Android Studio混淆代码

# 一．混淆配置

minifyEnabled true // 是否混淆

proguardFiles 'proguard-rules.pro' //混淆配置文件

# 二.混淆规则

## 1.保证类名不被混淆

两个常用的混淆命令，注意一颗星表示只是保持该包下的类名，而子包下的类名还是会被混淆；两颗星表示把本包和所含子包下的类名都保持；

-keep class cn.hadcn.test.\*\*

-keep class cn.hadcn.test.\*

以上方法保证类名不被混淆，但里面的具体方法和变量命名还是变了

## 2.保证类名以及方法名，变量名不被混淆

# 保留我们使用的四大组件，自定义的Application等等这些类不被混淆

# 因为这些子类都有可能被外部调用

-keep public class \* extends android.app.Activity

-keep public class \* extends android.app.Appliction

-keep public class \* extends android.app.Service

-keep public class \* extends android.content.BroadcastReceiver

-keep public class \* extends android.content.ContentProvider

-keep public class \* extends android.app.backup.BackupAgentHelper

-keep public class \* extends android.preference.Preference

-keep public class \* extends android.view.View

-keep public class com.android.vending.licensing.ILicensingService

## 三．哪些不要混淆

一般以下情况都会不混淆：   
1.使用了自定义控件那么要保证它们不参与混淆   
2.使用了枚举要保证枚举不被混淆   
3.对第三方库中的类不进行混淆   
4.运用了反射的类也不进行混淆   
5.使用了 Gson 之类的工具要使 JavaBean 类即实体类不被混淆   
6.在引用第三方库的时候，一般会标明库的混淆规则的，建议在使用的时候就把混淆规则添加上去，免得到最后才去找   
7.有用到 WebView 的 JS 调用也需要保证写的接口方法不混淆，原因和第一条一样   
8.Parcelable 的子类和 Creator 静态成员变量不混淆，否则会产生 Android.os.BadParcelableException 异常

# 四、基本的混淆模板

**-keepclassmembers** class fqcn.of.javascript.interface.for.webview {  
 public \*;  
}  
  
#指定代码的压缩级别  
**-optimizationpasses** 5  
  
#包明不混合大小写  
**-dontusemixedcaseclassnames**#不去忽略非公共的库类  
**-dontskipnonpubliclibraryclasses** #优化 不优化输入的类文件  
**-dontoptimize** #预校验  
**-dontpreverify** #混淆时是否记录日志  
**-verbose** # 混淆时所采用的算法  
**-optimizations** !code/simplification/arithmetic,!field/\*,!class/merging/\*  
  
#保护注解  
**-keepattributes** \*Annotation\*  
  
# 保持哪些类不被混淆  
**-keep** public class \* extends android.app.Fragment  
**-keep** public class \* extends android.app.Activity  
**-keep** public class \* extends android.app.Application  
**-keep** public class \* extends android.app.Service  
**-keep** public class \* extends android.content.BroadcastReceiver  
**-keep** public class \* extends android.content.ContentProvider  
**-keep** public class \* extends android.app.backup.BackupAgentHelper  
**-keep** public class \* extends android.preference.Preference  
**-keep** public class com.android.vending.licensing.ILicensingService  
#如果有引用v4包可以添加下面这行  
**-keep** public class \* extends android.support.v4.app.Fragment  
  
  
#忽略警告  
**-ignorewarning**##记录生成的日志数据,gradle build时在本项目根目录输出##  
#apk 包内所有 class 的内部结构  
**-dump** proguard/class\_files.txt  
#未混淆的类和成员  
**-printseeds** proguard/seeds.txt  
#列出从 apk 中删除的代码  
**-printusage** proguard/unused.txt  
#混淆前后的映射  
**-printmapping** proguard/mapping.txt  
########记录生成的日志数据，gradle build时 在本项目根目录输出-end######  
  
#如果引用了v4或者v7包  
**-dontwarn** android.support.\*\*  
  
####混淆保护自己项目的部分代码以及引用的第三方jar包library-end####  
  
  
  
#保持 native 方法不被混淆  
**-keepclasseswithmembernames** class \* {  
 native <methods>;  
}  
  
#保持自定义控件类不被混淆  
**-keepclasseswithmembers** class \* {  
 public <init>(android.content.Context, android.util.AttributeSet);  
}  
  
#保持自定义控件类不被混淆  
**-keepclassmembers** class \* extends android.app.Activity {  
 public void \*(android.view.View);  
}  
  
**-keep** public class \* extends android.view.View {  
 public <init>(android.content.Context);  
 public <init>(android.content.Context, android.util.AttributeSet);  
 public <init>(android.content.Context, android.util.AttributeSet, int);  
 public void set\*(...);  
}  
  
#保持 Parcelable 不被混淆  
**-keep** class \* implements android.os.Parcelable {  
 public static final android.os.Parcelable$Creator \*;  
}  
  
#保持 Serializable 不被混淆  
**-keepnames** class \* implements java.io.Serializable  
  
#保持 Serializable 不被混淆并且enum 类也不被混淆  
**-keepclassmembers** class \* implements java.io.Serializable {  
 static final long serialVersionUID;  
 private static final java.io.ObjectStreamField[] serialPersistentFields;  
 !static !transient <fields>;  
 !private <fields>;  
 !private <methods>;  
 private void writeObject(java.io.ObjectOutputStream);  
 private void readObject(java.io.ObjectInputStream);  
 java.lang.Object writeReplace();  
 java.lang.Object readResolve();  
}  
  
#保持枚举 enum 类不被混淆  
**-keepclassmembers** enum \* {  
 public static \*\*[] values();  
 public static \*\* valueOf(java.lang.String);  
}  
  
**-keepclassmembers** class \* {  
 public void \*ButtonClicked(android.view.View);  
}  
  
#不混淆资源类  
**-keepclassmembers** class \*\*.R$\* {  
 public static <fields>;  
}  
  
#避免混淆泛型 如果混淆报错建议关掉  
#-keepattributes Signature  
  
#移除Log类打印各个等级日志的代码，打正式包的时候可以做为禁log使用，这里可以作为禁止log打印的功能使用，另外的一种实现方案是通过BuildConfig.DEBUG的变量来控制  
#-assumenosideeffects class android.util.Log {  
# public static \*\*\* v(...);  
# public static \*\*\* i(...);  
# public static \*\*\* d(...);  
# public static \*\*\* w(...);  
# public static \*\*\* e(...);  
#}  
  
  
  
###########################################  
# 保留我们自定义控件（继承自View）不被混淆  
**-keep** public class \* extends android.view.View{  
 \*\*\* get\*();  
 void set\*(\*\*\*);  
 public <init>(android.content.Context);  
 public <init>(android.content.Context, android.util.AttributeSet);  
 public <init>(android.content.Context, android.util.AttributeSet, int);  
}  
  
# 保留我们自定义控件（继承自ImageView）不被混淆  
**-keep** public class \* extends android.widget.ImageView{  
 \*\*\* get\*();  
 void set\*(\*\*\*);  
 public <init>(android.content.Context);  
 public <init>(android.content.Context, android.util.AttributeSet);  
 public <init>(android.content.Context, android.util.AttributeSet, int);  
}  
  
# 保留我们自定义控件（继承自RecyclerView）不被混淆  
**-keep** public class \* extends android.support.v7.widget.RecyclerView{  
 \*\*\* get\*();  
 void set\*(\*\*\*);  
 public <init>(android.content.Context);  
 public <init>(android.content.Context, android.util.AttributeSet);  
 public <init>(android.content.Context, android.util.AttributeSet, int);  
}  
  
# 保留我们自定义控件（继承自ViewGroup）不被混淆  
**-keep** public class \* extends android.view.ViewGroup{  
 \*\*\* get\*();  
 void set\*(\*\*\*);  
 public <init>(android.content.Context);  
 public <init>(android.content.Context, android.util.AttributeSet);  
 public <init>(android.content.Context, android.util.AttributeSet, int);  
}  
  
# 保留我们自定义控件（继承自PtrClassicFrameLayout 上拉刷下，下拉加载）不被混淆  
**-keep** public class \* extends com.chanven.lib.cptr.PtrClassicFrameLayout{  
 \*\*\* get\*();  
 void set\*(\*\*\*);  
 public <init>(android.content.Context);  
 public <init>(android.content.Context, android.util.AttributeSet);  
 public <init>(android.content.Context, android.util.AttributeSet, int);  
}  
  
  
# 保留我们自定义控件（继承自RadioButton）不被混淆  
**-keep** public class \* extends android.widget.RadioButton{  
 \*\*\* get\*();  
 void set\*(\*\*\*);  
 public <init>(android.content.Context);  
 public <init>(android.content.Context, android.util.AttributeSet);  
 public <init>(android.content.Context, android.util.AttributeSet, int);  
}  
  
# 保留我们自定义控件（switch）不被混淆  
**-keep** public class \* extends android.widget.Switch{  
 \*\*\* get\*();  
 void set\*(\*\*\*);  
 public <init>(android.content.Context);  
 public <init>(android.content.Context, android.util.AttributeSet);  
 public <init>(android.content.Context, android.util.AttributeSet, int);  
}  
  
  
# 保留我们自定义控件（FrameLayout）不被混淆  
**-keep** public class \* extends android.widget.FrameLayout{  
 \*\*\* get\*();  
 void set\*(\*\*\*);  
 public <init>(android.content.Context);  
 public <init>(android.content.Context, android.util.AttributeSet);  
 public <init>(android.content.Context, android.util.AttributeSet, int);  
}  
  
# 保留我们自定义控件（RelativeLayout）不被混淆  
**-keep** public class \* extends android.widget.RelativeLayout{  
 \*\*\* get\*();  
 void set\*(\*\*\*);  
 public <init>(android.content.Context);  
 public <init>(android.content.Context, android.util.AttributeSet);  
 public <init>(android.content.Context, android.util.AttributeSet, int);  
}  
  
# 保留我们自定义控件（RelativeLayout）不被混淆  
**-keep** public class \* extends android.app.Dialog{  
 \*\*\* get\*();  
 void set\*(\*\*\*);  
 public <init>(android.content.Context);  
 public <init>(android.content.Context, android.util.AttributeSet);  
 public <init>(android.content.Context, android.util.AttributeSet, int);

}

网上的混淆

#############################################

#

# 对于一些基本指令的添加

#

#############################################

# 代码混淆压缩比，在0~7之间，默认为5，一般不做修改

-optimizationpasses 5

# 混合时不使用大小写混合，混合后的类名为小写

-dontusemixedcaseclassnames

# 指定不去忽略非公共库的类

-dontskipnonpubliclibraryclasses

# 这句话能够使我们的项目混淆后产生映射文件

# 包含有类名->混淆后类名的映射关系

-verbose

# 指定不去忽略非公共库的类成员

-dontskipnonpubliclibraryclassmembers

# 不做预校验，preverify是proguard的四个步骤之一，Android不需要preverify，去掉这一步能够加快混淆速度。

-dontpreverify

# 保留Annotation不混淆

-keepattributes \*Annotation\*,InnerClasses

# 避免混淆泛型

-keepattributes Signature

# 抛出异常时保留代码行号

-keepattributes SourceFile,LineNumberTable

# 指定混淆是采用的算法，后面的参数是一个过滤器

# 这个过滤器是谷歌推荐的算法，一般不做更改

-optimizations !code/simplification/cast,!field/\*,!class/merging/\*

#############################################

#

# Android开发中一些需要保留的公共部分

#

#############################################

# 保留我们使用的四大组件，自定义的Application等等这些类不被混淆

# 因为这些子类都有可能被外部调用

-keep public class \* extends android.app.Activity

-keep public class \* extends android.app.Appliction

-keep public class \* extends android.app.Service

-keep public class \* extends android.content.BroadcastReceiver

-keep public class \* extends android.content.ContentProvider

-keep public class \* extends android.app.backup.BackupAgentHelper

-keep public class \* extends android.preference.Preference

-keep public class \* extends android.view.View

-keep public class com.android.vending.licensing.ILicensingService

# 保留support下的所有类及其内部类

-keep class android.support.\*\* {\*;}

# 保留继承的

-keep public class \* extends android.support.v4.\*\*

-keep public class \* extends android.support.v7.\*\*

-keep public class \* extends android.support.annotation.\*\*

# 保留R下面的资源

-keep class \*\*.R$\* {\*;}

# 保留本地native方法不被混淆

-keepclasseswithmembernames class \* {

native <methods>;

}

# 保留在Activity中的方法参数是view的方法，

# 这样以来我们在layout中写的onClick就不会被影响

-keepclassmembers class \* extends android.app.Activity{

public void \*(android.view.View);

}

# 保留枚举类不被混淆

-keepclassmembers enum \* {

public static \*\*[] values();

public static \*\* valueOf(java.lang.String);

}

# 保留我们自定义控件（继承自View）不被混淆

-keep public class \* extends android.view.View{

\*\*\* get\*();

void set\*(\*\*\*);

public <init>(android.content.Context);

public <init>(android.content.Context, android.util.AttributeSet);

public <init>(android.content.Context, android.util.AttributeSet, int);

}

# 保留Parcelable序列化类不被混淆

-keep class \* implements android.os.Parcelable {

public static final android.os.Parcelable$Creator \*;

}

# 保留Serializable序列化的类不被混淆

-keepclassmembers class \* implements java.io.Serializable {

static final long serialVersionUID;

private static final java.io.ObjectStreamField[] serialPersistentFields;

!static !transient <fields>;

!private <fields>;

!private <methods>;

private void writeObject(java.io.ObjectOutputStream);

private void readObject(java.io.ObjectInputStream);

java.lang.Object writeReplace();

java.lang.Object readResolve();

}

# 对于带有回调函数的onXXEvent、\*\*On\*Listener的，不能被混淆

-keepclassmembers class \* {

void \*(\*\*On\*Event);

void \*(\*\*On\*Listener);

}

# webView处理，项目中没有使用到webView忽略即可

-keepclassmembers class fqcn.of.javascript.interface.for.webview {

public \*;

}

-keepclassmembers class \* extends android.webkit.webViewClient {

public void \*(android.webkit.WebView, java.lang.String, android.graphics.Bitmap);

public boolean \*(android.webkit.WebView, java.lang.String);

}

-keepclassmembers class \* extends android.webkit.webViewClient {

public void \*(android.webkit.webView, jav.lang.String);

}