Sumário

1 Segurança e Otimização de Aplicativos APK

Objetivos: - Encolhimento (shrinking) - Segurança (Ofuscação) - Otimização (Apks Menores e mais performáticos)

1.1 O que é?

ProGuard é um projeto Open-Source Java. Isso significa que não se limita apenas ao Android e você usá-lo em suas aplicações. Ele foi importado para o AndroidSDK para que funcione nessa plataforma também fazendo as 3 tarefas anteriores.

- Encolhimento: Remover códigos desnecessários e duplicados.
- Ofuscação: Trocar os nomes das propriedades para ficarem mais enxutas.
- Otimização do ByteCode Java.

1.2 Como funciona?

Há dezenas de formas para definir um arquivo de configuração do ProGuard e vamos falar dele mais pra frente. Agora, o que precisa ficar ligado é que o ProGuard vai criar e otimizar as classes de 2 maneiras. 1. Criar um pacote com suas classes a as dependências de terceiros como classes Helpers, Frameworks de Networking, Imagens e etc. (Lib.jar | Lib.aar) 2. Criar um pacote com as classes do Framework android (android.jar) de forma separada que o seu projeto precisa acessar (Context, Activity, etc).

1.2.1 Execução do ProGuard

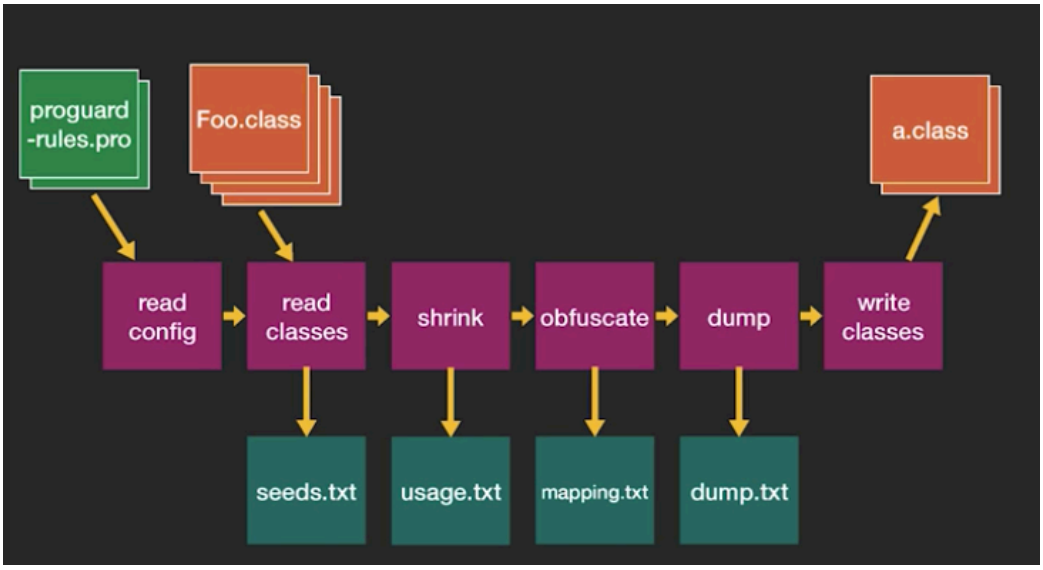
1. Como o ProGuard otimiza e ofusca quase tudo no projeto, você define o que ele NÃO DEVE mexer e manter do jeito que está. Por exemplo: em aplicações Java, nós temos o entry-point sendo o `static void main`. No Android, esse ponto de entrada é a Activity com método onCreate. Por isso, você deve especificar para manter esses caras intactos no arquivo **seed.txt** co.tiagoaguiar.demo.MainActivity co.tiagoaguiar.demo.MainActivity: onCreate(android.os.Bundle)

2. Depois que ele encontrar o EntryPoint, o ProGuard irá encolher o seu projeto tentando remover código que não estão sendo usados. Então, ele faz uma leitura de todo o seu código para encontrar variáveis, métodos e classes desnecessárias para remover.

Com isso ele cria um arquivo chamado **usage.txt** que mostra código NÃO USADOS (é eu sei, está escrito usage e deveria ser unused. Mas!) co.tiagoaguiar.demo.Helper: static String abc co.tiagoaguiar.demo.ClasseNaoUsada

3. Agora chegou a hora de ofuscar o seu código. Nesse ponto, o ProGuard irá renomear suas classes, métodos e variáveis para algo como *a, b, c, etc* e armazenar as mudanças no arquivo **mapping.txt** . Guarde esse arquivo pois iremos falar mais dele e o por quê ele é tão importante assim.

4. Por fim, ainda temos o arquivo **dump.txt** que é um arquivo enorme sobre o seu projeto e cria o **.jar** do seu projeto com as classes.



1.3 Regras a Serem Mantidas

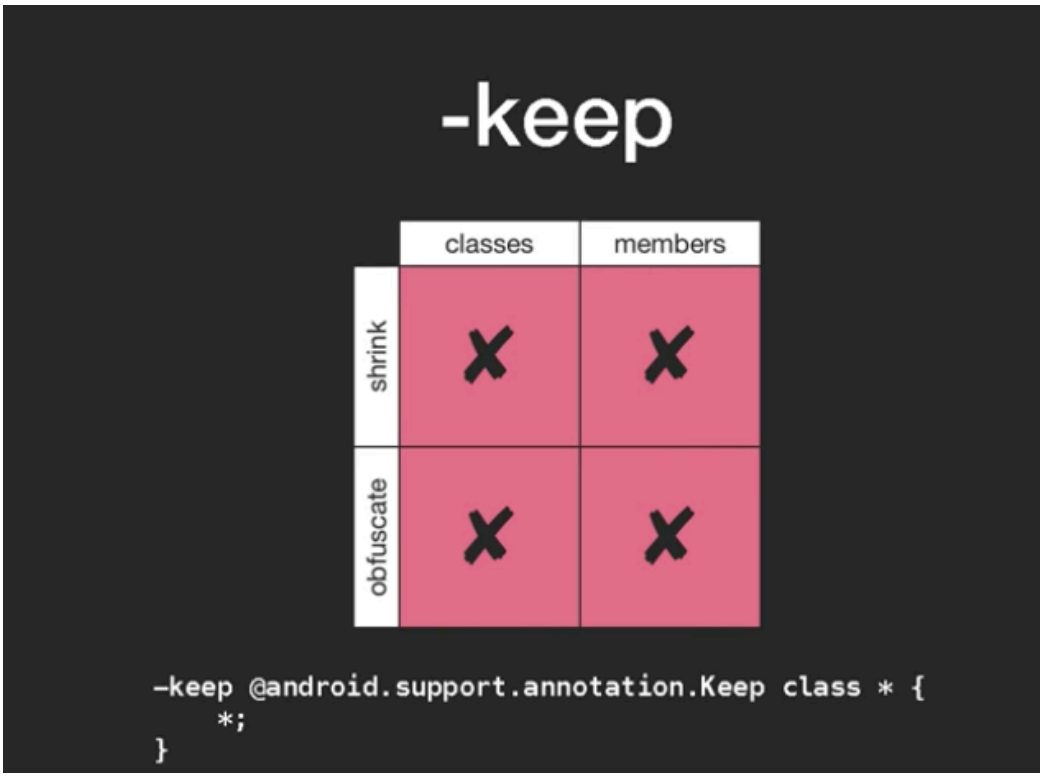
As regras são definidas nos arquivos: - proguardFiles 'proguard-rules.pro' - proguardFiles getDefaultProguardFile('proguard-android.txt') - consumerProguardFiles 'proguard-rules.pro' > Usada por Libs Veja o README da biblioteca para ver como implementar o proguard dela.

Ativando o ProGuard

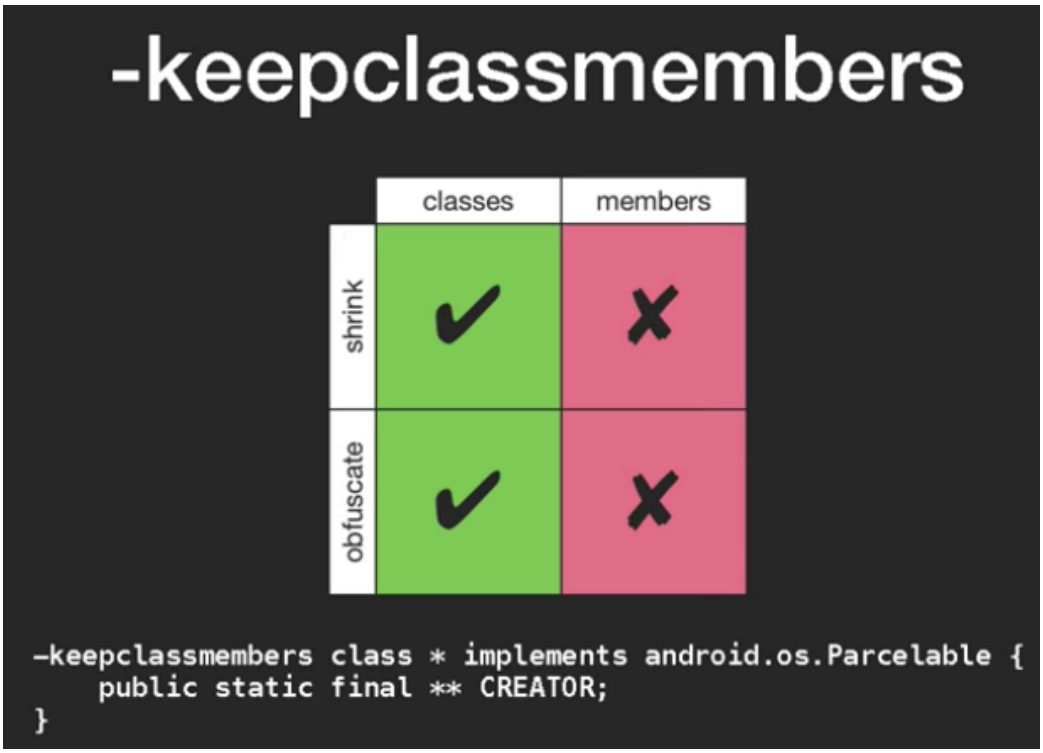
```
release {
    minifyEnabled true
}
```

- Entry-Point
- Reflection

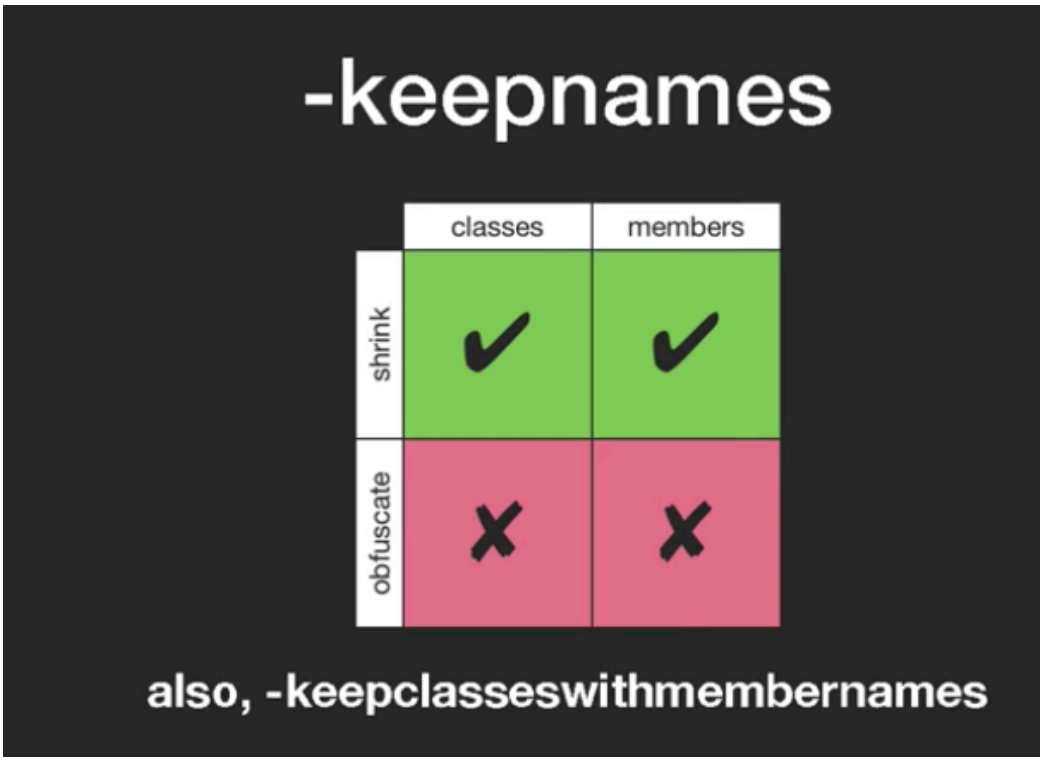
Há varias maneiras (ver link proguard), mas o essencial é: - -keep



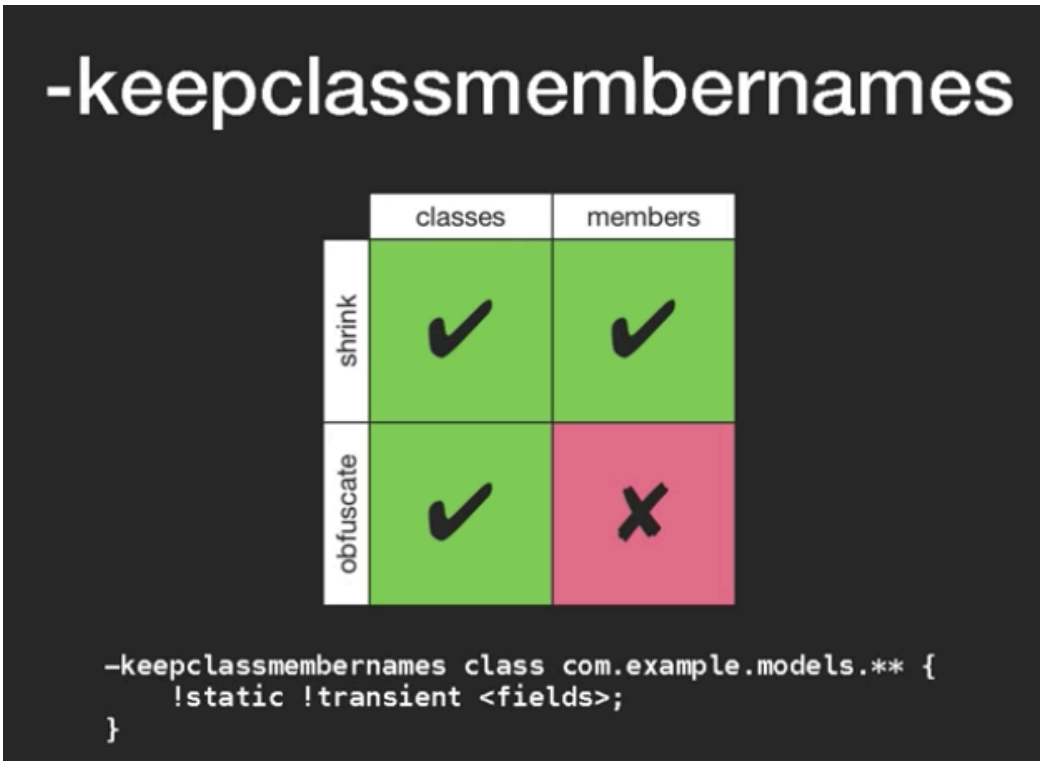
- -keepclassmembers


-keepclassmembers
classes
members
shrink
obfuscate
-keepclassmembers class * implements android.os.Parcelable {
public static final ** CREATOR;
}

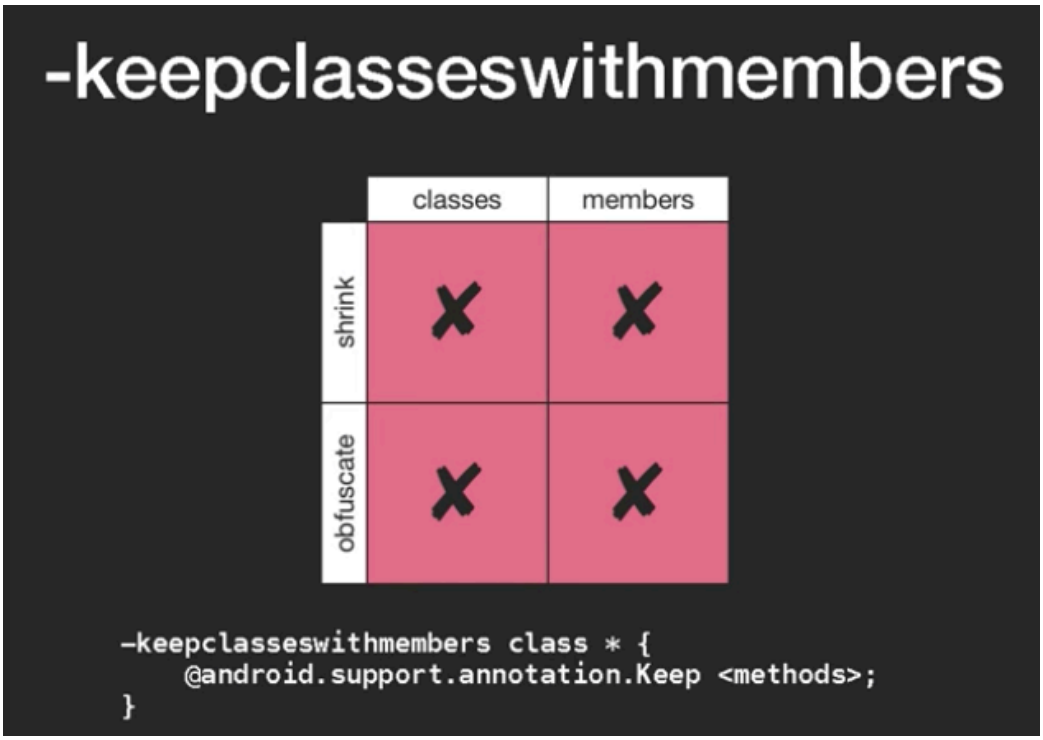
- -keepnames



- -keepclassmembersnames

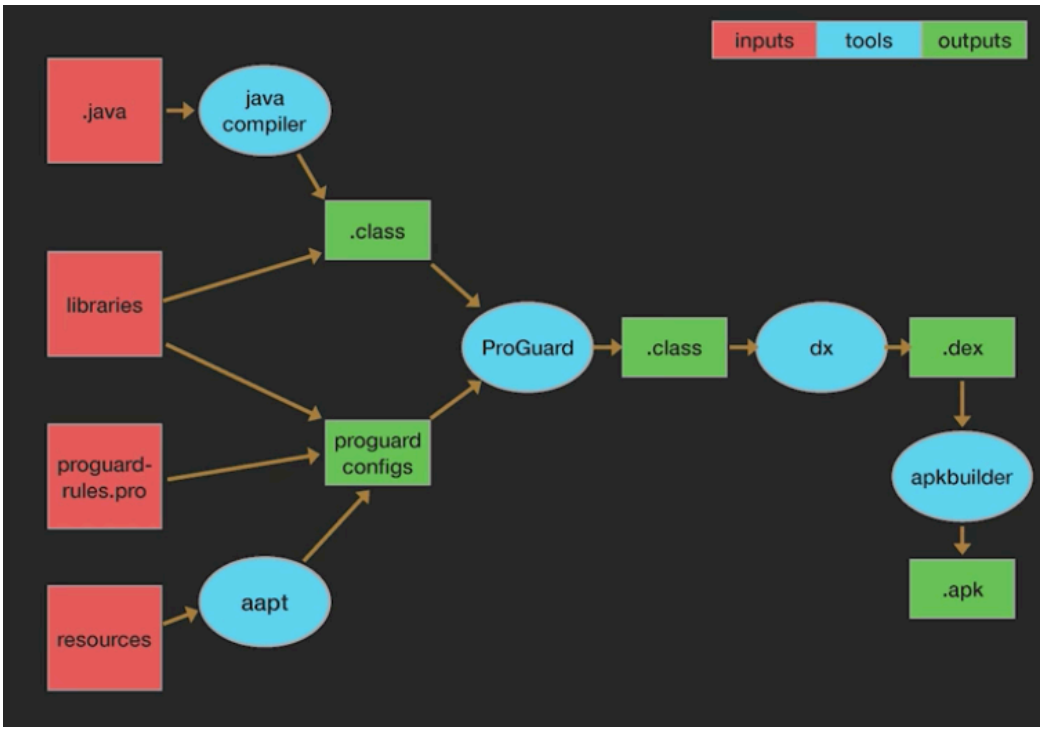


- -keepclasseswithmemebers

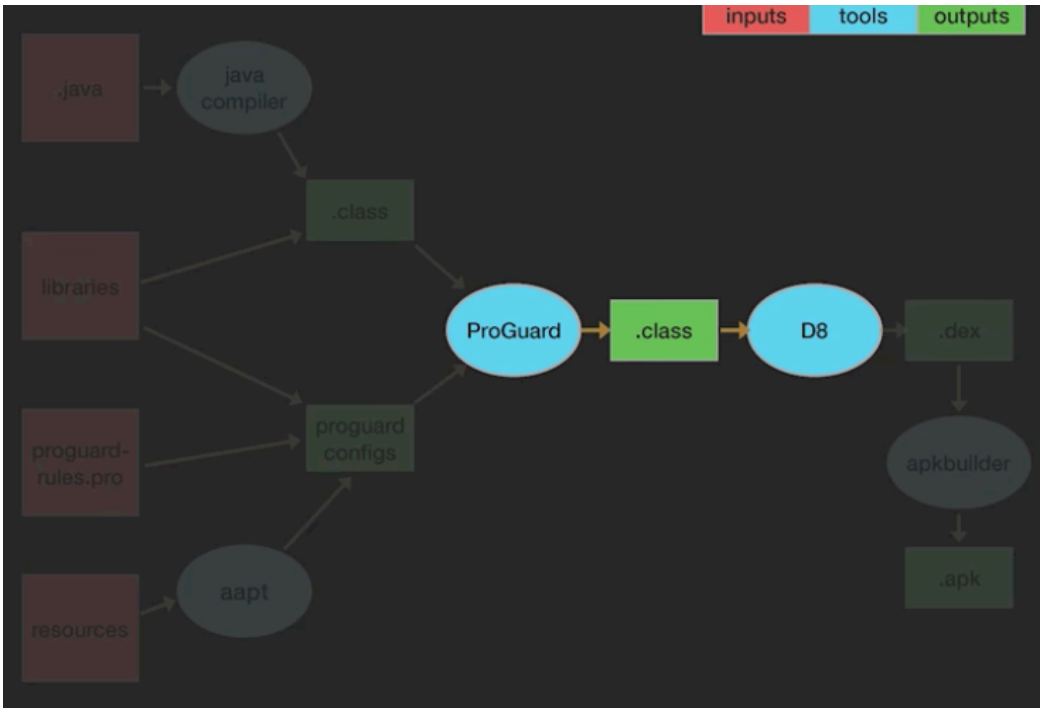


- -keepclasseswithmemebersnames

Todo o processo aqui.



Dx agora foi trocado por D8.



E o Proguard Foi substituído pelo R8 Proguard > .class > D8 = R8 / D8 Faz a mesma coisa mas melhor! Removeram o `dumps.txt` Outros nomes como "shrinking" agora é "three-shaking". "obfuscation" agora é "minification".

Para ativar, acesse `gradle.properties` e adicionar o `android.enableR8=true`. Desative se tiver problemas (ainda em desenvolvimento).

1.4 Minificando Classes e suas Propriedades

Quando usamos o pro guard. Classes podem se chamar a, b, c etc. Dificultando a engenharia reversa. Além de outras otimizações.

Estimasse que há X porcento de apps publicados que não se preocupam com segurança!!

1.5 Muitos Bugs

"Soluções" que sugerem - "…Desabilite o ProGuard que tudo vai funcionar" - - dontwarn * NÃO FAÇA ISSO

1.6 Erros e Debugging

Lembra do arquivo **mapping.txt**. Ele será útil aqui para desofuscar o seu stack trace e encontrar os eventuais bugs no seu projeto. Por essa razão ao fazer o upload do APK na loja, temos a opção de adicionar o **mapping.txt**. Isso facilita você a depurar e ainda manter o seu projeto seguro.

Outra maneira de usar o mapping.txt é com um script que se encontra dentro do próprio SDK: `$ANDROID_SDK/tools/proguard/bin/retrace.sh mapping.txt stack trace.txt` (Há a versão UI no `proguardgui.sh`)

1.7 Estou 100% seguro?

Não. Mas você tem uma camada a mais, além das existente para conseguir evitar uma engenharia reversa e de quebra, ter um apk menor.

> Média de tamanho dos apps nas lojas