1. 打印日志的关键流程：

-dontpreverify

#混淆时是否记录日志

-verbose

#apk包内所有的class的内部结构

-dump class\_files.txt

#未混淆的类和成员

-printseeds seeds.txt

#列出从类中删除的代码

-printusage unused.txt

#混淆前后的映射

-printmapping mapping.txt

2、一下情况不能使用混淆。

（1）反射中使用的元素，需要保证类名、方法名、属性名不变，否则混淆后会反射不了

（2）最好不让一些bean对象混淆

（3）四大组件不建议混淆，四大组件必须在AndroidManifest 中注册声明，而混淆后包名会发生改变，这样不符合四大组件的注册机制。

（4）注解不能混淆，很多场景下被用于在运行时发射一些元素。

（5）不嗯给你混淆枚举中的value 、valueOf方法，因为这两个方法是静态添加到代码中运行，也会被反射使用，所以无法混淆这两种方法

（6）JNI调用Java方法，需要通过类名和方法名构成的地址形成

（7）Java使用Native 方法，Native 是c/c++ 编写的，方法是无法一同混淆的

（8）JS 调用Java方法

（9）WebView中JavaScript的调用方法不能混淆

（10） 第三方库建议使用其自身的混淆规则

（11）Parcelable 的子类和Creator 的静态成员变量不混淆，否则会出现Android.os.BadParcelableException 异常

等等

**组件化混淆**

多个module创建之后，都会自带一个proguard-rule.pro的自定义混淆文件。每个module都可以有自己的混淆规则；但在组件化中，如果每个module都用自身混淆规则就会出现问题。解决方法：

1. 只在Application module中生成的时候混淆一次。其他module都关闭混淆，混淆规则都放在Application module 中
2. 当Application module 混淆的是够，启动一个命令将引用的多个module的proguard-rule.pro文件合成，然后覆盖Application module中的混淆文件,这种方式可以将混淆条件解耦到每个module中，但是需要编写Gradle 命令来配置操作，每次生成都会添加合成操作，也会对编译效率造成影响
3. Library module 自身拥有将proguard-rule.pro文件打包到aar中的设置，如下:

defaultConfig{

consumerProguardFiles ‘proguard-rules.pro’

}

consumerProguardFiles 的属性:

1》 proguard.txt文件会包含在aar文件中

2》这些proguard配置会在混淆时使用

3》此配置针对aar进行混淆配置

4》此配置支队库文件有效，对应用程序无效

当Application module将全部的代码汇总混淆的时候，Llibrary module会被打包为release aar，然后被引用汇总，通过proguard.txt规则各自混淆，保证混淆一次.