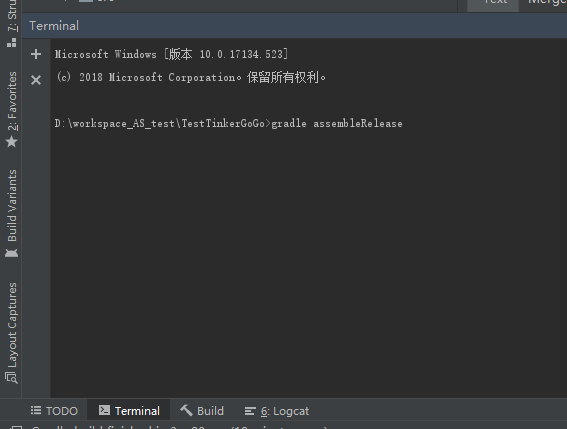
### 编写BUG

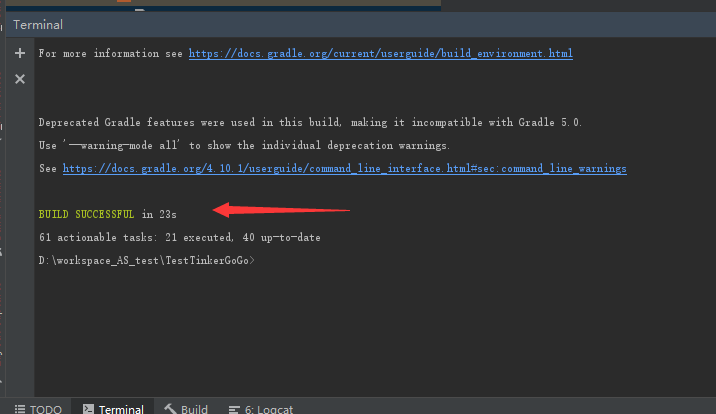
给textview 设置一个文本，当作这个页面有bug，稍后要修改这个文本当作补丁热修复成功。

public class MainActivity extends AppCompatActivity {  
  
 TextView tv;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 tv = findViewById(R.id.*tv*);  
 tv.setText("这是一个BUG 页面");  
 }  
}

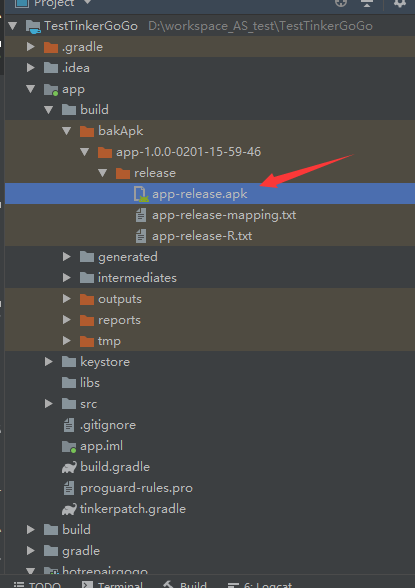
### 生成基础包

使用 **gradle assembleRelease** Task（在之前的配置gradle环境请自行实现）





基础包生成成功后会在build路径下找到自己的包



### 安装带BUG的apk



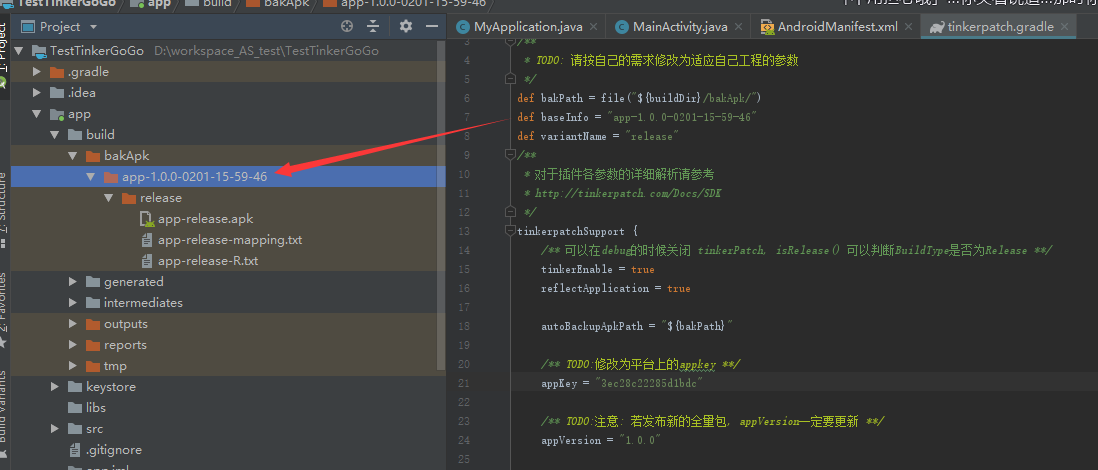
### 生成补丁包

#### 修改Activity（BUG）

public class MainActivity extends AppCompatActivity {  
  
 TextView tv;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 tv = findViewById(R.id.*tv*);  
 tv.setText("这是热修复后的包");  
 }  
}

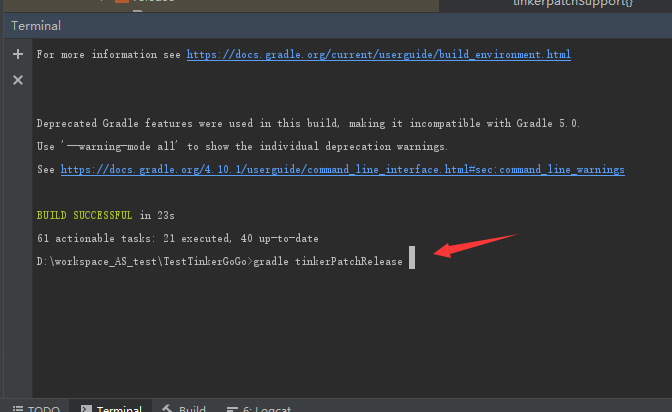
#### 定位全量包路径

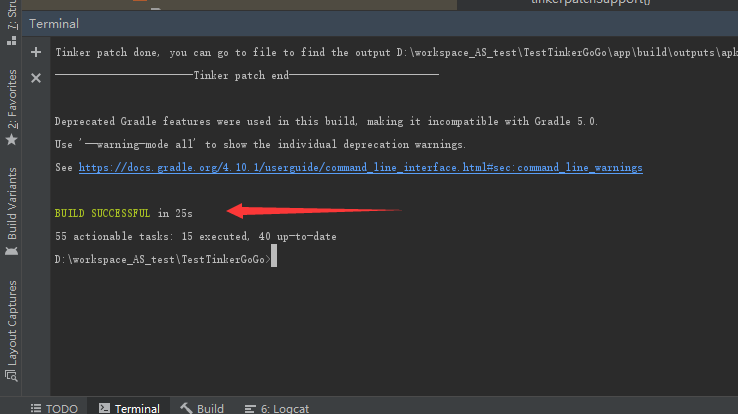
回到tinkerpatch.gradle 文件，baseInfo修改为刚才apk的目录文件夹，



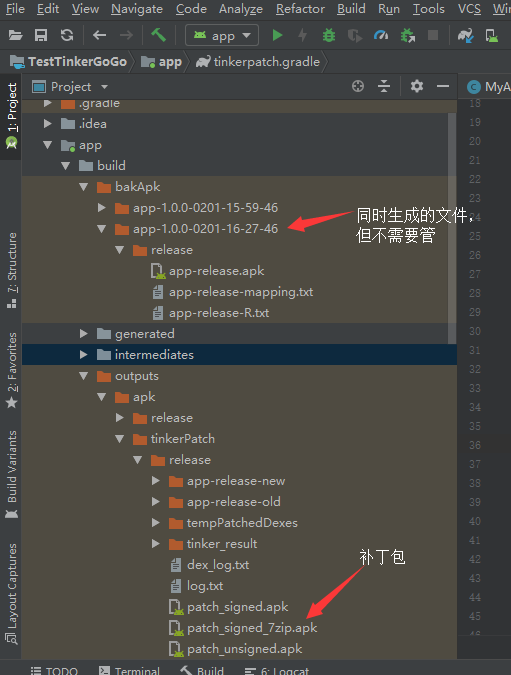
#### 打包

使用 **gradle tinkerPatchRelease** Task





在outputs/tinkerPatch 文件夹下找到patch\_signed\_7zip.apk，即就是所需要的补丁包

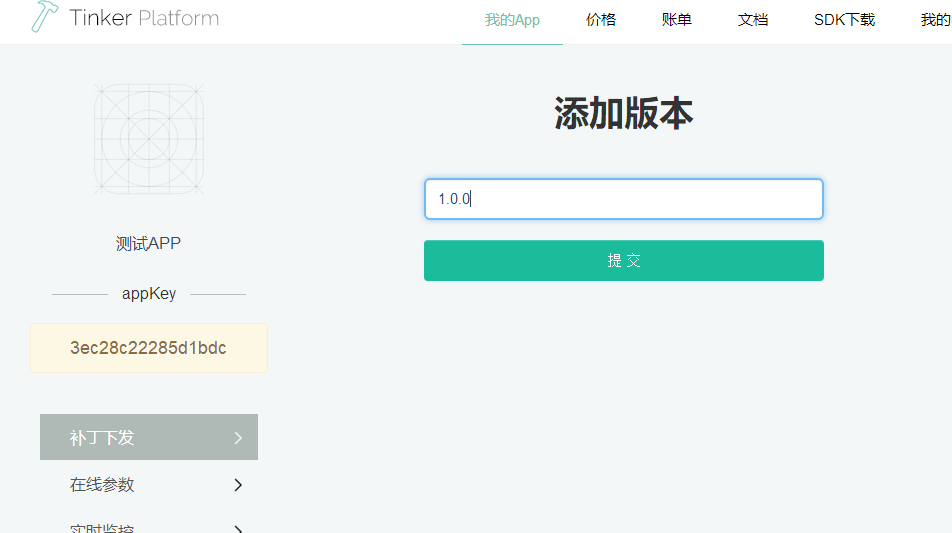


### 上传补丁包

添加APP版本



补丁的版本







### 测试效果

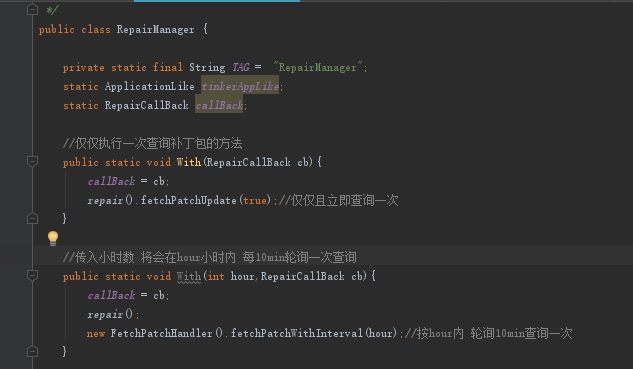
打开app，发现没变化，因为这次打开app只是下载了补丁，并没有重新编译补丁文件，再次打开app即可以看到效果。（安装也要时间，耐心点）

备注

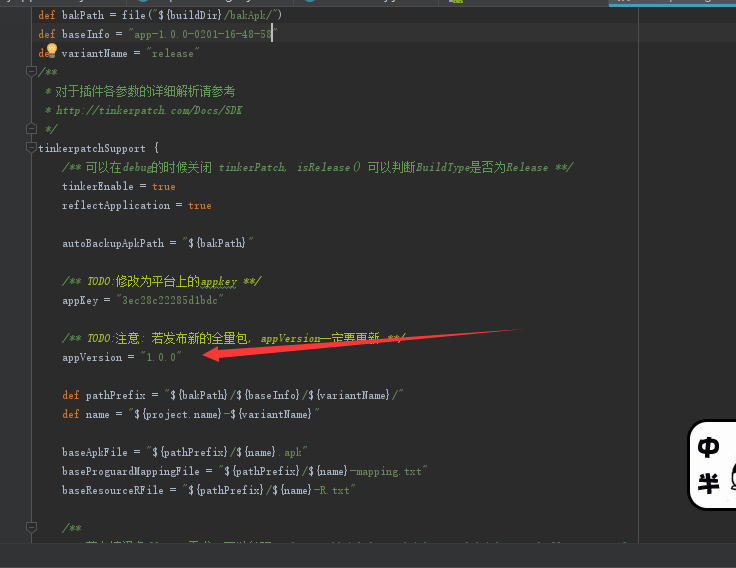
# 建议

## 关于查询补丁包的方案

上述demo使用的是RepairManager.With(RepairCallBack cb)方法 ，你可以使用另外一个重载的方法

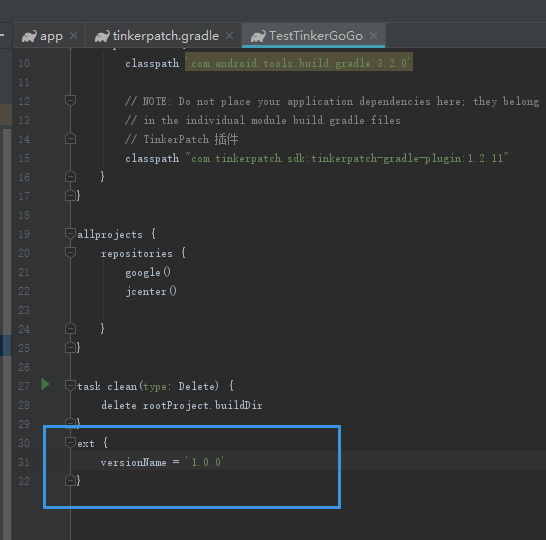


## 关于跟随基础包版本

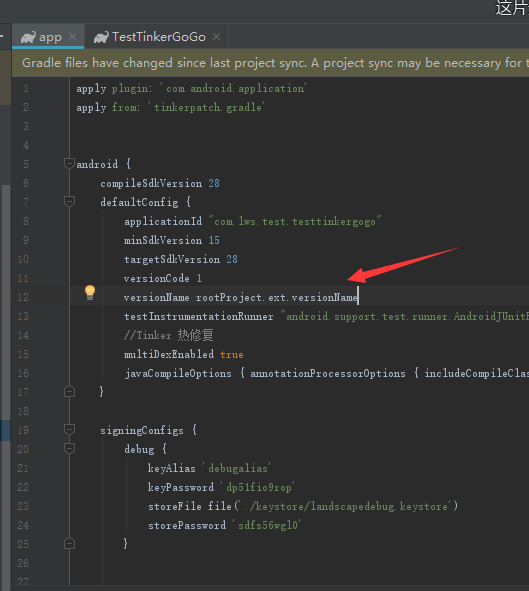
如果每次上线新的基础包到市场，那么appVersion也必须上升一个版本

在此为了以防万一忘记修改这里，建议这么做

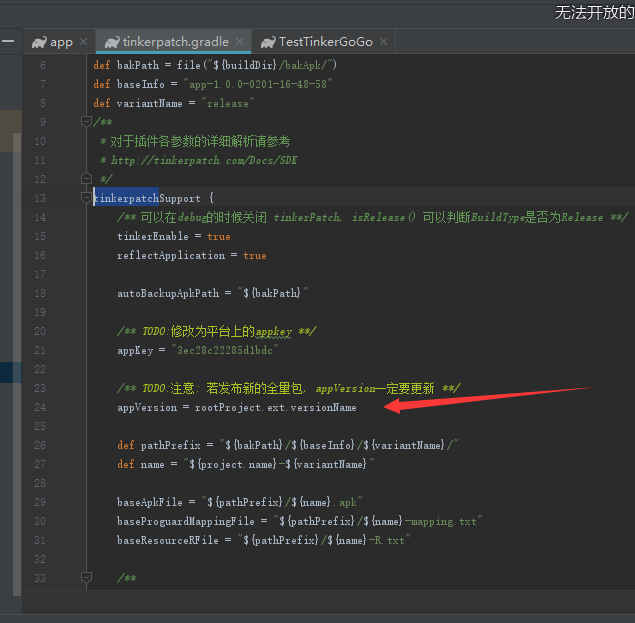
在工程的build.gradle文件 管理app的版本名称，



然后在app 的 build.gradle版本名称引用



同时tinkerpatch.gradle文件里



这样 下次更新全量包，只要修改工程的build.gradle文件的版本名即可，在官网平台上传补丁的时候，也是使用这个版本名，请注意

## 基础包文件夹请保存并管理好

整个文件夹保存，即bakApk文件夹，（起码保留到app-1.0.0-0201-15-59-46 文件夹），毕竟生成补丁包需要借助基础包，不然临时打补丁包时无法找到基础包使用，保存了基础包文件夹后， 下次无论哪个开发者需要打补丁，把保留的文件夹复制到工程里，都能依靠工程的tinkerpatch.gradle文件修改baseInfo即可生成补丁包。

